Partiality and Container Monads

Tarmo Uustalu[1] and Niccolò Veltri[1,2]

[1] Dept. of Software Science, Tallinn University of Technology,
Akadeemia tee 21B, 12618 Tallinn, Estonia
[2] IT University of Copenhagen,
Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark
{tarmo,niccolo}@cs.ioc.ee

Abstract. We investigate monads of partiality in Martin-Löf type theory, following Moggi's general monad-based method for modelling effectful computations. These monads are often called lifting monads and appear in category theory with different but related definitions. In this paper, we unveil the relationship between containers and lifting monads. We show that the lifting monads usually employed in type theory can be specified in terms of containers. Moreover, we give a precise characterization of containers whose interpretations carry a lifting monad structure. We show that these conditions are tightly connected with Rosolini's notion of dominance. We provide several examples, putting particular emphasis on Capretta's delay monad and its quotiented variant, the non-termination monad.

1 Introduction

Martin-Löf type theory is a total language, meaning that every definable function is necessarily total. This is a fundamental requirement guaranteeing the consistency of the type system. Therefore the implementation of partial functions in type theory requires the employment of ad-hoc techniques. A comprehensive overview of such techniques has been recently given by Bove et al. [5].

In this work, we investigate monads of partiality, following Moggi's general monad-based method for modelling effectful computations [16]. The study of monads of partiality in type theory, in particular aimed at the representation of possibly non-terminating computations (non-termination from iteration), has been an active area of research in recent years [7, 8, 3, 14].

Monads of partiality have also been intensively studied in category theory. These monads are often called *lifting monads* and appear in the literature with different but related definitions. The oldest one, originated in the area of topos theory, is the notion of partial map classifier [15]. A monad T on a category $\mathbb{C}$ is a partial map classifier if every partial map $f : X \rightharpoonup Y$ in $\mathbb{C}$, to be thought of as a total map from a certain subset of X into Y, is in one-to-one correspondence with a map $\hat{f} : X \to T\,Y$. Another notion of lifting monad is given by Cockett and Lack's classifying monads [10]. They can be thought of as abstract partial map classifiers, in the sense that their Kleisli category is an abstract category

of partial maps, viz., a restriction category [9]. For a base category with finite products, Bucalo et al. [6] have given another notion of lifting monad, that they call equational lifting monads. These monads enjoy a more concise algebraic characterization than classifying monads. Moreover, equational lifting monads are a subclass of classifying monads.

In this paper, we investigate the connection between lifting monads and containers [1] in type theory. We noticed that the monads of partiality most commonly employed in type theory can be specified by containers:

- The maybe monad can be defined as $\mathsf{Maybe}\, X =_{\mathrm{df}} \Sigma s : \mathsf{Bool}.\, ((s = \mathsf{ok}) \to X)$ where $\mathsf{Bool} = \{\mathsf{ok}, \mathsf{err}\}$.
- The different definitions of the monad for non-termination [8, 3] are isomorphic to $X_\perp =_{\mathrm{df}} \Sigma s : \mathsf{S}.\, ((s = \top) \to X)$ where S is a type called the Sierpinski set [13] or Rosolini's dominance [19].
- The full partial map classifier, classifying all partial maps, can be defined as $\mathsf{PMC}\, X =_{\mathrm{df}} \Sigma s : \Omega.\, ((s = 1) \to X)$ where Ω is the type of all propositions. In the type theory that we consider, $(s = 1) = s$.

The types Bool, S and Ω have to be thought as types of truth values classifying decidable, semidecidable and all subobjects respectively. The specific elements $\mathsf{ok} : \mathsf{Bool}$, $\top : \mathsf{S}$ and $1 : \Omega$ are the truth values corresponding to truth.

Ahman et al. [2, 21] have given an algebraic characterization of containers whose interpretations carry a monad structure. We build on their work and give an algebraic characterization of containers whose interpretations carry a lifting monad structure. We show that different notions of lifting monad give rise to different characterizations. In particular, we give a new algebraic description of dominances, a notion first introduced by Rosolini for the specification of partial map classifiers in topos theory [19].

Our motivation for studying the relationship between containers and lifting monads comes from the fact that different notions of lifting monad are difficult to get an intuition for and compare in the abstract. Containers are a more concrete setting where the fine differences between them become easier to see and appreciate. For partiality specifically, containers provide a separation between reasoning about the partiality effect of a computation (which is all about shapes) and its functional content (which is about assignments of values to positions).

There is also some simplification to the metatheory of non-termination. The container approach allowed Chapman et al. [8] to define the non-termination monad in homotopy type theory avoiding the axiom of countable choice by using standard higher inductive types. This was an improvement over Altenkirch et al.'s [3] earlier solution that required higher inductive-inductive types.

The paper is organized as follows. In Section 2, we give an overview of some notion of lifting monads that appear in category theory. In Section 3, we revise containers and mnd-containers (containers interpreting into monads). In Section 4, we derive algebraic conditions on containers that makes their interpretations carry a lifting monad structure. Moreover, we give a new algebraic description of dominances. In Section 5, we give several examples, with special emphasis on the non-termination monad, used to classify partial maps with

semidecidable domain of definedness in type theory. Finally, in Section 6, we draw some conclusions and we discuss future work.

We have fully formalized the development of the paper in the dependently typed programming language Agda [17]. The code is available at `http://cs.ioc.ee/~niccolo/partialcont/`; it uses Agda version 2.5.2 and Agda Standard Library version 0.13.

The Type-Theoretical Framework We work in Martin-Löf type theory extended with the following extensional concepts: function extensionality (pointwise equal functions are equal) and proposition extensionality (logically equivalent propositions are equal). Equivalently, we could work in homotopy type theory, where function and proposition extensionality are consequences of the univalence axiom.

We assume uniqueness of identity proofs, which corresponds to working with 0-truncated types in homotopy type theory.

We revise a couple of basic definitions. A type X is a *proposition*, if any two of its inhabitants are equal: $\mathsf{isProp}\, X =_{\mathrm{df}} \Pi x_1\, x_2 : X.\, x_1 = x_2$. A type X is called *contractible*, if there exists $x : X$ such that every other element of X is equal to x: $\mathsf{isContr}\, X =_{\mathrm{df}} \Sigma x : X.\, \Pi x' : X.\, x = x'$.

2 Lifting Monads

In this section, we revise the different definitions of lifting monads appearing in category theory and how these notions relate to each other.

A few words on notation. Given a monad (T, η, μ), we write $f^* : T\,X \to T\,Y$ for the Kleisli extension of the map $f : X \to T\,Y$. Moreover, we write $g \diamond f$ for the composition of $f : X \to T\,Y$ and $g : Y \to T\,Z$ in the Kleisli category of T.

We start with Cockett and Lack's classifying monads [10].

Definition 1. An *almost-classifying monad* on a category $\mathbb{C}$ is a monad (T, η, μ) with an operation $\overline{(-)}$, called *restriction*, sending any map $f : X \to T\,Y$ into a map $\overline{f} : X \to T\,X$ subject to the following conditions:

CM1 $f \diamond \overline{f} = f$,
CM2 $\overline{g} \diamond \overline{f} = \overline{f} \diamond \overline{g}$,
CM3 $\overline{g} \diamond \overline{f} = g \diamond \overline{f}$,
CM4 $\overline{g} \diamond f = f \diamond \overline{g \diamond f}$,
CM5 $\overline{\eta_Y \circ h} = \eta_X$, for $h : X \to Y$.

We call it a *classifying monad*, if it also satisfies

CM6 $\overline{\mathsf{id}_{TX}} = T\eta_X$.

The restriction of a map $f : X \to TY$ should be thought of as a "partial identity function" on X, a kind of a specification, in the form of a map, of the "domain of definedness" of f (which need not be present in the category as an object).

Conditions CM1-4 stipulate that the Kleisli category of T is a restriction category. Condition CM5 states that pure maps, i.e., maps in the base category $\mathbb{C}$, are total. The additional condition CM6 of a classifying monad is more technical. It was postulated by Cockett and Lack in order to connect classifying monads and partial map classifiers, or more generally, classified restriction categories and classified $\mathcal{M}$-categories (Theorem 3.6 of [10]), $\mathcal{M}$-categories being Robinson and Rosolini's [18] framework for partiality.

Together with CM5, the condition CM4 implies CM1:

$$f \diamond \overline{f} = f \diamond \overline{\eta_Y \diamond f} \overset{\text{CM4}}{=} \overline{\eta_Y} \diamond f \overset{\text{CM5}}{=} \eta_Y \diamond f = f$$

A more specific notion of lifting monads is given by effective classifying monads [10]. Recall that a natural transformation is called Cartesian, if all of its naturality squares are pullbacks. A monad (T, η, μ) is said to be Cartesian, if η and μ are Cartesian.

Definition 2. A classifying monad $(T, \eta, \mu, \overline{(-)})$ is called *effective*, if η is Cartesian, pullbacks along η_X exist and are preserved by T.

An effective classifying monad is always Cartesian. Effective classifying monads are the same thing as *partial map classifiers* (Theorem 5.8 of [10]). This means that, given an effective classifying monad $(T, \eta, \mu, \overline{(-)})$, there exists an isomorphism $\mathbb{C}(X, TY) \cong \mathsf{Par}_{\mathcal{M}}(\mathbb{C})(X, Y)$, where the category $\mathsf{Par}_{\mathcal{M}}(\mathbb{C})$ has the same objects of $\mathbb{C}$ and it has spans

$$X \xleftarrow{\ m\ } X' \xrightarrow{\ f\ } Y$$

as morphisms between X and Y, where the left leg m is a monic map belonging to the collection $\mathcal{M}$. Intuitively, an element of $\mathsf{Par}_{\mathcal{M}}(\mathbb{C})(X, Y)$ is a partial map from X to Y. The left leg of the span specifies its domain of definedness.

Bucalo et al. [6] introduced the notion of equational lifting monad. Recall that a monad (T, η, μ) on a category $\mathbb{C}$ with finite products, equipped with a left strength $\psi_{X,Y} : X \times TY \to T(X \times Y)$, is called *commutative*, if the following diagram commutes:

$$\begin{array}{ccc} TX \times TY & \xrightarrow{\ \psi_{TX,Y}\ } & T(TX \times Y) \\ {\scriptstyle \phi_{X,TY}}\downarrow & & \downarrow{\scriptstyle \phi^*_{X,Y}} \\ T(X \times TY) & \xrightarrow[\ \psi^*_{X,Y}\ ]{} & T(X \times Y) \end{array} \qquad \textbf{(CommM)}$$

where the right strength $\phi_{X,Y} : TX \times Y \to T(X \times Y)$ is defined as $\phi_{X,Y} =_{\text{df}} T\,\mathsf{swap} \circ \psi_{X,Y} \circ \mathsf{swap}$.

Definition 3. A commutative monad (T, η, μ, ψ) is called an *equational lifting monad*, if the following diagram commutes:

$$\begin{array}{ccc} T\,X & \xrightarrow{\quad\Delta\quad} & T\,X \times T\,X \\ {\scriptstyle T\,\Delta}\downarrow & & \downarrow{\scriptstyle \psi_{T\,X,X}} \\ T\,(X \times X) & \xrightarrow[T\,(\eta_X \times \mathsf{id}_X)]{} & T\,(T\,X \times X) \end{array} \qquad (\textbf{EqLM})$$

Every equational lifting monad is canonically a classifying monad. Its restriction operation is defined with the aid of the strength:

$$\overline{f} =_{\mathrm{df}} \; X \xrightarrow{\langle \mathsf{id}_X, f\rangle} X \times T\,Y \xrightarrow{\psi_{X,Y}} T\,(X \times Y) \xrightarrow{T\,\pi_0} T\,X \qquad (\star)$$

Cockett and Lack showed that there exist classifying monads on categories with finite products which are not equational lifting monads [10].

Notice that, in order to construct an almost-classifying monad, we can relax condition EqLM above and consider Cockett and Lack's copy monads [11].

Definition 4. A *copy monad* is a commutative monad (T, η, μ, ψ) for which the following diagram commutes:

$$\begin{array}{ccc} T\,X & \xrightarrow{\quad\Delta\quad} & T\,X \times T\,X \\ {\scriptstyle T\,\Delta}\downarrow & & \downarrow{\scriptstyle \psi_{T\,X,X}} \\ T\,(X \times X) & \xleftarrow[\phi^*_{X,X}]{} & T\,(T\,X \times X) \end{array} \qquad (\textbf{CopyM})$$

Every equational lifting monad is a copy monad:

$$\phi^* \circ \psi \circ \Delta = \phi^* \circ T\langle \eta, \mathsf{id}\rangle = (\phi \circ (\eta \times \mathsf{id}) \circ \Delta)^* = (\eta \circ \Delta)^* = T\Delta$$

Every copy monad is canonically an almost-classifying monad. Its restriction operation is defined as the one of equational lifting monads $(\star)$.

3 Container Monads

Containers [1] serve as a "syntax" of a wide class of set functors; we call these set functors *container functors*. Many constructions on set functors can be carried out on the level of containers. Their concreteness makes containers a useful tool for enumerative combinatorics of container functors with special structure or properties.

In this paper, we are specifically interested in container functors with a monad structure. These were characterized by Ahman et al. [2, 21].

A *container* is given by a set S (of shapes) and an S-indexed family P of sets (of positions in each shape).

A container determines a set functor $[\![S, P]\!]^{\mathrm{c}} =_{\mathrm{df}} T$ where $T\,X =_{\mathrm{df}} \Sigma s : S.\, P\,s \to X$ and $T\,f =_{\mathrm{df}} \lambda(s, v).\,(s, f \circ v)$.

We will not discuss container morphisms and their interpretation here. But containers form a category **Cont** and interpretation $[\![-]\!]^{\mathrm{c}}$ makes a fully-faithful functor between **Cont** and $[\mathbf{Set}, \mathbf{Set}]$.

There is an identity container defined by $\mathsf{Id}^{\mathrm{c}} =_{\mathrm{df}} (1, \lambda *.\, 1)$. Containers can be composed, composition is defined by $(S, P) \cdot^{\mathrm{c}} (S', P') =_{\mathrm{df}} (\Sigma s : S.\, P\,s \to S', \lambda(s, v).\, \Sigma p : P\,s.\, P'\,(v\,p))$.

Identity and composition of containers provide a monoidal category structure on **Cont**. Interpretation $[\![-]\!]^{\mathrm{c}}$ is a monoidal functor between $(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \cdot^{\mathrm{c}})$ and $([\mathbf{Set}, \mathbf{Set}], \mathsf{Id}, \cdot)$.

We call a *mnd-container* a container (S, P) with operations

- $\mathsf{e} : S$,
- $\bullet : \Pi s : S.\, (P\ s \to S) \to S$,
- $q_0 : \Pi s : S.\, \Pi v : P\,s \to S.\, P\,(s \bullet v) \to P\,s$,
- $q_1 : \Pi s : S.\, \Pi v : P\,s \to S.\, \Pi p : P\,(s \bullet v).\, P\,(v\,(v \mathbin{\nwarrow_s} p))$

(where we write $q_0\,s\,v\,p$ as $v \mathbin{\nwarrow_s} p$ and $q_1\,s\,v\,p$ as $p \mathbin{\nearrow_v} s$) satisfying

- $s \bullet (\lambda\_.\, \mathsf{e}) = s$,
- $\mathsf{e} \bullet (\lambda\_.\, s) = s$,
- $(s \bullet v) \bullet (\lambda p''.\, w\,(v \mathbin{\nwarrow_s} p'')\,(p'' \mathbin{\nearrow_v} s)) = s \bullet (\lambda p'.\, v\,p' \bullet w\,p')$,
- $(\lambda\_.\, \mathsf{e}) \mathbin{\nwarrow_s} p = p$,
- $p \mathbin{\nearrow_{\lambda\_.\, s}} \mathsf{e} = p$,
- $v \mathbin{\nwarrow_s} ((\lambda p''.\, w\,(v \mathbin{\nwarrow_s} p'')\,(p'' \mathbin{\nearrow_v} s)) \mathbin{\nwarrow_{s \bullet v}} p) = (\lambda p'.\, v\,p' \bullet w\,p') \mathbin{\nwarrow_s} p$,
- $((\lambda p''.\, w\,(v \mathbin{\nwarrow_s} p'')\,(p'' \mathbin{\nearrow_v} s)) \mathbin{\nwarrow_{s \bullet v}} p) \mathbin{\nearrow_v} s =$
  $\mathsf{let}\ u \leftarrow \lambda p'.\, v\,p' \bullet w\,p'\ \mathsf{in}\ w\,(u \mathbin{\nwarrow_s} p) \mathbin{\nwarrow_{v\,(u \nwarrow_s p)}} (p \mathbin{\nearrow_u} s)$,
- $p \mathbin{\nearrow_{\lambda p''.\, w\,(v \nwarrow_s p'')\,(p'' \nearrow_v s)}} (s \bullet v) =$
  $\mathsf{let}\ u \leftarrow \lambda p'.\, v\,p' \bullet w\,p'\ \mathsf{in}\ (p \mathbin{\nearrow_u} s) \mathbin{\nearrow_{w\,(u \nwarrow_s p)}} v\,(u \mathbin{\nwarrow_s} p)$.

The data $(\mathsf{e}, \bullet)$ are like a monoid structure on S modulo the 2nd argument of the multiplication being not an element of S, but a function from Ps to S where s is the 1st argument. Similarly, introducing the visual $\nwarrow, \nearrow$ notation for the data q_0, q_1 helps us see that they are reminiscent of a biaction (a pair of agreeing right and left actions) of this monoid-like structure on P. But here a further difference is also that P is not a set, but a S-indexed family of sets.

An mnd-container $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ interprets into a monad $[\![S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow]\!]^{\mathrm{mc}} =_{\mathrm{df}} (T, \eta, \mu)$ where

- $T =_{\mathrm{df}} [\![S, P]\!]^{\mathrm{c}}$
- $\eta\,x =_{\mathrm{df}} (\mathsf{e}, \lambda p.\, x)$
- $\mu\,(s, v) =_{\mathrm{df}} \mathsf{let}\ (v_0\,p, v_1\,p) \leftarrow v\,p\ \mathsf{in}\ (s \bullet v_0, \lambda p.\, v_1\,(v_0 \mathbin{\nwarrow_s} p)\,(p \mathbin{\nearrow_{v_0}} s))$

We omit the definition of mnd-container morphisms and their interpretation. We note that mnd-containers form a category **MCont** whose identity and composition are inherited from **Cont**. Interpretation $[\![-]\!]^{\mathrm{mc}}$ is a fully-faithful functor between **MCont** and **Monad**(**Set**).

Mnd-containers are in a bijection with containers whose interpretation carries a monad structure. The reason is, we could say, that they are in a bijection with monoid objects in the monoidal category $(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \cdot^{\mathrm{c}})$.

Moving from sets of objects to categories, one can observe that the functor $[\![-]\!]^{\mathrm{mc}} : \mathbf{MCont} \to \mathbf{Monad}(\mathbf{Set})$ is the pullback of the fully-faithful functor $[\![-]\!]^{\mathrm{c}} : \mathbf{Cont} \to [\mathbf{Set}, \mathbf{Set}]$ along $U : \mathbf{Monad}(\mathbf{Set}) \to [\mathbf{Set}, \mathbf{Set}]$ and the category $\mathbf{MCont}$ is isomorphic to the category of monoids in $(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \cdot^{\mathrm{c}})$.

$$\begin{array}{ccccc}
\mathbf{MCont} & & \xrightarrow{U} & & \mathbf{Cont} \qquad (\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \cdot^{\mathrm{c}}) \\
\cong \mathbf{Monoid}(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \cdot^{\mathrm{c}}) & & & & \\
\Big\downarrow{\scriptstyle \text{f.f. } [\![-]\!]^{\mathrm{mc}}} & & & & \Big\downarrow \quad \xleftarrow{U} \quad \Big\downarrow{\scriptstyle [\![-]\!]^{\mathrm{c}} \text{ f.f.}} \\
\mathbf{Monad}(\mathsf{Set}) & & \xrightarrow{U} & & [\mathsf{Set}, \mathsf{Set}] \qquad ([\mathsf{Set}, \mathsf{Set}], \mathsf{Id}, \cdot) \\
\cong \mathbf{Monoid}([\mathsf{Set}, \mathsf{Set}], \mathsf{Id}, \cdot) & & & &
\end{array}$$

As an example, we look at the container syntax for the list functor and the standard monad structure on it.

The relevant container is (S, P) where $S =_{\mathrm{df}} \mathbb{N}$, $P\, s =_{\mathrm{df}} [0..s)$. The corresponding functor is T where $T\, X =_{\mathrm{df}} \Sigma s : \mathbb{N}.\, [0..s) \to X \cong \mathsf{List}\, X$.

This container extends to an mnd-container by

- $\mathsf{e} =_{\mathrm{df}} 1$,
- $s \bullet v =_{\mathrm{df}} \sum_{p:[0..s)} v\, p$,
- $v \nwarrow_s p =_{\mathrm{df}}$ greatest $p_0 : [0..s)$ such that $\sum_{p':[0..p_0)} v\, p' \leq p$,
- $p \nearrow_v s =_{\mathrm{df}} p - \sum_{p':[0..v \nwarrow_s p)} v\, p'$

The corresponding monad structure on T is the standard list monad with $\eta\, x =_{\mathrm{df}} [x]$, $\mu\, xss =_{\mathrm{df}} \mathsf{concat}\, xss$.

The list mnd-container example can be generalized in the following way. Let $(O, \#, \mathsf{id}, \circ)$ be some non-symmetric operad, i.e., let O be a set of operations, $\# : O \to \mathbb{N}$ a function fixing the arity of each operation and $\mathsf{id} : O$ and $\circ : \Pi o : O.\, (\# o \to O) \to O$ an identity operation and a parallel composition operator, with $\# \mathsf{id} = 1$ and $\# (o \circ v) = \sum_{i:[0..\# o)} \# (v\, i)$, satisfying the equations of a non-symmetric operad. We can take $S =_{\mathrm{df}} O$, $P\, o =_{\mathrm{df}} [0..\# o)$, $\mathsf{e} =_{\mathrm{df}} \mathsf{id}$, $\bullet =_{\mathrm{df}} \circ$ and $\nwarrow$, $\nearrow$ analogously to the definition of the (standard) list mnd-container. This choice of $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ gives a mnd-container. The list mnd-container corresponds to a special case where there is exactly one operation for every arity, in which situation we can w.l.o.g. take $O = \mathbb{N}$, $\# o = o$. Keeping this generalization of the list monad example in mind, we can think of mnd-containers as a generalization of non-symmetric operads where the argument places of an operation are identified nominally rather than positionally, operations may also have infinite arities, and, importantly, arguments may be discarded and duplicated in composition.

Altenkirch and Pinyo [4] have proposed to think of an mnd-container $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ as a "lax" $(1, \Sigma)$ type universe à la Tarski, namely, to view S

as a universe of types ("codes for types"), P as an assignment of a set to each type, e as a type 1, $\bullet$ as a Σ-type former, $\nwarrow$ and $\nearrow$ as first and second projections from the denotation of a Σ-type. The laxity here is that we have functions $\lambda\_.\, * : P\,\mathsf{e} \to 1$ and $\lambda p.\,(v \nwarrow_s p, p \nearrow_v s) : P\,(s \bullet v) \to \Sigma p : P\,s.\,P\,(v\,p)$, but they are not isomorphisms.

The interpretation of an mnd-container $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ is a Cartesian monad if and only if the functions $\lambda\_.\, * : P\,\mathsf{e} \to 1$ and $\lambda p.\,(v \nwarrow_s p, p \nearrow_v s) : P\,(s \bullet v) \to \Sigma p : P\,s.\,P\,(v\,p)$ are isomorphisms. (We will prove the condition for Cartesianness of η in Lemma 11).

The list mnd-container satisfies these condition and the list monad is Cartesian.

Cartesian mnd-containers can be thought of as (proper) operads with possibly infinitary operations and as (proper) $(1, \Sigma)$ type universes.

4 Containers of Partiality

Let $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ be an mnd-container and let (T, η, μ) be its interpretation into a monad. We now derive conditions on the mnd-container under which the monad is a lifting monad (in some sense or other).

4.1 Classifying Monads

We start with classifying monads. We restrict our attention to classifying monads whose restriction operation is defined as in $(\star)$.

We have that $\overline{f}\,x = (s, \lambda\_.\, x)$, if $f\,x = (s, h)$. If additionally $g\,x = (s', h')$, then $g^*\,(\overline{f}\,x) = (s \bullet (\lambda\_.\, s'), \lambda\_.\, h'(p \nearrow_{\lambda\_.\, s'} s))$.

We already know that CM1 is derivable from CM4 and CM5.

Lemma 1. *CM2 holds if and only if the following condition holds:*

A2 $s \bullet (\lambda\_.\, s') = s' \bullet (\lambda\_.\, s)$

Proof. Given $f : X \to T\,Y$, $g : X \to T\,Z$, $x : X$, let $(s, h) = f\,x$ and $(s', h') = g\,x$. We have:

$$\overline{g}^*\,(\overline{f}\,x) = \overline{g}^*\,(s, \lambda\_.\, x) = (s \bullet (\lambda\_.\, s'), \lambda\_.\, x)$$

$$\overline{f}^*\,(\overline{g}\,x) = \overline{f}^*\,(s', \lambda\_.\, x) = (s' \bullet (\lambda\_.\, s), \lambda\_.\, x)$$

It is easy to see that A2 implies CM2. Vice versa, CM2 implies A2 by choosing $f, g : 1 \to T\,1$, $f * =_{\mathrm{df}} (s, \lambda\_.\, *)$ and $g * =_{\mathrm{df}} (s', \lambda\_.\, *)$.

□

Lemma 2. *CM3 holds always.*

Proof. Given $f : X \to T\,Y$, $g : X \to T\,Z$, $x : X$, let $(s, h) = f\,x$ and $(s', h') = g\,x$. We have $\overline{g}^*\,(\overline{f}\,x) = (s \bullet (\lambda\_.\, s'), \lambda\_.\, x)$ and $\overline{g^* \circ \overline{f}}\,x = (s \bullet (\lambda\_.\, s', \lambda\_.\, x)$. □

Lemma 3. *CM4 holds if and only if the following two conditions hold:*

A4 $(s \bullet v) \bullet (\lambda\_.\, s) = s \bullet v$
A4' $p \nearrow_{\lambda\_.\, s} (s \bullet v) = v \nwarrow_s p$

Proof. Given $f : X \to T\,Y$, $g : Y \to T\,Z$, $x : X$, let $(s, h) = f\,x$ and $v = \pi_0 \circ g \circ h : P\,s \to S$. We have:

$$\overline{g}^*\,(f\,x) = \overline{g}^*\,(s,h) = (s \bullet v, \lambda p.\, h\,(v \nwarrow_s p))$$

$$f^*\,(\overline{g^* \circ f}\,x) = f^*\,(s \bullet v, \lambda\_.x) = ((s \bullet v) \bullet (\lambda\_.\, s), \lambda p.\, h\,(p \nearrow_{\lambda\_.\, s} (s \bullet v)))$$

It is easy to see that A4 and A4' imply CM4. Vice versa, CM4 implies A4 and A4' by choosing $f : 1 \to T\,(P\,s)$, $f * =_{\mathrm{df}} (s, \mathsf{id})$ and $g : P\,s \to T\,1$, $g\,p =_{\mathrm{df}} (v\,p, \lambda\_.\, *)$. □

Lemma 4. *CM5 holds always.*

Proof. Immediate. □

It is worth noting that the condition A2 is similar to commutativity and condition A4 to left regularity of a binary operation.

In summary, we obtain the following characterization of containers whose interpretation carry an almost-classifying monad structure.

Theorem 1. *The interpretation of an mnd-container $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ is an almost-classifying monad with restriction operation defined as in ($\star$) if and only if it satisfies A2, A4 and A4'.*

We proceed to classifying monads and analyze condition CM6.

Lemma 5. *CM6 holds if and only if the following two conditions hold:*

B $P\,s \to s = \mathsf{e}$
C $\mathsf{isProp}\,(P\,s)$

Proof. ($\Rightarrow$) Assume CM6. Let $s : S$, we define $t : T\,(P\,s)$, $t =_{\mathrm{df}} (s, \mathsf{id}_{P\,s})$. We have:

$$\overline{\mathsf{id}_{T\,(P\,s)}}\,t = (s, \lambda\_.\, (s, \mathsf{id}_{P\,s}))$$

$$T\,\eta_1\,t = (s, (\lambda q.\, (\mathsf{e}, (\lambda\_.\, q))))$$

CM6 implies, in particular, that the functions $g =_{\mathrm{df}} \lambda\_.\, (s, \mathsf{id}_{P\,s})$ and $h =_{\mathrm{df}} \lambda q.\, (\mathsf{e}, \lambda\_.\, q)$ of type $P\,s \to T\,(P\,s)$ are equal.
In order to conclude B, let $p : P\,s$. Applying both functions g and h to p, we obtain $(s, \mathsf{id}_{P\,s}) = (\mathsf{e}, \lambda\_.\, p)$, and in particular $s = \mathsf{e}$.
In order to conclude C, let $p, q : P\,s$. Applying both functions g and h to p, we obtain $(s, \mathsf{id}_{P\,s}) = (\mathsf{e}, \lambda\_.\, p)$. Applying both functions g and h to q, we obtain $(s, \mathsf{id}_{P\,s}) = (\mathsf{e}, \lambda\_.\, q)$. In particular, we have $(\mathsf{e}, \lambda\_.\, p) = (\mathsf{e}, \lambda\_.\, q)$ and therefore $p = q$.

($\Leftarrow$) Assume B and C. Let $(s, g) : T\,X$. We have $\overline{\mathsf{id}_{T\,(P\,s)}}\,(s, g) = (s, \lambda\_.\,(s, g))$ and $T\,\eta_X\,(s, g) = (s, \lambda p.\,(\mathsf{e}, \lambda\_.\,g\,p))$. B and C imply that these two terms are equal.

□

For container monads, we have that CM6 implies CM4.

Lemma 6. *B implies A4; C implies A4'.*

Proof. Assume B. Let $s : S$, $v : Ps \to S$.

$$(s \bullet v) \bullet (\lambda\_.\,s) \overset{\mathrm{B}}{=} (s \bullet v) \bullet (\lambda\_.\,\mathsf{e}) = s \bullet v$$

Assuming also C, we derive immediately A4', since the latter is an equation between positions. □

In summary, we obtain the following characterization of containers whose interpretations carry a classifying monad structure.

Theorem 2. *The interpretation of an mnd-container $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ is a classifying monad with restriction operation defined as in ($\star$) if and only if it satisfies A2, B and C.*

As we have shown in the Lemma 6, condition C implies the position equation A4'. Analogously, condition C trivializes all position equations from the definition of mnd-container.

4.2 Copy Monads and Equational Lifting Monads

We now move to the characterization of containers whose interpretations carry a copy monad structure or an equational lifting monad structure.

The left strength ψ, when applied to a term $(s, h) : T\,X$, returns $\psi\,(x, (s, h)) = (s, \lambda p.\,(x, h\,p))$, while the right strength ϕ symmetrically returns $\phi\,((s, h), y) = (s, \lambda p.\,(h\,p, y))$.

Lemma 7. *CommM holds if and only if both A2 and the following condition hold:*

A2' $(\lambda\_.\,s') \nwarrow_s p = p \nearrow_{\lambda\_.\,s} s'$

Proof. Given $(s, h) : T\,X$ and $(s', h') : T\,Y$, we have:

$$\begin{aligned}\psi^*\,(\phi\,((s, h), (s', h'))) &= \psi^*\,(s, \lambda p.\,(h\,p, (s', h'))) \\ &= (s \bullet (\lambda\_.\,s'), \lambda p.\,(h\,(\lambda\_.s' \nwarrow_s p), h'\,(p \nearrow_{\lambda\_.s'} s)))\end{aligned}$$

$$\begin{aligned}\phi^*\,(\psi\,((s, h), (s', h'))) &= \phi^*\,(s', \lambda p.\,((s, h), h'\,p)) \\ &= (s' \bullet (\lambda\_.\,s), \lambda p.\,(h\,(p \nearrow_{\lambda\_.s} s'), h'\,(\lambda\_.s \nwarrow_{s'} p)))\end{aligned}$$

It is easy to see that A2 and A2' imply CommM. Vice versa, CommM implies both A2 and A2' by choosing $h =_{\mathrm{df}} \mathsf{id}_{P\,s}$ and $h' =_{\mathrm{df}} \mathsf{id}_{P\,s'}$. □

Lemma 8. *CopyM holds if and only if the following conditions hold:*

A1 $s \bullet (\lambda\_.\, s) = s$
A1' $p \nearrow_{\lambda\_.\, s} s = p$
A1" $(\lambda\_.\, s) \nwarrow_s p = p$

Proof. Given $(s, h) : T\, X$, we have:

$$\phi^* (\psi ((s, h), (s, h))) = (s \bullet (\lambda\_.\, s), \lambda p.\, (h\, (p \nearrow_{\lambda\_.\, s} s), h\, (\lambda\_.\, s \nwarrow_s p)))$$

$$T\, \Delta\, (s, h) = (s, \lambda p.\, (h\, p, h\, p))$$

It is easy to check that A1, A1' and A1" imply CopyM. Vice versa, CopyM implies A1, A1' and A1" by choosing $h =_{\mathrm{df}} \mathsf{id}_{P\, s}$. □

Condition A1 is a version of idempotence of a binary operation.

In summary, we obtain the following characterization of containers whose interpretations carry a copy monad structure.

Theorem 3. *The interpretation of an mnd-container $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ is a copy monad if and only if it satisfies A1, A1', A1", A2 and A2'.*

The following lemma shows that the set of conditions characterizing almost-classifying monads in Theorem 1 implies the set of conditions characterizing copy monads in Theorem 3.

Lemma 9. *A4 implies A1. A4 and A4' imply A1' and A1". Moreover, conditions A2, A4 and A4' imply A2'.*

Proof. Assume A4. Given $s : S$, we have:

$$s \bullet (\lambda\_.\, s) = (s \bullet (\lambda\_.\, \mathsf{e})) \bullet (\lambda\_.\, s) \overset{\text{A4}}{=} s \bullet (\lambda\_.\, \mathsf{e}) = s$$

Assume also A4'. Given $s : S$ and $p : P\, (s \bullet (\lambda\_.\, s))$, we have:

$$p \nearrow_{\lambda\_.\, s} s = p \nearrow_{\lambda\_.\, s} (s \bullet (\lambda\_.\, \mathsf{e})) \overset{\text{A4'}}{=} (\lambda\_.\, \mathsf{e}) \nwarrow_s p = p$$

$$(\lambda\_.\, s) \nwarrow_s p \overset{\text{A4'}}{=} p \nearrow_{\lambda\_.\, s} (s \bullet (\lambda\_.\, s)) \overset{\text{A1}}{=} p \nearrow_{\lambda\_.\, s} s = p$$

Now assume also A2. Given $s, s' : S$ and $p : P\, (s \bullet (\lambda\_.\, s'))$, we have:

$$\begin{aligned} p \nearrow_{\lambda\_.\, s} s' &\overset{\text{A1'}}{=} (p \nearrow_{\lambda\_.\, s' \bullet (\lambda\_.\, s)} (s' \bullet (\lambda\_.\, s))) \nearrow_{\lambda\_.\, s} s' \\ &= p \nearrow_{\lambda\_.\, s} ((s' \bullet (\lambda\_.\, s)) \bullet (\lambda\_.\, s')) \\ &\overset{\text{A4}}{=} p \nearrow_{\lambda\_.\, s} (s' \bullet (\lambda\_.\, s)) \overset{\text{A2}}{=} p \nearrow_{\lambda\_.\, s} (s \bullet (\lambda\_.\, s')) \\ &\overset{\text{A4'}}{=} (\lambda\_.\, s') \nwarrow_s p \end{aligned}$$

□

Remember that every copy monad is canonically an almost-classifying monad. Theorems 1 and 3 together with Lemma 9 tell us that the converse holds, if the underlying functor of the monad is specified by a container and if restriction is given as in $(\star)$.

Corollary 1. *The interpretation of an mnd-container is a copy monad if and only if it is an almost-classifying monad with restriction operation defined as in* $(\star)$*.*

We now move on to equational lifting monads.

Lemma 10. *EqLM holds if and only if B and C hold.*

Proof. Given $(s, h) : T\,X$, we have:

$$\psi\,(\Delta\,(s,h)) = (s, \lambda p.\,((s,h), h\,p))$$

$$T\,(\eta \times \mathsf{id})\,(T\,\Delta\,(s,h)) = (s, (\eta \times \mathsf{id}) \circ \Delta \circ h) = (s, \lambda p.\,((\mathsf{e}, \lambda\_.\,h\,p), h\,p))$$

It is easy to check that B and C imply EqLM. Vice versa, EqLM implies both B and C by choosing $h =_{\mathrm{df}} \mathsf{id}_{P\,s}$. □

Analogously to Lemma 6, it is easy to see C implies A1', A1", A2', as these are all position equations.

In summary, we obtain the following characterization of containers whose interpretations carry an equational lifting monad structure.

Theorem 4. *The interpretation of an mnd-container* $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ *is an equational lifting monad if and only if it satisfies A2, B and C.*

Remember that every equational lifting monad is canonically a classifying monad. Theorems 2 and 4 tell us that the converse holds, if the monad is specified by a container and if restriction is given as in $(\star)$.

Corollary 2. *The interpretation of an mnd-container is an equational lifting monad if and only if it is a classifying monad with restriction operation defined as in* $(\star)$*.*

4.3 Effective Classifying Monads

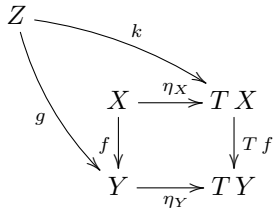
We move to effective classifying monads. As usual, we restrict our attention to classifying monads whose restriction operation is defined as in $(\star)$.

In **Set**, all pullbacks exist and any container functor preserves all of them. This means that in the category of sets and functions, an effective classifying monad is a classifying monad with Cartesian η.

Lemma 11. *The unit* η *is Cartesian if and only if the following condition holds:*

D $\mathsf{isContr}\,(P\,\mathsf{e})$

Proof. ($\Leftarrow$) Assume D. We are given the following data:



We have to construct a unique map $u : Z \to X$ such that $f \circ u = g$ and $\eta_X \circ u = k$. Given $z : Z$, if $k\,z = (s, h)$, then the commutativity of the outermost square gives us $s = \mathsf{e}$ and $f\,(h\,p) = g\,z$ for all $p : P\,\mathsf{e}$, since we have:

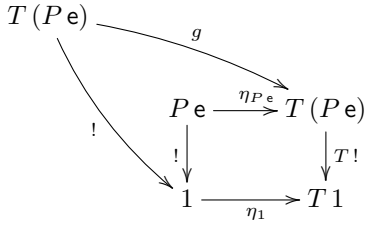
$$\eta_Y\,(g\,z) = (\mathsf{e}, \lambda\_.\, g\,z)$$

$$T\,f\,(k\,z) = (s, f \circ h)$$

By D, we are given an element $p : P\,\mathsf{e}$, so we define $u\,z =_{\mathrm{df}} h\,p$. We immediately have $f\,(u\,z) = g\,z$. Moreover, $\eta_X\,(u\,z) = (\mathsf{e}, \lambda\_.\, h\,p)$ which is equal to $k\,z$ since $h\,p = h\,q$ for all $q : P\,\mathsf{e}$ by D.

Let $u' : Z \to X$ be another mediating map. In particular, we have $\eta_X\,(u'\,z) = k\,z$, which implies $u'\,z = h\,p = u\,z$.

($\Rightarrow$) Assume that η is Cartesian. We consider the following diagram:



where $g\,(s, h) =_{\mathrm{df}} (\mathsf{e}, \mathsf{id}_{P\,e})$. Since the outermost diagram commutes, there exists a unique mediating map $u : T\,(P\,\mathsf{e}) \to P\,\mathsf{e}$. In particular, $P\,\mathsf{e}$ is inhabited by $u\,(\mathsf{e}, \mathsf{id}_{P\,\mathsf{e}})$. It remains to show that $P\,\mathsf{e}$ is a proposition. Let $(s, h) : T\,(P\,\mathsf{e})$. We know that $\eta_{P\,\mathsf{e}}\,(u\,(s, h)) = g\,(s, h)$, i.e., $(\mathsf{e}, \lambda\_.\, u\,(s, h)) = (\mathsf{e}, \mathsf{id}_{P\,\mathsf{e}})$. Then for all $p : P\,\mathsf{e}$, we have $p = u\,(s, h)$. So in particular $p = u\,(\mathsf{e}, \mathsf{id}_{P\,\mathsf{e}})$. Therefore $P\,\mathsf{e}$ is contractible.

□

Theorem 2 and Lemma 11 tell us that the interpretation of an mnd-container $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ is an effective classifying monad with restriction operation defined as in ($\star$) if and only if satisfies A2, B, C and D.

Conditions B, C and D, when considered together, are very powerful. Assuming conditions B and D, we obtain that $P\,s$ is logically equivalent to $s = \mathsf{e}$. The left-to-right direction is given by B. Vice versa, if $s = \mathsf{e}$, then $P\,s = P\,\mathsf{e}$ and therefore $P\,s$ holds by D. Assuming also C (and remembering that we are

assuming proposition extensionality), we obtain $P\,s = (s = \mathsf{e})$. Therefore, for containers whose interpretation extends to an effective classifying monad, P cannot be chosen, it is uniquely determined.

Mnd-containers whose interpretation carry an effective classifying monad structure can be given a more concise characterization.

Definition 5. An *effective classifier* is a set S equipped with operations

- $\mathsf{e} : S$
- $\bullet : \Pi s : S.\, ((s = \mathsf{e}) \to S) \to S$
- $q_0 : \Pi s : S.\, \Pi v : (s = \mathsf{e}) \to S.\, s \bullet v = \mathsf{e} \to s = \mathsf{e}$
- $q_1 : \Pi s : S.\, \Pi v : (s = \mathsf{e}) \to S.\, \Pi p : s \bullet v = \mathsf{e} \to v\,(q_0\,s\,v\,p) = \mathsf{e}$

satisfying $s \bullet (\lambda\_.\, \mathsf{e}) = s$ (i.e., the first shape equation of an mnd-container, henceforth denoted **RU**) and A2.

Theorem 5. *The interpretation of an mnd-container $(S, P, \mathsf{e}, \bullet, \upharpoonleft, \upharpoonright)$ is an effective classifying monad with restriction operation defined as in ($\star$) (so that, in particular, $P\,s = (s = \mathsf{e})$) if and only if $(S, \mathsf{e}, \bullet \upharpoonleft, \upharpoonright)$ is an effective classifier.*

Proof. We only need to show the right-to-left direction. In particular, we only need to check that, using RU and A2, it is possible to derive all the equations of mnd-containers. We know that the position equations hold since $s = \mathsf{e}$ is a proposition for all $s : S$.

The first shape equation is RU, so we only need to derive the other two. Notice that RU and A2 imply that the type family $\lambda s.\ (s = \mathsf{e})$ is injective, i.e., if $s = \mathsf{e} \leftrightarrow s' = \mathsf{e}$ then $s = s'$. In fact, from $s = \mathsf{e} \to s' = \mathsf{e}$ and $s' = \mathsf{e} \to s = \mathsf{e}$, we obtain two proofs, $r_1 : s \bullet (\lambda\_.\, \mathsf{e}) = s \bullet (\lambda\_.\, s')$ and $r_2 : s' \bullet (\lambda\_.\, \mathsf{e}) = s' \bullet (\lambda\_.\, s)$. Therefore we have:

$$s \overset{\text{RU}}{=} s \bullet (\lambda\_.\, \mathsf{e}) \overset{r_1}{=} s \bullet (\lambda\_.\, s') \overset{\text{A2}}{=} s' \bullet (\lambda\_.\, s) \overset{r_2}{=} s' \bullet (\lambda\_.\, \mathsf{e}) \overset{\text{RU}}{=} s' \tag{1}$$

Then, in order to prove $\mathsf{e} \bullet (\lambda\_.\, s) = s$, it is sufficient to construct the two implications $s = \mathsf{e} \to \mathsf{e} \bullet (\lambda\_.\, s) = \mathsf{e}$ and $\mathsf{e} \bullet (\lambda\_.\, s) = \mathsf{e} \to s = \mathsf{e}$. The first implication follows from RU, the second from the type of q_0.

The third shape equation is similarly derived employing the injectivity of $\lambda s.\ (s = \mathsf{e})$. □

4.4 Dominances

Traditionally, in topos theory partial map classifiers are specified by dominances, introduced by Rosolini in his PhD thesis [19]. Dominances are used to characterize the domain of definedness of partial maps, generalizing the notion of subobject classifier. In type theory, the notion of dominance was reformulated by Escardó and Knapp [14]. Here we give a definition which is equivalent to theirs.

Definition 6. A *dominance* is a set with operations

- $[\![-]\!] : D \to \Omega$,

– $1_D : D$,
– $\Sigma_D : \Pi X : D.\, (\llbracket X \rrbracket \to D) \to D$

satisfying

– $\mathsf{isInj}\, \llbracket - \rrbracket$,
– $\llbracket 1_D \rrbracket = 1$,
– $\llbracket \Sigma_D\, X\, Y \rrbracket = \Sigma x : \llbracket X \rrbracket.\, \llbracket Y\, x \rrbracket$.

The predicate isInj states the injectivity of a function. The type Ω is the type of all propositions, $\Omega =_{\mathrm{df}} \Sigma X : \mathcal{U}.\, \mathsf{isProp}\, X$.

In other words, a dominance is a set of (unique codes of) propositions closed under 1 and Σ. Interestingly, the notion of dominance is equivalent to the one of effective classifier.

Theorem 6. *A set carries a dominance structure if and only it carries an effective classifier structure.*

Proof. Let D be a set with a dominance structure.

First notice that, by the injectivity of $\llbracket - \rrbracket$ and the fact that $\llbracket 1_D \rrbracket = 1$, we have $\llbracket X \rrbracket = (X = 1_D)$. In fact, By injectivity of $\llbracket - \rrbracket$, we have $(\llbracket X \rrbracket = \llbracket 1_D \rrbracket) \leftrightarrow (X = 1_D)$. By proposition extensionality therefore $(\llbracket X \rrbracket = \llbracket 1_D \rrbracket) = (X = 1_D)$. Because $\llbracket 1_D \rrbracket = 1$, we have $(\llbracket X \rrbracket = \llbracket 1_D \rrbracket) = (\llbracket X \rrbracket = 1)$. If $\llbracket X \rrbracket$, then $\llbracket X \rrbracket \leftrightarrow 1$, so by propositional extensionality also $\llbracket X \rrbracket = 1$. Conversely, If $\llbracket X \rrbracket = 1$, then trivially $\llbracket X \rrbracket \leftrightarrow 1$ and further $\llbracket X \rrbracket$. So $\llbracket X \rrbracket \leftrightarrow (\llbracket X \rrbracket = 1)$. Hence by propositional extensionality one last time, $\llbracket X \rrbracket = (\llbracket X \rrbracket = 1)$. In summary, $\llbracket X \rrbracket = (\llbracket X \rrbracket = 1) = (\llbracket X \rrbracket = \llbracket 1_D \rrbracket) = (X = 1_D)$.

The type D is equipped with the following effective classifier structure:

– $\mathsf{e} =_{\mathrm{df}} 1_D$,
– $X \bullet Y =_{\mathrm{df}} \Sigma_D\, X\, Y$,
– q_0 and q_1 are definable, since we have $\llbracket \Sigma_D\, X\, Y \rrbracket = \Sigma x : \llbracket X \rrbracket.\, \llbracket Y\, x \rrbracket$, which is equivalent to $(\Sigma_D\, X\, Y = 1_D) \leftrightarrow \Sigma p : (X = 1_D).\, (Y\, p = 1_D)$. Using the latter logical equivalence, it is easy to prove both RU and A2.

Vice versa, let S be a set with an effective classifier structure. Then it is equipped with the following dominance structure:

– $\llbracket s \rrbracket =_{\mathrm{df}} (s = \mathsf{e})$,
– $1_D =_{\mathrm{df}} \mathsf{e}$,
– $\Sigma_D\, s\, v =_{\mathrm{df}} s \bullet v$,
– $\llbracket - \rrbracket$ is injective. The proof of this fact corresponds to Equation (1) in the proof of Theorem 5,
– $\llbracket 1_D \rrbracket = 1$ is trivially true,
– $\llbracket \Sigma_D\, s\, v \rrbracket = \Sigma p : \llbracket s \rrbracket.\, \llbracket v\, p \rrbracket$ holds. Indeed, using proposition extensionality, we only have to show that the two propositions are logically equivalent. Unfolding the definitions, we have that the left-to-right implications $s \bullet v = \mathsf{e} \to \Sigma p : s = \mathsf{e}.\, v\, p = \mathsf{e}$ corresponds precisely to the conjunction of q_0 and q_1. For the right-to-left implication, if $p : s = \mathsf{e}$ and $v\, p = \mathsf{e}$, then we have:

$$s \bullet v \overset{p,q}{=} \mathsf{e} \bullet (\lambda\_.\, \mathsf{e}) \overset{\mathrm{RU}}{=} \mathsf{e}$$

□

Theorem 6 shows that effective classifiers are the same as dominances. But the definition of effective classifier is more concise and easier to work with.

5 Examples

Imposed simultaneously, the characterizing conditions of effective classifying monads, i.e., conditions A2, B, C and D, are very strong and constrain the monad very much. But when some of them are dropped, more examples arise. In this section, we give examples of lifting monads specified by containers.

Maybe Monad, Exception Monad The maybe monad $T\,X =_{\mathrm{df}} X + 1$, or the exception monad with one exception, can be alternatively defined as the interpretation of the container $S =_{\mathrm{df}} \{\mathsf{ok}, \mathsf{err}\}$, $P\,\mathsf{ok} =_{\mathrm{df}} 1$, $P\,\mathsf{err} =_{\mathrm{df}} 0$. The mnd-container data are given by $\mathsf{e} =_{\mathrm{df}} \mathsf{ok}$, $\mathsf{ok} \bullet v =_{\mathrm{df}} v\,*$, $\mathsf{err} \bullet \_ =_{\mathrm{df}} \mathsf{err}$. These data satisfy all of A2, B, C and D. In fact, the maybe monad is the partial map classifier classifying partial maps with decidable domain of definedness.

One could ask if the exception monad with more than one exception carries the structure of a lifting monad. The answer is no. The exception monad with two exceptions, i.e., $T\,X =_{\mathrm{df}} X + 2$, can be alternatively defined as the interpretation of the container $S =_{\mathrm{df}} \{\mathsf{ok}, \mathsf{err}_0, \mathsf{err}_1\}$, $P\,\mathsf{ok} =_{\mathrm{df}} 1$, $P\,\mathsf{err}_i =_{\mathrm{df}} 0$. The mnd-container data are given by $\mathsf{e} =_{\mathrm{df}} \mathsf{ok}$, $\mathsf{ok} \bullet v =_{\mathrm{df}} v\,*$, $\mathsf{err}_i \bullet \_ =_{\mathrm{df}} \mathsf{err}_i$. These data satisfy B, C and D, but falsify A2, since $\mathsf{err}_0 \bullet (\lambda\_.\, \mathsf{err}_1) \neq \mathsf{err}_1 \bullet (\lambda\_.\, \mathsf{err}_0)$.

Delay Monad The underlying functor of Capretta's delay monad [7] is defined as the following coinductive type:

$$\frac{x : X}{\mathsf{now}\,x : \mathsf{D}\,X} \qquad \frac{c : \mathsf{D}\,X}{\mathsf{later}\,c : \mathsf{D}\,X}$$

It can be alternatively defined as the interpretation of the container $S =_{\mathrm{df}} \mathsf{co}\mathbb{N}$, $P\,n =_{\mathrm{df}} n{\downarrow}$ where $\mathsf{co}\mathbb{N} =_{\mathrm{df}} \mathsf{D}\,1$ is the set of conatural numbers and $\downarrow$ is the convergence predicate defined inductively by the rules

$$\frac{}{\mathsf{0}^{\downarrow} : 0{\downarrow}} \qquad \frac{n{\downarrow}}{\mathsf{suc}^{\downarrow} : (\mathsf{suc}\,n){\downarrow}}$$

We can define $\mathsf{e} =_{\mathrm{df}} 0$, $0 \bullet v =_{\mathrm{df}} v\,\mathsf{0}^{\downarrow}$, $\mathsf{suc}\,n \bullet v =_{\mathrm{df}} \mathsf{suc}\,(n \bullet v \circ \mathsf{suc}^{\downarrow})$. These data specify the monad structure introduced by Capretta [7], which validate A2, C and D, but neither A4 nor B. A4 fails since $n \bullet (\lambda\_.\, m) = n + m$, and therefore generally $(n \bullet v) \bullet (\lambda\_.\, n) \neq n \bullet v$. B fails since $n{\downarrow}$ does not generally imply $n = 0$.

If we instead define $\mathsf{suc}\,n \bullet v =_{\mathrm{df}} \mathsf{suc}\,(n \bullet \mathsf{pred} \circ v \circ \mathsf{suc}^{\downarrow})$, we also validate A4. In fact, now we have $n \bullet (\lambda\_.\, m) = \mathsf{max}\,(n, m)$. Therefore $(n \bullet v) \bullet (\lambda\_.\, n) = n \bullet v$. With this alternative definition of the operation $\bullet$, the delay datatype carries an almost-classifying monad structure.

Non-termination Monad To modify the delay monad example to validate B too, we can quotient $\mathsf{co}\mathbb{N}$ by weak bisimilarity $\approx$ defined by $n \approx m =_{\mathrm{df}} n{\downarrow} \leftrightarrow m{\downarrow}$, i.e., $S =_{\mathrm{df}} \mathsf{co}\mathbb{N}/{\approx}$. The new set S is typically called the *Sierpinski set* [13] or *Rosolini's dominance* [19]. The definitions of P and e are easily adjusted to this change: $P\,n =_{\mathrm{df}} n = [0]$, where $[m]$ indicate the equivalence class of m wrt. the equivalence relation $\approx$, and $\mathsf{e} =_{\mathrm{df}} [0]$. In order to lift the operation $\bullet$ to the quotient, it is necessary to invoke a choice principle. Chapman et al. [8] solve this issue assuming the axiom of countable choice, a semi-classical principle recently proved to be non-derivable in type theory [12]. A different solution is given by Escardó and Knapp [14] assuming the axiom of propositional choice from Rosolini propositions, i.e., propositions X for which the type $\|\Sigma n : \mathsf{co}\mathbb{N}.\,(n{\downarrow} \leftrightarrow X)\|$ is inhabited, where $\|A\|$ is the propositional truncation of the type A [20, Ch. 6.9]. It does not matter which version of $\bullet$ we lift to the quotient, addition or maximum, the resulting monad structure on the quotient is the same.

We call the interpretation of this mnd-container the *non-termination monad*.[3] It is the partial map classifier classifying partial maps with semidecidable domain of definedness.

The Sierpinski set can also be defined from scratch in homotopy type theory as a higher inductive type without recourse to the choice principle.

A first construction like this was by Altenkirch et al. [3] using higher inductive-inductive types. They defined a datatype $X_\perp$ as the free ω-complete pointed partial order on a type X. The join constructor $\bigcup$ has to take as input increasing streams of elements in $X_\perp$ (arbitrary streams do not have joins). Since this constructor has to take an increasingness proof as an argument, they had to use higher inductive-inductive types. They showed that the type $X_\perp$ is isomorphic to $\mathsf{D}\,X/{\approx}$ under the assumption of countable choice. The Sierpinski arises as $1_\perp$.

An alternative construction of the Sierpinski set was given by Chapman et al. [8] using standard higher inductive types. They defined the Sierpinski set S as the free countably-complete join semilattice on the unit type 1. This is to say that S is the countable powerset of 1. They noticed that the Sierpinski set can be specified using a join constructor $\bigvee$ that takes as input arbitrary streams of elements of $1_\perp$, not only the increasing ones. They showed that the types S and $1_\perp$ are isomorphic, which in turns also shows that S is isomorphic to $\mathsf{co}\mathbb{N}/{\approx}$ under the assumption of countable choice.

Full Partial Map Classifier The *full partial map classifier*, i.e., the partial map classifier classifying all partial maps, can be defined as the interpretation of the container $S =_{\mathrm{df}} \Omega$, $P\,X =_{\mathrm{df}} X$ where Ω is the type of all propositions introduced in Section 4.4.[4] The mnd-container data is given by $\mathsf{e} =_{\mathrm{df}} 1$, $X \bullet Y =_{\mathrm{df}} \Sigma x : X.\,Y\,x$. Notice that the latter type is a proposition, when X is a

[3] It is also called the partiality monad, but we want to use a more specific term here.

[4] Notice that, if $X : \mathcal{U}$, then $[\![S, P]\!]^{\mathrm{c}}\,X : \mathcal{U}_1$, i.e., $[\![S, P]\!]^{\mathrm{c}}$ would not be an endofunctor. But if $X : \mathcal{U}_1$, then also $[\![S, P]\!]^{\mathrm{c}}\,X : \mathcal{U}_1$. Therefore we think of $[\![S, P]\!]^{\mathrm{c}}$ as a monad on $\mathcal{U}_1$.

proposition and $Y\,x$ is a proposition for all $x : X$. These data satisfy all of A2, B, C and D.

Terminal Monad The terminal monad $T\,X =_{\mathrm{df}} 1$ can be alternatively defined as the interpretation of the container $S =_{\mathrm{df}} 1$, $P\,* =_{\mathrm{df}} 0$. The mnd-container data are trivial. These data satisfy A2, B and C, but falsify D, since $P\,*$ does not hold. The terminal monad is an example of a classifying monad that is not effective.

6 Conclusions

In this paper, we continued the study of monads of partiality in Martin-Löf type theory. In particular, we studied the connection between such monads and containers. Building on work by Ahman et al. [2, 21] on container comonads and container monads, we gave an algebraic characterization of containers whose interpretation carries a lifting monad structure.

We considered several notion of lifting monad from category theory: classifying monads [10], equational lifting monads [6], copy monads [11] and effective classifying monads [10], corresponding to partial map classifiers [15]. A byproduct of our investigation is the result that, for container monads, equational lifting monads are the same as classifying monads with restriction operation defined as in ($\star$). Similarly, copy monads are the same as almost-classifying monads with restriction operation defined as in ($\star$). Moreover, we discovered a new concise alternative definition of dominance [19] in type theory.

Recently, Uustalu and Veltri [22] introduced ω-complete pointed classifying monads, which are those classifying monads whose Kleisli category is ω**CPPO**-enriched wrt. the "less defined than" order on homsets induced by the restriction operation. These monads are used for modelling potential non-termination. Examples include the non-termination monad of Section 5, which is the initial ω-complete pointed classifying monad, and the full partial map classifier of Section 5. In future work, we plan to extend the characterization developed in this paper to containers whose interpretation carries a ω-complete pointed classifying monad structure. When considering the effective case, those containers should correspond to dominances closed under countable joins.

Acknowledgements We are thankful to Thorsten Altenkirch and Martín Escardó for discussions and valuable hints.

This research was supported by the ERDF funded Estonian national CoE project EXCITE and the Estonian Ministry of Education and Research institutional research grant IUT33-13.

References

1. Abbott, M., Altenkirch, T., Ghani, N.: Containers: constructing strictly positive types. Theor. Comput. Sci., 342(1), 3–27, 2005.

2. Ahman, D., Chapman, J., Uustalu, T.: When is a container a comonad? Log. Methods Comput. Sci., 10(3), article 14 (2014)
3. Altenkirch, T., Danielsson, N. A., Kraus, N.: Partiality, revisited: the partiality monad as a quotient inductive-inductive type. In: Esparza, J., Murawski, A. (eds.) FoSSaCS 2017, LNCS, vol. 10203, pp. 534–549. Springer, Heidelberg (2017)
4. Altenkirch, T., Pinyo, G.: Monadic containers and universes (abstract). In: Kaposi, A. (ed.) Abstracts of 23rd Int. Conf. on Types for Proofs and Programs, TYPES 2017, pp. 20–21. Eötvös Lórand University, Budapest (2017)
5. Bove, A., Krauss, A., M. Sozeau, M.: Partiality and recursion in interactive theorem provers—an overview. Math. Struct. Comput. Sci., 26(1), 38–88 (2016)
6. Bucalo, A., Führmann, C., Simpson, A.: An equational notion of lifting monad. Theor. Comput. Sci., 294(1–2), 31–60 (2003)
7. Capretta, V.: General recursion via coinductive types. Log. Methods Comput. Sci., 1(2), article 1 (2005)
8. Chapman, J., Uustalu, T., Veltri, N.: Quotienting the delay monad by weak bisimilarity. Math. Struct. Comput. Sci. (to appear)
9. Cockett, J. R. B., Lack, S.: Restriction categories I: categories of partial maps. Theor. Comput. Sci., 270(1–2), 223–259 (2002)
10. Cockett, J. R. B., Lack, S.: Restriction categories II: partial map classification. Theor. Comput. Sci, 294(1–2), 61–102 (2003)
11. Cockett, J. R. B., Lack, S.: Restriction categories III: colimits, partial limits, and extensivity. Math. Struct. Comput. Sci., 17(4), 775–817 (2007)
12. Coquand, T., Mannaa, B., Ruch, F.: Stack semantics of type theory. In: Proc. of 32th Ann. ACM/IEEE Symp. on Logic in Computer Science, LICS 2017 (2017)
13. Escardó, M. H.: Synthetic topology of data types and classical spaces. Electron. Notes Theor. Comput. Sci., 87, 21–156 (2004)
14. Escardó, M. H., Knapp, C. M.: Partial elements and recursion via dominances in univalent type theory. In: Goranko, V., Dam, M. (eds.) Proc. of 26th EACSL Ann. Conf. on Computer Science Logic, CSL 2017, Leibniz Int. Proc. in Informatics, vol. 82, article 21. Dagstuhl Publishing, Saarbrücken/Wadern (2017)
15. Johnstone, P. T.: Topos Theory. London Math. Soc. Monographs, vol. 10. Cambridge University Press (1977)
16. Moggi, E.: Notions of computation and monads. Inf. Comput., 93(1), 55–92 (1991)
17. Norell, U.: Dependently typed programming in Agda. In: Koopman, P., Plasmeijer, R., Swierstra, S. D. (eds.) AFP 2008, LNCS, vol. 5832, pp. 230–266. Springer, Heidelberg (2009)
18. Robinson, E., Rosolini, G.: Categories of partial maps. Inf. Comput., 79(2), 95–130 (1988)
19. Rosolini, G.: Continuity and Effectiveness in Topoi. DPhil thesis, University of Oxford (1986)
20. Univalent Foundations Program, The: Homotopy Type theory: Univalent Foundations of Mathematics. Institute for Advanced Study, Princeton, NJ (2013) `http://homotopytypetheory.org/book`
21. Uustalu, T.: Container combinatorics: monads and lax monoidal functors. In: Mousavi, M. R., Sgall, J. (eds.) TTCS 2017, LNCS. Springer, Heidelberg (to appear)
22. Uustalu, T., Veltri, N.: The delay monad and restriction categories. In: Hung, D. V., Kapur, D. (eds.) ICTAC 2017, LNCS, vol. 10580. Springer, Heidelberg (to appear)