Container Combinatorics: Monads and Lax Monoidal Functors

Tarmo Uustalu

▶ To cite this version:

Tarmo Uustalu. Container Combinatorics: Monads and Lax Monoidal Functors. 2nd International Conference on Topics in Theoretical Computer Science (TTCS), Sep 2017, Tehran, Iran. pp.91-105, 10.1007/978-3-319-68953-1_8 . hal-01760638

HAL Id: hal-01760638

https://inria.hal.science/hal-01760638

Submitted on 6 Apr 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Container Combinatorics: Monads and Lax Monoidal Functors

Tarmo Uustalu

Dept. of Software Science, Tallinn University of Technology
Akadeemia tee 21B, 12618 Tallinn, Estonia,
`tarmo@cs.ioc.ee`

Abstract. Abbott et al.'s containers are a "syntax" for a wide class of set functors in terms of shapes and positions. Containers whose "denotation" carries a comonad structure can be characterized as directed containers, or containers where a shape and a position in it determine another shape, intuitively a subshape of this shape rooted by this position. In this paper, we develop similar explicit characterizations for container functors with a monad structure and container functors with a lax monoidal functor structure as well as some variations. We argue that this type of characterizations make a tool, e.g., for enumerating the monad structures or lax monoidal functors that some set functor admits. Such explorations are of interest, e.g., in the semantics of effectful functional programming languages.

1 Introduction

Abbott et al.'s containers [1], a notational variant of polynomials, are a "syntax" for a wide class of set functors. They specify set functors in terms of shapes and positions. The idea is that an element of $F\,X$ should be given by a choice of a shape and an element of X for each of the positions in this shape; e.g., an element of $\mathsf{List}\,X$ is given by a natural number (the length of the list) and a matching number of elements of X (the contents of the list). Many constructions of set functors can be carried out on the level of containers, for example the product, coproduct of functors, composition and Day convolution of functors etc. One strength of containers is their usefulness for enumerating functors with specific structure or properties or with particular properties. It should be pointed out from the outset that containers are equivalent to simple polynomials in the sense of Gambino, Hyland and Kock [9,10,13,8], except that in works on polynomials one is often mainly interested in Cartesian polynomial morphisms whereas in works on containers general container morphisms are focussed on. The normal functors of Girard [11] are more constrained: a shape can only have finitely many positions.

Ahman et al. [3,4] sought to find a characterization of those containers whose interpretation carries a comonad structure in terms of some additional structure on the container, using that comonads are comonoids in the monoidal category of set functors. This additional structure, of what they called directed containers, turned out to be very intuitive: every position in a shape determines another shape, intuitively the subshape corresponding to this position; every shape has a distinguished root position; and positions in a subshape can be translated into positions in the shape. Directed containers are in fact the same as small categories, yet directed container morphisms are not functors, but cofunctors in the sense of Aguiar [2].

In this paper, we develop similar characterizations of container functors with a monad structure and those with a lax monoidal structure. We use that both monads and lax monoidal endofunctors are monoids in the category of set endofunctors wrt. its composition resp. Day convolution monoidal structures and that both monoidal structures are available also on the category of containers and preserved by interpretation into set functors. The relevant specializations of containers, which we here call mnd-containers and lmf-containers, are very similar, whereby every mnd-container turns out to also define an lmf-container.

Our motivation for this study is from programming language semantics and functional programming. Strong monads are a generally accepted means for organizing effects in functional programming since Moggi's seminal works. That strong lax monoidal endofunctors have a similar application was noticed first by McBride and Paterson [12] who called them applicative functors. That lax monoidal functors are the same as monoids in the Day convolution monoidal structure on the category of functors (under some assumptions guaranteeing that this monoidal structure is present) was noticed in this context by Capriotti and Kaposi [7]. It is sometimes of interest to find all monad or lax monoidal functor structures that a particular functor admits. Containers are a good tool for such explorations. We demonstrate this on a number of standard examples.

The paper is organized as follows. In Section 2, we review containers and directed containers as an explicit characterization of those containers whose interpretations carries a comonad structure. In Section 3, we analyze containers whose interpretation is a monad. In Section 4, we contrast this with an analysis of containers whose interpretation is a lax monoidal functor. In Section 5, we consider some specializations of monads and monoidal functors, to conclude in Section 6.

To describe our constructions on containers, we use type-theoretically inspired syntax, as we need dependent function and pair types throughout. For conciseness of presentation, we work in an informal extensional type theory, but everything we do can be formalized in intensional type theory. "Minor" ("implicit") arguments of functions are indicated as subscripts in Π-types, λ-abstractions and applications to enhance readability (cf. the standard notation for components of natural transformations). We use pattern-matching lambda-abstractions; _ is a "don't care" pattern.

The paper is a write-up of material that was presented by the author at the SSGEP 2015 summer school in Oxford[1], but was not published until now.

[1] See the slides at `http://cs.ioc.ee/~tarmo/ssgep15/`.

2 Containers, Directed Containers

2.1 Containers

We begin by a condensed review of containers [1].

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

A *container morphism* between two containers (S, P) and (S', P') is given by operations $t : S \to S'$ (the shape map) and $q : \Pi_{s:S}.\, P'\,(t\,s) \to P\,s$ (the position map). Note that while the shape map goes in the forward direction, the position map for a given shape goes in the backward direction.

The identity container morphism on (S, P) is $(\mathsf{id}_S, \lambda_s.\, \mathsf{id}_{P\,s})$. The composition of container morphisms $(t, q) : (S, P) \to (S', P')$ and $(t', q') : (S', P') \to (S'', P'')$ is $(t' \circ t, \lambda_s.\, q_s \circ q'_{t\,s})$. Containers and container morphisms form a category **Cont**.

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

A container morphism (t, q) between containers (S, P) and (S', P') interprets into a natural transformation $[\![t, q]\!]^{\mathrm{c}} = \tau$ between $[\![S, P]\!]^{\mathrm{c}}$ and $[\![S', P']\!]^{\mathrm{c}}$ where $\tau\,(s, v) = (t\,s, v \circ q_s)$.

Interpretation $[\![-]\!]^{\mathrm{c}}$ is a fully-faithful functor from **Cont** to $[\mathbf{Set}, \mathbf{Set}]$.

For example, the list functor can be represented by the container (S, P) where $S = \mathbb{N}$, because the shape of a list is a number—its length, and $P\,s = [0..s)$, as a position in a list of length s is a number between 0 and s, with the latter excluded. We have $[\![S, P]\!]^{\mathrm{c}}\,X = \Sigma s : \mathbb{N}.\,[0..s) \to X \cong \mathsf{List}\,X$, reflecting that to give a list amounts to choosing a length together with the corresponding number of elements. The list reversal function is represented by the container endomorphism (t, q) on (S, P) where $t\,s = s$, because reversing a list yields an equally long list, and $q_s\,p = s - p$, as the element at position p in the reversed list is the element at position $s - p$ in the given list. But the list self-append function is represented by (t, q) where $t\,s = s + s$ and $q_s\,p = p \mod s$.

There is an identity container defined by $\mathsf{Id}^{\mathrm{c}} = (1, \lambda *.\, 1)$. Containers can be composed, composition is defined by $(S, P) \cdot^{\mathrm{c}} (S', P') = (\Sigma s : S.\, P\,s \to S', \lambda(s, v).\, \Sigma p : P\,s.\, P'\,(v\,p))$. Identity and composition of containers provide a monoidal category structure on **Cont**.

Interpretation $[\![-]\!]^{\mathrm{c}}$ is a monoidal functor from $(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \cdot^{\mathrm{c}})$ to the strict monoidal category $([\mathbf{Set}, \mathbf{Set}], \mathsf{Id}, \cdot)$. Indeed, $\mathsf{Id}\,X = X \cong \Sigma * : 1.\, 1 \to X = [\![\mathsf{Id}^{\mathrm{c}}]\!]^{\mathrm{c}}\,X$ and $([\![S, P]\!]^{\mathrm{c}} \cdot [\![S', P']\!]^{\mathrm{c}})\,X = [\![S, P]\!]^{\mathrm{c}}\,([\![S', P']\!]^{\mathrm{c}}\,X) \cong \Sigma s : S.\, P\,s \to \Sigma s' : S'.\, P'\,s' \to X \cong \Sigma(s, v) : (\Sigma s : S.\, P\,s \to S').\,(\Sigma p : P\,s.\, P'\,(v\,p)) \to X = [\![(S, P) \cdot^{\mathrm{c}} (S', P')]\!]^{\mathrm{c}}\,X$.

Another monoidal category structure on **Cont** is symmetric. Define Hancock's tensor by $(S, P) \circledast^{\mathrm{c}} (S', P') = (S \times S', \lambda(s, s').\, P\,s \times P'\,s)$. Now $(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \circledast^{\mathrm{c}})$ form a symmetric monoidal category.

Interpretation $[\![-]\!]^{\mathrm{c}}$ is a symmetric monoidal functor from $(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \circledast^{\mathrm{c}})$ to the symmetric monoidal category $([\mathbf{Set}, \mathbf{Set}], \mathsf{Id}, \circledast)$ where $\circledast$ is the Day convolution defined by $(F \circledast G)\, Z = \int^{X,Y} (X \times Y \to Z) \times (F\, X \times G\, Y)$. Indeed,

$$\begin{aligned}
&[\![S, P]\!]^{\mathrm{c}} \circledast [\![S', P']\!]^{\mathrm{c}}\, Z \\
&= \int^{X,Y} (X \times Y \to Z) \times ((\Sigma s : S.\, P\, s \to X) \times (\Sigma s' : S'.\, P'\, s' \to Y)) \\
&\cong \Sigma(s, s') : S \times S'.\, \int^{X,Y} (X \times Y \to Z) \times ((P\, s \to X) \times (P'\, s' \to Y)) \\
&\cong \Sigma(s, s') : S \times S'.\, P\, s \times P'\, s' \to Z \\
&= [\![(S, P) \circledast^{\mathrm{c}} (S, P)]\!]^{\mathrm{c}}\, Z
\end{aligned}$$

2.2 Directed Containers

Next we review directed containers as a characterization those containers whose interpretation carries a comonad structure; we rely on [3,4].

A *directed container* is defined as a container (S, P) with operations

- $\downarrow\, : \Pi s : S.\, P\, s \to S$ (the subshape corresponding to a position in a shape),
- $\mathsf{o} : \Pi_{s:S}.\, P\, s$ (the root position), and
- $\oplus : \Pi_{s:S}.\, \Pi p : P\, s.\, P\, (s \downarrow p) \to P\, s$ (translation of a position in a position's subshape)

satisfying

- $s \downarrow \mathsf{o}_s = s$
- $s \downarrow (p \oplus_s p') = (s \downarrow p) \downarrow p'$
- $p \oplus_s \mathsf{o}_{s \downarrow p} = p$
- $\mathsf{o}_s \oplus_s p = p$
- $(p \oplus_s p') \oplus_s p'' = p \oplus_s (p' \oplus_{s \downarrow p} p'')$

The data $(\mathsf{o}, \oplus)$ resemble a monoid structure on P. However, P is not a set, but a family of sets, and $\oplus$ operates across the family. Similarly, $\downarrow$ resembles a right action of $(P, \mathsf{o}, \oplus)$ on S. When none of $P\, s$, o_s, $p \oplus_s p'$ depends on s, these data form a proper monoid structure and a right action.

A *directed container morphism* between two directed containers $(S, P, \downarrow, \mathsf{o}, \oplus)$ and $(S', P', \downarrow', \mathsf{o}', \oplus')$ is a morphism (t, q) between the underlying containers satisfying

- $t\, (s \downarrow q_s\, p) = t\, s \downarrow' p$
- $\mathsf{o}_s = q_s\, \mathsf{o}'_{t\, s}$
- $q_s\, p \oplus_s q_{s \downarrow q_s\, p}\, p' = q_s\, (p \oplus'_{t\, s} p')$

Directed containers form a category **DCont** whose identities and composition are inherited from **Cont**.

A directed container $(S, P, \downarrow, \mathsf{o}, \oplus)$ interprets into a comonad $[\![S, P, \downarrow, \mathsf{o}, \oplus]\!]^{\mathrm{dc}} = (D, \varepsilon, \delta)$ where

- $D = [\![S, P]\!]^{\mathrm{c}}$
- $\varepsilon\, (s, v) = v\, \mathsf{o}_s$
- $\delta\, (s, v) = (s, \lambda p.\, (s \downarrow p, \lambda p'.\, v\, (p \oplus_s p')))$

A directed container morphism (t, q) between $(S, P, \downarrow, \mathsf{o}, \oplus)$ and $(S', P', \downarrow', \mathsf{o}', \oplus')$ interprets into a comonad morphism $\llbracket t, q \rrbracket^{\mathrm{dc}} = \llbracket t, q \rrbracket^{\mathrm{c}}$ between $\llbracket S, P, \downarrow, \mathsf{o}, \oplus \rrbracket^{\mathrm{dc}}$ and $\llbracket S', P', \downarrow', \mathsf{o}', \oplus' \rrbracket^{\mathrm{dc}}$.

$\llbracket - \rrbracket^{\mathrm{dc}}$ is a fully-faithful functor between $\mathbf{DCont}$ and $\mathbf{Comonad}(\mathbf{Set})$. Moreover, the functor $\llbracket - \rrbracket^{\mathrm{dc}}$ is the pullback of the fully-faithful functor $\llbracket - \rrbracket^{\mathrm{c}} : \mathbf{Cont} \to [\mathbf{Set}, \mathbf{Set}]$ along $U : \mathbf{Comonad}(\mathbf{Set}) \to [\mathbf{Set}, \mathbf{Set}]$ and the category $\mathbf{DCont}$ is isomorphic to the category of comonoids in $(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \cdot^{\mathrm{c}})$.

$$\begin{tikzcd}
\begin{array}{c}\mathbf{DCont}\\ \cong \mathbf{Comonoid}(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \cdot^{\mathrm{c}})\end{array} \arrow[r, "U"] \arrow[dd, "\text{f.f.}\quad \llbracket - \rrbracket^{\mathrm{dc}}"'] & \mathbf{Cont} \arrow[dd] & (\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \cdot^{\mathrm{c}}) \arrow[dd, "\llbracket - \rrbracket^{\mathrm{c}}\quad \text{f.f.}"] \\
 & {} & {} \arrow[l, mapsto, "U"'] \\
\begin{array}{c}\mathbf{Comonad}(\mathbf{Set})\\ \cong \mathbf{Comonoid}([\mathbf{Set}, \mathbf{Set}], \mathsf{Id}, \cdot)\end{array} \arrow[r, "U"] & {[\mathbf{Set}, \mathbf{Set}]} & ([\mathbf{Set}, \mathbf{Set}], \mathsf{Id}, \cdot)
\end{tikzcd}$$

Here are some standard examples of directed containers and corresponding comonads.

Nonempty list functor (free semigroup functor) Let $D\,X = \mathsf{NEList}\,X = \mu Z.\, X \times (1 + Z) \cong \Sigma s : \mathbb{N}.\, [0..s] \to X$. We have $D\,X \cong \llbracket S, P \rrbracket^{\mathrm{c}}\, X$ for $S = \mathbb{N}$, $P\,s = [0..s]$.

The container (S, P) carries a directed container structure given by $s \downarrow p = s - p$, $\mathsf{o}_s = 0$, $p \oplus_s p' = p + p'$. Note that all three operations are well-defined: $p \leq s$ implies that $s - p$ is well-defined; $0 \leq s$; and $p \leq s$ and $p' \leq s - p$ imply $p + p' \leq s$.

The corresponding comonad has $\varepsilon\,(x : xs) = x$ (the head of xs), $\delta\,[x] = [[x]]$, $\delta\,(x : xs) = (x : xs) : \delta\, xs$ (the nonempty list of all nonempty suffixes of xs).

There are other directed container structures on (S, P). One is given by $s \downarrow p = s$, $\mathsf{o}_s = 0$, $p \oplus_s p' = (p + p') \mod s$. This directed container interprets into the comonad defined by $\varepsilon\, xs = \mathsf{hd}\, xs$, $\delta\, xs = \mathsf{shifts}\, xs$ (the nonempty list of all cyclic shifts of xs).

Exponent functor Let $D\,X = U \to X \cong 1 \times (U \to X)$ for some set U. We have $D\,X \cong \llbracket S, P \rrbracket^{\mathrm{c}}\, X$ for $S = 1$, $P\,{*} = U$.

Directed container structures on $\llbracket S, P \rrbracket^{\mathrm{c}}$ are in a bijection with monoid structures on U. Given a monoid structure $(\mathsf{i}, \otimes)$, the corresponding directed container structure is given by $* \downarrow p = *$, $\mathsf{o}_* = \mathsf{i}$, $p \oplus_* p' = p \otimes p'$.

The corresponding comonad has $\varepsilon\, f = f\, \mathsf{i}$, $\delta\, f = \lambda p.\, \lambda p'.\, f\,(p \otimes p')$.

Via the isomorphism $\mathsf{Str}\, X = \nu Z.\, X \times Z \cong \mathbb{N} \to X$, the special case of $(U, \mathsf{i}, \otimes) = (\mathbb{N}, 0, +)$ corresponds to the familiar stream comonad defined by $D\,X = \mathsf{Str}\,X$, $\varepsilon\, xs = \mathsf{hd}\, xs$ (the head of xs), $\delta\, xs = xs : \delta\,(\mathsf{tl}\, xs)$ (the stream of all suffixes of xs). A different special case $(U, \mathsf{i}, \otimes) = (\mathbb{N}, 1, *)$ corresponds to a different stream comonad given by $\varepsilon\, xs = \mathsf{hd}\,(\mathsf{tl}\, xs)$, $\delta\, xs = \mathsf{samplings}\, xs$ (the stream of all samplings of xs, where by the sampling of a stream $[x_0, x_1, x_2, \ldots]$ at rate p we mean the stream $[x_0, x_p, x_{p*2}, \ldots]$).

Product functor Let $D\,X = V \times X = V \times (1 \to X)$ for some set V. We have that $T\,X \cong [\![S,P]\!]^{\mathrm{c}}\,X$ for $S = V$, $P\,\_ = 1$.

Evidently there is exactly one directed container structure on (S,P); it is given by $s \downarrow * = s$, $\mathsf{o}_s = *$, $* \oplus_s * = *$.

The corresponding comonad has $\varepsilon\,(v,x) = x$, $\delta\,(v,x) = (v,(v,x))$.

We defined directed containers as containers with specific additional structure. But they are in a bijection (up to isomorphism) with something much more familiar—small categories. Indeed, a directed container $(S,P,\downarrow,\mathsf{o},\oplus)$ defines a small category as follows: the set of objects is S, the set of maps between s and s' is $\Sigma p : P\,s.\,(s \downarrow p = s')$; the identities and composition are given by o and $\oplus$. Any small category arises from a directed container uniquely in this fashion. The free category on a set V of objects (the discrete category with V as the set of objects), for example, arises from the directed container for the product comonad for V. However, directed container morphisms do not correspond to functors, since the shape map and position map of a container morphism go in opposite directions. A directed container morphism is reminiscent of a split opcleavage, except that, instead of a functor, it relies on an object mapping without an accompanying functorial action and accordingly the lift maps cannot be required to be opCartesian. A directed container morphism is a cofunctor (in the opposite direction) in the sense of Aguiar [2]. The category of directed containers is equivalent to the opposite of the category of small categories and cofunctors.

3 Containers ∩ Monads

There is no reason why the analysis of container functors with comonad structure could not be repeated for other types of functors with structure, the most obvious next candidate target being monads. The additional structure on containers corresponding to monads was sketched already in the original directed containers work [3]. Here we discuss the same characterization in detail.

We define an *mnd-container* to be a container (S,P) with operations

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

where we write $q_0\,s\,v\,p$ as $v \nwarrow_s p$ and $q_1\,s\,v\,p$ as $p \nearrow_v s$, satisfying

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

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

We can see that 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 $P\,s$ 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 a further difference is also that P is not a set, but a S-indexed family of sets.

We also define an *mnd-container morphism* between $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ and $(S', P', \mathsf{e}', \bullet', \nwarrow', \nearrow')$ to be a container morphism (t, q) between (S, P) and (S', P') such that

– $t\,\mathsf{e} = \mathsf{e}'$
– $t\,(s \bullet v) = t\,s \bullet' (t \circ v \circ q_s)$
– $v \nwarrow_s q_{s \bullet v}\,p = q_s\,((t \circ v \circ q_s) \nwarrow'_{t\,s} p)$
– $q_{s \bullet v}\,p \nearrow_v s = q_{v\,(v \nwarrow_s q_{s \bullet v}\,p)}\,(p \nearrow'_{t \circ v \circ q_s} (t\,s))$

Mnd-containers form a category **MCont** whose identity and composition are inherited from **Cont**.

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

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

Every mnd-container morphism (t, q) between $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ and $(S', P', \mathsf{e}', \bullet', \nwarrow', \nearrow')$ interprets into a monad morphism $[\![t, q]\!]^{\mathrm{mc}} = [\![t, q]\!]^{\mathrm{c}}$ between $[\![S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow]\!]^{\mathrm{mc}}$ and $[\![S', P', \mathsf{e}', \bullet', \nwarrow', \nearrow']\!]^{\mathrm{mc}}$.

$[\![-]\!]^{\mathrm{mc}}$ is a fully-faithful functor between **MCont** and **Monad**(**Set**). Moreover, the functor $[\![-]\!]^{\mathrm{mc}}$ is the pullback of the fully-faithful functor $[\![-]\!]^{\mathrm{c}}$: **Cont** $\to$ [**Set**, **Set**] along U : **Monad**(**Set**) $\to$ [**Set**, **Set**] and the category **MCont** is isomorphic to the category of monoids in (**Cont**, Id^{c}, $\cdot^{\mathrm{c}}$).

$$\begin{tikzcd}
\begin{array}{c}\mathbf{MCont}\\ \cong \mathbf{Monoid}(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \cdot^{\mathrm{c}})\end{array} \arrow[r, "U"] \arrow[d, "\text{f.f.}\ \ {[\![-]\!]^{\mathrm{mc}}}"'] & \mathbf{Cont} \arrow[d, ""{name=L}] & (\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \cdot^{\mathrm{c}}) \arrow[d, ""{name=R}, "{[\![-]\!]^{\mathrm{c}}}\ \ \text{f.f.}"] \\
\begin{array}{c}\mathbf{Monad}(\mathbf{Set})\\ \cong \mathbf{Monoid}([\mathbf{Set}, \mathbf{Set}], \mathsf{Id}, \cdot)\end{array} \arrow[r, "U"] & {[\mathbf{Set}, \mathbf{Set}]} & ([\mathbf{Set}, \mathbf{Set}], \mathsf{Id}, \cdot)
\arrow[from=R, to=L, maps to, "U"']
\end{tikzcd}$$

We consider as examples some containers interpreting into functors with a monad structure used in programming language semantics or functional programming.

Coproduct functor Let $T\,X = X + E$ for some set E. We have that $T\,X \cong [\![S, P]\!]^{\mathrm{c}}\,X$ for $S = 1 + E$, $P\,(\mathsf{inl}\,*) = 1$, $P\,(\mathsf{inr}\,\_) = 0$.

In a hypothetical mnd-container structure on (S, P), we cannot have $\mathsf{e} = \mathsf{inr}\,e_0$ for some $e_0 : E$, since then $P\,\mathsf{e} = 0$, but all elements of $0 \to S \cong 1$ are equal, in particular, $\lambda\_.\,\mathsf{inl}\,* = \lambda\_.\,\mathsf{inr}\,e_0 : 0 \to S$, so the 2nd mnd-container equation $\mathsf{e} \bullet (\lambda\_.\,s) = s$ cannot hold for both $s = \mathsf{inl}\,*$ and $s = \mathsf{inr}\,e_0$.

Therefore it must be that $\mathsf{e} = \mathsf{inl}\,*$. By the 2nd mnd-container equation then $\mathsf{inl}\,* \bullet v = \mathsf{e} \bullet (\lambda *.\,v\,*) = v\,*$ (since $P\,(\mathsf{inl}\,*) = 1$) whereas $\mathsf{inr}\,e \bullet v = \mathsf{inr}\,e \bullet (\lambda\_.\,\mathsf{e}) = \mathsf{inr}\,e$ by the 1st mnd-container equation (since $P\,(\mathsf{inr}\,e) = 0$).

To have $p : P\,(s \bullet v)$ is only possible, if $s = \mathsf{inl}\,*$, $v = \lambda *.\,\mathsf{inl}\,*$. In this case, $P\,(s \bullet v) = 1$ and $p = *$, and we can define $v \nwarrow_s p = *$ and $p \nearrow_v s = *$.

This choice of $(\mathsf{e}, \bullet, \nwarrow, \nearrow)$ satisfies all 8 equations of a mnd-container.

We see that the container (S, P) carries exactly one mnd-container structure. The corresponding monad structure on T is that of the exception monad, with $\eta\,x = \mathsf{inl}\,x$, $\mu\,(\mathsf{inl}\,c) = c$, $\mu\,(\mathsf{inr}\,e) = \mathsf{inr}\,e$.

List functor (free monoid functor) Let T be the list functor: $T\,X = \mathsf{List}\,X = \mu Z.\,1 + X \times Z \cong \Sigma s : \mathbb{N}.\,[0..s) \to X$. We have that $T\,X \cong [\![S, P]\!]^{\mathrm{c}}\,X$ for $S = \mathbb{N}$, $P\,s = [0..s)$.

The container (S, P) carries the following mnd-container structure:

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

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

This is not the only mnd-container structure available on (S, P). Another is $\mathsf{e} = 1$, $s \bullet \lambda\_.\,1 = s$, $1 \bullet \lambda 0.\,s = s$, $s \bullet v = 0$ otherwise, $\lambda\_.\,1 \nwarrow_s p = p$, $\lambda 0.\,s \nwarrow_1 p = 0$, $p \nearrow_{\lambda\_.\,1} s = 0$, $p \nearrow_{\lambda 0.\,s} 1 = p$.

The corresponding monad structure on T has $\eta\,x = [x]$, $\mu\,[[x_0], \ldots, [x_{v\,0-1}]] = [x_0, \ldots, x_{v\,0-1}]$, $\mu\,[xs] = xs$, $\mu\,xss = []$ otherwise.

Exponent functor Let $T\,X = U \to X$ for some set U and $S = 1$, $P\,* = U$.

There is exactly one mnd-container structure on (S, P) given by

- $\mathsf{e} = *$
- $* \bullet (\lambda\_.\,*) = *$
- $(\lambda\_.\,*) \nwarrow_* p = p$
- $p \nearrow_{\lambda\_.\,*} * = p$

Indeed, first note that the 1st to 3rd equations of an mnd-container are trivialized by $S = 1$. Further, $S = 1$ and the 4th and 5th equations force the definitions of $\nwarrow$ and $\nearrow$ and the remaining equations hold.

The corresponding monad structure on T is given by $\eta\,x = \lambda u.\,x$, $\mu\,f = \lambda u.\,f\,u\,u$. This is the well-known reader monad.

Product functor Let $T\,X = V \times X$ for some set V and $S = V$, $P\,\_ = 1$.

Any mnd-container structure on (S, P) must be of the form

- $\mathsf{e} = \mathsf{i}$
- $s \bullet (\lambda *.\, s') = s \otimes s'$
- $(\lambda *.\, s') \nwarrow_s * = *$
- $* \nearrow_{\lambda *.\, s'} s = *$

for some $\mathsf{i} : V$ and $\otimes : V \to V \to V$. The 1st to 3rd equations of an mnd-container reduce to the equations of a monoid while the remaining equations are trivialized by $P\,\_ = 1$. So mnd-container structures on (S, P) are in a bijective correspondence with monoid structures on V.

The corresponding monad structures on T have $\eta\, x = (\mathsf{i}, x)$, $\mu\,(p, (p', x)) = (p \otimes p', x)$. They are the writer monads for the different monoid structures on V.

Underlying functor of the state monad Let $T\,X = U \to U \times X \cong (U \to U) \times (U \to X)$ for some set U. We have $T\,X \cong [\![S, P]\!]^{\mathrm{c}}\, X$ for $S = U \to U$ and $P\,\_ = U$.

The container (S, P) admits the mnd-container structure defined by

- $\mathsf{e} = \lambda p.\, p$
- $s \bullet v = \lambda p.\, v\, p\, (s\, p)$
- $v \nwarrow_s p = p$
- $p \nearrow_v s = s\, p$

The corresponding monad structure on T is that of the state monad for U, given by $\eta\, x = \lambda u.\, (u, x)$ and $\mu\, f = \lambda u.\, \mathsf{let}\ (u', g) \leftarrow f\, u'\ \mathsf{in}\ g\, u'$.

This mnd-container structure is not unique; as a simplest variation, one can alternatively choose $s \bullet v = \lambda p.\, v\, p\, (s^n\, p)$, $p \nearrow_v s = s^n\, p$ for some fixed $n : \mathbb{N}$, with s^n denoting n-fold iteration of s.

Underlying functor of update monads Let $T\,X = U \to V \times X \cong (U \to V) \times (U \to X)$ for some sets U and V. We have $T\,X \cong [\![S, P]\!]^{\mathrm{c}}\, X$ for $S = U \to V$ and $P\,\_ = U$.

If $(\mathsf{i}, \otimes)$ is a monoid structure on V and $\downarrow$ its right action on U, then the container (S, P) admits the mnd-container structure defined by

- $\mathsf{e} = \lambda\_.\, \mathsf{i}$
- $s \bullet v = \lambda p.\, s\, p \otimes v\, p\, (s\, p)$
- $v \nwarrow_s p = p$
- $p \nearrow_v s = p \downarrow s\, p$

The corresponding monad structure on T is that of the update monad [5] for U, $(V, \mathsf{i}, \otimes)$ and $\downarrow$ given by $\eta\, x = \lambda u. (\mathsf{i}, x)$ and $\mu\, f = \lambda u.\, \mathsf{let}\ (p, g) \leftarrow f\, u; (p', x) \leftarrow g\, (u \downarrow p)\ \mathsf{in}\ (p \otimes p', x)$.

It should be clear that not every monad structure on T arises from some $(\mathsf{i}, \otimes)$ and $\downarrow$ in this manner.

The list functor 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 $S = O$, $P\, o = [0..\# o)$, $\mathsf{e} = \mathsf{id}$, $\bullet = \circ$ and $\nwarrow$, $\nearrow$ as in the definition of the (standard) list mnd-container. This choice of $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ gives an 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 version of non-symmetric operads where the argument places of an operation are identified nominally rather than positionally and operations may also have infinite arities.

Altenkirch and Pinyo [6] 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 set 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 is that there are no constructors for the denotations of 1 and Σ-types, and of course the equations governing the interaction of the constructors and the eliminators are then not enforced either. Thus 1 need not really denote the singleton set and Σ-types need not denote dependent products.

4 Containers ∩ Lax Monoidal Functors

We proceed to analyzing containers whose interpretation carries a lax monoidal functor structure wrt. the $(1, \times)$ monoidal category structure on **Set**. We will see that the corresponding additional structure on containers is very similar to that for monads, but simpler.

Recall that a *lax monoidal functor* between monoidal categories $(\mathcal{C}, I, \otimes)$ and $(\mathcal{C}', I', \otimes')$ is defined as a functor F between $\mathcal{C}$ and $\mathcal{C}'$ with a map $\mathsf{m}^0 : I' \to F I$ and a natural transformation with components $\mathsf{m}_{X,Y} : F X \otimes' F Y \to F(X \otimes Y)$ cohering with the unitors and associators of the two categories. A *lax monoidal transformation* between two lax monoidal functors $(F, \mathsf{m}^0, \mathsf{m})$ and $(F', \mathsf{m}^{0\prime}, \mathsf{m}')$ is a natural transformation $\tau : F \to F'$ such that $\tau_I \circ \mathsf{m}^0 = \mathsf{m}^{0\prime}$ and $\tau_{X \otimes Y} \circ \mathsf{m}_{X,Y} = \mathsf{m}'_{X,Y} \circ \tau_X \otimes' \tau_Y$.

We define an *lmf-container* as a container (S, P) with operations

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

where we write $q_0\, s\, s'\, p$ as $s' \nwarrow_s p$ and $q_1\, s\, s'\, p$ as $p \nearrow_{s'} s$, satisfying

- $\mathsf{e} \bullet s = s$
- $s = s \bullet e$
- $(s \bullet s') \bullet s'' = s \bullet (s' \bullet s'')$

– $\mathsf{e} \nwarrow_s p = p$
– $p \nearrow_s \mathsf{e} = p$
– $s' \nwarrow_s (s'' \nwarrow_{s \bullet s'} p) = (s' \bullet s'') \nwarrow_s p$
– $(s'' \nwarrow_{s \bullet s'} p) \nearrow_{s'} s = s'' \nwarrow_{s'} (p \nearrow_{s' \bullet s''} s)$
– $p \nearrow_{s''} (s \bullet s') = (p \nearrow_{s' \bullet s''} s) \nearrow_{s''} s'$

Differently from the mnd-container case, the data $(S, \mathsf{e}, \bullet)$ of a lmf-container form a proper monoid. The data $(\nwarrow, \nearrow)$ resemble a biaction of $(S, \mathsf{e}, \bullet)$.

We also define an *lmf-container morphism* between $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ and $(S', P', \mathsf{e}', \bullet', \nwarrow', \nearrow')$ to be a container morphism (t, q) between (S, P) and (S', P') such that

– $t\, \mathsf{e} = \mathsf{e}'$
– $t\, (s \bullet s') = t\, s \bullet' t\, s'$
– $s' \nwarrow_s q_{s \bullet s'}\, p = q_s\, (t\, s' \nwarrow'_{t\, s}\, p)$
– $q_{s \bullet s'}\, p \nearrow_{s'} s = q_{s'}\, (p \nearrow'_{t\, s'}\, t\, s)$

Lmf-containers form a category $\mathbf{LCont}$ whose identity and composition are inherited from $\mathbf{Cont}$.

Every lmf-container $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ interprets into a lax monoidal endofunctor $[\![S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow]\!]^{\mathrm{lc}} = (F, \mathsf{m}^0, \mathsf{m})$ on $(\mathbf{Set}, 1, \times)$ where

– $F = [\![S, P]\!]^{\mathrm{c}}$
– $\mathsf{m}^0 * = (\mathsf{e}, \lambda\_. *)$
– $\mathsf{m}\, ((s, v), (s', v')) = (s \bullet s', \lambda p.\, (v\, (s' \nwarrow_s p), v'\, (p \nearrow_{s'} s)))$

Every lmf-container morphism (t, q) between $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ and $(S', P', \mathsf{e}', \bullet', \nwarrow', \nearrow')$ interprets into a lax monoidal transformation $[\![t, q]\!]^{\mathrm{lc}} = [\![t, q]\!]^{\mathrm{c}}$ between $[\![S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow]\!]^{\mathrm{mc}}$ and $[\![S', P', \mathsf{e}', \bullet', \nwarrow', \nearrow']\!]^{\mathrm{lc}}$.

$[\![-]\!]^{\mathrm{lc}}$ is a fully-faithful functor between $\mathbf{LCont}$ and the category $\mathbf{LMF}(\mathbf{Set})$ of lax endofunctors on $(\mathbf{Set}, 1, \times)$. The functor $[\![-]\!]^{\mathrm{lc}}$ is the pullback of the fully-faithful functor $[\![-]\!]^{\mathrm{c}} : \mathbf{Cont} \to [\mathbf{Set}, \mathbf{Set}]$ along $U : \mathbf{LMF}(\mathbf{Set}) \to [\mathbf{Set}, \mathbf{Set}]$. The category $\mathbf{LCont}$ is isomorphic to the category of monoids in $(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \circledast^{\mathrm{c}})$.

$$\begin{tikzcd}
\begin{array}{c}\mathbf{LCont} \\ \cong \mathbf{Monoid}(\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \circledast^{\mathrm{c}})\end{array} \arrow[r, "U"] \arrow[d, "\text{f.f.} \quad [\![-]\!]^{\mathrm{lc}}"'] & \mathbf{Cont} \arrow[d, ""{name=A}] & (\mathbf{Cont}, \mathsf{Id}^{\mathrm{c}}, \circledast^{\mathrm{c}}) \arrow[d, "{[\![-]\!]^{\mathrm{c}} \quad \text{f.f.}}", ""'{name=B}] \\
\begin{array}{c}\mathbf{LMF}(\mathbf{Set}) \\ \cong \mathbf{Monoid}([\mathbf{Set}, \mathbf{Set}], \mathsf{Id}, \circledast)\end{array} \arrow[r, "U"] & {[\mathbf{Set}, \mathbf{Set}]} & ([\mathbf{Set}, \mathbf{Set}], \mathsf{Id}, \circledast)
\arrow[from=B, to=A, "U"', |->]
\end{tikzcd}$$

The similarity between the additional structures on containers for monads and lax monoidal functors may at first appear unexpected, but the reasons become clearer, if one compares the types of the "accumulating" Kleisli extension $\lambda(c, f).\, \mu\, (T\, (\lambda x.\, T\, (\lambda y.\, (x, y))\, (f\, x))\, c) : T\, X \times (X \to T\, Y) \to T\, (X \times Y)$ and the monoidality constraint $\mathsf{m} : F\, X \times F\, Y \to F\, (X \times Y)$.

It is immediate from the definitions that any mnd-container $(S, P, \mathsf{e}, \bullet, \nwarrow, \nearrow)$ carries an lmf-container structure $(\mathsf{e}', \bullet', \nwarrow', \nearrow')$ given by

– $\mathsf{e}' = \mathsf{e}$
– $s \bullet' s' = s \bullet (\lambda\_.\, s')$
– $s' \nwarrow'_s p = (\lambda\_.\, s') \nwarrow'_s p$
– $p \nearrow'_{s'} s = p \nearrow'_{\lambda\_.\, s'} s$

This is in agreement with the theorem that any strong monad defines a strong lax monoidal functor. Since any set functor is uniquely strong and all natural transformations between set functors are strong, the strength assumption and conclusion trivialize in our setting.

Another immediate observation is that, for any lmf-container structure $(\mathsf{e}, \bullet, \nwarrow, \nearrow)$ on (S, P), there is also a reverse lmf-container structure $(\mathsf{e}', \bullet', \nwarrow', \nearrow')$ given by

– $\mathsf{e}' = \mathsf{e}$
– $s \bullet' s' = s' \bullet s$
– $s' \nwarrow'_s p = p \nearrow_s s'$
– $p \nearrow'_{s'} s = s \nwarrow_{s'} p$

The corresponding statement about lax monoidal functors holds for any symmetric monoidal category.

Let us now revisit our example containers and see which lmf-container structures they admit.

Coproduct functor Let $T\,X = X + E$ for some set E and $S = 1 + E$, $P\,(\mathsf{inl}\,*) = 1$, $P\,(\mathsf{inr}\,\_) = 0$.

Any lmf-container structure on (S, P) must have $\mathsf{e} = \mathsf{inl}\,*$. Indeed, if it were the case $\mathsf{e} = \mathsf{inr}\,e_0$ for some $e_0 : E$, then we would have $\mathsf{inr}\,e_0 \bullet \mathsf{inl}\,* = \mathsf{inl}\,*$ by the 1st lmf-container equation. But then $q_0\,(\mathsf{inr}\,e_0)\,(\mathsf{inl}\,*) : 1 \to 0$, which cannot be.

Similarly, for all $e_0 : E$, $s : S$, it must be that $\mathsf{inr}\,e_0 \bullet s \neq \mathsf{inl}\,*$ and $s \bullet \mathsf{inr}\,e_0 \neq \mathsf{inl}\,*$. Hence, by the 1st and 2nd lmf-container equations, it must be the case that $\mathsf{inl}\,* \bullet s = s$, $\mathsf{inr}\,e \bullet \mathsf{inl}\,* = \mathsf{inr}\,e$, $\mathsf{inr}\,e \bullet \mathsf{inr}\,e' = \mathsf{inr}\,(e \otimes e')$. The 3rd lmf-container equation forces that $\otimes$ is a semigroup structure on E. The other lmf-container equations hold trivially. Therefore, lmf-container structures on (S, P) are in a bijection with semigroup structures on E.

The corresponding lax monoidal functors have $\mathsf{m}^0\,* = \mathsf{inl}\,*$, $\mathsf{m}\,(\mathsf{inl}\,x, \mathsf{inl}\,x') = \mathsf{inl}\,(x, x')$, $\mathsf{m}\,(\mathsf{inl}\,x, \mathsf{inr}\,e) = \mathsf{inr}\,e$, $\mathsf{m}\,(\mathsf{inr}\,e, \mathsf{inl}\,x) = \mathsf{inr}\,e$, $\mathsf{m}\,(\mathsf{inr}\,e, \mathsf{inr}\,e') = \mathsf{inr}\,(e \otimes e')$.

The unique mnd-container structure on (S, P) corresponds to the particular case of the left zero semigroup, i.e., the semigroup where $e \otimes e' = e$.

List functor Let $T\,X = \mathsf{List}\,X$ and $S = \mathbb{N}$, $P\,s = [0..s)$.

The standard mnd-container structure on (S, P) gives this lmf-container structure:

– $\mathsf{e} = 1$
– $s \bullet s' = s * s'$
– $s' \nwarrow_s p = p \;\mathrm{div}\; s'$
– $p \nearrow_{s'} s = p \;\mathrm{mod}\; s'$

The corresponding lax monoidal functor structure on T is given by $\mathsf{m}^0 \, * = [*]$, $\mathsf{m}\,(xs, ys) = [(x,y) \mid x \leftarrow xs, y \leftarrow ys]$.

The other mnd-container structure we considered gives $\mathsf{e} = 1$, $s \bullet 1 = s$, $1 \bullet s = s$, $s \bullet s' = 0$ otherwise, $1 \nwarrow_s p = p$, $s \nwarrow_1 p = 0$, $p \nearrow_1 s = 0$, $p \nearrow_s 1 = p$.

The corresponding lax monoidal functor structure on T is $\mathsf{m}^0 \, * = [*]$, $\mathsf{m}\,(xs, [y]) = [(x,y) \mid x \leftarrow xs]$, $\mathsf{m}\,([x], ys) = [(x,y) \mid y \leftarrow ys]$, $\mathsf{m}\,(xs, ys) = []$ otherwise.

But there are further lmf-container structures on (S, P) that do not arise from an mnd-container structure, for example this:

- $\mathsf{e} = 1$
- $s \bullet s' = s \min s'$
- $s' \nwarrow_s p = p$
- $p \nearrow_{s'} s = p$

The corresponding lax monoidal functor structure is $\mathsf{m}^0 \, * = [*]$, $\mathsf{m}\,(xs, ys) = \mathsf{zip}\,(xs, ys)$.

Exponent functor Let $T\,X = U \to X$ for some set U and $S = 1$, $P \, * = U$.

There is exactly one lmf-container structure on (S, P) given by

- $\mathsf{e} = *$
- $* \bullet * = *$
- $* \nwarrow_* p = p$
- $p \nearrow_* * = p$

and that is the lmf-container given by the unique mnd-container structure.

The corresponding lax monoidal functor structure on T is given by $\mathsf{m}^0 \, * = \lambda u.\, *$, $\mathsf{m}\,(f, f') = \lambda u.\,(f\,u, f'\,u)$.

Product functor Let $T\,X = V \times X$ for some set V and $S = V$, $P\,\_ = 1$.

Any lmf-container structure on (S, P) must be of the form

- $\mathsf{e} = \mathsf{i}$
- $s \bullet s' = s \otimes s'$
- $s' \nwarrow_s * = *$
- $* \nearrow_{s'} s = *$

for $(\mathsf{i}, \otimes)$ a monoid structure on V, so the only lmf-container structures are those given by mnd-structures.

The corresponding lax monoidal functor structures on T are given by $\mathsf{m}^0 \, * = (\mathsf{i}, *)$, $\mathsf{m}\,((p, x), (p', x')) = (p \otimes p', (x, x'))$.

Similarly to the monad case, we can generalize the list functor example. Now we are interested in relaxation of non-symmetric operads where parallel composition is only defined when the given n operations composed with the given n-ary operation are all the same, i.e., we have O a set of operations, $\# : O \to \mathbb{N}$ a function fixing the arity of each operation and $\mathsf{id} : O$ and $\circ : O \to O \to O$

an identity operation and a parallel composition operator, with $\#\,\mathsf{id} = 1$ and $\#\,(o \circ o') = \#\,o * \#\,o'$, satisfying the equations of an ordinary non-symmetric operad. If we now choose $S = O$, $P\,o = [0..\#\,o)$, $\mathsf{e} = \mathsf{id}$, $\bullet = \circ$ and take $\nwarrow$, $\nearrow$ as in the definition of the standard list lmf-container, we get a non-symmetric operad in this relaxed sense.

Under the lax type universe view, an lmf-container is a lax $(1, \times)$-universe, i.e., it is only closed under non-dependent lax Σ-types.

5 Further specializations

There are numerous special types of monads and lax monoidal functors that can be analyzed similarly. Here are some examples.

The lax monoidal functor interpreting an lmf-container is symmetric (i.e., satisfies $F\,\sigma_{X,Y} \circ \mathsf{m}_{X,Y} = \mathsf{m}_{Y,X} \circ \sigma_{FX,FY}$) if and only if the lmf-container is identical to its reverse, i.e., it satisfies

- $s \bullet s' = s' \bullet s$,
- $s' \nwarrow_s p = p \nearrow_s s'$

In this case, the monoid $(S, \mathsf{e}, \bullet)$ is commutative and each of the two action-like operations $\nwarrow$, $\nearrow$ determines the other.

The monad interpreting an mnd-container is commutative (which reduces to the corresponding lax monoidal functor being symmetric) if and only if

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

Note that, in this case, $\nwarrow$ and $\nearrow$ are constrained, but not to the degree of fully determining each other.

The monad interpreting an mnd-container is Cartesian (which means that all naturality squares of η and μ are pullbacks) if and only if

- the function $\lambda\_.\, * : P\,\mathsf{e} \to 1$ is an isomorphism
- for any $s : S$ and $v : P\,s \to S$, the function $\lambda p.\, (v \nwarrow_s p, p \nearrow_v s) : P\,(s \bullet v) \to \Sigma p : P\,s.\, P\,(v\,p)$ is an isomorphism.

Such mnd-containers with additional conditions are proper $(1, \Sigma)$-type universes: 1 and Σ-types denote the singleton set and dependent products.

6 Conclusion

We showed that the containers whose interpretation into a set functor carries a monad or a lax monoidal functor structure admit explicit characterizations similar to the directed container (or small category) characterization of those containers whose interpretation is a comonad. It was not surprising that such characterizations are possible, as we could build on the very same observations that were used in the analysis of the comonad case. But the elaboration of the characterizations is, we believe, novel. We also believe that it provides useful insights into the nature of monad or lax monoidal functor structures on container functors. In particular, it provides some clues on why monads and lax monoidal functors on **Set** and, more generally, in the situation of canonical strengths enjoy analogous properties. In future work, we would like to reach a better understanding of the connections of containers to operads.

Acknowledgments I am very grateful to Thorsten Altenkirch, Pierre-Louis Curien, Conor McBride, Niccolò Veltri for discussions. Paul-André Melliès pointed me to Aguiar's work. The anonymous reviewers of TTCS 2017 provided very useful feedback. This work was supported by the Estonian Ministry of Education and Research institutional research grant IUT33-13.

References

1. Abbott, M., Altenkirch, A., Ghani, N.: Containers: constructing strictly positive types. Theor. Comput. Sci., 342(1), 3–27 (2005) doi: 10.1016/j.tcs.2005.06.002
2. Aguiar, M.: Internal Categories and Quantum Groups. PhD thesis. Cornell University, Ithaca, NY (1997) `http://www.math.cornell.edu/~maguiar/thesis2.pdf`
3. Ahman, D., Chapman, J., Uustalu, T.: When is a container a comonad? Log. Methods Comput. Sci., 10(3), article 14 (2014) doi: 10.2168/lmcs-10(3:14)2014
4. Ahman, D., Uustalu, T.: Directed containers as categories. In: Atkey, R., Krishnaswami, N. (eds.) Proc. of 6th Wksh. on Mathematically Structured Functional Programming, MSFP 2016, Electron. Proc. in Theor. Comput. Sci., vol. 207, pp. 89–98. Open Publishing Assoc., Sydney (2016) doi: 10.4204/eptcs.207.5
5. Ahman, D., Uustalu, T.: Update monads: cointerpreting directed containers. In: Matthes, R., Schubert, A. (eds.) Proc. of 19th Conf. on Types for Proofs and Programs, Leibniz Int. Proc. in Inf., vol. 26, pp. 1–23. Dagstuhl Publishing, Saarbrücken/Wadern (2014) doi: 10.4230/lipics.types.2013.1
6. 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)
7. Capriotti, P., Kaposi, A.: Free applicative functors. In: Levy, P., Krishnaswami, N. (eds.) Proc. of 5th Wksh. on Mathematically Structured Functional Programming, MSFP 2014, Electron. Proc. in Theor. Comput. Sci., vol. 153, pp. 2–30. Open Publishing Assoc., Sydney (2014) doi: 10.4204/eptcs.153.2
8. Curien, P.-L.: Syntactic presentation of polynomial functors. Note (May 2017)
9. Gambino, N, Hyland, M.: Wellfounded trees and dependent polynomial functors. In: Berardi, S., Coppo, M., Damiani, F. (eds.) Revised Selected Papers from Int. Wksh. on Types for Proofs and Programs, TYPES 2003, Lect. Notes in Comput. Sci., vol. 2075, pp. 210–225. Springer (2004) doi: 10.1007/978-3-540-24849-1_14
10. Gambino, N., Kock, J.: Polynomial functors and polynomial monads. Math. Proc. Cambridge Philos. Soc. 154(1), 153–192 (2013) doi: 10.1017/s0305004112000394
11. Girard, J.-Y.: Normal functors, power series and lambda-calculus. Ann. Pure Appl. Log., 37(2), pp. 129–177 (1988) doi: 10.1016/0168-0072(88)90025-5
12. McBride, C. Paterson, R.: Applicative programming with effects. J. Funct. Program., 18(1), 1–13 (2008) doi: 10.1017/s0956796807006326
13. Weber, M.: Polynomials in categories with pullbacks. Theor. Appl. Categ., 30, pp. 533–598 (2015) `http://www.tac.mta.ca/tac/volumes/30/16/30-16abs.html`