Theoretical Computer Science 99 (1992) 141-155
Elsevier

Monads and algebras in the semantics of partial data types*

Philip S. Mulry**
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Communicated by P. Freyd
Received June 1989
Revised May 1990

Introduction

Programming language semantics plays an important role in the development of software technology by providing a coherent approach to the design and implementation of programming languages. By avoiding ad hoc techniques, such an approach seeks to provide an environment suitable for the interpretation of powerful yet flexible languages capable of greater modularity and polymorphic behavior. This in turn should lead to greater code reuse, reliability and maintenance so crucial to the future developement of large software systems.

Algebraic methods have long provided a plenitude of roles in both defining issues and solving problems in the foundations of programming language semantics. In this paper we focus attention on the use of algebraic structures, particularly categories of algebras and monads, to describe partial data types and their corresponding computations.

It has become increasingly evident with the proliferation of semantic categories that no one category is ideal for all purposes. Instead, what is proposed in this paper in the context of partial types is a *systematic approach* which constructs several categories of algebras and their corresponding adjunctions. The semantic content is then derived not only from the individual categories but from their functorial relationships.

Recent attempts at dealing with partial operations on data types have led to the construction of categories of partial maps [11, 12]. Earlier work had focused on the

* Partially supported by NSF Grants CCR8706333, CCR9002251.
** Permanent Address: Department of Computer Science, Colgate University, Hamilton, NY 13346.

use of refinements of the partial map classifier in a topos setting to describe partial data types and operations [7]. Working in a cartesian closed category and utilizing the refinements existing in the associated topos, we exploit their monadic structure to construct categories of algebras of partial types including the well-known Kleisli and Eilenberg-Moore categories of algebras. In the case of the Kleisli category of algebras we show a coreflection exists from the Kleisli category to our original category. Utilizing special properties of the monad, a model of a partial computational calculus is produced which is governed by a call-by-value semantics. This has connections with recent work in [6]. Also, we show that partial cartesian closed categories arise in this manner, thus providing a general framework for results in [11].

Turning in a different direction we address the troublesome issue of divergence by constructing Eilenberg-Moore categories of algebras for various monadic refinements. In addition to being injective objects, the algebra objects are well behaved with respect to a notion of divergence imposed by the monad. In particular, in the case of semantic categories the corresponding categories of algebras form reflective subcategories. Also, the coreflection generated by the Kleisli category factors through this model connecting up the two approaches to partiality. Important examples such as domains, effective domans and partial equivalence relations with requirements of continuity and effectivity are easily interpreted in this setting.

In this paper we have proposed a general program for interpreting partial data types by constructing categories of algebras for special monads. The general ideas outlined in this approach underlie much of the fundamental work in standard semantic examples such as domains, yet work equally well in other settings as well. Much work remains to be done as the rich structure of the monadic refinements has barely been explored or applied. Future work will also focus on the intrinsic ordering on the partial data types implicit in the constructions as well as the strong connections of this work to operational semantics.

1. Partial map classifiers and monads

In this section we provide an introduction to various ideas and constructions such as categories of algebras and refinements of the partial map classifier, utilized in the remaining parts of the paper. The treatment is not intended to be exhaustive but should be sufficient to understand what follows.

Let $\boldsymbol{C}$ be an arbitrary category with endofunctor T.

Definition 1.1. $\langle T, \eta, \mu \rangle$ is a monad on $\boldsymbol{C}$ where there exist natural transformations $\eta : id \to T$ and $\mu : T^2 \to T$ so that

(1) $\mu \circ \eta_T = id_T = \mu \circ T\eta$,

(2) $\mu \circ \mu_T = \mu \circ T\mu$.

Example 1.2. Let G be a group and T the endofunctor $G \times \_$ of **SET** (i.e. $TX = G \times X$). T is a monad where $\eta(X) = (1, X)$ and $\mu(g_1, g_2, X) = (g_1 g_2, X)$.

Definition 1.3. $X \in \boldsymbol{C}$ is an algebra for monad T if there exists a map $TX \xrightarrow{h} X$ in $\boldsymbol{C}$ so that $h \circ \eta = id$ and $h \circ Th = h \circ \mu$. The category of T-algebras $\boldsymbol{C}^T$ has objects $\langle X, h \rangle$ and arrows $\langle X, h \rangle \to \langle Y, h' \rangle$ so that the diagram commutes

$$\begin{array}{ccc} TX & \xrightarrow{Tf} & TY \\ \downarrow{\scriptstyle h} & & \downarrow{\scriptstyle h'} \\ X & \xrightarrow{f} & Y. \end{array}$$

Example 1.4. Returning to Example 1.2, set X is a T algebra when it is a G-set, i.e. a set with action h.

Note. It has become common in computer science to also call a T-algebra on the category $\boldsymbol{C}$ a pair $\langle X, h \rangle$, $TX \xrightarrow{h} X$, where T is an arbitrary endofunctor of $\boldsymbol{C}$. Consequently, this weaker notion does not have the equational requirements $h \circ \eta = id$ or $h \circ Th = h \circ \mu$. For the purposes of this paper a T-algebra will mean exclusively the stronger notion.

In addition to $\boldsymbol{C}^T$ there is another canonical choice for a category associated to monad T, namely $\boldsymbol{C}_T$, the Kleisli category of T. Objects of $\boldsymbol{C}_T$ are often denoted by X_T where $X \in \boldsymbol{C}$. Arrows in $\boldsymbol{C}_T$ correspond to maps $X \xrightarrow{f} TY$ in $\boldsymbol{C}$ or equivalently maps $TX \to TY$ denoted $f^\flat$ where $f^\flat = \mu_Y \circ Tf$ and $f^\flat \circ \eta_X = f$. We say that map $TX \xrightarrow{g} TY$ satisfies condition $(\flat)$ if $g = f^\flat$ for some $X \xrightarrow{f} TY$. The category $\boldsymbol{C}_T$ can be fully embedded inside $\boldsymbol{C}^T$ via the monad T and in this setting can be thought of as the subcategory of free algebras. By abuse of notation we often denote objects in $\boldsymbol{C}_T$ by $X, Y, \ldots$ as in $\boldsymbol{C}$ where the context should make clear which category is being assumed. For example, by the above we have the correspondence $Hom_{\boldsymbol{C}_T}(X, Y) \cong Hom_{\boldsymbol{C}}(X, TY) \cong Hom^\flat_{\boldsymbol{C}}(TX, TY)$.

Some care must be taken with composition in $\boldsymbol{C}_T$ so that domains and codomains agree. Thus for example the composition $X \xrightarrow{f} TY$ with $Y \xrightarrow{g} TZ$ is $(g^\flat \circ f)^\flat$. For any Y, η_Y acts as the identity map so we have the equation $(g^\flat \circ \eta_Y)^\flat = g^\flat$. Further details about this category can be found in [4] or [5].

Notation 1.5. For notational convenience we will denote $\boldsymbol{C}^T$ by $\mathscr{A}\boldsymbol{C}$ and $\boldsymbol{C}_T$ by $\mathscr{K}\boldsymbol{C}$ where the monad should be clear from the context.

We now focus our attention on a particular monad which plays a special role in the semantics of partial computation. Recall that a topos $\mathscr{E}$ is a cartesian closed

category with finite limits (i.e., $\mathscr{E}$ has a terminal object and pullbacks) and a truth value object Ω acting as a subobject classifier. In particular this implies for any $Y \in \mathscr{E}$ there is an injective object, $\tilde{Y}$, called the partial map classifier for Y, along with a mono map $Y \xrightarrow{\eta_Y} \tilde{Y}$ which satisfies the following universal property: for any partial map $(X, m) \xrightarrow{f} Y$ (i.e., a pair of maps

$$\begin{array}{ccc} A & \xrightarrow{f} & Y \\ \downarrow{\scriptstyle m} & & \\ X & & \end{array}$$

where m is mono) there exists a unique extension $X \xrightarrow{\bar{f}} \tilde{Y}$ making the diagram a pullback:

$$\begin{array}{ccc} A & \xrightarrow{f} & Y \\ \downarrow{\scriptstyle m} & & \downarrow{\scriptstyle \eta} \\ X & \xrightarrow{\bar{f}} & \tilde{Y}. \end{array}$$

Example 1.6. In the category Set, $\tilde{Y} = Y \cup \{*\}$ and for $x \in X$,

$$\bar{f}(x) = \begin{cases} fx & \text{if } x \in A, \\ * & \text{if } x \notin A. \end{cases}$$

The universal property of partial map classifiers makes $(\tilde{\ })$ into a functor on $\mathscr{E}$. In fact, this functor can exist in more general settings, but as we shall point out, its meaning remains strongly tied to an associated topos.

The connection between partial map classifiers and computation was not exploited for some time because in general there are too many partial maps in a topos. The decisive step was first taken in [7] where it was recognized that partial map classifiers exist for different truth value objects. This idea was exploited in an effective setting to provide a framework for the description of computable objects and partially computable maps.

The objects which we call *partial truth value objects* are subobjects of Ω, $\Omega_p \hookrightarrow \Omega$, satisfying (i) true $\in \Omega_p$ and (ii) Ω_p is closed under subobject composition (i.e. if $A \hookrightarrow B$ and $B \hookrightarrow C$ are both classified by Ω_p then so is the composite $A \hookrightarrow C$). More recently this process had been axiomatized in [12] where a different characterization of partial truth value object appears and is called a dominance.

The following result is implicit in [7].

Theorem 1.7. *For any partial truth value subobject Ω_p of Ω there exists a corresponding partial map classifier $(\tilde{})_p$.*

Proof. A partial truth value subobject Ω_p generates for any $Y \in \mathscr{E}$ the following pullback diagram:

$$\begin{array}{ccc} Y & \longrightarrow & 1 \\ \downarrow & & \downarrow{\scriptstyle true} \\ \tilde{Y}_p & \longrightarrow & \Omega_p \\ \downarrow & & \downarrow \\ \tilde{Y} & \longrightarrow & \Omega. \end{array}$$

Thus a map $X \to \tilde{Y}_p$ corresponds to a partial map $(X, m) \xrightarrow{f} Y$ where the domain of f, $A \xrightarrow{m} X$, is classified by Ω_p.

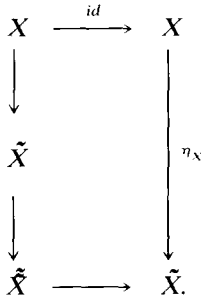
The proof indicates the intimate connection between the refinement of the partial map classifier and the corresponding refinement in the domain of the partial map, namely the subobject $A \hookrightarrow X$ is not only classified by Ω but by Ω_p. When we wish to emphasize this we will use the notation $A_p \hookrightarrow X$. Note in particular that since $true \in \Omega_p$, $Y_p \xrightarrow{\eta_Y} \tilde{Y}_p$. Examples of the next theorem will emphasize the utility of these notions. First, however, we now make explicit the connection to our earlier work.

Theorem 1.8. *$(\tilde{})$ and more generally $(\tilde{})_p$ are monads on $\mathscr{E}$.*

Proof. This is a well-known result for $(\tilde{})$ and the result carries over easily for $(\tilde{})_p$. A short proof would simply consist of noting that there exists an adjunction $\mathscr{E} \underset{(\tilde{})_p}{\overset{i}{\rightleftarrows}} Part(\mathscr{E})$ where $i \dashv (\tilde{})_p$ and $Part(\mathscr{E})$ is the category with objects in $\mathscr{E}$ and arrows consisting of p-partial maps. The left adjoint is the inclusion map. Since adjunctions generate monads we are done. □

For notational ease we denote a partial map from A to B by $A \rightharpoonup B$ or $A_p \rightharpoonup B$ if we wish to emphasize the map corresponds to a map $A \to \tilde{B}_p$. We point out some particulars of the last theorem. The η for the monad is the η defined by the adjunction and is also the η in the description of the partial map classifier. The map μ_X is the

unique map guaranteed by the diagram



Intuitively, of course, $(\tilde{\ })_p$ is a lifting relative to p-truth values.

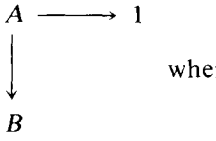
Example 1.9. In **SET** $\tilde{\tilde{X}} = X \cup \{*_1\} \cup \{*_2\}$ and μ collapses the two distinguished points to one. Note that Ω in Set is just the two element set $\{0, 1\}$ and thus has only two nontrivial subobjects, namely itself and $1 \xrightarrow{true} \Omega$. The latter collapses the lifting to an identity, thus there is only one partial map classifier, namely the obvious one described in Example 1.5.

We do not mean to suggest that the only interesting examples are toposes. In fact, semantic categories, which are not toposes, will be the primary source of our examples. Recall that any category $\boldsymbol{C}$ can be embedded inside its stack or presheaf topos, $S^{\boldsymbol{C}^{op}}$, via the Yoneda embedding $Y: \boldsymbol{C} \hookrightarrow S^{\boldsymbol{C}^{op}}$. Y is the full and faithful functor defined by $Y(X) = Hom_{\boldsymbol{C}}(\ , X)$. Suppose T is an endomap of $S^{\boldsymbol{C}^{op}}$ that restricts to an endomap of $\boldsymbol{C}$ (i.e. $YT' = TY$ where T' is an endomap of $\boldsymbol{C}$); then the following lemma will be helpful in what follows.

Lemma 1.10. *Suppose a monad T for $S^{\boldsymbol{C}^{op}}$ restricts to an endomap of $\boldsymbol{C}$. Then T is a monad for $\boldsymbol{C}$ as well.*

Proof. The proof is an easy consequence of the fact Y is full and faithful. □

Example 1.11. If $\boldsymbol{C}$ is the category of domains **DOM** then the endomap $(\ )_\perp$ is a monad on **DOM** where $(\ )_\perp$ denotes the lifting map, i.e. $D_\perp = D \cup \{\uparrow\}$ where $\uparrow \leqslant d$ for all $d \in D$. The unit map is the usual inclusion of a domain into its lifting while μ collapses the two new bottoms to one. **DOM** has no partial map classifier because **DOM** has too few limits (e.g., pullbacks). In particular, $(\ )_\perp$ cannot act like a partial map classifier. For example, the partial map



where $A = 0 \to 1$ and $B = 0 \to 1 \to 2$

has no extension to $(1)_\perp$. $S^{\mathbf{DOM}^{op}}$, however, does have a partial map classifier since it is a topos. While this partial map classifier is too broad for our needs, the

continuous refinement $(\tilde{})_s$ generated by restricting to partial maps on Scott-open subobjects is well defined in the topos. This endomap $(\tilde{})_s$ restricts to the monad $(\ )_\perp$ on **DOM**.

Example 1.12. If we replace **DOM** in the previous example with the category of effective domains **EDOM** then the corresponding monad must now be effective. Consequently we consider the monad $(\tilde{})_{re}$ which corresponds to effective lifting. For example, in the topos, $\tilde{1}_{re}$ refers to effective Sierpinski space, $\tilde{1}_{re}^{N}$ refers to the domain of r.e. sets and $\tilde{N}_{re}^{N}$ refers to the domain of partial recursive functions. As before, this monad is the restriction to **EDOM** of a (now effective) refinement of the partial map classifier in the topos. More immediate to our interests, $(\tilde{})_{re}$ is also the restriction to **EDOM** of the monad of the same name on **PER**, the category of partial equivalence relations on N (see [2]). Note that the notion of restriction makes sense here since **EDOM** is fully embedded inside **PER**. For example, $\tilde{1}_{re}$ mentioned above corresponds to the (total) equivalence relation generated by K, the r.e. nonrecursive set associated with the Halting Problem. **PER** has enjoyed considerable attention recently in connection with semantic constructions.

The concept of an effective lifting endofunctor was first introduced for yet another topos, **REC**, the recursive topos where it was shown to be a refinement of the partial map classifier. In fact, this lifting also restricts to the effective lifting on **PER** mentioned above. Also **REC** possesses many other such refinements corresponding to different computational levels of lifting. The interested reader should consult [7] for details.

Example 1.13. The notion of an ordered natural number object, *o.n.n.o.*, was devised in [9] to formalize the categorical role of the natural numbers in fixed point constructions. While *n.n.o.*'s do not generally exist in semantic categories, *o.n.n.o.*'s do. In the case of **DOM**, the initial algebra of the monad $(\tilde{})$ is the *o.n.n.o.* $N^{\infty} = 0 \to 1 \to 2 \to \cdots \to \infty$, while in **EDOM** (or **PER**) the *o.n.n.o.* is the initial algebra of monad $(\tilde{})_{re}$.

2. Partial maps and the Kleisli category

In this section we focus our attention on the Kleisli category of free algebras associated with $(\tilde{})_p$ (for ease of exposition we generally omit the p). We construct in this case a computational calculus which is weaker than the usual lambda calculus. This computational calculus models a call by value calculus of partial types and also provides a monadic view to *pccc*'s and partial types as found in [6] and [11]. Through specific examples we provide effective and continuous refinements to the monad thus allowing interpretation/comparison with standard semantic examples.

We begin by considering what a partial computation in a category $\boldsymbol{C}$ might be. Let $\boldsymbol{C}$ be a cartesian closed category. For $A, B \in \boldsymbol{C}$ a partial computation from A to B could be considered a partial map $A \rightharpoonup B$ or equivalently, if $\boldsymbol{C}$ has a partial map classifier, a map $A \to \tilde{B}$. Generally speaking, $\boldsymbol{C}$ will have neither partial map classifiers nor sufficient structure to describe partial maps, so at the very least we can consider a partial computation as a map $YA \to \widetilde{YB}$ in $S^{\boldsymbol{C}^{\mathrm{op}}}$. If $(\tilde{\ })$ in $S^{\boldsymbol{C}^{\mathrm{op}}}$ restricts to $\boldsymbol{C}$ however, then the resulting monadic structure is sufficient for our needs.

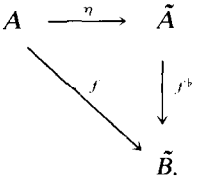
Convention 2.1. For the remainder of the paper we assume $(\tilde{\ })$ (or more generally $(\tilde{\ })_p$) restricts to an endomap of $\boldsymbol{C}$ (i.e., $Y(\tilde{\ }) = (\tilde{\ })Y$ where by abuse of notation $(\tilde{\ })$ refers also to the restricted endomap of $\boldsymbol{C}$).

Once again we note that Convention 2.1 and Lemma 1.11 imply $(\tilde{\ })$ is a monad on $\boldsymbol{C}$ (but does not imply it is a partial map classifier).

Theorem 2.2. *$\boldsymbol{C}$ is a coreflective subcategory of the Kleisli category $\mathcal{K}\boldsymbol{C}$ with coreflection $(\tilde{\ })$.*

Proof. For any monad T we have an adjunction $\boldsymbol{C} \underset{G}{\overset{F}{\rightleftarrows}} \mathcal{K}\boldsymbol{C}$ where $FX = X_T$ and $GX_T = TX$. For the special case of $T = (\tilde{\ })$ the unit map η_X is mono for all X in $\boldsymbol{C}$ and so the left adjoint F is an inclusion. □

Composition in $\mathcal{K}\boldsymbol{C}$ is described as in Section 1. The partial maps $A \overset{f}{\rightharpoonup} \tilde{B}$ and $B \overset{g}{\rightharpoonup} \tilde{C}$ have composite $(g^\flat \circ f)^\flat$ where $f^\flat$ is the unique map guaranteed by the monad



In order for this category to handle partial computation on products the monad $(\tilde{\ })$ must be strong in the sense of [3].

Definition 2.3. T is a strong monad for $\boldsymbol{C}$ if there exists a natural transformation $t_{A,B} : A \times TB \to T(A \times B)$ so that

(1) $t_{A,B} \circ (id \times \eta_B) = \eta_{A \times B}$,

(2) $t_{A,B} \circ (id \times \mu_B) = \mu_{A \times B} \circ T(t_{A,B}) \circ t_{A,TB}$,

and t respects the isomorphisms $1 \times A \cong A$, $(A \times B) \times C \cong A \times (B \times C)$.

Theorem 2.4. *Any* $(\tilde{\ })_p$ *is a strong monad.*

Proof. In fact there is perhaps a more intuitive natural transformation ψ defined by $\psi_{A,B}: A \times B \to (\widetilde{A \times B})$ where $\psi_{A,B}$ is the unique map guaranteed by the diagram

$$\begin{array}{ccc} A \times B & \xrightarrow{id} & A \times B \\ \downarrow{\scriptstyle \eta_A \times \eta_B} & & \downarrow{\scriptstyle \eta_{A \times B}} \\ \tilde{A} \times \tilde{B} & \xrightarrow{\psi_{A,B}} & A \times B \end{array}$$

From ψ one can construct t easily as follows:

$$t_{A,B} = \psi_{A,B} \circ \eta_A \times id.$$

Since natural transformations are closed under composition, t is also natural. It is straightforward to check that the above equations hold. □

Example 2.5. It is fairly obvious why we need the monad to be strong. The composition of $A \xrightarrow{f} B \times C$ and $B \times C \xrightarrow{g} D$ in $\mathcal{K}\boldsymbol{C}$ is handled as usual namely $g^\flat \circ f: A \to \tilde{D}$ in $\boldsymbol{C}$. The composition $A \xrightarrow{\langle f_1, f_2 \rangle} \tilde{B} \times \tilde{C}$, $B \times C \xrightarrow{g} \tilde{D}$ creates a type clash if we try to compose $g^\flat \circ \langle f_1, f_2 \rangle$. We can, however, compose $g^\flat \circ \psi_{B,C} \circ \langle f_1, f_2 \rangle: A \to \tilde{D}$. This is particularly useful when interpreting *let* in the computational calculus.

As pointed out in [6], when T is a strong monad, $\boldsymbol{C}_T$ interprets a computational calculus where programs correspond to maps in $\boldsymbol{C}_T$. Thus by Theorem 2.4 $\mathcal{K}\boldsymbol{C}$ provides a categorical model for a computational calculus of partial data types. In fact, this calculus is weaker than a typed lambda calculus since $\mathcal{K}\boldsymbol{C}$ is generally not cartesian closed. More precisely, the calculus lacks the ability to lambda abstract although in a concrete sense there is a ready higher type interpretation with $\boldsymbol{C}$ (or $\boldsymbol{S}^{\boldsymbol{C}^{op}}$) nearby. Recalling the usual adjunction we have in $\boldsymbol{C}$, $Hom_{\boldsymbol{C}}(A \times B, C) \cong Hom_{\boldsymbol{C}}(A, C^B)$. Working now on partial types, we have $Hom_{\mathcal{K}\boldsymbol{C}}(A \times B, C) \cong Hom_{\boldsymbol{C}}(A \times B, \tilde{C}) \cong Hom_{\boldsymbol{C}}(A, \tilde{C}^B)$. Thus we see the only possible candidate for a partial function data type is $\tilde{C}^B$ which must be interpreted in $\boldsymbol{C}$, not $\mathcal{K}\boldsymbol{C}$. This approach to partial data types was first used in [7] to model data types of varying effectiveness and also agrees with the more recent notion of function type in the context of *pccc*'s [11]. Many details of the connections between categories of the form $\mathcal{K}\boldsymbol{C}$ and *pccc*'s as well as much of the history of this development can be found in [10], including the next theorem.

Theorem 2.6. *For any refinement* $(\tilde{\ })_p$, $\mathcal{K}\boldsymbol{C}$ *is a pccc.*

The semantics associated to the computational calculus of partial data types can be made continuous or effective by prudent choice of category $\boldsymbol{C}$ and monad refinement. We first emphasize a special feature of our semantics. The following is a joint result with Harper.

Theorem 2.7. *If the partial map* $A \xrightarrow{f} \tilde{B} \xrightarrow{g^\flat} \tilde{C}$ *factors through* η_C *then* f *must factor through* η_B.

Proof. We have the pullback

$$\begin{array}{ccc} D & \longrightarrow & C \\ \downarrow & & \downarrow \\ B & \xrightarrow{g} & \tilde{C} \end{array}$$

in $S^{\mathbf{C}^{\mathrm{op}}}$. Since $g^\flat = \mu \circ \tilde{g}$, D is also the pullback of C along $g^\flat$. If $g^\flat f$ factors through C then f factors through D and thus through η_B. By Yoneda the factorization now holds in $\boldsymbol{C}$. □

The last theorem indicates that the composite of two computations outputs only if the first computation outputs. Consequently, the corresponding semantics respects call-by-value semantics.

Example 2.8. In the computational model found in [1] a new bar type is introduced to model partial computation. In joint work with Harper we have found that Theorem 2.7 holds for this model although now, interestingly, the underlying semantics is call-by-name.

Example 2.9. DOM is a cartesian closed category and maps $D \to E$ in $\mathcal{K}$**DOM** correspond to domain maps $D \to \tilde{E}$. By Example 1.11 **DOM** does not have partial map classifiers nor sufficient structure for partial maps so we consider the continuous refinement $(\tilde{\ })_s$ in the topos which restricts to $(\ )_\perp$ in **DOM**. Thus maps $D \to E$ in $\mathcal{K}$**DOM** correspond to domain maps $D \to E_\perp$ and $\mathcal{K}$**DOM** is a *pccc* coinciding with the semantic treatment of partial maps and data types found in [11]. In the effective case maps $D \to E$ in $\mathcal{K}$**EDOM** correspond to partial effective domain maps $D \to E_\perp$ in **EDOM** where $(\ )_\perp$ is now the restriction of $(\tilde{\ })_{\mathrm{re}}$.

Example 2.10. In $\mathcal{K}$**PER** the map $N \to N$ corresponds to map $N \to \tilde{N}_{\mathrm{re}}$ or $\tilde{N}_{\mathrm{re}} \to \tilde{N}_{\mathrm{re}}$ in **PER** which in turn is a partial recursive function. Thus the maps $X \to Y$ in $\mathcal{K}$**PER** are partially computable (in X and Y).

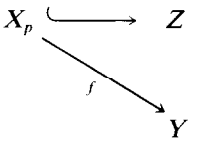
3. Partial maps and the Eilenberg–Moore category

In the last section the Kleisli category of algebras provided one useful environment for the interpretation of partiality. In fact, $\boldsymbol{C}$ was the coreflective subcategory of $\boldsymbol{C}$ via $(\tilde{\ })$. By looking in the dual direction, namely the Eilenberg–Moore category for $\boldsymbol{C}$, we can narrow this interpretation. In particular when we consider semantic categories the usual adjunction reduces to a reflection onto categories with well-behaved divergence.

Once again let us note that $\boldsymbol{C}$ is a cartesian closed category and the monad $(\tilde{\ })_p$ restricts to an endomap of $\boldsymbol{C}$.

We recall the definition of injective now relativized to a given refinement of the monad.

Definition 3.1. Y in $\boldsymbol{C}$ is p-injective if for any diagram



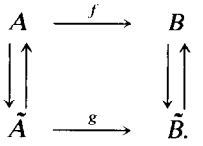
there exists an extension $Z \to Y$ making the diagram commute.

We remind the reader that all results that follow relativize for any refinement $(\tilde{\ })_p$ although for convenience we drop the p.

Theorem 3.2. *For* $X \in \boldsymbol{C}$, X *is injective iff* X *is a retract of* $\tilde{X}$ *with section* η_X.

Proof. If X is injective then id_X has an extension along $\eta_X : X \to \tilde{X}$ which provides the retraction. Conversely for any X, $\tilde{X}$ is injective in the stack topos and the retract of an injective is injective. Since $(\tilde{\ })$ restricts to $\boldsymbol{C}$ and Y is full and faithful, X is injective in $\boldsymbol{C}$ as well. □

Let $\mathcal{I}\boldsymbol{C}$ denote the full subcategory of $\boldsymbol{C}$ consisting of injective objects. By the theorem one can associate to each injective object a (nonunique) pair of maps $A \underset{q_A}{\overset{\eta_A}{\rightleftarrows}} \tilde{A}$. Also arrows $A \xrightarrow{f} B$ in $\mathcal{I}\boldsymbol{C}$ generate commutative diagrams



where g can be found for any f simply by letting $g = \eta_B \circ f \circ q_A$. This diagram is very suggestive of $(\tilde{\ })$-algebras where g is replaced by $\tilde{f}$.

Let $\mathcal{A}\boldsymbol{C}$ denote the Eilenberg-Moore category of algebras for monad $(\tilde{\ })$, and let $\tilde{A} \xrightarrow{q_A} A$ denote the algebra map. We begin by recalling a few well-known facts. Let U denote the forgetful functor $\mathcal{A}\boldsymbol{C} \xrightarrow{U} \boldsymbol{C}$.

Theorem 3.3. *The forgetful functor* U *has a left adjoint taking elements* X *of* $\boldsymbol{C}$ *to* $\tilde{X}$.

Proof. This is the special case of the well-known adjunction $\boldsymbol{C} \underset{U}{\overset{F}{\rightleftarrows}} \mathcal{A}\boldsymbol{C}$ for an arbitrary monad T where $FX = \langle TX, \mu_X \rangle$. □

In general, $\mathcal{A}\boldsymbol{C}$ is not cartesian closed nor is U generalty full or an inclusion as the next example illustrates.

Example 3.4. If $\boldsymbol{C}$ is **SET** then the monad $(\tilde{\ })$ adds a base point so $\mathscr{A}$**SET** is equivalent to pointed sets with base point preserving maps. The singleton set is both initial and terminal and so $\mathscr{A}$**SET** has a zero object and thus cannot be cartesian closed. If set X has two distinct elements x_1, x_2 then the maps $q_i : \tilde{X} \to X$ taking the base point to x_i, $i = 1, 2$ provide two different algebra structures. Thus U is not an inclusion. Finally, a map $U(X, q_X) \to U(Y, q_Y)$ is an arbitrary set function so U is clearly not full.

Despite the above example, $\mathscr{A}\boldsymbol{C}$ does inherit some structure from $\boldsymbol{C}$ as the next well-known theorem indicates.

Theorem 3.5. *Limits in $\mathscr{A}\boldsymbol{C}$ agree with those in $\boldsymbol{C}$.*

Proof. This follows immediately from the fact that the forgetful functor $\mathscr{A}\boldsymbol{C} \to \boldsymbol{C}$ creates limits. Specifically if $\langle A_i, q_i \rangle$ is a diagram of algebras then the limit diagram $\textit{limit}\, A_i \xrightarrow{\Pi_i} A_i$ in $\boldsymbol{C}$ produces the algebra map $\langle q_i \circ \tilde{\Pi}_i \rangle : \widetilde{\textit{limit}\, A_i} \to \textit{limit}\, A_i$ which is the desired limit in $\mathscr{A}\boldsymbol{C}$. □

In the case of categories utilized in semantics we have structure that is not present in the previous example such as continuity and effectiveness. We utilize some of this structure in the next result.

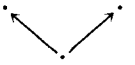
Theorem 3.6. $\mathscr{A}$**DOM** *is a reflective subcategory of* **DOM** *with reflection* $(\tilde{\ })$. *Further* $\mathscr{A}$**DOM** *is equivalent to the category of domains with strict maps.*

Proof. Returning to Example 1.12, **DOM** is a *ccc* with monad $(\tilde{\ })$. The retract in **DOM** of a domain is again a domain and conversely every domain is a retract of its lifting. Thus $\mathscr{A}$**DOM** has the same objects as **DOM**; moreover, maps $D \xrightarrow{\perp} E$ in $\mathscr{A}$**DOM** now satisfy $fq_D = q_E\tilde{f}$. Since q_D is monotone, $q_D(\uparrow) \leqslant q_D(\eta_D(\perp_D)) = \perp_D$, so $q_D(\uparrow) = \perp_D$. Consequently $f(\perp_D) = fq_D(\uparrow) = q_E\tilde{f}(\uparrow) = q_E(\uparrow) = \perp_E$ and f must be a strict (bottom-preserving) map. Similarly any strict map f satisfies $q_E\tilde{f} = fq_D$. Thus $\mathscr{A}$**DOM** is equivalent to the category of domains with strict maps. Finally by Theorem 3.3 the forgetful functor U has a left adjoint which acts like $(\tilde{\ })$. By the above discussion U is now an inclusion, since each domain object has a unique associated algebra structure. Thus we have a (nonfull) reflection. □

The above result is not restricted to **DOM** but works for similar semantic categories as well such as the categories of algebraic c.p.o.'s and SFP objects. As the next example shows, without the requirement of an algebra map we lose uniqueness.

Example 3.7. Let the map $D \xrightarrow{\perp} E$ be a map of domains where $f = \lambda d.e$ for some $e \neq \perp_E$ in E. There are at least two possible extensions of f to $\tilde{D}$, namely $\lambda d : \tilde{D}.e$ and the map that takes $\uparrow_D$ to $\perp_E$ and all other d to e. The latter map is an algebra map while the former is not.

Example 3.8. The reflection of **DOM** into $\mathscr{A}$**DOM** produces an easy proof that **DOM** does not have coproducts. If D and E are both trivial one point domains then the coproduct of the reflective images of D and E in $\mathscr{A}$**DOM** (which does exist) is just the domain



Clearly this domain is not the reflective image of any domain in **DOM** and so, since the reflection preserves coproducts, $D+E$ cannot exist in **DOM**.

Corollary 3.9. $\mathscr{A}$**EDOM** *is a reflective subcategory of* **EDOM** *with reflection* $(\tilde{\ })_{\mathrm{re}}$.

Proof. Since all maps in **EDOM** are effective they are, in particular, continuous. Thus the only candidate for an algebra structure for an element of **EDOM** is that presented in the proof of Theorem 3.6. Thus, all maps in $\mathscr{A}$**EDOM** must again be strict and likewise U is an inclusion.

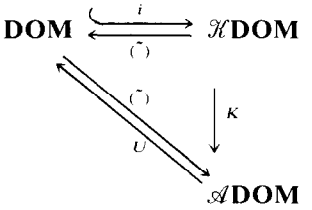
Example 3.10. Returning to Example 1.12 we now consider the category $\mathscr{A}$**PER** for monad $(\tilde{\ })_{\mathrm{re}}$. Just as **EDOM** is fully embedded inside **PER**, likewise $\mathscr{A}$**EDOM** fully embeds inside $\mathscr{A}$**PER**. The objects of this category come equipped with an intrinsic effective structure induced by the existence of the algebra map. For example, the algebras of r.e. sets and partial recursive functions exist in $\mathscr{A}$**PER**. The *n.n.o.* N, however, found in **PER**, is no longer in $\mathscr{A}$**PER** but the *o.n.n.o.* N^{∞} (Example 1.13) is. With its close affinity to **PER**, a category of considerable recent interest, this is a category worthy of further investigation. The issue of strictness is not addressed here since no partial order has been imposed on **PER**. A partial order, however, naturally exists for certain objects in **PER**, including those in $\mathscr{A}$**PER** due to the intrinsic effective structure present. This has been the subject of recent work in the foundations of semantics [2].

Maps $X \to Y$ in $\mathscr{K}\boldsymbol{C}$ correspond to maps $X \to \tilde{Y}$ or equivalently maps $\tilde{X} \to \tilde{Y}$ satisfying condition ($\flat$) in $\boldsymbol{C}$. In light of Example 3.7 we add for emphasis that the latter map fixes $\uparrow$. If X is of the form $\tilde{Z}$, however, then maps $f: \tilde{Z} \to \tilde{Y}$ exist in $\mathscr{K}\boldsymbol{C}$ which do not preserve $\uparrow$. One can make this map strict by extending it to $\tilde{X} = \tilde{Z}$. We make this discussion more precise in the next well-known theorem.

Theorem 3.11. *The coreflection* $\mathscr{K}\boldsymbol{C} \to \boldsymbol{C}$ *factors through* $\mathscr{A}\boldsymbol{C}$.

Proof. $\mathscr{K}\boldsymbol{C}$ and $\mathscr{A}\boldsymbol{C}$ are the initial and terminal categories providing adjunctions with $\boldsymbol{C}$ that generate the monad. Since $\boldsymbol{C}$ is coreflective in $\mathscr{K}\boldsymbol{C}$, the factorization of the coreflection is just the usual comparison functor $K: \mathscr{K}\boldsymbol{C} \to \mathscr{A}\boldsymbol{C}$ followed by the forgetful functor U. For any $X \in \mathscr{K}\boldsymbol{C}$, $KX = (\tilde{X}, \mu_X)$ in $\mathscr{A}\boldsymbol{C}$ and K takes maps $X \xrightarrow{f} Y$ in $\mathscr{K}\boldsymbol{C}$ to $f^{\flat} = \mu_Y \circ \tilde{f}$ in $\mathscr{A}\boldsymbol{C}$. Each component of the composition is an algebra map and therefore so is $f^{\flat}$. □

Example 3.12. If C is **DOM** then we have the following diagram of adjunctions:



where $Ki = (\tilde{\ })$ and $U \circ K = (\tilde{\ })$. Consider the map in $\mathscr{K}$**DOM** corresponding to the map $f = \lambda d.e.: D \to E_\perp$ in **DOM** where $e \neq \perp$. One makes this nonstrict map strict by extending f to $D_\perp$ and simply sending the new bottom, $\perp_D$ to $\perp_E$. This is precisely $K(f) = \mu \circ \widetilde{\lambda d.e.}: D_\perp \to E_\perp$.

Conclusion

In this paper we have approached the topic of the semantics of partial data types by constructing and studying categories of algebras. While modeling separate semantic viewpoints, these categories are functorially related and represent extreme (initial vs. terminal) models of partiality. The monads for these algebras, while well known in certain cases, were shown to arise as restrictions of refinements of partial map classifiers in corresponding toposes, thus demonstrating the intimate connection between different approaches to partiality. These categories of algebras have arisen naturally in examples such as domains, yet the approach of this paper is considerably more general. In particular, the rich structure of the monadic refinements found in toposes has yet to be sufficiently exploited at the semantic level. For example, in [7] a whole hierarchy of such monads is described. In fact, such refinements have already been utilized in work in effective semantics found in [2]. Also, for the sake of individual examples, we have restricted our discussion to cartesian closed categories. The treatment can be easily moved to a more general setting.

We have intentionally avoided introducing partial orders to the discussion since from our viewpoint they can be explicitly derived from the constructions. This is the topic of another paper, however, and instead we chose to exhibit examples with pre-existing or easily describable partial order. Finally, there are strong connections between the work in this paper and issues in fixed point semantics and operational semantics. Work on these concerns will appear elsewhere.

Acknowledgment

Conversations with Peter Freyd, Bob Harper, William Lawvere and Dana Scott have been very helpful. In particular I would like to express my gratitude to Dana Scott for providing the opportunity to visit Carnegie Mellon University during which time this paper was written.

References

[1] R.L. Constable and S.P. Smith, Computational foundations of basic recursive function theory, in: *Proc. LICS*, Scotland, 1988, 360-371.
[2] P. Freyd, P. Mulry, G. Rosolini and D. Scott, Extensional PERS, in: *Proc. LICS*, University of Pennsylvania, 1990, 346-354.
[3] A. Kock, Strong functors and monoidal monads, *Arch. Math. (Basel)* **23** (1972) 113-120.
[4] S. MacLane, *Categories for the Working Mathematician*, Graduate Texts in Mathematics (Springer, Berlin, 1971).
[5] E. Manes, *Algebraic Theories*, Graduate Texts in Mathematics (Springer, Berlin, 1976).
[6] E. Moggi, Computational lambda-calculus and monads, *Proc. LICS*, Asilomar, 1990, 14-23.
[7] P.S. Mulry, Generalized Banach-Mazur functionals in the topos of recursive sets, *J. Pure Appl. Algebra* **26** (1982) 71-83.
[8] P.S. Mulry, A categorical approach to the theory of computation, *Ann. Pure Appl. Logic* **43** (1989) 293-305.
[9] P.S. Mulry, Categorical fixed point semantics, *Theoret. Comput. Sci.* 70 (1990) 85-97.
[10] P.S. Mulry, Partial map classifiers and pccc's. Carneigie Mellon Technical Report, to appear.
[11] G. Plotkin, Denotational semantics with partial functions, Lecture Notes for CSLI, 1985.
[12] G. Rosolini, Continuity and effectiveness in topoi, PhD. thesis, Oxford, 1986.