Logical Methods in Computer Science
Vol. 10(3:14)2014, pp. 1–48
www.lmcs-online.org

Submitted Dec. 5, 2012
Published Sep. 03, 2014

WHEN IS A CONTAINER A COMONAD? *

DANEL AHMAN [a], JAMES CHAPMAN [b], AND TARMO UUSTALU [c]

[a] Laboratory for Foundations of Computer Science, School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, United Kingdom
e-mail address: d.ahman@ed.ac.uk

[b,c] Institute of Cybernetics at Tallinn University of Technology, Akadeemia tee 21, 12618 Tallinn, Estonia
e-mail address: {james,tarmo}@cs.ioc.ee

Abstract. Abbott, Altenkirch, Ghani and others have taught us that many parameterized datatypes (set functors) can be usefully analyzed via container representations in terms of a set of shapes and a set of positions in each shape. This paper builds on the observation that datatypes often carry additional structure that containers alone do not account for. We introduce directed containers to capture the common situation where every position in a data-structure determines another data-structure, informally, the sub-data-structure rooted by that position. Some natural examples are non-empty lists and node-labelled trees, and data-structures with a designated position (zippers). While containers denote set functors via a fully-faithful functor, directed containers interpret fully-faithfully into comonads. But more is true: every comonad whose underlying functor is a container is represented by a directed container. In fact, directed containers are the same as containers that are comonads. We also describe some constructions of directed containers. We have formalized our development in the dependently typed programming language Agda.

1. Introduction

Containers, as introduced by Abbott, Altenkirch and Ghani [2] are a neat representation for a wide class of parameterized datatypes (set functors) in terms of a set of shapes and a set of positions in each shape. They cover lists, colists, streams, various kinds of trees, etc. Containers can be used as a "syntax" for programming with these datatypes and reasoning about them, as can the strictly positive datatypes and polynomial functors of Dybjer [14],

2012 ACM CCS: [**Software and its engineering**]: Software notations and tools—General programming languages—Language features—Data types and structures; [**Theory of computation**]: Semantics and reasoning—Program semantics—Categorical semantics.

Key words and phrases: containers, comonads, datatypes, dependently typed programming, Agda.

* This article is a revised and expanded version of the FoSSaCS 2012 conference paper [5].

[a,b,c] The authors were supported by the ERDF funded CoE project EXCS, the Estonian Ministry of Education and Research target-financed theme no. 0140007s12, and the Estonian Science Foundation grants no. 9219 and 9475.

DOI:10.2168/LMCS-10(3:14)2014

Moerdijk and Palmgren [24], Gambino and Hyland [15], and Kock [23]. The theory of this class of datatypes is elegant, as they are well-behaved in many respects.

This paper proceeds from the observation that datatypes often carry additional structure that containers alone do not account for. We introduce directed containers to capture the common situation in programming where every position in a data-structure determines another data-structure, informally, the sub-data-structure rooted by that position. Some natural examples of such data-structures are non-empty lists and node-labelled trees, and data-structures with a designated position or focus (zippers). In the former case, the sub-data-structure is a sublist or a subtree. In the latter case, it is the whole data-structure but with the focus moved to the given position.

We show that directed containers are no less neat than containers. While containers denote set functors via a fully-faithful functor, directed containers interpret fully-faithfully into comonads. They admit some of the constructions that containers do, but not others: for instance, two directed containers cannot be composed in general. Our main result is that every comonad whose underlying functor is the interpretation of a container is the interpretation of a directed container. So the answer to the question in the title of this paper is: a container is a comonad exactly when it is a directed container. In more precise terms, the category of directed containers is the pullback of the forgetful functor from the category of comonads to that of set functors along the interpretation functor of containers. This also means that a directed container is the same as a comonoid in the category of containers.

In the core of the paper, we study directed containers on **Set**. Toward the end of the paper we point it out that the development could also be carried out more generally in locally Cartesian closed categories (LCCCs) and yet more generally in categories with pullbacks.

In our mathematics, we use syntax similar to the dependently typed functional programming language Agda [26, 27]. If some function argument will be derivable in most contexts, we mark it as implicit by enclosing it/its type in braces in the function's type declaration and either give this argument in braces or omit it in the definition and applications of the function.

We have formalized the central parts of the theory presented in Agda. The development is available at `http://cs.ioc.ee/~danel/dcont.html`.

Structure of the Article. In Section 2, we review the basic theory of containers, showing also some examples. We introduce containers and their interpretation into set functors. We show some constructions of containers such as the coproduct of containers. In Section 3, we revisit our examples and introduce directed containers as a specialization of containers and describe their interpretation into comonads. Our main result, that a container is a comonad exactly when it is directed, is the subject of Section 3.3. In Section 4, we look at some constructions, in particular the cofree directed container and the focussed container (zipper) construction. In addition, we also introduce strict directed containers and construct the product of two strict directed containers in the category of directed containers. Intuitively, a strict directed container is a directed container where no position in a non-root subshape of a shape translates to its root. In Section 5, we ask whether a similar characterization is possible for containers that are monads and hint that this is the case. In Section 6, we show that interpreting the opposite of the category of directed containers into set functors gives monads. In Section 7, we hint how the directed container

theory (presented in the paper for **Set**) could be developed in the more general setting of categories with pullbacks. We briefly summarize related work in Section 8 and conclude with outlining some directions for future work in Section 9. The proofs of the main results of Sections 3 and 4 appear in Appendices A and B.

We spend a section on the background theory of containers as they are central for our paper but relatively little known, but assume that the reader knows about comonads, monoidal categories and comonoids.

Differences from the FoSSaCS 2012 Conference Version. This article is a revised and expanded version of the FoSSaCS 2012 conference paper [5]. We have added many of the proofs that were omitted from the conference version. We have rearranged the different constructions on directed containers into a separate section, namely Section 4. In Section 4.3, we give a detailed discussion of cofree directed containers. In Section 4.2, which is entirely new, we define strict directed containers and coideal comonads and give an explicit formula for the product of two strict directed containers.

Likewise entirely new are the sections on cointerpreting directed containers in Section 6 and directed containers in categories with pullbacks in Section 7.

2. Containers

We begin with a recap of containers. We introduce the category of containers and the fully-faithful functor into the category of set functors defining the interpretation of containers and show that these are monoidal. We also recall some basic constructions of containers. For proofs of the propositions in this section and further information, we refer the reader to Abbott et al. [2, 1].

2.1. **Containers.** Containers are a form of "syntax" for datatypes. A *container* $S \lhd P$ is given by a set $S : \mathsf{Set}$ of *shapes* and a shape-indexed family $P : S \to \mathsf{Set}$ of *positions*. Intuitively, shapes are "templates" for data-structures and positions identify "blanks" in these templates that can be filled with data.

Example 2.1. The datatype of lists is represented by $S \lhd P$ where the shapes $S = \mathsf{Nat}$ are the possible lengths of lists and the positions $P\,s = \mathsf{Fin}\,s = \{0, \ldots, s-1\}$ provide s places for data in lists of length s. Non-empty lists are obtained by letting $S = \mathsf{Nat}$ and $P\,s = \mathsf{Fin}\,(s+1)$ (so that shape s has $s+1$ rather than s positions).

Example 2.2. Streams are characterized by a single shape with natural number positions: $S = 1 = \{*\}$ and $P\,* = \mathsf{Nat}$. The singleton datatype has one shape and one position: $S = 1$, $P\,* = 1$.

A *morphism* between containers $S \lhd P$ and $S' \lhd P'$ is a pair $t \lhd q$ of maps $t : S \to S'$ and $q : \Pi\{s : S\}.\, P'\,(t\,s) \to P\,s$ (the shape map and position map). Note how the positions are mapped backwards. The intuition is that, if a function between two datatypes does not look at the data, then the shape of a data-structure given to it must determine the shape of the data-structure returned and the data in any position in the shape returned must come from a definite position in the given shape.

Examples 2.3.

- The head function, sending a non-empty list to a single data item, is determined by the maps $t : \mathsf{Nat} \to 1$ and $q : \Pi\{s : \mathsf{Nat}\}.\, 1 \to \mathsf{Fin}\,(s+1)$ defined by $t\,\_ = *$ and $q\,* = 0$.
- The tail function, sending a non-empty list to a list, is represented by $t : \mathsf{Nat} \to \mathsf{Nat}$ and $q : \Pi\{s : \mathsf{Nat}\}.\, \mathsf{Fin}\,s \to \mathsf{Fin}\,(s+1)$ defined by $t\,s = s$ and $q\,p = p + 1$.
- For the function dropping every second element of a non-empty list, the shape and position maps $t : \mathsf{Nat} \to \mathsf{Nat}$ and $q : \Pi\{s : \mathsf{Nat}\}.\, \mathsf{Fin}\,(s \div 2 + 1) \to \mathsf{Fin}\,(s+1)$ are $t\,s = s \div 2$ and $q\,p = p * 2$.
- For self-append of a non-empty list, they are $t : \mathsf{Nat} \to \mathsf{Nat}$ and $q : \Pi\{s : \mathsf{Nat}\}.\, \mathsf{Fin}\,(s * 2 + 2) \to \mathsf{Fin}\,(s+1)$ defined by $t\,s = s * 2 + 1$ and $q\,\{s\}\,p = p \mod (s+1)$.
- For reversal of non-empty lists, they are $t : \mathsf{Nat} \to \mathsf{Nat}$ and $q : \Pi\{s : \mathsf{Nat}\}.\, \mathsf{Fin}\,(s+1) \to \mathsf{Fin}\,(s+1)$ defined by $t\,s = s$ and $q\,\{s\}\,p = s - p$.

(See Prince et al. [28] for more similar examples.)

The *identity* morphism $\mathsf{id}^{\mathrm{c}}\{C\}$ on a container $C = S \lhd P$ is defined by $\mathsf{id}^{\mathrm{c}} = \mathsf{id}\,\{S\} \lhd \lambda\{s\}.\, \mathsf{id}\,\{P\,s\}$. The *composition* $h \circ^{\mathrm{c}} h'$ of container morphisms $h = t \lhd q$ and $h' = t' \lhd q'$ is defined by $h \circ^{\mathrm{c}} h' = t \circ t' \lhd \lambda\{s\}.\, q'\,\{s\} \circ q\,\{t'\,s\}$. Composition of container morphisms is associative, identity is the unit.

Proposition 2.4. *Containers form a category* **Cont**.

2.2. **Interpretation of Containers.** To map containers into datatypes made of data-structures that have the positions in some shape filled with data, we must equip containers with a "semantics".

For a container $C = S \lhd P$, we define its *interpretation* $[\![C]\!]^{\mathrm{c}} : \mathsf{Set} \to \mathsf{Set}$ on sets by $[\![C]\!]^{\mathrm{c}}\,X = \Sigma s : S.\, P\,s \to X$, so that $[\![C]\!]^{\mathrm{c}}\,X$ consists of pairs of a shape and an assignment of an element of X to each of the positions in this shape, reflecting the intuitive reading that shapes are "templates" for datatypes and positions identify "blanks" in these templates that can be filled in with data. The interpretation $[\![C]\!]^{\mathrm{c}} : \forall\{X\}, \{Y\}.\, (X \to Y) \to (\Sigma s : S.\, P\,s \to X) \to \Sigma s : S.\, P\,s \to Y$ of C on functions is defined by $[\![C]\!]^{\mathrm{c}}\,f\,(s, v) = (s, f \circ v)$. It is straightforward that $[\![C]\!]^{\mathrm{c}}$ preserves identity and composition of functions, so it is a set functor (as any datatype should be).

Example 2.5. Our example containers denote the datatypes intended. If we let C be the container of lists, we have $[\![C]\!]^{\mathrm{c}}\,X = \Sigma s : \mathsf{Nat}.\, \mathsf{Fin}\,s \to X \cong \mathsf{List}\,X$. The container of streams interprets into $\Sigma * : 1.\, \mathsf{Nat} \to X \cong \mathsf{Nat} \to X \cong \mathsf{Str}\,X$. Etc.

A morphism $h = t \lhd q$ between containers $C = S \lhd P$ and $C = S' \lhd P'$ is interpreted as a natural transformation between $[\![C]\!]^{\mathrm{c}}$ and $[\![C']\!]^{\mathrm{c}}$, i.e., as a polymorphic function $[\![h]\!]^{\mathrm{c}} : \forall\{X\}.\, (\Sigma s : S.\, P\,s \to X) \to \Sigma s' : S'.\, P'\,s' \to X$ that is natural. It is defined by $[\![h]\!]^{\mathrm{c}}\,(s, v) = (t\,s, v \circ q\,\{s\})$. $[\![-]\!]^{\mathrm{c}}$ preserves the identities and composition of container morphisms.

Example 2.6. The interpretation of the container morphism h for the list head function is $[\![h]\!]^{\mathrm{c}} : \forall\{X\}.\, (\Sigma s : \mathsf{Nat}.\, \mathsf{Fin}\,(s+1) \to X) \to \Sigma * : 1.\, 1 \to X$ defined by $[\![h]\!]^{\mathrm{c}}\,(s, v) = (*, \lambda *.\, v\,0)$.

Proposition 2.7. $[\![-]\!]^{\mathrm{c}}$ *is a functor from* **Cont** *to* $[\mathbf{Set}, \mathbf{Set}]$.

Every natural transformation between container interpretations is the interpretation of some container morphism. For containers $C = S \lhd P$ and $C' = S' \lhd P'$, a natural transformation τ between $[\![C]\!]^{\mathrm{c}}$ and $[\![C']\!]^{\mathrm{c}}$, i.e., a polymorphic function $\tau : \forall\{X\}.\, (\Sigma s : S.\, P\,s \to X) \to \Sigma s' : S'.\, P'\,s' \to X$ that is natural, can be "quoted" to a container morphism

$\ulcorner \tau \urcorner^{\mathrm{c}} = t \lhd q$ between C and C' where $t : S \to S'$ and $q : \Pi\{s : S\}.\, P'\,(t\,s) \to P\,s$ are defined by $\ulcorner \tau \urcorner^{\mathrm{c}} = (\lambda s.\, \mathsf{fst}\,(\tau\,\{P\,s\}\,(s, \mathsf{id}))) \lhd (\lambda\{s\}.\, \mathsf{snd}\,(\tau\,\{P\,s\}\,(s, \mathsf{id})))$.

For any container morphism h, $\ulcorner [\![h]\!]^{\mathrm{c}} \urcorner^{\mathrm{c}} = h$, and, for any natural transformation τ and τ' between container interpretations, $\ulcorner \tau \urcorner^{\mathrm{c}} = \ulcorner \tau' \urcorner^{\mathrm{c}}$ implies $\tau = \tau'$.

Proposition 2.8. $[\![-]\!]^{\mathrm{c}}$ *is fully faithful.*

2.3. **Monoidal Structure.** We have already seen the *identity* container $\mathsf{Id}^{\mathrm{c}} = 1 \lhd \lambda *.\, 1$. The *composition* $C_0 \cdot^{\mathrm{c}} C_1$ of containers $C_0 = S_0 \lhd P_0$ and $C_1 = S_1 \lhd P_1$ is the container $S \lhd P$ defined by $S = \Sigma s : S_0.\, P_0\,s \to S_1$ and $P\,(s, v) = \Sigma p_0 : P_0\,s.\, P_1\,(v\,p_0)$. It has as shapes pairs of an outer shape s and an assignment of an inner shape to every position in s. The positions in the composite container are pairs of a position p in the outer shape and a position in the inner shape assigned to p. The (horizontal) composition $h_0 \cdot^{\mathrm{c}} h_1$ of container morphisms $h_0 = t_0 \lhd q_0$ and $h_1 = t_1 \lhd q_1$ is the container morphism $t \lhd q$ defined by $t\,(s, v) = (t_0\,s, t_1 \circ v \circ q_0\,\{s\})$ and $q\,\{s, v\}\,(p_0, p_1) = (q_0\,\{s\}\,p_0, q_1\,\{v\,(q_0\,\{s\}\,p_0)\}\,p_1)$. The horizontal composition preserves the identity container morphisms and the (vertical) composition of container morphisms, which means that $- \cdot^{\mathrm{c}} -$ is a bifunctor.

Cont has isomorphisms $\rho : \forall\{C\}.\, C \cdot^{\mathrm{c}} \mathsf{Id}^{\mathrm{c}} \to C$, $\lambda : \forall\{C\}.\, \mathsf{Id}^{\mathrm{c}} \cdot^{\mathrm{c}} C \to C$ and $\alpha : \forall\{C, C', C''\}.\, (C \cdot^{\mathrm{c}} C') \cdot^{\mathrm{c}} C'' \to C \cdot^{\mathrm{c}} (C' \cdot^{\mathrm{c}} C'')$, given by $\rho = \lambda(s, v).\, s \lhd \lambda\{s, v\}.\, \lambda p.\, (p, *)$, $\lambda = \lambda(*, v).\, v * \lhd \lambda\{*, v\}.\, \lambda p.\, (*, p)$ and $\alpha = \lambda((s, v), v').\, (s, \lambda p.\, (v\,p, \lambda p'.\, v'\,(p, p'))) \lhd \lambda\{(s, v), v'\}.\, \lambda(p, (p', p'')).\, ((p, p'), p'')$. They satisfy Mac Lane's coherence conditions.

Proposition 2.9. *The category* **Cont** *is a monoidal category.*

There are also natural isomorphisms $\mathsf{e} : \mathsf{Id} \to [\![\mathsf{Id}^{\mathrm{c}}]\!]^{\mathrm{c}}$ and $\mathsf{m} : \forall\{C_0, C_1\}.\, [\![C_0]\!]^{\mathrm{c}} \cdot [\![C_1]\!]^{\mathrm{c}} \to [\![C_0 \cdot^{\mathrm{c}} C_1]\!]^{\mathrm{c}}$ defined by $\mathsf{e}\,x = (*, \lambda *.\, x)$ and $\mathsf{m}\,(s, v) = ((s, \lambda p.\, \mathsf{fst}\,(v\,p)), \lambda(p, p').\, \mathsf{snd}\,(v\,p)\,p')$ satisfying the appropriate coherence conditions.

Proposition 2.10. *The functor* $[\![-]\!]^{\mathrm{c}}$ *is a monoidal functor.*

2.4. **Constructions of Containers.** Containers are closed under various constructions such as products, coproducts and constant exponentiation, preserved by interpretation.

Products. For two containers $C_0 = S_0 \lhd P_0$ and $C_1 = S_1 \lhd P_1$, their *product* $C_0 \times C_1$ is the container $S \lhd P$ defined by $S = S_0 \times S_1$ and $P\,(s_0, s_1) = P_0\,s_0 + P_1\,s_1$. It holds that $[\![C_0 \times C_1]\!]^{\mathrm{c}} \cong [\![C_0]\!]^{\mathrm{c}} \times [\![C_1]\!]^{\mathrm{c}}$.

Coproducts. The *coproduct* $C_0 + C_1$ of containers $C_0 = S_0 \lhd P_0$ and $C_1 = S_1 \lhd P_1$ is the container $S \lhd P$ defined by $S = S_0 + S_1$, $P\,(\mathsf{inl}\,s) = P_0\,s$ and $P\,(\mathsf{inr}\,s) = P_1\,s$. It is the case that $[\![C_0 + C_1]\!]^{\mathrm{c}} \cong [\![C_0]\!]^{\mathrm{c}} + [\![C_1]\!]^{\mathrm{c}}$.

Exponentials. For a set $K \in \mathsf{Set}$ and a container $C_0 = S_0 \lhd P_0$, the *exponential* $K \to C_0$ is the container $S \lhd P$ where $S = K \to S_0$ and $P\,f = \Sigma k : K.\, P\,(f\,k)$. We have that $[\![K \to C_0]\!]^{\mathrm{c}} \cong K \to [\![C_0]\!]^{\mathrm{c}}$.

3. Directed Containers

We now proceed to our contribution, directed containers. We define the category of directed containers and a fully-faithful functor interpreting directed containers as comonads, and discuss some examples and constructions.

3.1. **Directed Containers.** Parametrized datatypes often carry some additional structure that is worth making explicit. For example, each node in a list or non-empty list defines a sublist (a suffix). In container terms, this corresponds to every position in a shape determining another shape, the subshape corresponding to this position. The theory of containers alone does not account for such additional structure. Directed containers, studied in the rest of this paper, axiomatize subshapes and translation of positions in a subshape into the global shape.

Definition 3.1. A *directed container* is a container $S \lhd P$ together with three operations

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

satisfying the following two shape equations and three position equations:

(1) $\forall\{s\}.\, s \downarrow \mathsf{o} = s$,
(2) $\forall\{s, p, p'\}.\, s \downarrow (p \oplus p') = (s \downarrow p) \downarrow p'$,
(3) $\forall\{s, p\}.\, p \oplus \{s\}\, \mathsf{o} = p$,
(4) $\forall\{s, p\}.\, \mathsf{o}\,\{s\} \oplus p = p$,
(5) $\forall\{s, p, p', p''\}.\, (p \oplus \{s\}\, p') \oplus p'' = p \oplus (p' \oplus p'')$.

(Using $\oplus$ as an infix operation, we write the first, implicit, argument next to the operation symbol when we want to give it explicitly.) Modulo the fact that the positions involved come from different sets, laws 3–5 are the laws of a monoid. In the special case $S = 1$, we have exactly one set of positions, namely $P\,*$, and that is a monoid. If S is general, but $s \downarrow p$ does not depend on p (in this case $s \downarrow p = s$ thanks to law 1), then each $P\, s$ is a monoid. (One might also notice that laws 1–2 bear similarity to the laws of a monoid action. If none of $P\, s$, $\mathsf{o}\,\{s\}$, $p \oplus \{s\}\, p'$ depends on s, then we have one single monoid and $\downarrow$ is then a right action of that monoid on S.)

To help explain the operations and laws, we sketch in Fig. 1 a data-structure with nested sub-data-structures.

The global shape s is marked with a solid boundary and has a root position $\mathsf{o}\,\{s\}$. Then, any position p in s determines a shape $s' = s \downarrow p$, marked with a dotted boundary, to be thought of as the subshape of s given by this position. The root position in s' is $\mathsf{o}\,\{s'\}$. Law 3 says that its translation $p \oplus \mathsf{o}\,\{s'\}$ into a position in shape s is p, reflecting the idea that the subshape given by a position should have that position as the root.

By law 1, the subshape $s \downarrow \mathsf{o}\,\{s\}$ corresponding to the root position $\mathsf{o}\{s\}$ in the global shape s is s itself. Law 4, which is only well-typed thanks to law 1, stipulates that the translation of position p in $s \downarrow \mathsf{o}\,\{s\}$ into a position in s is just p (which is possible, as $P\,(s \downarrow \mathsf{o}\,\{s\}) = P\, s$).

A further position p' in s' determines a shape $s'' = s' \downarrow p'$. But p' also translates into a position $p \oplus p'$ in s and that determines a shape $s \downarrow (p \oplus p')$. Law 2 says that s'' and $s \downarrow (p \oplus p')$ are the same shape, which is marked by a dashed boundary in the

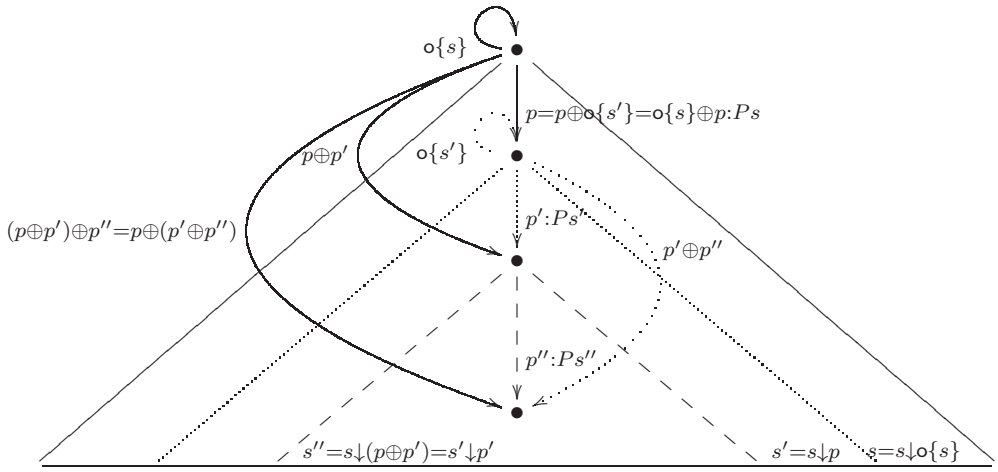


Figure 1: A data-structure with two nested sub-data-structures

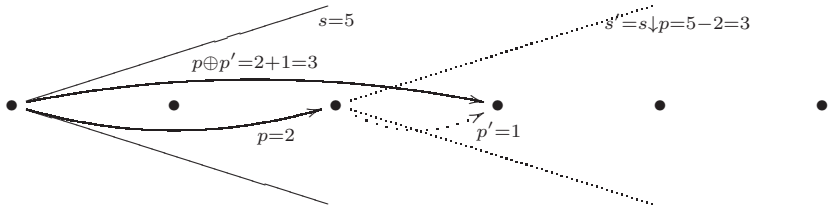


Figure 2: The shape and positions for non-empty lists of length 6

figure. Finally, law 5 (well-typed only because of law 2) says that the two alternative ways to translate a position p'' in shape s'' into a position in shape s agree with each other.

Example 3.2. Lists cannot form a directed container, as the shape 0 (for the empty list), having no positions, has no possible root position.

But the container of *non-empty lists* (with $S = \mathsf{Nat}$ and $P\,s = \mathsf{Fin}\,(s+1)$) is a directed container with respect to *non-empty suffixes* as sublists. The subshape given by a position p in a shape s (for lists of length $s+1$) is the shape of the corresponding suffix, given by $s \downarrow p = s - p$. The root $\mathsf{o}\,\{s\}$ is the position 0 of the head node. A position in the global shape is recovered from a position p' in the subshape of the position p by $p \oplus p' = p + p'$.

Fig. 2 shows an example of the shape and positions of a non-empty list with length 6, i.e., with shape $s = 5$. This figure also shows that the subshape determined by a position $p = 2$ in the global shape s is $s' = s \downarrow p = 5 - 2 = 3$ and a position $p' = 1$ in s' is rendered as the position $p \oplus p' = 2 + 1 = 3$ in the initial shape. Clearly one could also choose prefixes as subshapes and the last node of a non-empty list as the root, but this gives an isomorphic directed container.

Example 3.3. Non-empty lists also give rise to an entirely different directed container structure that has *cyclic shifts* as "sublists" (this example was suggested to us by Jeremy Gibbons). The subshape at each position is the global shape ($s \downarrow p = s$). The root is

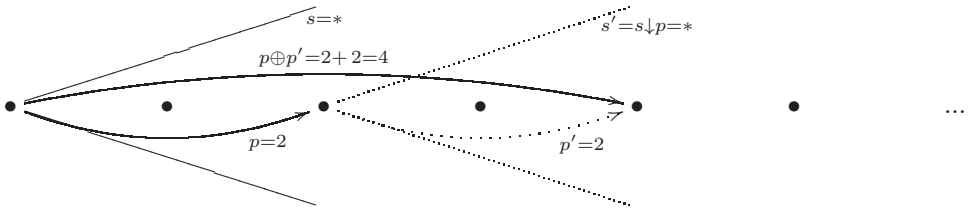


Figure 3: The shape and positions for streams

still $\mathsf{o}\,\{s\} = 0$. The interesting part is that translation into the global shape of a subshape position is defined by $p \oplus \{s\}\, p' = (p + p') \mod (s + 1)$, satisfying all the required laws.

Example 3.4. The container of *streams* ($S = 1$, $P\,* = \mathsf{Nat}$) carries a very trivial directed container structure given by $* \downarrow p = *$, $\mathsf{o} = 0$ and $p \oplus p' = p + p'$. Fig. 3 shows how a position $p = 2$ in the only possible global shape $s = *$ and a position $p' = 2$ in the equal subshape $s' = s \downarrow p = *$ give back a position $p + p' = 4$ in the global shape.

This directed container is nothing else than the monoid $(\mathsf{Nat}, 0, +)$ seen as a directed container.

Similarly to the theory of containers, one can also define morphisms between directed containers.

Definition 3.5. A *morphism* between two directed containers $(S \lhd P, \downarrow, \mathsf{o}, \oplus)$ and $(S' \lhd P', \downarrow', \mathsf{o}', \oplus')$ is a morphism $t \lhd q$ between the containers $S \lhd P$ and $S' \lhd P'$ that satisfies three laws:

(1) $\forall \{s, p\}.\, t\,(s \downarrow q\, p) = t\, s \downarrow'\, p$,
(2) $\forall \{s\}.\, \mathsf{o}\,\{s\} = q\,(\mathsf{o}'\,\{t\, s\})$,
(3) $\forall \{s, p, p'\}.\, q\, p \oplus \{s\}\, q\, p' = q\,(p \oplus'\, \{t\, s\}\, p')$.

In the special case $S = S' = 1$, laws 2 and 3 are the laws of a monoid morphism.

Recall the intuition that t determines the shape of the data-structure that some given data-structure is sent to and q identifies for every position in the data-structure returned a position in the given data-structure. These laws say that the positions in the sub-data-structure for any position in the resulting data-structure must map back to positions in the corresponding sub-data-structure of the given data-structure. This means that they can receive data only from those positions, other flows are forbidden. Morphisms between directed containers representing node-labelled tree datatypes are exactly upwards accumulations—this was one of the motivations for choosing the name 'directed containers'.

Example 3.6. The container representations of the head and drop-even functions for non-empty lists are directed container morphisms for the directed container of non-empty lists and suffixes (and the identity directed container). But those of self-append and reversal are not.

Example 3.7. For the directed container of non-empty lists and cyclic shifts, not only the representations of the head and drop-even functions but also the self-append function are directed container morphisms.

The identities and composition of **Cont** can give the identities and composition for directed containers, since for every directed container $E = (C, \downarrow, \mathsf{o}, \oplus)$, the identity container morphism $\mathsf{id}^{\mathrm{c}}\,\{C\}$ is a directed container morphism and the composition $h \circ^{\mathrm{c}} h'$ of two directed container morphisms is also a directed container morphism.

Proposition 3.8. *Directed containers form a category* **DCont**.

3.2. **Interpretation of Directed Containers.** As directed containers are containers with some operations obeying some laws, a directed container should denote not just a set functor, but a set functor with operations obeying some laws. The correct domain of denotation for directed containers is provided by comonads on sets.

Definition 3.9. Given a directed container $E = (S \lhd P, \downarrow, \mathsf{o}, \oplus)$, we define its *interpretation* $[\![E]\!]^{\mathrm{dc}}$ to be the set functor $D = [\![S \lhd P]\!]^{\mathrm{c}}$ (i.e., the interpretation of the underlying container) together with two natural transformations

$$\begin{aligned}
&\varepsilon : \forall \{X\}.\, (\Sigma s : S.\, P\, s \to X) \to X \\
&\varepsilon\, (s, v) = v\, (\mathsf{o}\, \{s\}) \\
&\delta : \forall \{X\}.\, (\Sigma s : S.\, P\, s \to X) \to \Sigma s : S.\, P\, s \to \Sigma s' : S.\, P\, s' \to X \\
&\delta\, (s, v) = (s, \lambda p.\, (s \downarrow p, \lambda p'.\, v\, (p \oplus \{s\}\, p')))
\end{aligned}$$

The directed container laws ensure that the natural transformations ε, δ make the counit and comultiplication of a comonad structure on D.

Intuitively, the counit extracts the data at the root position of a data-structure (e.g., the head of a non-empty list), the comultiplication, which produces a data-structure of data-structures, replaces the data at every position with the sub-data-structure corresponding to this position (e.g., the corresponding suffix or cyclic shift).

The interpretation $[\![h]\!]^{\mathrm{dc}}$ of a morphism h between directed containers $E = (C, \downarrow, \mathsf{o}, \oplus)$, $E' = (C', \downarrow', \mathsf{o}', \oplus')$ is defined by $[\![h]\!]^{\mathrm{dc}} = [\![h]\!]^{\mathrm{c}}$ (using that h is a container morphism between C and C'). The directed container morphism laws ensure that this natural transformation between $[\![C]\!]^{\mathrm{c}}$ and $[\![C']\!]^{\mathrm{c}}$ is also a comonad morphism between $[\![E]\!]^{\mathrm{dc}}$ and $[\![E']\!]^{\mathrm{dc}}$.

Since the category **Comonads**(**Set**) inherits its identities and composition from [**Set**, **Set**], the functor $[\![-]\!]^{\mathrm{dc}}$ also preserves the identities and composition.

Proposition 3.10. $[\![-]\!]^{\mathrm{dc}}$ *is a functor from* **DCont** *to* **Comonads**(**Set**).

Similarly to the case of natural transformations between container interpretations, one can also "quote" comonad morphisms between directed container interpretations into directed container morphisms. For any directed containers $E = (C, \downarrow, \mathsf{o}, \oplus)$, $E' = (C', \downarrow', \mathsf{o}', \oplus')$ and any morphism τ between the comonads $[\![E]\!]^{\mathrm{dc}}$ and $[\![E']\!]^{\mathrm{dc}}$ (which is a natural transformation between $[\![C]\!]^{\mathrm{c}}$ and $[\![C']\!]^{\mathrm{c}}$), the container morphism $\ulcorner \tau \urcorner^{\mathrm{dc}} = \ulcorner \tau \urcorner^{\mathrm{c}}$ between the underlying containers C and C' is also a directed container morphism between E and E'. The directed container morphism laws follow from the comonad morphism laws.

From what we already know about interpretation and quoting of container morphisms, it is immediate that $\ulcorner [\![h]\!]^{\mathrm{dc}} \urcorner^{\mathrm{dc}} = h$ for any directed container morphism h and that $\ulcorner \tau \urcorner^{\mathrm{dc}} = \ulcorner \tau' \urcorner^{\mathrm{dc}}$ implies $\tau = \tau'$ for any comonad morphisms τ and τ' between directed container interpretations.

Proposition 3.11. $[\![-]\!]^{\mathrm{dc}}$ *is fully faithful.*

The *identity container* $\mathsf{Id}^{\mathrm{c}} = 1 \lhd \lambda *. 1$ extends trivially to an identity directed container whose denotation is isomorphic to the identity comonad. But, similarly to the situation with functors and comonads, composition of containers fails to yield a composition monoidal structure on **DCont**.

We have elsewhere [7] shown that, similarly to the functors and comonads case [10], the composition of the underlying containers of two directed containers carries a *compatible* directed container structure if and only if there is a *distributive law* between these directed containers. Compatible compositions of directed containers turn out to generalize Zappa-Szép products of monoids [33, 11], with distributive laws playing the role of matching pairs of mutual actions.

3.3. **Containers ∩ Comonads = Directed Containers.** Since not every functor can be represented by a container, there is no point in asking whether every comonad can be represented as a directed container. An example of a natural comonad that is not a directed container is the cofree comonad on the finite powerset functor $\mathcal{P}_{\mathrm{f}}$ (node-labelled non-well-founded strongly-extensional trees) where the carrier of this comonad is not a container ($\mathcal{P}_{\mathrm{f}}$ is also not a container). But, what about those comonads whose underlying functor is an interpretation of a container? It turns out that any such comonad does indeed define a directed container that is obtained as follows.

Given a comonad (D, ε, δ) and a container $C = S \lhd P$ such that $D = [\![C]\!]^{\mathrm{c}}$, the counit ε and comultiplication δ induce container morphisms

$$\begin{aligned} &h^{\varepsilon} : C \to \mathsf{Id}^{\mathrm{c}} \\ &h^{\varepsilon} = t^{\varepsilon} \lhd q^{\varepsilon} = \ulcorner \mathsf{e} \circ \varepsilon \urcorner^{\mathrm{c}} \\ &h^{\delta} : C \to C \cdot^{\mathrm{c}} C \\ &h^{\delta} = t^{\delta} \lhd q^{\delta} = \ulcorner \mathsf{m}\,\{C, C\} \circ \delta \urcorner^{\mathrm{c}} \end{aligned}$$

using that $[\![-]\!]^{\mathrm{c}}$ is fully faithful. From (D, ε, δ) satisfying the laws of a comonad we can prove that $(C, h^{\varepsilon}, h^{\delta})$ satisfies the laws of a comonoid in **Cont** (i.e., an object in **Comonoids**(**Cont**)). Further, we can define

$$\begin{aligned} &s \downarrow p = \mathsf{snd}\,(t^{\delta}\, s)\, p \\ &\mathsf{o}\,\{s\} = q^{\varepsilon}\{s\} * \\ &p \oplus \{s\}\, p' = q^{\delta}\, \{s\}\, (p, p') \end{aligned}$$

and the comonoid laws further enforce the laws of the directed container for $(C, \downarrow, \mathsf{o}, \oplus)$.

It may seem that the maps t^{ε} and $\mathsf{fst} \circ t^{\delta}$ are not used in the directed container structure, but $t^{\varepsilon} : S \to 1$ contains no information $(\forall \{s\}.\, t^{\varepsilon}\, s = *)$ and the comonad/comonoid right counital law forces that $\forall \{s\}.\, \mathsf{fst}\,(t^{\delta}\, s) = s$, which gets used in the proofs of each of the five directed container laws. The latter fact is quite significant. It tells us that the comultiplication δ of any comonad whose underlying functor is the interpretation of a container preserves the shape of a given data-structure as the outer shape of the data-structure returned.

The situation is summarized as follows.

Proposition 3.12. *Any comonad (D, ε, δ) and container C such that $D = [\![C]\!]^{\mathrm{c}}$ determine a directed container $\lceil (D, \varepsilon, \delta), C \rceil$.*

Proposition 3.13. $\lceil [\![C, \downarrow, \mathsf{o}, \oplus]\!]^{\mathrm{dc}}, C \rceil = (C, \downarrow, \mathsf{o}, \oplus)$.

Proposition 3.14. $[\![\lceil (D, \varepsilon, \delta), C \rceil]\!]^{\mathrm{dc}} = (D, \varepsilon, \delta)$.

These observations combine into the following theorem.

Proposition 3.15. *The following is a pullback in* **CAT***:*

$$\begin{array}{ccc} \mathbf{DCont} & \xrightarrow{U} & \mathbf{Cont} \\ {\scriptstyle [\![-]\!]^{\mathrm{dc}}}\downarrow{\scriptstyle \text{f.f.}} & & {\scriptstyle [\![-]\!]^{\mathrm{c}}}\downarrow{\scriptstyle \text{f.f.}} \\ \mathbf{Comonads}(\mathbf{Set}) & \xrightarrow{U} & [\mathbf{Set},\mathbf{Set}] \end{array}$$

A structured way to prove this theorem is to first note that a pullback is provided by **Comonoids**(**Cont**) and then verify that **Comonoids**(**Cont**) is isomorphic to **DCont**.

Sam Staton pointed it out to us that the proof of the first part only hinges on **Cont** and $[\mathbf{Set},\mathbf{Set}]$ being monoidal categories and $[\![-]\!]^{\mathrm{c}} : \mathbf{Cont} \to [\mathbf{Set},\mathbf{Set}]$ being a fully faithful monoidal functor. Thus we actually establish a more general fact, viz., that for any two monoidal categories $\mathcal{C}$ and $\mathcal{D}$ and a fully-faithful monoidal functor $F : \mathcal{C} \to \mathcal{D}$, the pullback of F along the forgetful functor $U : \mathbf{Comonoids}(\mathcal{D}) \to \mathcal{D}$ is $\mathbf{Comonoids}(\mathcal{C})$.

In summary, we have seen that the interpretation of a container carries the structure of a comonad exactly when it extends to a directed container.

4. Constructions of Directed Containers

We now show some constructions of directed containers. While some standard constructions of containers extend to directed containers, others do not.

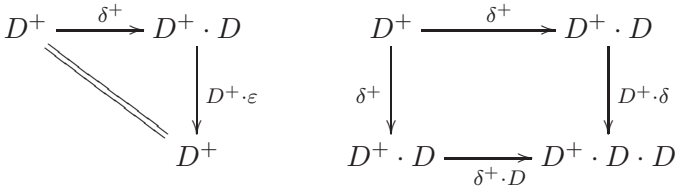
4.1. **Coproducts of Directed Containers.** Given two directed containers $E_0 = (S_0 \lhd P_0, \downarrow_0, \mathsf{o}_0, \oplus_0)$, $E_1 = (S_1 \lhd P_1, \downarrow_1, \mathsf{o}_1, \oplus_1)$, their coproduct is $E = (S \lhd P, \downarrow, \mathsf{o}, \oplus)$ where the underlying container $C = S \lhd P$ is the coproduct of containers $C_0 = S_0 \lhd P_0$ and $C_1 = S_1 \lhd P_1$. All of the directed container operations are defined either using $\downarrow_0, \mathsf{o}_0, \oplus_0$ or $\downarrow_1, \mathsf{o}_1, \oplus_1$ depending on the given shape. This means that the subshape operation is given by $\mathsf{inl}\, s \downarrow p = \mathsf{inl}\,(s \downarrow_0 p)$ and $\mathsf{inr}\, s \downarrow p = \mathsf{inr}\,(s \downarrow_1 p)$, the root position is given by $\mathsf{o}\,\{\mathsf{inl}\, s\} = \mathsf{o}_0\,\{s\}$ and $\mathsf{o}\,\{\mathsf{inr}\, s\} = \mathsf{o}_1\,\{s\}$ and the subshape position translation operation is given by $p \oplus \{\mathsf{inl}\, s\}\, p' = p \oplus_0 \{s\}\, p'$ and $p \oplus \{\mathsf{inr}\, s\}\, p' = p \oplus_1 \{s\}\, p'$. The interpretation of E is isomorphic to the coproduct of comonads $[\![E_0]\!]^{\mathrm{dc}}$ and $[\![E_1]\!]^{\mathrm{dc}}$.

Proposition 4.1. *E defined above is a coproduct of the given directed containers E_0 and E_1. It interprets to a coproduct of the comonads $[\![E_0]\!]^{\mathrm{dc}}$ and $[\![E_1]\!]^{\mathrm{dc}}$, whose underlying functor is isomorphic to $[\![C_0]\!]^{\mathrm{c}} + [\![C_1]\!]^{\mathrm{c}}$.*

4.2. **Products of (Strict) Directed Containers.** There is no general way to endow the product of the underlying containers of two directed containers $E_0 = (S_0 \lhd P_0, \downarrow_0, \mathsf{o}_0, \oplus_0)$ and $E_1 = (S_1 \lhd P_1, \downarrow_1, \mathsf{o}_1, \oplus_1)$ with the structure of a directed container. One can define $S = S_0 \times S_1$ and $P\,(s_0, s_1) = P_0\, s_0 + P_1\, s_1$, but there are two choices o_0 and o_1 for o. Moreover, there is no general way to define $p \oplus p'$. But this should not be surprising, as the product of the underlying functors of two comonads is not generally a comonad. Also, the product of two comonads would not be a comonad structure on the product of the underlying functors.

However, for monads it is known that, although the coproduct of two arbitrary monads may not always exist and is generally relatively difficult to construct explicitly [22], there is a feasible explicit formula for the coproduct of two ideal monads [17]. The duality with comonads gives a formula for the product of two coideal comonads.

Definition 4.2. A *coideal comonad* on **Set** is given by a functor $D^+ : \mathbf{Set} \to \mathbf{Set}$ and a natural transformation $\delta^+ : D^+ \to D^+ \cdot D$ such that the diagrams below commute



for a functor $D : \mathbf{Set} \to \mathbf{Set}$ and natural transformations $\varepsilon : D \to \mathsf{Id}$ and $\delta : D \to D \cdot D$ defined by

- $DX = X \times D^+X$
- $\varepsilon : \forall\{X\}.\, X \times D^+X \to X$
  $\varepsilon = \mathsf{fst}$
- $\delta : \forall\{X\}.\, X \times D^+X \to D\,X \times D^+\,(D\,X)$
  $\delta = \langle \mathsf{id}, \delta^+ \circ \mathsf{snd} \rangle$

The design of this definition ensures that the data (D, ε, δ) make a comonad as soon as the data (D^+, δ^+) satisfy the coideal comonad laws.[1]

Given two coideal comonads (D_0^+, δ_0^+) and (D_1^+, δ_1^+), the functor D given by

- $D\,X = \overline{D_0^+}\,X \times \overline{D_1^+}\,X$

where

- $(\overline{D_0^+}\,X, \overline{D_1^+}\,X) = \nu(Z_0, Z_1).\,(D_0^+\,(X \times Z_1), D_1^+\,(X \times Z_0))$

(assuming the existence of the final coalgebra) carries a coideal comonad structure that is a product, in the category of all comonads, of the given ones.

Next we define the corresponding specialization of directed containers and give an explicit product construction for this case. A strict directed container is, intuitively, a directed container where no position in a non-root subshape of a shape translates to its root, i.e., $p \oplus p'$ should not be o when $p \neq \mathsf{o}$.

Definition 4.3. A *strict directed container* is specifiable by the data

- $S : \mathsf{Set}$
- $P^+ : S \to \mathsf{Set}$
- $\downarrow^+ : \Pi s : S.\, P^+\,s \to S$
- $\oplus^+ : \Pi\{s : S\}.\, \Pi p : P^+\,s.\, P^+\,(s \downarrow^+ p) \to P^+\,s$

satisfying the laws

(1) $\forall\{s, p, p'\}.\, s \downarrow^+ (p \oplus^+ p') = (s \downarrow^+ p) \downarrow^+ p'$
(2) $\forall\{s, p, p', p''\}.\, (p \oplus^+ \{s\}\, p') \oplus^+ p'' = p \oplus^+ (p' \oplus^+ p'')$

It induces a directed container $(S \lhd P, \downarrow, \mathsf{o}, \oplus)$ via

[1] The term 'coideal comonad' is motivated by (D^+, δ^+) being a right comodule of the comonad (D, ε, δ). For the same concept, also the term 'ideal comonad' has been used.

- $P\,s = \mathsf{Maybe}\,(P^+ s)$
- $s \downarrow \mathsf{nothing} = s$
  $s \downarrow \mathsf{just}\,p = s \downarrow^+ p$
- $\mathsf{o} = \mathsf{nothing}$
- $\mathsf{nothing} \oplus p = p$
  $\mathsf{just}\,p \oplus \mathsf{nothing} = \mathsf{just}\,p$
  $\mathsf{just}\,p \oplus \mathsf{just}\,p' = \mathsf{just}\,(p \oplus^+ p')$

Similarly to coideal comonads, the design of this definition also ensures that the data $(S \lhd P, \downarrow, \mathsf{o}, \oplus)$ make a directed container as soon as the data $(S \lhd P^+, \downarrow^+, \oplus^+)$ satisfy the strict directed container laws.[2]

Strict directed containers are the pullback of the interpretation of directed containers and the inclusion of coideal comonads into comonads.

Notice that the special case $S = 1$ describes monoids without right-invertible non-unit elements (such monoids are trivially also without left-invertible non-unit elements; they arise from adding a unit to a semigroup freely). For example, the datatype of lists and suffixes is a strict directed container; on the other hand, the datatype of lists and cyclic shifts is not.

We take inspiration from the construction of the product of two coideal comonads and construct the product of two strict directed containers.

Given two strict directed containers $E_0 = (S_0 \lhd P_0^+, \downarrow_0^+, \oplus_0^+)$ and $E_1 = (S_1 \lhd P_1^+, \downarrow_1^+, \oplus_1^+)$, we define the data $E = (S \lhd P^+, \downarrow^+, \oplus^+)$ by

- $S = \overline{S_0} \times \overline{S_1}$
  where
  $(\overline{S_0}, \overline{S_1}) = \nu(Z_0, Z_1).\,(\Sigma s_0 : S_0.\,P_0^+ s_0 \to Z_1, \Sigma s_1 : S_1.\,P_1^+ s_1 \to Z_0)$

- $P^+(s_0, s_1) = \overline{P_0^+} s_0 + \overline{P_1^+} s_1$
  where
  $(\overline{P_0^+}, \overline{P_1^+}) = \mu(Z_0, Z_1).$
  $(\lambda(s_0, v_0).\,\Sigma p_0 : P_0^+\,s_0.\,\mathsf{Maybe}\,(Z_1\,(v_0\,p_0)), \lambda(s_1, v_1).\,\Sigma p_1 : P_1^+ s_1.\,\mathsf{Maybe}\,(Z_0\,(v_1\,p_1)))$

- $\downarrow^+ : \Pi s : S.\,P^+ s \to S$
  $(s_0, s_1) \downarrow^+ \mathsf{inl}\,p = s_0\,\overline{\downarrow_0^+}\,p$
  $(s_0, s_1) \downarrow^+ \mathsf{inr}\,p = s_1\,\overline{\downarrow_1^+}\,p$
  where
  $\overline{\downarrow_0^+} : \Pi s : \overline{S_0}.\,\overline{P_0^+}\,s \to S$
  $\overline{\downarrow_1^+} : \Pi s : \overline{S_1}.\,\overline{P_1^+}\,s \to S$
  (by mutuual recursion)
  $(s_0, v_0)\,\overline{\downarrow_0^+}\,(p_0, \mathsf{nothing}) = ((s_0 \downarrow_0^+ p_0, \lambda p.\,v_0\,(p_0 \oplus_0^+ p)), v_0\,p_0)$

[2]You may notice a small "mismatch" between the definitions of strict directed containers and coideal comonads. We have given $\oplus^+$ the type $\Pi\{s : S\}.\,\Pi p : P^+ s.\,P^+\,(s \downarrow^+ p) \to P^+ s$ while the δ^+ has type $D^+ \cdot D \to D^+$, not $D^+ \cdot D^+ \to D^+$. The reason is that the first option for the type of δ^+ is more general and really the "correct" one for comonads. For comonads whose underlying functors are containers, however, the corresponding type $\Pi\{s : S\}.\,\Pi p : P^+ s.\,P\,(s \downarrow^+ p) \to P^+ s$ buys no additional generality.

$(s_0, v_0)\ \overline{\downarrow_0^+}\ (p_0, \mathsf{just}\, p) = v_0\, p_0\ \overline{\downarrow_1^+}\ p$
$(s_1, v_1)\ \overline{\downarrow_1^+}\ (p_1, \mathsf{nothing}) = (v_1\, p_1, (s_1\ \downarrow_1^+\ p_1, \lambda p.\, v_1\, (p_1 \oplus_1^+ p)))$
$(s_1, v_1)\ \overline{\downarrow_1^+}\ (p_1, \mathsf{just}\, p) = v_1\, p_1\ \overline{\downarrow_0^+}\ p$

- $\oplus^+ \colon \Pi\{s : S\}.\, \Pi p : \overline{P^+}\, s.P^+(s \downarrow^+ p) \to P^+ s$
  $\mathsf{inl}\, p \oplus^+ p' = \mathsf{inl}\, (p\ \overline{\oplus_0^+}\ p')$
  $\mathsf{inr}\, p \oplus^+ p' = \mathsf{inr}\, (p\ \overline{\oplus_1^+}\ p')$
  where
  $\overline{\oplus_0^+} \colon \Pi\{s : \overline{S_0}\}.\, \Pi p : \overline{P_0^+}\, s.\, P^+(s\ \overline{\downarrow_0^+}\ p) \to \overline{P_0^+}\, s$
  $\overline{\oplus_1^+} \colon \Pi\{s : \overline{S_1}\}.\, \Pi p : \overline{P_1^+}\, s.\, P^+(s\ \overline{\downarrow_1^+}\ p) \to \overline{P_1^+}\, s$
  (by mutual recursion)
  $(p_0, \mathsf{nothing})\ \overline{\oplus_0^+}\ \mathsf{inl}\, (p_0', p_1') = (p_0 \oplus_0^+ p_0', p_1')$
  $(p_0, \mathsf{nothing})\ \overline{\oplus_0^+}\ \mathsf{inr}\, p = (p_0, \mathsf{just}\, p)$
  $(p_0, \mathsf{just}\, p_1)\ \overline{\oplus_0^+}\ p = (p_0, \mathsf{just}\, (p_1\ \overline{\oplus_1^+}\ p))$
  $(p_1, \mathsf{nothing})\ \overline{\oplus_1^+}\ \mathsf{inr}\, (p_1', p_0') = (p_1 \oplus_1^+ p_1', p_0')$
  $(p_1, \mathsf{nothing})\ \overline{\oplus_1^+}\ \mathsf{inl}\, p = (p_1, \mathsf{just}\, p)$
  $(p_1, \mathsf{just}\, p_0)\ \overline{\oplus_1^+}\ p = (p_1, \mathsf{just}\, (p_0\ \overline{\oplus_0^+}\ p))$

Proposition 4.4. *E is a product, in the category of all directed containers, of the strict directed containers E_0 and E_1. It interprets to a product, in the category of all comonads, of their interpreting coideal comonads.*

The definitions above a considerable amount of detail, but the intuition behind them is not difficult. The product of two strict directed containers generalizes the coproduct of two monoids without non-unit right-invertible elements. The elements of this monoid are finite alternating sequences of non-unit elements of the two given monoids. The definitions above arrange for alternations of a similar nature.

4.3. **Cofree Directed Containers.** Given a container $C_0 = S_0 \lhd P_0$, let us define $E = (S \lhd P, \downarrow, \mathsf{o}, \oplus)$ by

- $S = \nu Z.\, \Sigma s : S_0.P_0\, s \to Z$
- $P = \mu Z.\, \lambda(s, v).\, 1 + \Sigma p : P_0\, s.\, Z\, (v\, p)$
- (by recursion)
  $(s, v) \downarrow \mathsf{inl}\, * = (s, v)$
  $(s, v) \downarrow \mathsf{inr}\, (p, p') = v\, p \downarrow p'$
- $\mathsf{o}\, \{s, v\} = \mathsf{inl}\, *$
- (by recursion)
  $\mathsf{inl}\, * \oplus \{s, v\}\, p'' = p''$
  $\mathsf{inr}\, (p, p') \oplus \{s, v\}\, p'' = \mathsf{inr}\, (p, p' \oplus \{v\, p\}\, p'')$

Proposition 4.5. *E is a cofree directed container on C_0. It interprets into a cofree comonad on the functor $[\![C_0]\!]^{\mathrm{c}}$, which has its underlying functor isomorphic to $D\, X = \nu Z.\, X \times [\![C_0]\!]^{\mathrm{c}}\, Z$.*

Example 4.6. In the special case $S_0 = 1$, we get that $S = \nu Z.\, \Sigma * : 1.\, P_0\, * \to Z \cong 1$ and this example degenerates to the free monoid on a given set $P_0\, *$, i.e., the monoid of lists over

$P_0 *$ (with the empty list as the unit and concatenation as the multiplication operation). This directed container interprets into the comonad of nonwellfounded node-labelled $P_0 *$-branching trees.

4.4. **Cofree Recursive Directed Containers.** A recursive comonad is a coideal comonad (D^+, δ^+) such that, for any map $f : D^+ (X \times Y) \to Y$, there exists a unique map $f^\dagger : D^+ X \to Y$ such that

$$\begin{array}{ccc} D^+ X & \xrightarrow{\quad f^\dagger \quad} & Y \\ {\scriptstyle \delta^+}\downarrow & & \uparrow{\scriptstyle f} \\ D^+ (D X) & \xrightarrow[D^+ (X \times f^\dagger)]{} & D^+ (X \times Y) \end{array}$$

Recursive directed containers are the pullback of the interpretation of strict directed containers and the inclusion of recursive comonads into coideal comonads.

Now the cofree recursive directed container on a given container C is obtained by replacing the ν in the definition of the shape set S of the cofree directed container with μ. The interpretation has its underlying functor isomorphic to $D X = \mu Z. X \times [\![C]\!]^{\mathrm{c}} Z$, which is the cofree recursive comonad on $[\![C]\!]^{\mathrm{c}}$.

While cofree directed containers represent datatypes of node-labelled nonwellfounded trees, cofree recursive directed containers correspond to node-labelled wellfounded trees. The simplest interesting example is the datatype of non-empty lists (with its suffixes structure), which is represented by the cofree recursive directed container on the "maybe" container $1 + 1 \lhd \lambda\{(\mathsf{inl}\, *).\, 0\ ;\ (\mathsf{inr}\, *).\, 1\}$, i.e., two shapes, one with no positions, the other with one position.

4.5. **Data-structures with a Focus.** Below we discuss directed containers equipped a notion of focus. We present a construction for turning any container into a directed container with a designated focus. We also show that the zipper types of Huet [20] have a direct representation as directed containers.

Focussing. Any container $C_0 = S_0 \lhd P_0$ defines a directed container $E = (S \lhd P, \downarrow, \mathsf{o}, \oplus)$ as follows. We take $S = \Sigma s : S_0.\, P_0\, s$, so that a shape is a pair of a shape s, the "shape proper", and an arbitrary position p in that shape, the "focus". We take $P\,(s, p) = P_0\, s$, so that a position in the shape (s, p) is a position in the shape proper s, irrespective of the focus. The subshape determined by position p' in shape (s, p) is given by keeping the shape proper but changing the focus: $(s, p) \downarrow p' = (s, p')$. The root in the shape (s, p) is the focus p, so $\mathsf{o}\,\{s, p\} = p$. Finally, we take the translation of positions from the subshape (s, p') given by position p' to shape (s, p) to be the identity, by defining $p' \oplus \{s, p\}\, p'' = p''$. All directed container laws are satisfied.

The directed container E so obtained interprets into the canonical comonad structure on the functor $\partial [\![C_0]\!]^{\mathrm{c}} \times \mathsf{Id}$, where ∂F denotes the derivative of the functor F. (For derivatives of set functors and containers, see Abbott et al. [4].)

Differently from, e.g., the cofree directed container construction, this construction is not a functor from **Cont** to **DCont**. Instead, it is a functor from the category of containers and Cartesian container morphisms (where position maps are bijections).

Zippers. Inductive (tree-like) datatypes with a designated focus position are isomorphic to the zipper types of Huet [20]. A zipper data-structure encodes a tree with a focus as a pair of a context and a tree. The tree is the subtree of the global tree rooted by the focus and the context encodes the rest of the global tree. On zippers, changing the focus is supported via local navigation operations for moving one step down into the tree or up or aside into the context.

Zipper datatypes are directly representable as directed containers. We illustrate this on the example of zippers for lists (which are, in fact, the same as zippers for non-empty lists, as one cannot focus on a position in the empty list).

Example 4.7. A list zipper is a pair of a list (the context) and a non-empty list (the suffix determined by the focus position). Accordingly, by defining $S = \mathsf{Nat} \times \mathsf{Nat}$, the shape of a zipper is a pair (s_0, s_1) where s_0 is the shape of the context and s_1 is the shape of the suffix. For positions, it is convenient to choose $P\,(s_0, s_1) = \{-s_0, \ldots, s_1\}$ by allocating the negative numbers in the interval for positions in the context and non-negative numbers for positions in the suffix. The root position is $\mathsf{o}\,\{s_0, s_1\} = 0$, i.e., the focus. The subshape for each position is given by $(s_0, s_1) \downarrow p = (s_0 + p, s_1 - p)$ and translation of subshape positions by $p \oplus \{s_0, s_1\}\, p' = p + p'$.

Fig. 4 gives an example of a non-empty list with focus with its shape fixed to $s = (5, 6)$. It should be clear from the figure how the $\oplus$ operation works on positions $p = 4$ and $p' = -7$ to get back the position $p \oplus p' = -3$ in the initial shape. The subshape operation $\downarrow$ works as follows: $s \downarrow p$ gives back a subshape $s' = (9, 2)$ and $s \downarrow (p \oplus p')$ gives $s'' = (2, 9)$.

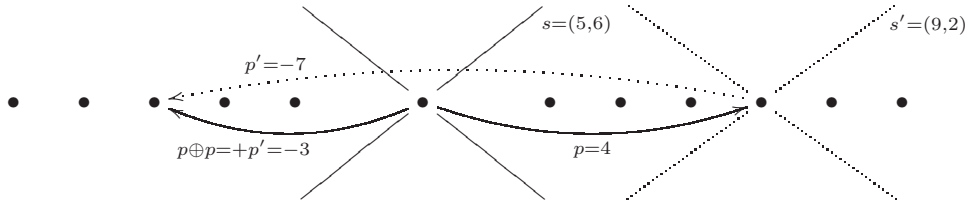


Figure 4: The shape and positions for non-empty lists of length 12 focussed at position 5

The isomorphism of the directed container representation of the list zipper datatype and the directed container of focussed lists is $t : \mathsf{Nat} \times \mathsf{Nat} \to \Sigma s : \mathsf{Nat}.\, \{0, \ldots, s - 1\}$, $t\,(s_0, s_1) = (s_0 + s_1 + 1, s_0)$, $q : \Pi\{(s_0, s_1) : \mathsf{Nat} \times \mathsf{Nat}\}.\, \{0, \ldots, s_0 + s_1\} \to \{-s_0, \ldots, s_1\}$, $q\,\{s_0, s_1\}\, p = p - s_0$.

We refrain here from delving deeper into the topic of derivatives and zippers, leaving this discussion for another occasion.

5. Containers ∩ Monads = ?

Given that comonads whose underlying functor is the interpretation of a container are the same as directed containers, it is natural to ask whether a similar characterization is possible for monads whose underlying functor can be represented as a container. The answer is "yes", but the additional structure is more involved than that of directed containers.

Given a container $C = S \lhd P$, the structure (η, μ) of a monad on the functor $T = [\![C]\!]^{\mathrm{c}}$ is interdefinable with the following structure on C

- $\mathsf{e} : S$ (for the shape map for η),
- $\bullet : \Pi s : S.(P\,s \to S) \to S$ (for the shape map for μ),
- $\nwarrow : \Pi\{s : S\}.\,\Pi v : P\,s \to S.P\,(s \bullet v) \to P\,s$ and
- $\nearrow : \Pi\{s : S\}.\,\Pi v : P\,s \to S.\Pi p : P\,(s \bullet v).\,P\,(v\,(v \nwarrow \{s\}p))$ (both for the position map for μ)

subject to three shape equations and five position equations. Perhaps not unexpectedly, this amounts to having a monoid structure on C. We refrain from a more detailed discussion of this variation of the concept of containers.

Example 5.1. To get some intuition, consider the monad structure on the datatype of lists. The unit is given by singleton lists and multiplication is flattening a list of lists by concatenation. For the list container $S = \mathsf{Nat}$, $P\,s = \mathsf{Fin}\,s$, we get that $\mathsf{e} = 1$, $s \bullet v = \sum_{p:\mathsf{Fin}\,s} v\,p$, $v \nwarrow \{s\}\,p =$ [greatest $p' : \mathsf{Fin}\,s$ such that $\sum_{p'':\mathsf{Fin}\,p'} v\,p'' \leq p$] and $v \nearrow \{s\}\,p = p - \sum_{p'':\mathsf{Fin}\,(v \nwarrow \{s\}\,p)} v\,p''$. The reason is that the shape of singleton lists is e while flattening a list of lists with outer shape s and inner shape $v\,p$ for every position p in s results in a list of shape $s \bullet v$. For a position p in the shape of the flattened list, the corresponding positions in the outer and inner shapes of the given list of lists are $v \nwarrow \{s\}\,p$ and $v \nearrow \{s\}\,p$.

6. Cointerpreting Directed Containers into Monads

What we have just described is not the only way to relate containers to monads. In a recent work [8], we defined *cointerpretation* of containers as the functor $\langle\!\langle - \rangle\!\rangle^{\mathrm{c}} : \mathbf{Cont}^{\mathrm{op}} \to [\mathbf{Set}, \mathbf{Set}]$ given by

$$\langle\!\langle S \lhd P \rangle\!\rangle^{\mathrm{c}} X = \Pi s : S.\,Ps \times X \cong (\Pi s : S.\,P\,s) \times (S \to X)$$

Differently from $[\![-]\!]^{\mathrm{c}}$, the functor $\langle\!\langle - \rangle\!\rangle^{\mathrm{c}}$ is neither full nor faithful. It also fails to be monoidal for the monoidal structure on $\mathbf{Cont}^{\mathrm{op}}$ (taken from $\mathbf{Cont}$). But it is lax monoidal.

It is straightforward that $\mathbf{DCont}^{\mathrm{op}} \cong (\mathbf{Comonoids}(\mathbf{Cont}))^{\mathrm{op}} \cong \mathbf{Monoids}(\mathbf{Cont}^{\mathrm{op}})$. Lax monoidal functors send monoids to monoids. Hence $\langle\!\langle - \rangle\!\rangle^{\mathrm{c}}$ lifts to a functor $\langle\!\langle - \rangle\!\rangle^{\mathrm{dc}} : \mathbf{DCont}^{\mathrm{op}} \to \mathbf{Monads}(\mathbf{Set})$ that equips each set functor $\langle\!\langle S \lhd P \rangle\!\rangle^{\mathrm{c}}$ with a monad structure

$$\begin{aligned}
&\eta : \forall\{X\}.\,X \to \Pi s : S.\,P\,s \times X\\
&\eta\,x\,s = (\mathsf{o}\,\{s\}, x)\\
&\mu : \forall\{X\}.\,(\Pi s : S.\ P\,s \times \Pi s' : S.\ P\,s' \times X) \to \Pi s : S.\,P\,s \times X\\
&\mu\,f\,s = \mathsf{let}\ \{(p, g) = f\,s;\ (p', x) = g\,(s \downarrow p)\}\ \mathsf{in}\ (p \oplus p', x)
\end{aligned}$$

Due to the resemblance to compatible compositions of reader and writer monads, we call monads in the image of this functor "dependently typed update monads". It is instructive to think of shapes in S as states, positions in Ps as updates applicable to a state s (or programs safe to evaluate from state s), $s \downarrow p$ as the result of applying an update p to the state s (or the result of evaluating p from s), $\mathsf{o}\,\{s\}$ as the nil update in state s and $p \oplus p'$ as accumulation of two consecutive updates (skip and sequential composition).

Example 6.1. The directed container for the nonempty list comonad, $S = \mathsf{Nat}$, $P\,s = [0..s]$, $s \downarrow p = s - p$, $\mathsf{o} = 0$, $p \oplus p' = p + p'$, gives us a monad on the set functor T given by $T\,X = \Pi s : \mathsf{Nat}.\,[0..s] \times X$. The states are natural numbers; the updates applicable to a state s are numbers not greater than s; applying an update means decrementing the state.

We can see that directed containers are not more "comonadic" inherently than they are "monadic". We see them first of all as an algebraic-like structure in their own right, a generalization of monoids.

7. Directed Containers in Categories with Pullbacks

Container theory can be carried out in locally Cartesian closed categories (LCCCs)—the LCCC generalization of containers being well known under the name of polynomials [15, 23]—and even more generally in categories with pullbacks [32]. It is natural to expect the same of directed container theory.

This is the case indeed. The proofs in this paper can be seen as having been carried out in the internal language of an LCCC (with the assumptions of existence of initial algebras and final coalgebras corresponding to assumptions about availability of W- and M-types).

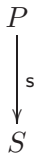
In the weaker setting of a category with pullbacks, one has to be a lot more careful. It is possible to define the concepts required from the first principles.

We show the definitions of the counterparts of directed containers and directed container morphisms; we call them "directed polynomials" and "directed polynomial morphisms" in the local scope of this section.

In all diagrams below, bullet-labelled nodes with a pair of unlabelled outgoing arcs denote pullbacks defined by a pair of maps that are given directly or constructed. Dashed arrows denote unique maps into a pullback. The polygon actually required to commute is marked with a small circular arrow.

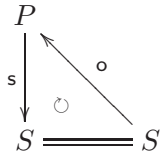
Given a category with pullbacks $\mathcal{C}$, a directed polynomial is given by

- two objects S and P ("sets" of shapes and positions) and an exponentiable map s (assigning every position a shape)



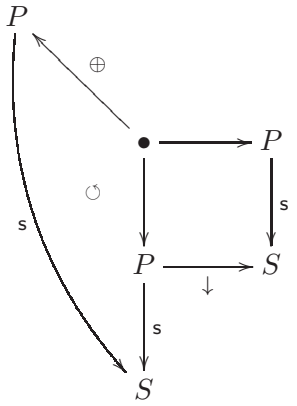
- a morphism $\downarrow$ picking out a shape for each position (the corresponding subshape)

- a map o picking out, for every shape, a position in that shape (the root position)



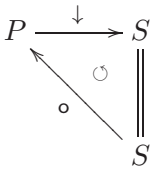
- a map $\oplus$ sending a position in a given, global shape and a position in the corresponding to subshape to a position in the global shape (translation of the subshape position to the

global shape)

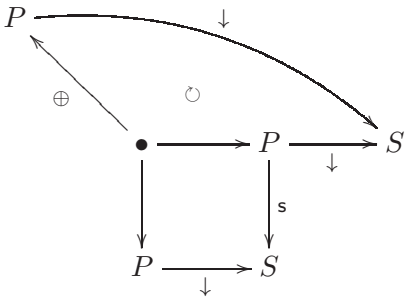


satisfying the following five laws:

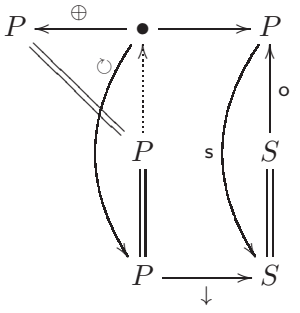
(1)



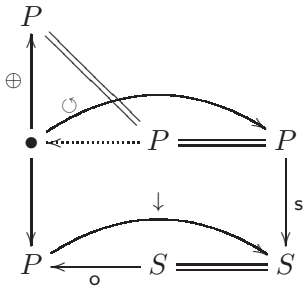
(2)



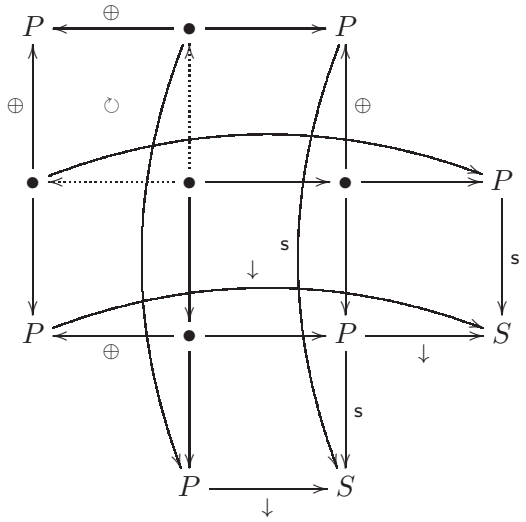
(3)



(4)

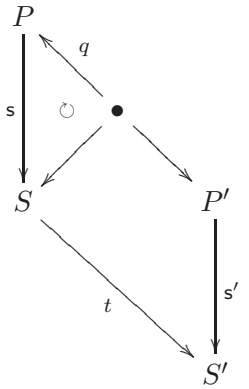


(5)



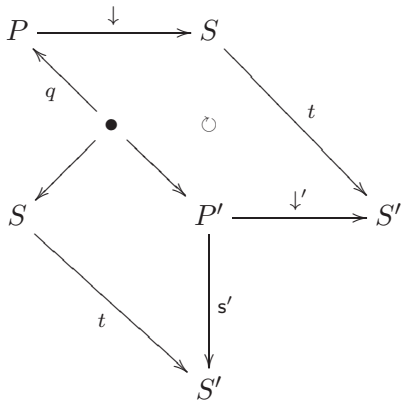
The data S, $\downarrow$, o, $\oplus$ here correspond to the homonymous data of a directed container while P and s together correspond to P. The five laws governing them correspond exactly to the five laws of a directed container.

A *morphism* between two directed polynomials $(S, P, \mathsf{s}, \downarrow, \mathsf{o}, \oplus)$ and $(S', P', \mathsf{s}', \downarrow', \mathsf{o}', \oplus')$ is given by two maps t and q (of shapes and positions)

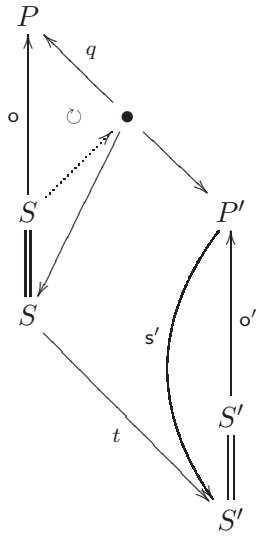


satisfying the following three laws:

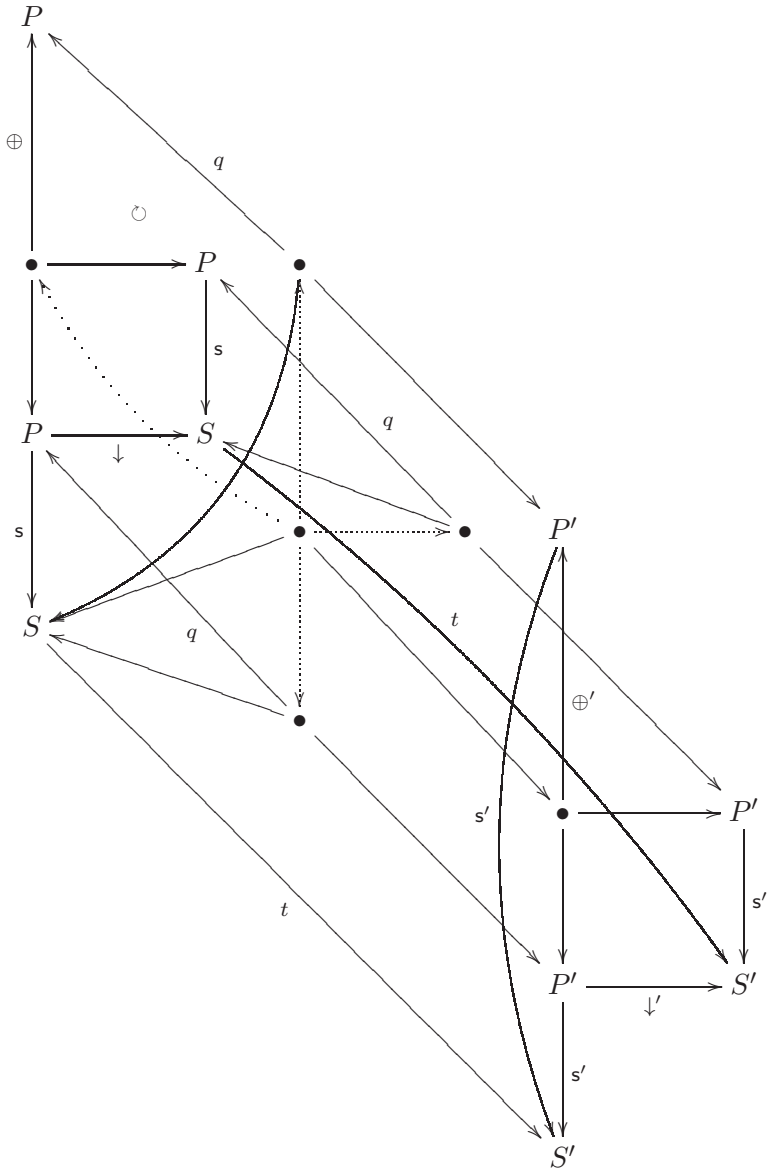
(1)



(2)



(3)



The data t, q correspond to the homonymous data of a directed container morphism and the three laws to the three laws of a directed container morphism.

In the special case $\mathcal{C} = \mathbf{Set}$, the definitions of a directed polynomial and directed polynomial morphism are equivalent to those of a directed container and directed container morphism.

Remarkably, the definition of a directed polynomial is completely symmetric in s and $\downarrow$—swapping them over we also get a directed polynomial. The definition of a directed polynomial morphism is symmetric, if q is an isomorphism.

The definitions of the interpretation of a directed polynomial resp. directed polynomial morphism into a comonad resp. comonad morphism require using distributivity pullbacks in $\mathcal{C}$ (or pullbacks in its slice categories).

8. Related Work

The core of this paper builds on the theory of containers as developed by Abbott, Altenkirch and Ghani [2, 1] to analyze strictly positive datatypes. Some generalizations of the concept of containers are the indexed containers of Altenkirch and Morris [9, 25] and the quotient containers of Abbott et al. [3]. In our work we look at a specialization of containers rather than a generalization. Recently [7], we have also studied compatible compositions of directed containers and how they generalize Zappa-Szép products [33] of two monoids.

Simple/indexed containers are intimately related to strongly positive datatypes/families and simple/dependent polynomial functors as appearing in the works of Dybjer [14], Moerdijk and Palmgren [24], Gambino and Hyland [15], Kock [23]. Girard's normal functors [18] and Joyal's analytic functors [21] functors are similar to containers resp. quotient containers, but only allow for finitely many positions in a shape. Gambino and Kock [16] also treat polynomial monads.

Abbott, Altenkirch, Ghani and McBride [4] have investigated derivatives of datatypes. Derivatives provide a systematic way to explain Huet's zipper type [20].

Brookes and Geva [12] and later Uustalu with coauthors [29, 30, 19, 13] have used comonads to analyze notions of context-dependent computation such as dataflow computation, attribute grammars, tree transduction and cellular automata. Uustalu and Vene's [31] observation of a connection between bottom-up tree relabellings and containers with extra structure started our investigation into directed containers.

9. Conclusions and Future Work

We introduced directed containers as a specialization of containers for describing a certain class of datatypes (data-structures where every position determines a sub-data-structure) that occur very naturally in programming. It was a pleasant discovery for us that directed containers are an entirely natural concept also from the mathematical point of view: they are the same as containers whose interpretation carries the structure of a comonad. They also generalize monoids in an interesting way. In a recent piece of work [6], we have witnessed that coalgebras of comonads interpreting directed containers are relevant for bidirectional transformations as a flavor of lenses ("dependently typed update lenses").

As future work, we intend to take a closer look at focussing and related concepts, such as derivatives. A curious special case of directed containers supports translation of the root of a shape into every subshape. Such bidirectional containers include, e.g., focussed containers and generalize groups in the same way as directed containers generalize monoids. We would like to find out if this specialization of directed containers is an interesting and useful concept. We wonder whether our explicit formula for the product of two directed containers can be scaled to the general, non-strict, case. Last, we would like to analyze containers that are monads more closely.

Acknowledgments. We are indebted to Thorsten Altenkirch, Jeremy Gibbons, Peter Morris, and Sam Staton for comments and suggestions. We thank our anonymous referees for the useful feedback that helped us improve the article.

References

[1] M. Abbott. *Categories of Containers*. PhD thesis, University of Leicester, 2003.

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

[3] M. Abbott, T. Altenkirch, N. Ghani, C. McBride. Constructing polymorphic programs with quotient types. In D. Kozen, ed., *Proc. of 7th Int. Conf. on Mathematics of Program Construction, MPC 2004*, vol. 3125 of *Lect. Notes in Comput. Sci.*, pp. 2–15. Springer, 2004.

[4] M. Abbott, T. Altenkirch, N. Ghani, C. McBride. ∂ is for data: differentiating data structures. *Fund. Inform.*, 65(1–2):1–28, 2005.

[5] D. Ahman, J. Chapman, T. Uustalu. When is a container a comonad? In L. Birkedal, ed., *Proc. of 15th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS 2012*, vol. 7213 of *Lect. Notes in Comput. Sci.*, pp. 74–88. Springer, 2012.

[6] D. Ahman, T. Uustalu. Coalgebraic update lenses. In B. Jacobs, A. Silva, S. Staton, eds., *Proc. of 30th Conf. on Mathematical Foundations of Programming Semantics, MFPS XXX*, *Electron. Notes in Theor. Comput. Sci.*, Elsevier, to appear.

[7] D. Ahman, T. Uustalu. Distributive laws of directed containers. *Progress in Informatics*, 10:3–18, 2013.

[8] D. Ahman, T. Uustalu. Update monads: cointerpreting directed containers. In R. Matthes, A. Schubert, eds., *Proc. of 19th Conf. on Types for Proofs and Programs, TYPES 2013*, vol. 26 of *Leibniz Int. Proc. in Inform.*, pp. 1–23. Dagstuhl Publishing, 2014.

[9] T. Altenkirch, P. Morris. Indexed containers. In *Proc. of 24th Ann. IEEE Symp. on Logic in Computer Science, LICS 2009*, pp. 277–285. IEEE CS Press, 2009.

[10] J. Beck. Distributive laws. In B. Eckmann, ed., *Seminar on Triples and Categorical Homology, ETH 1966/67*, vol. 80 of *Lect. Notes in Math.*, pp. 119–140. Springer, 1969.

[11] M. G. Brin. On the Zappa-Szép product. *Commun. in Algebra*, 33(2):393–424, 2005.

[12] S. Brookes, S. Geva. Computational comonads and intensional semantics. In M. P. Fourman, P. T. Johnstone, A. M. Pitts, eds., *Applications of Categories in Computer Science*, vol. 77 of *London Math. Society Lect. Note Series*, pp. 1–44. Cambridge Univ. Press, 1992.

[13] S. Capobianco, T. Uustalu. A categorical outlook on cellular automata. In J. Kari, ed., *Proc. of 2nd Symp. on Cellular Automata, JAC 2010*, vol. 13 of *TUCS Lecture Note Series*, pp. 88–89. Turku Centre for Comput. Sci., 2011.

[14] P. Dybjer. Representing inductively defined sets by wellorderings in Martin-Löf's type theory. *Theor. Comput. Sci.*, 176(1–2):329–335, 1997.

[15] N. Gambino, M. Hyland. Wellfounded trees and dependent polynomial functors. In S. Berardi, M. Coppo, F. Damiani, eds., *Revised Selected Papers from Int. Wksh. on Types for Proofs and Programs, TYPES 2003*, vol. 2075 of *Lect. Notes in Comput. Sci.*, pp. 210–225. Springer, 2004.

[16] N. Gambino, J. Kock. Polynomial functors and polynomial monads. *Math. Proc. of Cambridge Phil. Soc.*, 154(1):153–192, 2013.

[17] N. Ghani, T. Uustalu. Coproducts of ideal monads. *Theor. Inform. and Appl.*, 38(4): 321–342, 2004.

[18] J.-Y. Girard. Normal functors, power series and lambda-calculus. *Ann. of Pure and Appl. Logic*, 37(2):129–177, 1988.

[19] I. Hasuo, B. Jacobs, T. Uustalu. Categorical views on computations on trees. In L. Arge, C. Cachin, T. Jurdzinski, A. Tarlecki, eds., *Proc. of 34th Int. Coll. on Automata, Languages and Programming, ICALP 2007*, vol. 4596 of *Lect. Notes in Comput. Sci.*, pp. 619–630. Springer, 2007.

[20] G. Huet. The zipper. *J. of Funct. Program.*, 7(5):549–554, 1997.

[21] A. Joyal. Foncteurs analytiques et espèces de structures. In G. Labelle, P. Leroux, eds., *Combinatoire énumerative*, vol. 1234 of *Lect. Notes in Math.*, pp. 126–159. Springer, 1987.

[22] G. M. Kelly. A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves and so on. *Bull. of Austral. Math. Soc.*, 22(1):1–83, 1980.

[23] J. Kock. Polynomial functors and trees. *Int. Math. Research Notices*, 2011(3):609–673, 2011.

[24] I. Moerdijk, E. Palmgren. Wellfounded trees in categories. *Ann. of Pure and Appl. Logic* 104(1–3):189–218, 2000.

[25] P. Morris. Constructing Universes for Generic Programming. PhD thesis, University of Nottingham, 2007.

[26] U. Norell. Towards a Practical Programming Language Based on Dependent Type Theory. PhD thesis, Chalmers University of Technology, 2007.

[27] U. Norell. Dependently typed programming in Agda. In P. Koopman, R. Plasmeijer, and D. Swierstra, eds., *Revised Lectures from 6th Int. School on Advanced Functional Programming, AFP 2008*, vol. 5832 of *Lect. Notes in Comput. Sci.*, pp. 230–266. Springer, 2009.
[28] R. Prince, N. Ghani, C. McBride. Proving properties about lists using containers. In J. Garrigue, M. Hermenegildo, eds., *Proc. of 9th Int. Symp. on Functional and Logic Programming, FLOPS 2008*, vol. 4989 of *Lect. Notes in Comput. Sci.*, pp. 97–112. Springer, 2008.
[29] T. Uustalu, V. Vene. The essence of dataflow programming. In K. Yi, ed., *Proc. of 2nd Asian Symp. on Programming Languages and Systems, APLAS 2004*, vol. 3780 of *Lect. Notes in Comput. Sci.*, pp. 2–18. Springer, 2004.
[30] T. Uustalu, V. Vene. Comonadic functional attribute evaluation. In M. van Eekelen, ed., *Trends in Functional Programming 6*, pp. 145–162. Intellect, 2007.
[31] T. Uustalu, V. Vene. Comonadic notions of computation. In J. Adámek, C. Kupke, eds., *Proc. of 9th Int. Wksh. on Coalgebraic Methods in Computer Science, CMCS 2008*, vol. 203(5) of *Electron. Notes in Theor. Comput. Sci.*, pp. 263–284. Elsevier, 2008.
[32] M. Weber. Polynomials in categories with pullbacks. arXiv preprint, arXiv:1106.1983. 2011.
[33] G. Zappa. Sulla costruzione dei gruppi prodotto di due dati sottogruppi permutabili tra loro. In *Atti Secondo Congresso dell'Unione Matemática Italiana*, pp. 119–125. Edizioni Cremonense, Rome, 1942.

Appendix A. Proofs for Section 3

Proof of Proposition 3.10.

Proof. We must check that the interpretation $[\![E]\!]^{\mathrm{dc}} = (D, \varepsilon, \delta)$ of the given directed container $E = (C, \downarrow, \mathsf{o}, \oplus)$ is a comonad.
Proof of the right counital law:

$$\begin{array}{ll} & D\,\varepsilon\,(\delta\,(s,v)) \\ = & \quad \{\text{definition of } D\} \\ & (\lambda(s,v).\,(s, \lambda p.\,\varepsilon\,(v\,p)))\,(\delta\,(s,v)) \\ = & \quad \{\text{definitions of } \varepsilon,\ \delta\} \\ & (s, \lambda p.\,v\,(p \oplus \mathsf{o})) \\ = & \quad \{\text{directed container law 3}\} \\ & (s,v) \end{array}$$

Proof of the left counital law:

$$\begin{array}{ll} & \varepsilon\,(\delta\,(s,v)) \\ = & \quad \{\text{definitions of } \varepsilon,\ \delta\} \\ & (s \downarrow \mathsf{o}, \lambda p'.\,v\,(\mathsf{o} \oplus p')) \\ = & \quad \{\text{directed containers laws 1 and 4}\} \\ & (s,v) \end{array}$$

Proof of the coassociativity law:

$$\begin{array}{ll} & D\,\delta\,(\delta\,(s,v)) \\ = & \quad \{\text{definition of } D\} \\ & (\lambda(s,v).\,(s, \lambda p.\,\delta\,(v\,p)))\,(\delta\,(s,v)) \\ = & \quad \{\text{definition of } \delta\} \\ & (s, \lambda p.\,(s \downarrow p, \lambda p'.\,((s \downarrow p) \downarrow p', \lambda p''.\,v\,(p \oplus (p' \oplus p'')))))) \\ = & \quad \{\text{directed container laws 2 and 5}\} \\ & (s, \lambda p.\,(s \downarrow p, \lambda p'.\,(s \downarrow (p \oplus p'), \lambda p''.\,v\,((p \oplus p') \oplus p'')))) \\ = & \quad \{\text{definition of } \delta\} \\ & \delta\,(\delta\,(s,v)) \end{array}$$

We must also verify that the interpretation $[\![h]\!]^{\mathrm{dc}} = \tau$ of a morphism $h = t \lhd q$ between two directed containers $E = (C, \downarrow, \mathsf{o}, \oplus)$ and $E' = (C', \downarrow', \mathsf{o}', \oplus')$ is a comonad morphism between $[\![E]\!]^{\mathrm{dc}} = (D, \varepsilon, \delta)$ and $[\![E']\!]^{\mathrm{dc}} = (D', \varepsilon', \delta')$.
Proof of the counit preservation law:

$$\begin{array}{ll} & \varepsilon\,(s, v) \\ = & \quad \{\text{definition of } \varepsilon\} \\ & v\,\mathsf{o} \\ = & \quad \{\text{directed container morphism law 2}\} \\ & v\,(q\,\mathsf{o}') \\ = & \quad \{\text{definitions of } \tau,\ \varepsilon'\} \\ & \varepsilon'\,(\tau\,(s, v)) \end{array}$$

Proof of the comultiplication preservation law:

$$\begin{array}{ll} & D\,\tau\,(\tau\,(\delta\,(s, v))) \\ = & \quad \{\text{definition of } D\} \\ & (\lambda(s, v).\,(s, \lambda p.\,\tau\,(v\,p)))\,(\tau\,(\delta\,(s, v))) \\ = & \quad \{\text{definitions of } \tau,\ \delta\} \\ & (t\,s, \lambda p.\,(t\,(s \downarrow q\,p), \lambda p'.\,v\,(q\,p \oplus q\,p'))) \\ = & \quad \{\text{directed container morphism laws 1 and 3}\} \\ & (t\,s, \lambda p.\,(t\,s \downarrow' p, \lambda p'.\,v\,(q\,(p \oplus' p')))) \\ = & \quad \{\text{definitions of } \tau,\ \delta'\} \\ & \delta'\,(\tau\,(s, v)) \end{array}$$

□

Proof of Proposition 3.11.

Proof. From Proposition 2.8, we know that the interpretation of containers is fully faithful. It remains to show that, for directed containers $E = (C, \downarrow, \mathsf{o}, \oplus)$, $E' = (C', \downarrow', \mathsf{o}', \oplus')$ and a morphism τ between the comonads $[\![E]\!]^{\mathrm{dc}}$ and $[\![E']\!]^{\mathrm{dc}}$, the container morphism $h = t \lhd q = {\ulcorner}\tau{\urcorner}^{\mathrm{c}}$ between C and C' is also a directed container morphism between E and E'.

The counit and comultiplication ε and δ of the comonad $[\![E]\!]^{\mathrm{dc}}$ induce container morphisms $h^{\varepsilon} : C \to \mathsf{Id}^{\mathrm{c}}$ and $h^{\delta} : C \to C \cdot^{\mathrm{c}} C$ by $h^{\varepsilon} = t^{\varepsilon} \lhd q^{\varepsilon} = {\ulcorner}\mathsf{e} \circ \varepsilon{\urcorner}^{\mathrm{c}}$, $h^{\delta} = t^{\delta} \lhd q^{\delta} = {\ulcorner}\mathsf{m} \circ \delta{\urcorner}^{\mathrm{c}}$. Similarly ε' and δ' give us container morphisms $h^{\varepsilon'} : C' \to \mathsf{Id}^{\mathrm{c}}$ and $h^{\delta'} : C' \to C' \cdot^{\mathrm{c}} C'$ by $h^{\varepsilon'} = t^{\varepsilon'} \lhd q^{\varepsilon'} = {\ulcorner}\mathsf{e} \circ \varepsilon'{\urcorner}^{\mathrm{c}}$, $h^{\delta'} = t^{\delta'} \lhd q^{\delta'} = {\ulcorner}\mathsf{m} \circ \delta'{\urcorner}^{\mathrm{c}}$.

Let us express h^{ε} and h^{δ} directly in terms of $\downarrow$, o, $\oplus$.

First, from the definitions of ε, e we get

$$h^{\varepsilon} = {\ulcorner}\mathsf{e} \circ \varepsilon{\urcorner}^{\mathrm{c}} = {\ulcorner}\lambda(s, v).\,(*, \lambda *.\,v\,(\mathsf{o}\,\{s\}))\,{\urcorner}^{\mathrm{c}}$$

The definition of ${\ulcorner}-{\urcorner}^{\mathrm{c}}$ further gives us

$$\begin{array}{rcl} t^{\varepsilon}\,s & = & * \\ q^{\varepsilon}\,\{s\}\,* & = & \mathsf{o}\,\{s\} \end{array}$$

Second, the definitions of δ, m dictate that

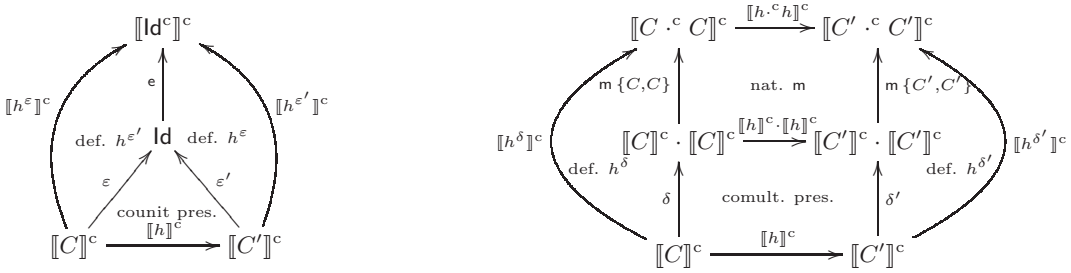
$$h^{\delta} = {\ulcorner}\mathsf{m}\,\{C, C\} \circ \delta{\urcorner}^{\mathrm{c}} = {\ulcorner}\lambda(s, v).\,(s, \lambda p.\,s \downarrow p), \lambda(p, p').\,v\,(p \oplus \{s\}\,p')\,{\urcorner}^{\mathrm{c}}$$

The definition of ${\ulcorner}-{\urcorner}^{\mathrm{c}}$ allows us to infer that

$$\begin{array}{rcl} t^{\delta}\,s & = & (s, \lambda p.\,s \downarrow p) \\ q^{\delta}\,\{s\}\,(p, p') & = & p \oplus \{s\}\,p' \end{array}$$

Analogous direct expressions in terms of $\downarrow'$, o', $\oplus'$ hold for $h^{\varepsilon'}$, $h^{\delta'}$.

Now, using $h^{\varepsilon}, h^{\delta}, h^{\varepsilon'}, h^{\delta'}$ above, we can repackage the two comonad morphism laws for $\tau = [\![h]\!]^{\mathsf{c}}$ in terms of container interpretations as depicted in the following two diagrams.



Going a step further, we can quote these two diagrams to get their reformulations in terms of containers, resulting in the two diagrams below.

We are now in a position to prove that $h = \ulcorner \tau \urcorner^{\mathsf{c}}$ satisfies directed container morphism laws.

From the counit preservation law by going clockwise we get that

$$s \longmapsto t^{\varepsilon}\, s$$

$$C \xrightarrow{\;h^{\varepsilon}\;} \mathsf{Id}^{\mathsf{c}}$$

$$q^{\varepsilon}\,\{s\}\,* \longleftarrow\!\shortmid\; *$$

and by going counter-clockwise

$$s \longmapsto ts \longmapsto t^{\varepsilon'}\,(t\, s)$$

$$C \xrightarrow{\;h\;} C' \xrightarrow{\;h^{\varepsilon'}\;} \mathsf{Id}^{\mathsf{c}}$$

$$q\,\{s\}\,(q^{\varepsilon'}\,\{t\, s\}\,*) \longleftarrow\!\shortmid\; q^{\varepsilon'}\,\{t\, s\}\,* \longleftarrow\!\shortmid\; *$$

which gives us the second directed container morphism law:

$$\mathsf{o}\,\{s\} = q^{\varepsilon}\,\{s\}\,* = q\,\{s\}\,(q^{\varepsilon'}\,\{t\, s\}\,*) = q\,\{s\}\,(\mathsf{o}'\,\{t\, s\})$$

Clockwise traversal of the comultiplication preservation law gives us that

$$s \longmapsto t^{\delta}\, s \longmapsto \begin{array}{l}(t\,(\mathsf{fst}\,(t^{\delta}\, s)),\\ \lambda p.\, t\,(\mathsf{snd}\,(t^{\delta}\, s)\\ (q\,\{\mathsf{fst}\,(t^{\delta}\, s)\}\, p)))\end{array}$$

$$C \xrightarrow{\;h^{\delta}\;} C \cdot^{\mathsf{c}} C \xrightarrow{\;h \cdot^{\mathsf{c}} h\;} C' \cdot^{\mathsf{c}} C'$$

$$\begin{array}{c}q^{\delta}\{s\}\\ (q\,\{\mathsf{fst}\,(t^{\delta}\, s)\}\, p,\\ q\,\{\mathsf{snd}\,(t^{\delta}\, s)\,(q\,\{\mathsf{fst}\,(t^{\delta}\, s)\}\, p)\}\, p')\end{array} \longleftarrow\!\shortmid\; \begin{array}{c}(q\,\{\mathsf{fst}\,(t^{\delta}\, s)\}\, p,\\ q\{\mathsf{snd}\,(t^{\delta}\, s)\,(q\,\{\mathsf{fst}\,(t^{\delta}\, s)\}\, p)\}\, p')\end{array} \longleftarrow\!\shortmid\; (p, p')$$

and counter-clockwise traversal that

$$\begin{array}{ccccc} s & \longmapsto & t\,s & \longmapsto & t^{\delta'}\,(t\,s) \\ C & \xrightarrow{\quad h \quad} & C' & \xrightarrow{\quad h^{\delta'} \quad} & C' \cdot^{\mathrm{c}} C' \end{array}$$

$$q\,\{s\}\,(q^{\delta'}\,\{t\,s\}\,(p,p')) \longleftarrow\!\shortmid\; q^{\delta'}\,\{t\,s\}\,(p,p') \longleftarrow\!\shortmid\; (p,p')$$

from where we can derive both the first and the third directed container morphism laws:

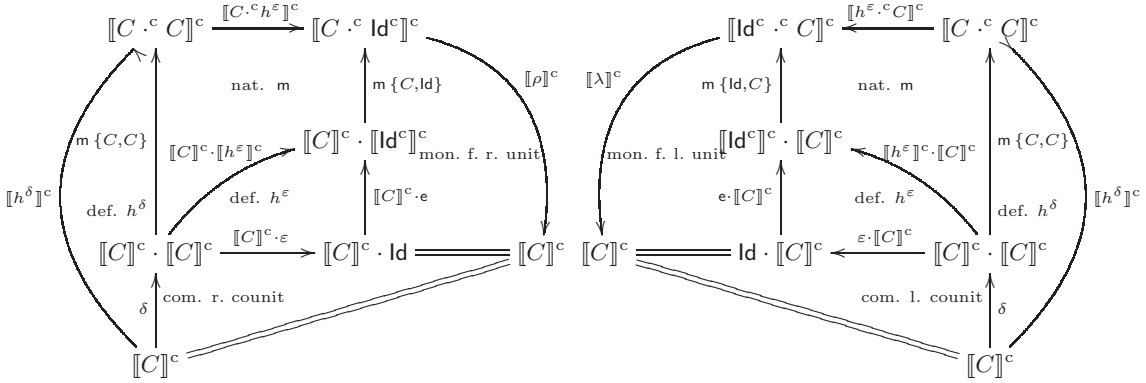
$$\begin{aligned} t\,(s \downarrow q\,\{s\}\,p) &= t\,(\mathsf{snd}\,(t^{\delta}\,s)\,(q\,\{s\}\,p)) = t\,(\mathsf{snd}\,(t^{\delta}\,s)\,(q\,\{\mathsf{fst}\,(t^{\delta}\,s)\}\,p)) \\ &= \mathsf{snd}\,(t^{\delta'}\,(t\,s))\,p = t\,s \downarrow' p \\ q\,\{s\}\,p \oplus \{s\}\,q\,\{s \downarrow q\,\{s\}\,p\}\,p' &= q^{\delta}\{s\}\,(q\,\{s\}\,p, q\,\{\mathsf{snd}\,(t^{\delta}\,s)\,(q\,\{s\}\,p)\}\,p') \\ &= q^{\delta}\{s\}\,(q\,\{\mathsf{fst}\,(t^{\delta}\,s)\}\,p, q\,\{\mathsf{snd}\,(t^{\delta}\,s)\,(q\,\{\mathsf{fst}\,(t^{\delta}\,s)\}\,p)\}\,p') \\ &= q\,\{s\}\,(q^{\delta'}\,\{t\,s\}\,(p,p')) = q\,\{s\}\,(p \oplus'\,\{t\,s\}\,p') \end{aligned}$$

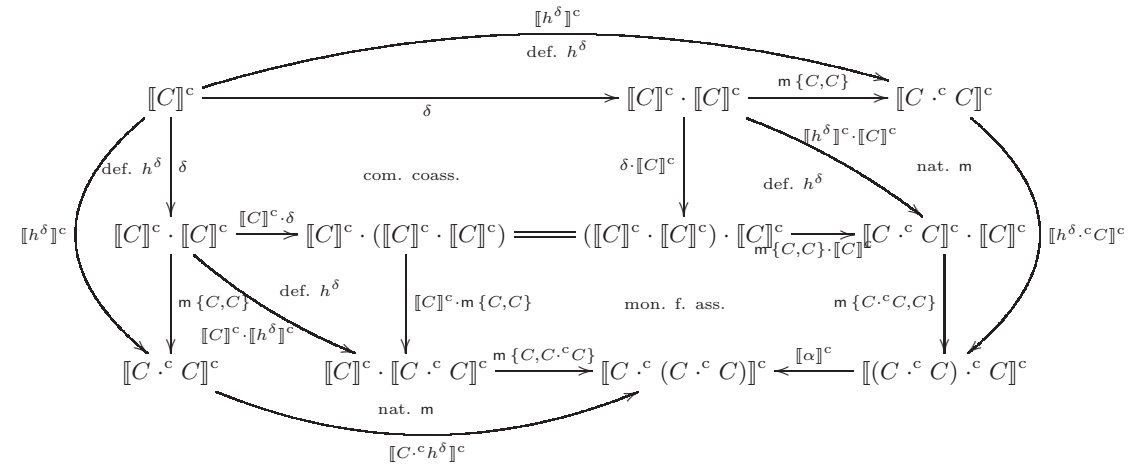
□

Proof of Proposition 3.12.

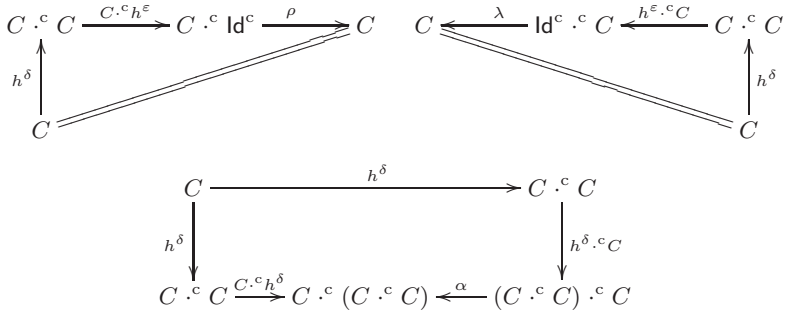
Proof. We need to verify that $(C, \downarrow, \mathsf{o}, \oplus)$ satisfies the directed container laws and can assume that (D, ε, δ) satisfies the comonad laws.

The comonad laws can be rewritten in terms of container interpretations as outlined in the following commuting diagrams:





Next we quote these three diagrams to get the comonad laws in terms of containers in the next three commuting diagrams.



From the comonad right counital law we get by going clockwise

$$\begin{array}{ccccccc} s & \longmapsto & t^\delta s & \longmapsto & (\mathsf{fst}\,(t^\delta\,s), & \longmapsto & \mathsf{fst}\,(t^\delta\,s) \\ & & & & \lambda\_.\,*) & & \end{array}$$

$$C \xrightarrow{h^\delta} C \cdot^{\mathsf{c}} C \xrightarrow{C\cdot^{\mathsf{c}} h^\varepsilon} C \cdot^{\mathsf{c}} \mathsf{Id}^{\mathsf{c}} \xrightarrow{\rho} C$$

$$\begin{array}{ccccccc} q^\delta\,\{s\}\,(p, & \longmapsto & (p, & \longmapsto & (p,*) & \longmapsto & p \\ q^\varepsilon\,\{\mathsf{snd}\,(t^\delta\,s)\,p\}\,*) & & q^\varepsilon\{\mathsf{snd}\,(t^\delta\,s)\,p\}\,*) & & & & \end{array}$$

from where it follows that δ preserves the shape of the given data-structure as the outer shape of the composite data-structure returned and that the third directed container law holds:

$$s = \mathsf{fst}\,(t^\delta\,s)$$

$$p = q^\delta\{s\}(p, q^\varepsilon\{\mathsf{snd}\,(t^\delta s)\,p\}\,*) = p \oplus \{s\}\,\mathsf{o}\,\{s \downarrow p\}$$

Similarly, from the comonad left counital law we get by going counter-clockwise

$$\begin{array}{ccccccc} s & \longmapsto & t^\delta s & \longmapsto & (*, \lambda *.\,\mathsf{snd}\,(t^\delta\,s) & \longmapsto & \mathsf{snd}\,(t^\delta\,s) \\ & & & & (q^\varepsilon\,\{\mathsf{fst}\,(t^\delta\,s)\})) & & (q^\varepsilon\,\{\mathsf{fst}\,(t^\delta\,s)\}\,*) \end{array}$$

$$C \xrightarrow{h^\delta} C \cdot^{\mathsf{c}} C \xrightarrow{h^\varepsilon\cdot^{\mathsf{c}} C} \mathsf{Id}^{\mathsf{c}} \cdot^{\mathsf{c}} C \xrightarrow{\lambda} C$$

$$\begin{array}{ccccccc} q^\delta\,\{s\} & \longmapsto & (q^\varepsilon\,\{\mathsf{fst}\,(t^\delta\,s)\}\,*, & \longmapsto & (*,p) & \longmapsto & p \\ (q^\varepsilon\,\{\mathsf{fst}\,(t^\delta\,s)\}\,*, p) & & p) & & & & \end{array}$$

from where the first and fourth directed container laws follow:

$$s = \mathsf{snd}\,(t^\delta\,s)\,(q^\varepsilon\,\{\mathsf{fst}\,(t^\delta\,s)\}\,*) = \mathsf{snd}\,(t^\delta\,s)\,(q^\varepsilon\,\{s\}) = s \downarrow \mathsf{o}\,\{s\}$$

$$p = q^\delta\{s\}\,(q^\varepsilon\,\{\mathsf{fst}\,(t^\delta\,s)\}\,*, p) = q^\delta\{s\}\,(q^\varepsilon\,\{s\}\,*, p) = \mathsf{o}\,\{s\} \oplus \{s\}\,p$$

The last two directed container laws are derivable from the comonad coassociativity law. By going clockwise we get

$$\begin{array}{ccccccc} s & \longmapsto & t^\delta s & \longmapsto & (t^\delta\,(\mathsf{fst}\,(t^\delta\,s)), & \longmapsto & (\mathsf{fst}\,(t^\delta\,(\mathsf{fst}\,(t^\delta\,s))), \\ & & & & \mathsf{snd}\,(t^\delta\,s)\circ & & (\lambda p.\,\mathsf{snd}\,(t^\delta\,(\mathsf{fst}\,(t^\delta s)))\,p, \\ & & & & (q^\delta\,\{\mathsf{fst}\,(t^\delta\,s)\})) & & \lambda p'.\,\mathsf{snd}\,(t^\delta\,s) \\ & & & & & & (q^\delta\,\{\mathsf{fst}\,(t^\delta\,s)\}\,(p,p')))) \end{array}$$

$$C \xrightarrow{h^\delta} C \cdot^{\mathsf{c}} C \xrightarrow{h^\delta\cdot^{\mathsf{c}} C} (C \cdot^{\mathsf{c}} C) \cdot^{\mathsf{c}} C \xrightarrow{\alpha} C \cdot^{\mathsf{c}} (C \cdot^{\mathsf{c}} C)$$

$$\begin{array}{ccccccc} (q^\delta\{s\} & \longmapsto & (q^\delta\{\mathsf{fst}\,(t^\delta\,s)\} & \longmapsto & ((p,p'),p'') & \longmapsto & (p,(p',p'')) \\ (q^\delta\,\{\mathsf{fst}\,(t^\delta\,s)\} & & (p,p'),p'') & & & & \\ (p,p'),p'') & & & & & & \end{array}$$

and by going counter-clockwise we get

$$s \longmapsto t^\delta s \longmapsto \begin{array}{l}(\mathsf{fst}\,(t^\delta\,s),\\ (\lambda p.\,\mathsf{fst}\,(t^\delta\,(\mathsf{snd}\,(t^\delta\,s)\,p)),\\ \lambda p'.\,\mathsf{snd}\,(t^\delta\,(\mathsf{snd}\,(t^\delta s)\,p))\,p'))\end{array}$$

$$C \xrightarrow{\;h^\delta\;} C \cdot^{\mathrm{c}} C \xrightarrow{\;C\cdot^{\mathrm{c}} h^\delta\;} C \cdot^{\mathrm{c}} (C \cdot^{\mathrm{c}} C)$$

$$\begin{array}{l}(q^\delta\,\{s\}\,(p,\\ q^\delta\,\{\mathsf{snd}\,(t^\delta\,s)\,p\}\,(p',p'')))\end{array} \longmapsfrom \begin{array}{l}(p,\\ q^\delta\,\{\mathsf{snd}\,(t^\delta\,s)\,p\}\,(p',p''))\end{array} \longmapsfrom (p,(p',p''))$$

from where the second and fifth directed container laws follow

$$s \downarrow (p \oplus p') = \mathsf{snd}\,(t^\delta\,s)\,(q^\delta\,\{s\}\,(p,p')) = \mathsf{snd}\,(t^\delta\,s)\,(q^\delta\,\{\mathsf{fst}\,(t^\delta\,s)\}\,(p,p')) =$$
$$\mathsf{snd}\,(t^\delta\,(\mathsf{snd}\,(t^\delta\,s)\,p))\,p' = (s \downarrow p) \downarrow p'$$
$$(p \oplus \{s\}\,p') \oplus \{s\}\,p'' = q^\delta\,\{s\}\,(q^\delta\,\{s\}\,(p',p''),p'') = q^\delta\,\{s\}\,(q^\delta\,\{\mathsf{fst}\,(t^\delta\,s)\}\,(p',p''),p'') =$$
$$q^\delta\,\{s\}\,(p,q^\delta\,\{\mathsf{snd}\,(t^\delta\,s)\,p\}\,(p',p'')) = p \oplus \{s\}\,(p' \oplus \{s \downarrow p\}\,p'')$$ □

Proof of Proposition 3.13.

Proof. By interpreting the given directed container $(C, \downarrow, \mathsf{o}, \oplus)$ we get a comonad (D, ε, δ) whereby $D = [\![C]\!]^{\mathrm{c}}$, $\varepsilon\,(s,v) = v\,(\mathsf{o}\,\{s\})$ and $\delta\,(s,v) = (s, \lambda p.\,(s \downarrow p, \lambda p'.\,v\,(p \oplus \{s\}\,p')))$.

From the comonad, we get a directed container $(C, \downarrow', \mathsf{o}', \oplus') = \lceil (D,\varepsilon,\delta), C \rceil$ by taking $s \downarrow' p = \mathsf{snd}\,(t^\delta\,s)\,p$, $\mathsf{o}'\,\{s\} = q^\varepsilon\,\{s\}\,*$, $p \oplus'\,\{s\}\,p' = q^\delta\,\{s\}\,(p,p')$.

This directed container must be equal to the original directed container $(C, \downarrow, \mathsf{o}, \oplus)$, i.e., we need to prove that $s \downarrow' p = s \downarrow p$ and $\mathsf{o}'\,\{s\} = \mathsf{o}\,\{s\}$ and $p \oplus' p' = p \oplus p'$.

By the definitions of e, m, $\ulcorner - \urcorner^{\mathrm{c}}$, for the container morphisms $t^\varepsilon \lhd q^\varepsilon = \ulcorner \mathsf{e} \circ \varepsilon \urcorner^{\mathrm{c}}$ and $t^\delta \lhd q^\delta = \ulcorner \mathsf{m} \circ \delta \urcorner^{\mathrm{c}}$ we have that

$$t^\varepsilon\,s = *$$
$$q^\varepsilon\,\{s\}\,* = \varepsilon\,(s, \mathsf{id})$$
$$t^\delta\,s = (\mathsf{fst}\,(\delta\,(s,\mathsf{id})), \lambda p.\,\mathsf{fst}\,(\mathsf{snd}\,(\delta\,(s,\mathsf{id}))\,p))$$
$$q^\delta\,\{s\}\,(p,p') = \mathsf{snd}\,(\mathsf{snd}\,(\delta\,(s,\mathsf{id}))\,p)\,p'$$

Using the definitions of $\downarrow'$, o', $\oplus'$, ε, δ, we calculate:

$$s \downarrow' p = \mathsf{snd}\,(t^\delta\,s)\,p = \mathsf{fst}\,(\mathsf{snd}\,(\delta\,(s,\mathsf{id}))\,p)) = s \downarrow p$$
$$\mathsf{o}'\,\{s\} = q^\varepsilon\,\{s\}\,* = \varepsilon\,(s,\mathsf{id}) = \mathsf{o}\,\{s\}$$
$$p \oplus'\,\{s\}\,p' = q^\delta\,\{s\}\,(p,p') = \mathsf{snd}\,(\mathsf{snd}\,(\delta\,(s,\mathsf{id}))\,p)\,p' = p \oplus \{s\}\,p'$$ □

Proof of Proposition 3.14.

Proof. The comonad (D, ε, δ) induces a directed container $(S \lhd P, \downarrow, \mathsf{o}, \oplus) = \lceil (D,\varepsilon,\delta), S \lhd P \rceil$ whereby

$$s \downarrow p = \mathsf{fst}\,(\mathsf{snd}\,(\delta\,(s,\mathsf{id}))\,p)$$
$$\mathsf{o}\,\{s\} = \varepsilon\,\{P\,s\}\,(s,\mathsf{id})$$
$$p \oplus \{s\}\,p' = \mathsf{snd}\,(\mathsf{snd}\,(\delta\,(s,\mathsf{id}))\,p)\,p'$$

By interpreting this directed container, we get a comonad $(D', \varepsilon', \delta') = [\![S \lhd P, \downarrow, \mathsf{o}, \oplus]\!]^{\mathrm{dc}}$ by taking $\varepsilon'\,(s,v) = v\,(\mathsf{o}\,\{s\})$ and $\delta'\,(s,v) = (s, \lambda p.\,(s \downarrow p, \lambda p'.\,v\,(p \oplus \{s\}\,p')))$.

This comonad must equal (D, ε, δ), i.e., we need to prove that $D' = D$ and $\varepsilon'\,\{X\}\,(s, v) = \varepsilon\,\{X\}\,(s, v)$ and $\delta'\,\{X\}\,(s, v) = \delta\,\{X\}\,(s, v)$.

First of all, from the definition of directed container interpretation, we know that the underlying functors are equal: $D' = [\![S \lhd P]\!]^{\mathrm{c}} = D$.

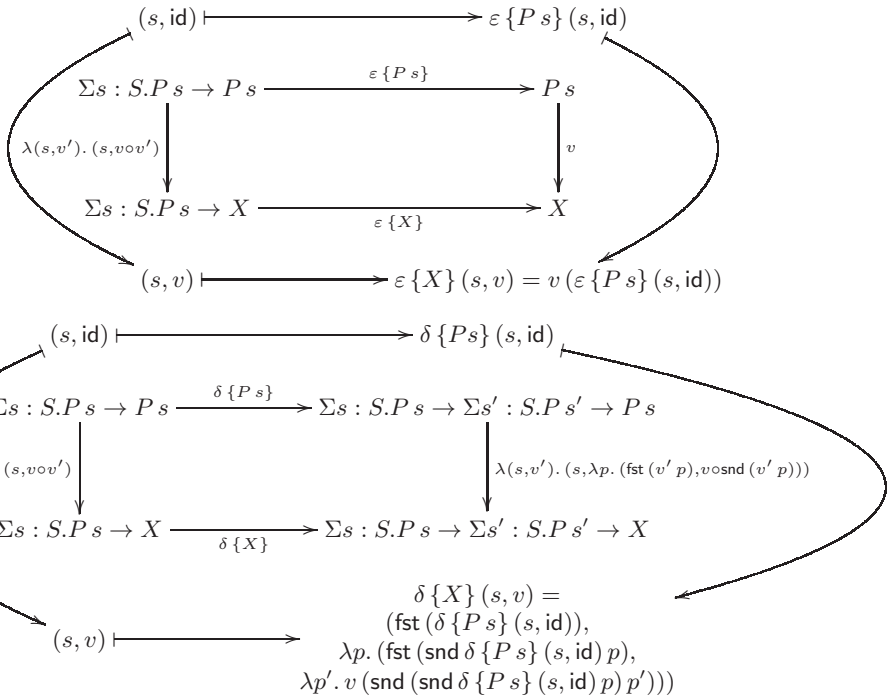
Using the definitions of ε', δ', $\downarrow$, o, $\oplus$ we can calculate

$$\varepsilon'\,\{X\}(s, v) = v\,(\mathsf{o}\,\{s\}) = v\,(q^{\varepsilon}\,\{s\}\,*) = v\,(\varepsilon\,\{P\,s\}\,(s, \mathsf{id}))$$

$$\delta'\,\{X\}(s, v) = (s, \lambda p.\, s \downarrow p\,,\ \lambda p'.\, v\,(p \oplus \{s\}\,p')) =$$

$$= (s, \lambda p.\, (\mathsf{fst}\,(\mathsf{snd}\,\delta\,\{P\,s\}\,(s, \mathsf{id})\,p), \lambda p'.\, v\,(\mathsf{snd}\,(\mathsf{snd}\,\delta\,\{P\,s\}\,(s, \mathsf{id})\,p)\,p')))$$

Now, because of naturality of ε and δ expressed in the diagrams



it is evident that the counit and comultiplication of (D, ε, δ) and $(D', \varepsilon', \delta')$ are equal:

$$\varepsilon'\,\{X\}\,(s, v) = \varepsilon\,\{X\}\,(s, v)$$

$$\delta'\,\{X\}\,(s, v) = \delta\,\{X\}\,(s, v)$$

□

Appendix B. Proofs for Section 4

Proof of Proposition 4.4.

We must show that the definitions yield a strict directed container that is a product of two given strict directed containers in the category of all directed containers.

We first check that E is a strict directed container.

Lemma B.1. *The data $\downarrow^+$ and $\oplus^+$ equip the container $S \lhd P^+$ with a strict directed container structure.*

Proof. Auxiliary statements $s \mathbin{\overline{\downarrow_0^+}} (p \mathbin{\overline{\oplus_0^+}} p') = (s \mathbin{\overline{\downarrow_0^+}} p) \downarrow^+ p'$ and $s \mathbin{\overline{\downarrow_1^+}} (p \mathbin{\overline{\oplus_1^+}} p') = (s \mathbin{\overline{\downarrow_1^+}} p) \downarrow^+ p'$ for law 1, by mutual induction on the two ps. We show only the cases for the first auxiliary statement; those of the second are symmetric.
Case $p = (p_0, \mathsf{nothing})$, $p' = \mathsf{inl}\,(p_0', \mathsf{nothing})$:

$$\begin{aligned}
& (s_0, v_0) \mathbin{\overline{\downarrow_0^+}} ((p_0, \mathsf{nothing}) \mathbin{\overline{\oplus_0^+}} \mathsf{inl}\,(p_0', \mathsf{nothing})) \\
= \quad & \{\text{definitions of } \overline{\downarrow_0^+}, \overline{\oplus_0^+}\} \\
& ((s_0 \downarrow_0^+ (p_0 \oplus_0^+ p_0'), \lambda p.\, v_0\,((p_0 \oplus_0^+ p_0') \oplus_0^+ p)), v_0\,(p_0 \oplus_0^+ p_0')) \\
= \quad & \{\text{strict directed container laws 1, 2}\} \\
& (((s_0 \downarrow_0^+ p_0) \downarrow_0^+ p_0', \lambda p.\, v_0\,(p_0 \oplus_0^+ (p_0' \oplus_0^+ p))), v_0\,(p_0 \oplus_0^+ p_0')) \\
= \quad & \{\text{definitions of } \overline{\downarrow_0^+}, \downarrow^+\} \\
& ((s_0, v_0) \mathbin{\overline{\downarrow_0^+}} (p_0, \mathsf{nothing})) \downarrow^+ \mathsf{inl}\,(p_0', \mathsf{nothing})
\end{aligned}$$

Case for $p = (p_0, \mathsf{nothing})$, $p' = \mathsf{inl}\,(p_0', \mathsf{just}\,p_1')$:

$$\begin{aligned}
& (s_0, v_0) \mathbin{\overline{\downarrow_0^+}} ((p_0, \mathsf{nothing}) \mathbin{\overline{\oplus_0^+}} \mathsf{inl}\,(p_0', \mathsf{just}\,p_1')) \\
= \quad & \{\text{definitions of } \overline{\downarrow_0^+}, \overline{\oplus_0^+}\} \\
& v_0\,(p_0 \oplus_0^+ p_0') \mathbin{\overline{\downarrow_1^+}} p_1' \\
= \quad & \{\text{definitions of } \overline{\downarrow_0^+}, \downarrow^+\} \\
& ((s_0, v_0) \mathbin{\overline{\downarrow_0^+}} (p_0, \mathsf{nothing})) \downarrow^+ \mathsf{inl}\,(p_0', \mathsf{just}\,p_1')
\end{aligned}$$

Case $p = (p_0, \mathsf{nothing})$, $p' = \mathsf{inr}\,p'$:

$$\begin{aligned}
& (s_0, v_0) \mathbin{\overline{\downarrow_0^+}} ((p_0, \mathsf{nothing}) \mathbin{\overline{\oplus_0^+}} \mathsf{inr}\,p') \\
= \quad & \{\text{definitions of } \overline{\downarrow_0^+}, \overline{\oplus_0^+}\} \\
& v_0\,p_0 \mathbin{\overline{\downarrow_1^+}} p' \\
= \quad & \{\text{definitions of } \overline{\downarrow_0^+}, \downarrow^+\} \\
& ((s_0, v_0) \mathbin{\overline{\downarrow_0^+}} (p_0, \mathsf{nothing})) \downarrow^+ \mathsf{inr}\,p'
\end{aligned}$$

Case $p = (p_0, \mathsf{just}\,p_1)$:

$$\begin{aligned}
& (s_0, v_0) \mathbin{\overline{\downarrow_0^+}} ((p_0, \mathsf{just}\,p_1) \mathbin{\overline{\oplus_0^+}} p') \\
= \quad & \{\text{definitions of } \overline{\downarrow_0^+}, \overline{\oplus_0^+}\} \\
& v_0\,p_0 \mathbin{\overline{\downarrow_1^+}} (p_1 \mathbin{\overline{\oplus_1^+}} p') \\
= \quad & \{\text{inductive hypothesis for } p_1\} \\
& (v_0\,p_0 \mathbin{\overline{\downarrow_1^+}} p_1) \downarrow^+ p') \\
= \quad & \{\text{definition of } \overline{\downarrow_0^+}\} \\
& ((s_0, v_0) \mathbin{\overline{\downarrow_0^+}} (p_0, \mathsf{just}\,p_1)) \downarrow^+ p'
\end{aligned}$$

Strict directed container law 1. Case $s = (s_0, s_1)$, $p = \mathsf{inl}\, p$:

$$\begin{array}{ll} & (s_0, s_1) \downarrow^+ (\mathsf{inl}\, p \oplus^+ p') \\ = & \quad \{\text{definition of } \oplus^+\} \\ & (s_0, s_1) \downarrow^+ \mathsf{inl}\, (p \mathbin{\overline{\oplus_0^+}} p') \\ = & \quad \{\text{definition of } \downarrow^+\} \\ & s_0 \mathbin{\overline{\downarrow_0^+}} (p \mathbin{\overline{\oplus_0^+}} p') \\ = & \quad \{\text{first aux. statement}\} \\ & (s_0 \mathbin{\overline{\downarrow_0^+}} p) \downarrow^+ p' \\ = & \quad \{\text{definition of } \downarrow^+\} \\ & ((s_0, s_1) \downarrow^+ \mathsf{inl}\, p) \downarrow^+ p' \end{array}$$

Case $p = \mathsf{inr}\, p$ is symmetric.

Auxiliary statements $(p \mathbin{\overline{\oplus_0^+}} p') \mathbin{\overline{\oplus_0^+}} p'' = p \mathbin{\overline{\oplus_0^+}} (p' \oplus^+ p'')$ and $(p \mathbin{\overline{\oplus_1^+}} p') \mathbin{\overline{\oplus_1^+}} p'' = p \mathbin{\overline{\oplus_1^+}} (p' \oplus^+ p'')$ for law 2, by mutual induction on the two ps. We show only the cases of the first statement. Case $p = (p_0, \mathsf{nothing})$, $p' = \mathsf{inl}\, (p'_0, \mathsf{nothing})$, $p'' = \mathsf{inl}\, (p''_0, p''_1)$:

$$\begin{array}{ll} & ((p_0, \mathsf{nothing}) \mathbin{\overline{\oplus_0^+}} \mathsf{inl}\, (p'_0, \mathsf{nothing})) \mathbin{\overline{\oplus_0^+}} \mathsf{inl}\, (p''_0, p''_1) \\ = & \quad \{\text{definition of } \oplus_0^+\} \\ & ((p_0 \oplus_0^+ p'_0) \oplus_0^+ p''_0, p''_1) \\ = & \quad \{\text{strict direct container law 2}\} \\ & (p_0 \oplus_0^+ (p'_0 \oplus_0^+ p''_0), p''_1) \\ = & \quad \{\text{definitions of } \oplus_0^+, \oplus^+\} \\ & (p_0, \mathsf{nothing}) \mathbin{\overline{\oplus_0^+}} (\mathsf{inl}\, (p'_0, \mathsf{nothing}) \oplus^+ \mathsf{inl}\, (p''_0, p''_1)) \end{array}$$

Case $p = (p_0, \mathsf{nothing})$, $p' = \mathsf{inl}\, (p'_0, \mathsf{nothing})$, $p'' = \mathsf{inr}\, p''$:

$$\begin{array}{ll} & ((p_0, \mathsf{nothing}) \mathbin{\overline{\oplus_0^+}} \mathsf{inl}\, (p'_0, \mathsf{nothing})) \mathbin{\overline{\oplus_0^+}} \mathsf{inr}\, p'' \\ = & \quad \{\text{definition of } \oplus_0^+\} \\ & (p_0 \oplus_0^+ p'_0, \mathsf{just}\, p'') \\ = & \quad \{\text{definitions of } \oplus_0^+, \oplus^+\} \\ & (p_0, \mathsf{nothing}) \mathbin{\overline{\oplus_0^+}} (\mathsf{inl}\, (p'_0, \mathsf{nothing}) \oplus^+ \mathsf{inr}\, p'') \end{array}$$

Case $p = (p_0, \mathsf{nothing})$, $p' = \mathsf{inl}\, (p'_0, \mathsf{just}\, p'_1)$:

$$\begin{array}{ll} & ((p_0, \mathsf{nothing}) \mathbin{\overline{\oplus_0^+}} \mathsf{inl}\, (p'_0, \mathsf{just}\, p'_1)) \mathbin{\overline{\oplus_0^+}} p'' \\ = & \quad \{\text{definition of } \oplus_0^+\} \\ & (p_0 \oplus_0^+ p'_0, \mathsf{just}\, (p'_1 \mathbin{\overline{\oplus_1^+}} p'')) \\ = & \quad \{\text{definitions of } \oplus_0^+, \oplus^+\} \\ & (p_0, \mathsf{nothing}) \mathbin{\overline{\oplus_0^+}} (\mathsf{inl}\, (p'_0, \mathsf{just}\, p'_1) \oplus^+ p'') \end{array}$$

Case $p = (p_0, \mathsf{nothing})$, $p' = \mathsf{inr}\, p'$:

$$\begin{array}{ll} & ((p_0, \mathsf{nothing}) \mathbin{\overline{\oplus_0^+}} \mathsf{inr}\, p') \mathbin{\overline{\oplus_0^+}} p'' \\ = & \quad \{\text{definition of } \oplus_0^+\} \\ & (p_0, \mathsf{just}\, (p' \mathbin{\overline{\oplus_1^+}} p'')) \\ = & \quad \{\text{definitions of } \oplus_0^+, \oplus^+\} \\ & (p_0, \mathsf{nothing}) \mathbin{\overline{\oplus_0^+}} (\mathsf{inr}\, p' \oplus^+ p'') \end{array}$$

Case $p = (p_0, \mathsf{just}\, p_1)$:

$$\begin{aligned}
& ((p_0, \mathsf{just}\, p_1)\ \overline{\oplus_0^+}\ p')\ \overline{\oplus_0^+}\ p'' \\
= \quad & \{\text{definition of } \overline{\oplus_0^+}\} \\
& (p_0, \mathsf{just}\, ((p_1\ \overline{\oplus_1^+}\ p')\ \overline{\oplus_1^+}\ p'')) \\
= \quad & \{\text{inductive hypothesis for } p_1\} \\
& (p_0, \mathsf{just}\, (p_1\ \overline{\oplus_1^+}\ (p' \oplus^+ p''))) \\
= \quad & \{\text{definition of } \overline{\oplus_0^+}\} \\
& (p_0, \mathsf{just}\, p_1)\ \overline{\oplus_0^+}\ (p' \oplus^+ p'')
\end{aligned}$$

Strict directed container law 2. Case $p = \mathsf{inl}\, p$:

$$\begin{aligned}
& (\mathsf{inl}\, p \oplus^+ p') \oplus^+ p'' \\
= \quad & \{\text{definition of } \oplus^+\} \\
& \mathsf{inl}\, (p\ \overline{\oplus_0^+}\ p') \oplus^+ p'' \\
= \quad & \{\text{definition of } \oplus^+\} \\
& \mathsf{inl}\, ((p\ \overline{\oplus_0^+}\ p')\ \overline{\oplus_0^+}\ p') \\
= \quad & \{\text{first aux. statement}\} \\
& \mathsf{inl}\, (p\ \overline{\oplus_0^+}\ (p' \oplus^+ p'')) \\
= \quad & \{\text{definition of } \oplus^+\} \\
& \mathsf{inl}\, p \oplus^+ (p' \oplus^+ p'')
\end{aligned}$$

Case $p = \mathsf{inr}\, p$ is symmetric. □

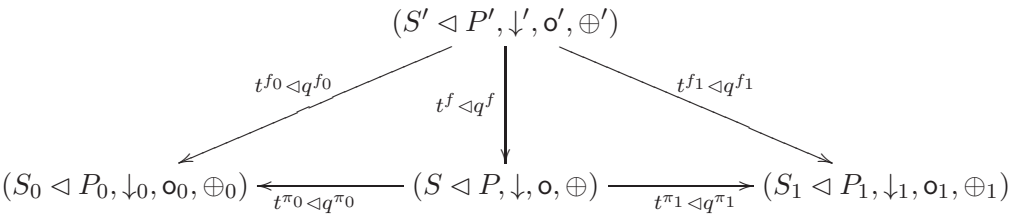
To check that E is a product of E_0 and E_1 we can either verify it directly that it satisfies the required universal property or prove that it interprets to a product of the interpreting comonads. Here we have chosen to pursue the first route.

For E to be a product of E_0 and E_1, it must come with directed container morphisms $\pi_0 = t^{\pi_0} \lhd q^{\pi_0} : E_0 \to E$, $\pi_1 = t^{\pi_1} \lhd q^{\pi_1} : E_1 \to E$. We claim that they can be defined by

- $t^{\pi_0} : S \to S_0$
  $t^{\pi_0}\, ((s_0, v_0), (s_1, v_1)) = s_0$
- $t^{\pi_1} : S \to S_1$
  $t^{\pi_1}\, ((s_0, v_0), (s_1, v_1)) = s_1$
- $q^{\pi_0} : \Pi\{s : S\}.\, P_0\, (t^{\pi_0}\, s) \to P\, s$
  $q^{\pi_0}\, \mathsf{nothing} = \mathsf{nothing}$
  $q^{\pi_0}\, (\mathsf{just}\, p) = \mathsf{just}\, (\mathsf{inl}\, (p, \mathsf{nothing}))$
- $q^{\pi_1} : \Pi\{s : S\}.\, P_1\, (t^{\pi_1}\, s) \to P\, s$
  $q^{\pi_1}\, \mathsf{nothing} = \mathsf{nothing}$
  $q^{\pi_1}\, (\mathsf{just}\, p) = \mathsf{just}\, (\mathsf{inr}\, (p, \mathsf{nothing}))$

Moreover, any directed container $E' = (S' \lhd P', \downarrow', \mathsf{o}', \oplus')$ with two directed container morphisms $f_0 = t^{f_0} \lhd q^{f_0} : E' \to E_0$ and $f_1 = t^{f_1} \lhd q^{f_1} : E' \to E_1$ must jointly determine a unique directed container morphism $f = t^f \lhd q^f : E' \to E$ such that the following two

triangles commute.



(†)

We claim that f is given by

- $t^f : S' \to S$
  $t^f\ s = (\overline{t^{f_0}}\ s, \overline{t^{f_1}}\ s)$
  where
  $\overline{t^{f_0}} : S' \to \overline{S_0}$
  $\overline{t^{f_1}} : S' \to \overline{S_1}$
  (by mutual corecursion)
  $\overline{t^{f_0}}\ s = (t^{f_0}\ s, \lambda p.\, \overline{t^{f_1}}\ (s \downarrow' q^{f_0}\ (\mathsf{just}\, p)))$
  $\overline{t^{f_1}}\ s = (t^{f_1}\ s, \lambda p.\, \overline{t^{f_0}}\ (s \downarrow' q^{f_1}\ (\mathsf{just}\, p)))$

- $q^f : \Pi\{s : S'\}.\, P\ (t^f\ s) \to P'\ s$
  $q^f\ \mathsf{nothing} = \mathsf{o}'$
  $q^f\ (\mathsf{just}\ (\mathsf{inl}\, p)) = \overline{q^{f_0}}\ p$
  $q^f\ (\mathsf{just}\ (\mathsf{inr}\, p)) = \overline{q^{f_1}}\ p$
  where
  $\overline{q^{f_0}} : \Pi\{s : S'\}.\, \overline{P_0^+}\ (\overline{t^{f_0}}\ s) \to P'\ s$
  $\overline{q^{f_1}} : \Pi\{s : S'\}.\, \overline{P_1^+}\ (\overline{t^{f_1}}\ s) \to P'\ s$
  (by mutual recursion)
  $\overline{q^{f_0}}\ (p_0, \mathsf{nothing}) = q^{f_0}\ (\mathsf{just}\, p_0)$
  $\overline{q^{f_0}}\ (p_0, \mathsf{just}\, p_1) = q^{f_0}\ (\mathsf{just}\, p_0) \oplus' \overline{q^{f_1}} p_1$
  $\overline{q^{f_1}}\ (p_1, \mathsf{nothing}) = q^{f_1}\ (\mathsf{just}\, p_1)$
  $\overline{q^{f_1}}\ (p_1, \mathsf{just}\, p_1) = q^{f_1}\ (\mathsf{just}\, p_1) \oplus' \overline{q^{f_0}} p_0$

Lemma B.2. *The container morphisms π_0, π_1 are directed container morphisms.*[3]

Proof. We give the proof only for π_0. The proof for π_1 is symmetric.
Directed container morphism law 1. Case $p = \mathsf{nothing}$:

$$\begin{array}{ll} & t^{\pi_0}\ (s \downarrow (q^{\pi_0}\ \mathsf{nothing})) \\ = & \quad \{\text{definition of } q^{\pi_0}\} \\ & t^{\pi_0}\ (s \downarrow \mathsf{nothing}) \\ = & \quad \{\text{definition of } \downarrow\} \\ & t^{\pi_0}\ s \\ = & \quad \{\text{definition of } \downarrow_0\} \\ & t^{\pi_0}\ s \downarrow_0 \mathsf{nothing} \end{array}$$

[3]They are in fact strict directed container morphisms, but we will not prove this here, as we have not defined this concept.

Case $p = \mathsf{just}\, p$:

$$
\begin{array}{ll}
 & t^{\pi_0}\,(((s_0, v_0), (s_1, v_1)) \downarrow q^{\pi_0}\,(\mathsf{just}\, p)) \\
= & \quad \{\text{definition of } q^{\pi_0}\} \\
 & t^{\pi_0}\,(((s_0, v_0), (s_1, v_1)) \downarrow \mathsf{just}\,(\mathsf{inl}\,(p, \mathsf{nothing}))) \\
= & \quad \{\text{definition of } \downarrow\} \\
 & t^{\pi_0}\,(((s_0, v_0), (s_1, v_1)) \downarrow^{+} \mathsf{inl}\,(p, \mathsf{nothing})) \\
= & \quad \{\text{definition of } \downarrow^{+}\} \\
 & t^{\pi_0}\,((s_0, v_0)\ \overline{\downarrow_0^{+}}\ (p, \mathsf{nothing})) \\
= & \quad \{\text{definition of } \overline{\downarrow_0^{+}}\} \\
 & t^{\pi_0}\,((s_0 \downarrow_0^{+} p, \lambda p'.\, v_0\,(p \oplus_0^{+} p')), v_0\, p) \\
= & \quad \{\text{definition of } t^{\pi_0}\} \\
 & s_0 \downarrow_0^{+} p \\
= & \quad \{\text{definition of } t^{\pi_0}\} \\
 & t^{\pi_0}\,((s_0, v_0), (s_1, v_1)) \downarrow_0^{+} p \\
= & \quad \{\text{definition of } \downarrow_0\} \\
 & t^{\pi_0}\,((s_0, v_0), (s_1, v_1)) \downarrow_0 \mathsf{just}\, p
\end{array}
$$

Directed container morphism law 2:

$$
\begin{array}{ll}
 & q^{\pi_0}\, \mathsf{o}_0 \\
= & \quad \{\text{definition of } \mathsf{o}_0\} \\
 & q^{\pi_0}\, \mathsf{nothing} \\
= & \quad \{\text{definition of } q^{\pi_0}\} \\
 & \mathsf{nothing} \\
= & \quad \{\text{definition of } \mathsf{o}\} \\
 & \mathsf{o}
\end{array}
$$

Directed container morphism law 3. Case $p = \mathsf{nothing}$:

$$
\begin{array}{ll}
 & q^{\pi_0}\,(\mathsf{nothing} \oplus_0 p') \\
= & \quad \{\text{definition of } \oplus_0\} \\
 & q^{\pi_0}\, p' \\
= & \quad \{\text{definition of } \oplus\} \\
 & \mathsf{nothing} \oplus q^{\pi_0}\, p'
\end{array}
$$

Case $p = \mathsf{just}\, p$, $p' = \mathsf{nothing}$:

$$
\begin{array}{ll}
 & q^{\pi_0}\,(\mathsf{just}\, p \oplus_0 \mathsf{nothing}) \\
= & \quad \{\text{definition of } \oplus_0\} \\
 & q^{\pi_0}\,(\mathsf{just}\, p) \\
= & \quad \{\text{definition of } q^{\pi_0}\} \\
 & \mathsf{just}\,(\mathsf{inl}\,(p, \mathsf{nothing})) \\
= & \quad \{\text{definition of } \oplus\} \\
 & \mathsf{just}\,(\mathsf{inl}\,(p, \mathsf{nothing})) \oplus \mathsf{nothing} \\
= & \quad \{\text{definition of } q^{\pi_0}\} \\
 & q^{\pi_0}\,(\mathsf{just}\, p) \oplus \mathsf{nothing}
\end{array}
$$

Case $p = \mathsf{just}\, p$, $p' = \mathsf{just}\, p'$:

$$
\begin{array}{ll}
 & q^{\pi_0}\,(\mathsf{just}\, p \oplus_0 \mathsf{just}\, p') \\
= & \quad \{\text{definition of } \oplus_0\} \\
 & q^{\pi_0}\,(\mathsf{just}\,(p \oplus_0^+ p')) \\
= & \quad \{\text{definition of } q^{\pi_0}\} \\
 & \mathsf{just}\,(\mathsf{inl}\,(p \oplus_0^+ p', \overline{\mathsf{nothing}})) \\
= & \quad \{\text{definition of } \overline{\oplus_0^+}\} \\
 & \mathsf{just}\,(\mathsf{inl}\,((p, \mathsf{nothing}) \,\overline{\oplus_0^+}\, \mathsf{inl}\,(p', \mathsf{nothing}))) \\
= & \quad \{\text{definition of } \oplus^+\} \\
 & \mathsf{just}\,(\mathsf{inl}\,(p, \mathsf{nothing}) \oplus^+ \mathsf{inl}\,(p', \mathsf{nothing})) \\
= & \quad \{\text{definition of } \oplus\} \\
 & \mathsf{just}\,(\mathsf{inl}\,(p, \mathsf{nothing}) \oplus \mathsf{just}\,(\mathsf{inl}\,(p', \mathsf{nothing})) \\
= & \quad \{\text{definition of } q^{\pi_0}\} \\
 & q^{\pi_0}\,(\mathsf{just}\, p) \oplus q^{\pi_0}\,(\mathsf{just}\, p'))
\end{array}
$$

□

Lemma B.3. *The container morphism $f = t^f \lhd q^f$ is a directed container morphism.*

Proof. Auxiliary statements $t^f\,(s \downarrow' \overline{q^{f_0}}\, p) = \overline{t^{f_0}}\, s\, \overline{\downarrow_0^+}\, p$ and $t^f\,(s \downarrow' \overline{q^{f_1}}\, p) = \overline{t^{f_1}}\, s\, \overline{\downarrow_1^+}\, p$ for law 1, by mutual induction on the ps, showing the cases of the first statement; those of the second one are symmetric. Case $p = (p_0, \mathsf{nothing})$.

$$
\begin{array}{ll}
 & t^f\,(s \downarrow' \overline{q^{f_0}}(p_0, \mathsf{nothing})) \\
= & \quad \{\text{definition of } \overline{q^{f_0}}\} \\
 & t^f\,(s \downarrow' q^{f_0}\,(\mathsf{just}\, p_0)) \\
= & \quad \{\text{definition of } t^f\} \\
 & (\overline{t^{f_0}}\,(s \downarrow' q^{f_0}\,(\mathsf{just}\, p_0)), \overline{t^{f_1}}\,(s \downarrow' q^{f_0}\,(\mathsf{just}\, p_0))) \\
= & \quad \{\text{definition of } \overline{t^{f_0}}\} \\
 & ((t^{f_0}\,(s \downarrow' q^{f_0}\,(\mathsf{just}\, p_0)), \lambda p.\, \overline{t^{f_1}}\,((s \downarrow' q^{f_0}\,(\mathsf{just}\, p_0)) \downarrow' q^{f_0}\,(\mathsf{just}\, p))), \overline{t^{f_1}}\,(s \downarrow' q^{f_0}\,(\mathsf{just}\, p_0))) \\
= & \quad \{\text{directed container law 2}\} \\
 & ((t^{f_0}\,(s \downarrow' q^{f_0}\,(\mathsf{just}\, p_0)), \lambda p.\, \overline{t^{f_1}}\,(s \downarrow' (q^{f_0}\,(\mathsf{just}\, p_0) \oplus' q^{f_0}\,(\mathsf{just}\, p)))), \overline{t^{f_1}}\,(s \downarrow' q^{f_0}\,(\mathsf{just}\, p_0))) \\
= & \quad \{\text{directed container morphism laws 1, 3}\} \\
 & ((t^{f_0}\, s \downarrow_0 \mathsf{just}\, p_0, \lambda p.\, \overline{t^{f_1}}\,(s \downarrow' (q^{f_0}\,(\mathsf{just}\, p_0 \oplus_0 \mathsf{just}\, p)))), \overline{t^{f_1}}\,(s \downarrow' q^{f_0}\,(\mathsf{just}\, p_0))) \\
= & \quad \{\text{definitions of } \downarrow_0, \oplus_0\} \\
 & ((t^{f_0}\, s \downarrow_0^+ p_0, \lambda p.\, \overline{t^{f_1}}\,(s \downarrow' (q^{f_0}\,(\mathsf{just}\,(p_0 \oplus_0^+ p)))), \overline{t^{f_1}}\,(s \downarrow' q^{f_0}\,(\mathsf{just}\, p_0))) \\
= & \quad \{\text{definition of } \overline{\downarrow_0^+}\} \\
 & (t^{f_0}\, s, \lambda p.\, \overline{t^{f_1}}\,(s \downarrow' q^{f_0}\,(\mathsf{just}\, p)))\, \overline{\downarrow_0^+}\, (p_0, \mathsf{nothing}) \\
= & \quad \{\text{definition of } \overline{t^{f_0}}\} \\
 & \overline{t^{f_0}}\, s\, \overline{\downarrow_0^+}\, (p_0, \mathsf{nothing})
\end{array}
$$

Case $p = (p_0, \mathsf{just}\, p_1)$:

$$\begin{array}{ll} & t^f\ (s \downarrow' \overline{q^{f_0}}\ (p_0, \mathsf{just}\, p_1)) \\ = & \quad \{\text{definition of } \overline{q^{f_0}}\} \\ & t^f\ (s \downarrow' (q^{f_0}\ (\mathsf{just}\, p_0) \oplus' \overline{q^{f_1}}\ p_1)) \\ = & \quad \{\text{directed container law 2 }\} \\ & t^f\ ((s \downarrow' q^{f_0}\ (\mathsf{just}\, p_0)) \downarrow' \overline{q^{f_1}}\ p_1) \\ = & \quad \{\text{inductive hypothesis for } p_1\} \\ & \overline{t^{f_1}}\ (s \downarrow' q^{f_0}\ (\mathsf{just}\, p_0))\ \overline{\downarrow_1^+}\ p_1 \\ = & \quad \{\text{definition of } \overline{\downarrow_0^+}\} \\ & (t^{f_0}\ s, \lambda p.\, \overline{t^{f_1}}\ (s \downarrow' q^{f_0}\ (\mathsf{just}\, p)))\ \overline{\downarrow_0^+}\ (p_0, \mathsf{just}\, p_1) \\ = & \quad \{\text{definition of } \overline{t^{f_0}}\} \\ & \overline{t^{f_0}}\ s\ \overline{\downarrow_0^+}\ (p_0, \mathsf{just}\, p_1) \end{array}$$

Directed container morphism law 1. Case $p = \mathsf{nothing}$:

$$\begin{array}{ll} & t^f\ (s \downarrow' q^f\ \mathsf{nothing}) \\ = & \quad \{\text{definition of } q^f\} \\ & t^f\ (s \downarrow' \mathsf{o}') \\ = & \quad \{\text{directed container law 1}\} \\ & t^f\ s \\ = & \quad \{\text{definition of } \downarrow\} \\ & t^f\ s \downarrow \mathsf{nothing} \end{array}$$

Case $p = \mathsf{just}\, (\mathsf{inl}\, p)$:

$$\begin{array}{ll} & t^f\ (s \downarrow' q^f\ (\mathsf{just}\, (\mathsf{inl}\, p))) \\ = & \quad \{\text{definition of } q^f\} \\ & t^f\ (s \downarrow' \overline{q^{f_0}}\ p) \\ = & \quad \{\text{aux. statement}\} \\ & \overline{t^{f_0}}\ s\ \overline{\downarrow_0^+}\ p \\ = & \quad \{\text{definition of } t^f, \downarrow\} \\ & t^f\ s \downarrow \mathsf{just}\, (\mathsf{inl}\, p) \end{array}$$

Case $p = \mathsf{just}\, (\mathsf{inr}\, p)$ is symmetric.
Directed container morphism law 2:

$$\begin{array}{ll} & q^f\ \mathsf{o} \\ = & \quad \{\text{definition of } \mathsf{o}\} \\ & q^f\ \mathsf{nothing} \\ = & \quad \{\text{definition of } q^f\} \\ & \mathsf{o}' \end{array}$$

Auxiliary statements $\overline{q^{f_0}}\, (p\ \overline{\oplus_0^+}\ p') = \overline{q^{f_0}}\, p\ \oplus'\ q^f\, (\mathsf{just}\, p')$ and $\overline{q^{f_1}}\, (p\ \overline{\oplus_1^+}\ p') = \overline{q^{f_1}}\, p\ \oplus'\ q^f\, (\mathsf{just}\, p')$ for law 3, by mutual induction on the ps, showing the cases of the first statement.

Case $p = (p_0, \mathsf{nothing})$, $p' = \mathsf{inl}\,(p'_0, \mathsf{nothing})$:

$$\begin{array}{ll} & \overline{q^{f_0}}\,((p_0, \mathsf{nothing})\ \overline{\oplus_0^+}\ \mathsf{inl}\,(p'_0, \mathsf{nothing})) \\ = & \quad \{\text{definition of } \overline{\oplus_0^+}\} \\ & \overline{q^{f_0}}\,(p_0 \oplus_0^+ p'_0, \mathsf{nothing}) \\ = & \quad \{\text{definition of } \overline{q^{f_0}}\} \\ & q^{f_0}\,(\mathsf{just}\,(p_0 \oplus_0^+ p'_0)) \\ = & \quad \{\text{definition of } \oplus_0\} \\ & q^{f_0}\,(\mathsf{just}\,p_0 \oplus_0 \mathsf{just}\,p'_0) \\ = & \quad \{\text{directed container morphism law 3}\} \\ & q^{f_0}\,(\mathsf{just}\,p_0) \oplus' q^{f_0}\,(\mathsf{just}\,p'_0) \\ = & \quad \{\text{definition of } \overline{q^{f_0}}\} \\ & \overline{q^{f_0}}\,(p_0, \mathsf{nothing}) \oplus' \overline{q^{f_0}}\,(p'_0, \mathsf{nothing}) \\ = & \quad \{\text{definition of } q^f\} \\ & \overline{q^{f_0}}\,(p_0, \mathsf{nothing}) \oplus' q^f\,(\mathsf{just}\,(\mathsf{inl}\,(p'_0, \mathsf{nothing}))) \end{array}$$

Case $p = (p_0, \mathsf{nothing})$, $p' = \mathsf{inl}\,(p'_0, \mathsf{just}\,p'_1)$:

$$\begin{array}{ll} & \overline{q^{f_0}}\,((p_0, \mathsf{nothing})\ \overline{\oplus_0^+}\ \mathsf{inl}(p'_0, \mathsf{just}\,p'_1)) \\ = & \quad \{\text{definition of } \overline{\oplus_0^+}\} \\ & \overline{q^{f_0}}\,(p_0 \oplus_0^+ p'_0, \mathsf{just}\,p'_1) \\ = & \quad \{\text{definition of } \overline{q^{f_0}}\} \\ & q^{f_0}\,(\mathsf{just}\,(p_0 \oplus_0^+ p'_0)) \oplus' \overline{q^{f_1}}\,p'_1 \\ = & \quad \{\text{definition of } \oplus_0\} \\ & q^{f_0}\,(\mathsf{just}\,p_0 \oplus_0 \mathsf{just}\,p'_0) \oplus' \overline{q^{f_1}}\,p'_1 \\ = & \quad \{\text{directed container morphism law 3}\} \\ & (q^{f_0}\,(\mathsf{just}\,p_0) \oplus' q^{f_0}\,(\mathsf{just}\,p'_0)) \oplus' \overline{q^{f_1}}\,p'_1 \\ = & \quad \{\text{directed container law 5}\} \\ & q^{f_0}\,(\mathsf{just}\,p_0) \oplus' (q^{f_0}\,(\mathsf{just}\,p'_0) \oplus' \overline{q^{f_1}}\,p'_1) \\ = & \quad \{\text{definition of } \overline{q^{f_0}}\} \\ & \overline{q^{f_0}}\,(p_0, \mathsf{nothing}) \oplus' \overline{q^{f_0}}\,(p'_0, \mathsf{just}\,p'_1) \\ = & \quad \{\text{definition of } q^f\} \\ & \overline{q^{f_0}}\,(p_0, \mathsf{nothing}) \oplus' q^f\,(\mathsf{just}\,(\mathsf{inl}\,(p'_0, \mathsf{just}\,p'_1))) \end{array}$$

Case $p = (p_0, \mathsf{nothing})$, $p' = \mathsf{inr}\,p'$:

$$\begin{array}{ll} & \overline{q^{f_0}}\,((p_0, \mathsf{nothing})\ \overline{\oplus_0^+}\ \mathsf{inr}\,p') \\ = & \quad \{\text{definition of } \overline{\oplus_0^+}\} \\ & \overline{q^{f_0}}\,(p_0, \mathsf{just}\,p') \\ = & \quad \{\text{definition of } \overline{q^{f_0}}\} \\ & q^{f_0}\,(\mathsf{just}\,p_0) \oplus' \overline{q^{f_1}}\,p' \\ = & \quad \{\text{definitions of } \overline{q^{f_0}} \text{ and } q^f\} \\ & \overline{q^{f_0}}\,(p_0, \mathsf{nothing}) \oplus' q^f\,(\mathsf{just}\,(\mathsf{inr}\,p')) \end{array}$$

Case $p = (p_0, \mathsf{just}\, p_1)$:

$$\begin{array}{ll} & \overline{q^{f_0}}\,((p_0,\, \mathsf{just}\, p_1)\ \overline{\oplus_0^+}\ p') \\ = & \quad \{\text{definition of } \overline{\oplus_0^+}\} \\ & \overline{q^{f_0}}\,(p_0,\, \mathsf{just}\,(p_1\ \overline{\oplus_1^+}\ p')) \\ = & \quad \{\text{definition of } \overline{q^{f_0}}\} \\ & q^{f_0}\,(\mathsf{just}\, p_0) \oplus' \overline{q^{f_1}}\,(p_1\ \overline{\oplus_1^+}\ p') \\ = & \quad \{\text{inductive hypothesis for } p_1\} \\ & q^{f_0}\,(\mathsf{just}\, p_0) \oplus' (\overline{q^{f_1}}\, p_1 \oplus' q^f\,(\mathsf{just}\, p')) \\ = & \quad \{\text{directed container law 5}\} \\ & (q^{f_0}\,(\mathsf{just}\, p_0) \oplus' \overline{q^{f_1}}\, p_1) \oplus' q^f\,(\mathsf{just}\, p') \\ = & \quad \{\text{definition of } \overline{q^{f_0}}\} \\ & \overline{q^{f_0}}\,(p_0,\, \mathsf{just}\, p_1) \oplus' q^f\,(\mathsf{just}\, p') \end{array}$$

Directed container law 3. Case $p = \mathsf{nothing}$:

$$\begin{array}{ll} & q^f\,(\mathsf{nothing} \oplus p') \\ = & \quad \{\text{definition of } \oplus\ \} \\ & q^f\, p' \\ = & \quad \{\text{directed container law 4}\} \\ & \mathsf{o}' \oplus' q^f\, p' \\ = & \quad \{\text{definition of } q^f\} \\ & q^f\ \mathsf{nothing} \oplus' q^f\, p' \end{array}$$

Case $p = \mathsf{just}\, p$, $p' = \mathsf{nothing}$:

$$\begin{array}{ll} & q^f\,(\mathsf{just}\, p \oplus \mathsf{nothing}) \\ = & \quad \{\text{definition of } \oplus\} \\ & q^f\,(\mathsf{just}\, p) \\ = & \quad \{\text{directed container law 3}\} \\ & q^f\,(\mathsf{just}\, p) \oplus' \mathsf{o}' \\ = & \quad \{\text{definition of } q^f\} \\ & q^f\,(\mathsf{just}\, p) \oplus' q^f\ \mathsf{nothing} \end{array}$$

Case $p = \mathsf{just}\,(\mathsf{inl}\, p)$, $p' = \mathsf{just}\, p'$:

$$\begin{array}{ll} & q^f\,(\mathsf{just}\,(\mathsf{inl}\, p) \oplus \mathsf{just}\, p') \\ = & \quad \{\text{definition of } \oplus\ \} \\ & q^f\,(\mathsf{just}\,(\mathsf{inl}\, p \oplus^+ p')) \\ = & \quad \{\text{definition of } \oplus^+\ \} \\ & q^f\,(\mathsf{just}\,(\mathsf{inl}\,(p\ \overline{\oplus_0^+}\ p'))) \\ = & \quad \{\text{definition of } q^f\} \\ & \overline{\overline{q^{f_0}}\,(p \oplus p')} \\ = & \quad \{\text{first aux. statement}\} \\ & \overline{\overline{q^{f_0}}\, p \oplus' q^f\,(\mathsf{just}\, p')} \\ = & \quad \{\text{definition of } \overline{q^{f_0}}\ \} \\ & q^f\,(\mathsf{just}\,(\mathsf{inl}\, p)) \oplus' q^f\,(\mathsf{just}\, p') \end{array}$$

Case $p = \mathsf{just}\,(\mathsf{inr}\, p)$, $p' = \mathsf{just}\, p'$ is symmetric. □

Lemma B.4. *The product triangles* (†) *commute, i.e.,* $\pi_0 \circ f = f_0$ *and* $\pi_1 \circ f = f_1$.

Proof. We verify only the left triangle $\pi_0 \circ f = f_0$. The right triangle is symmetric. Statement for shapes:

$$\begin{aligned} & t^{\pi_0}\,(t^f\ s) \\ = \quad & \{\text{definition of } t^{\pi_0}\} \\ & \mathsf{fst}\,(\mathsf{fst}\,(t^f\ s)) \\ = \quad & \{\text{definition of } t^f\} \\ & \mathsf{fst}\,(\overline{t^{f_0}}\ s) \\ = \quad & \{\text{definition of } \overline{t^{f_0}}\} \\ & t^{f_0}\ s \end{aligned}$$

Statement for positions. Case $p = \mathsf{nothing}$:

$$\begin{aligned} & q^f\,(q^{\pi_0}\ \mathsf{nothing}) \\ = \quad & \{\text{definition of } q^{\pi_0}\} \\ & q^f\,\mathsf{nothing} \\ = \quad & \{\text{definition of } q^f\} \\ & \mathsf{o}' \\ = \quad & \{\text{directed container morphism law 2}\} \\ & q^{f_0}\,\mathsf{o}_0 \\ = \quad & \{\text{definition of } \mathsf{o}_0\} \\ & q^{f_0}\ \mathsf{nothing} \end{aligned}$$

Case $p = \mathsf{just}\,p$:

$$\begin{aligned} & q^f\,(q^{\pi_0}\ (\mathsf{just}\,p)) \\ = \quad & \{\text{definition of } q^{\pi_0}\} \\ & q^f\ (\mathsf{just}\,(\mathsf{inl}\,(p,\ \mathsf{nothing}))) \\ = \quad & \{\text{definition of } q^f\} \\ & \overline{q^{f_0}}\,(p, \mathsf{nothing}) \\ = \quad & \{\text{definition of } \overline{q^{f_0}}\} \\ & q^{f_0}\ (\mathsf{just}\,p) \end{aligned}$$

□

Lemma B.5. *The directed container morphism $f = t^f \lhd q^f$ is unique, i.e., if there is a directed container morphism $h = t^h \lhd q^h : E' \to E$ such that $\pi_0 \circ h = f_0$ and $\pi_1 \circ h = f_1$, then $f = h$.*

Proof. Auxiliary statements $\overline{t^{f_0}}\, s = \mathsf{fst}\,(t^h\, s)$ and $\overline{t^{f_1}}\, s = \mathsf{snd}\,(t^h\, s)$ for shapes, by mutual coinduction, showing only the case of the first statement.

$$
\begin{aligned}
& \overline{t^{f_0}}\, s \\
= \quad & \{\text{definition of } \overline{t^{f_1}}\} \\
& (t^{f_0}\, s, \lambda p_0.\, \overline{t^{f_1}}\, (s \downarrow' q^{f_0}\, (\mathsf{just}\, p_0))) \\
= \quad & \{\text{assumption}\} \\
& (t^{\pi_0}\, (t^h\, s), \lambda p_0.\, \overline{t^{f_1}}\, (s \downarrow' q^{f_0}\, (\mathsf{just}\, p_0))) \\
= \quad & \{\text{coinductive hypothesis}\} \\
& (t^{\pi_0}\, (t^h\, s), \lambda p_0.\, \mathsf{snd}\, (t^h\, (s \downarrow' q^h\, (q^{\pi_0}\, (\mathsf{just}\, p_0))))) \\
= \quad & \{\text{directed container morphism law 1}\} \\
& (t^{\pi_0}\, (t^h\, s), \lambda p_0.\, \mathsf{snd}\, (t^h\, s \downarrow q^{\pi_0}\, (\mathsf{just}\, p_0))) \\
= \quad & \{\text{definition of } q^{\pi_0}\} \\
& (t^{\pi_0}\, (t^h\, s), \lambda p_0.\, \mathsf{snd}\, (t^h\, s \downarrow \mathsf{just}\, (\mathsf{inl}\, (p_0, \mathsf{nothing})))) \\
= \quad & \{\text{definition of } \downarrow\} \\
& (t^{\pi_0}\, (t^h\, s), \lambda p_0.\, \mathsf{snd}\, (t^h\, s \downarrow^+ \mathsf{inl}\, (p_0, \mathsf{nothing}))) \\
= \quad & \{\text{definition of } \downarrow^+\} \\
& (t^{\pi_0}\, (t^h\, s), \lambda p_0.\, \mathsf{snd}\, (\mathsf{fst}\, (t^h\, s)\ \overline{\downarrow_0^+}\ (p_0, \mathsf{nothing}))) \\
= \quad & \{\text{definition of } \overline{\downarrow_0^+}\} \\
& (t^{\pi_0}\, (t^h\, s), \lambda p_0.\, \mathsf{snd}\, (\mathsf{fst}\, (t^h\, s))\, p_0) \\
= \quad & \{\text{definition of } t^{\pi_0}\} \\
& \mathsf{fst}\, (t^h\, s)
\end{aligned}
$$

Statement for shapes, i.e., $t^f = t^h$:

$$
\begin{aligned}
& t^f\, s \\
= \quad & \{\text{definition of } t^f\} \\
& (\overline{t^{f_0}}\, s, \overline{t^{f_1}}\, s) \\
= \quad & \{\text{aux. statements}\} \\
& t^h\, s
\end{aligned}
$$

Auxiliary statements $\overline{q^{f_0}}\, p = q^h\, (\mathsf{just}\, (\mathsf{inl}\, p))$ and $\overline{q^{f_1}}\, p = q^h\, (\mathsf{just}\, (\mathsf{inr}\, p))$ for positions, by mutual induction on the ps, showing the cases of the first statement. Case $p = (p_0, \mathsf{nothing})$:

$$
\begin{aligned}
& \overline{q^{f_0}}\, (p_0, \mathsf{nothing}))) \\
= \quad & \{\text{definition of } \overline{q^{f_0}}\} \\
& q^{f_0}\, (\mathsf{just}\, p_0) \\
= \quad & \{\text{assumption}\} \\
& q^h\, (q^{\pi_0}\, (\mathsf{just}\, p_0)) \\
= \quad & \{\text{definition of } q^{\pi_0}\} \\
& q^h\, (\mathsf{just}\, (\mathsf{inl}\, (p_0, \mathsf{nothing})))
\end{aligned}
$$

Case $p = (p_0, \mathsf{just}\, p_1)$:

$$\begin{array}{ll} & \overline{q^{f_0}}\,(p_0, \mathsf{just}\, p_1))) \\ = & \quad \{\text{definition of } \overline{q^{f_0}}\} \\ & q^{f_0}\,(\mathsf{just}\, p_0) \oplus' \overline{q^{f_1}}\, p_1 \\ = & \quad \{\text{assumption}\} \\ & q^h\,(q^{\pi_0}\,(\mathsf{just}\, p_0))) \oplus' \overline{q^{f_1}}\, p_1 \\ = & \quad \{\text{inductive hypothesis for } p_1\} \\ & q^h\,(q^{\pi_0}\,(\mathsf{just}\, p_0))) \oplus' q^h\,(\mathsf{just}\,(\mathsf{inr}\, p_1)) \\ = & \quad \{\text{definition of } q^{\pi_0}\} \\ & q^h\,(\mathsf{just}\,(\mathsf{inl}\,(p_0, \mathsf{nothing}))) \oplus' q^h\,(\mathsf{just}\,(\mathsf{inr}\, p_1)) \\ = & \quad \{\text{directed container morphism law 3}\} \\ & q^h\,(\mathsf{just}\,(\mathsf{inl}\,(p_0, \mathsf{nothing})) \oplus \mathsf{just}\,(\mathsf{inr}\, p_1)) \\ = & \quad \{\text{definition of } \oplus\} \\ & q^h\,(\mathsf{just}\,(\mathsf{inl}\,(p_0, \mathsf{nothing}) \oplus^+ \mathsf{inr}\, p_1)) \\ = & \quad \{\text{definition of } \oplus^+\} \\ & q^h\,(\mathsf{just}\,(\mathsf{inl}\,((p_0, \mathsf{nothing}) \overline{\oplus_0^+}\ \mathsf{inr}\, p_1))) \\ = & \quad \{\text{definition of } \overline{\oplus_0^+}\} \\ & q^h\,(\mathsf{just}\,(\mathsf{inl}\,(p_0, \mathsf{just}\, p_1))) \end{array}$$

Statement for positions, i.e., $q^f = q^h$. Case $p = \mathsf{nothing}$:

$$\begin{array}{ll} & q^f\,\mathsf{nothing} \\ = & \quad \{\text{definition of } q^f\} \\ & \mathsf{o}' \\ = & \quad \{\text{directed container morphism law 2}\} \\ & q^h\,\mathsf{o} \\ = & \quad \{\text{definition of } \mathsf{o}\} \\ & q^h\,\mathsf{nothing} \end{array}$$

Case $p = \mathsf{just}\,(\mathsf{inl}\, p)$:

$$\begin{array}{ll} & q^f\,(\mathsf{just}\,(\mathsf{inl}\, p) \\ = & \quad \{\text{definition of } q^f\} \\ & \overline{q^{f_0}}\, p \\ = & \quad \{\text{first aux. statement}\} \\ & q^h\,(\mathsf{just}\,(\mathsf{inl}\, p)) \end{array}$$

Case $p = \mathsf{just}\,(\mathsf{inr}\, p)$ is symmetric. □

Proof of Proposition 4.5.

We must prove that $E = (S \lhd P, \downarrow, \mathsf{o}, \oplus)$ is a cofree directed container on the container $C_0 = S_0 \lhd P_0$.

Lemma B.6. *The data $\downarrow$, o, $\oplus$ provide a directed container structure on the container $C = S \lhd P$.*

Proof. Directed container law 1:

$$\begin{array}{ll} & (s, v) \downarrow \mathsf{o}\,\{s, v\} \\ = & \quad \{\text{definition of } \mathsf{o}\} \\ & (s, v) \downarrow (\mathsf{inl}\,*) \\ = & \quad \{\text{definition of } \downarrow\} \\ & (s, v) \end{array}$$

Directed container law 2 by induction on p. Case $p = \mathsf{inl}\,*$:

$$\begin{array}{ll} & (s, v) \downarrow (\mathsf{inl}\,* \oplus p') \\ = & \quad \{\text{definition of } \oplus\} \\ & (s, v) \downarrow p' \\ = & \quad \{\text{definition of } \downarrow\} \\ & ((s, v) \downarrow \mathsf{inl}\,*) \downarrow p' \end{array}$$

Case $p = \mathsf{inr}\,(p, p')$:

$$\begin{array}{ll} & (s, v) \downarrow (\mathsf{inr}\,(p, p') \oplus p'') \\ = & \quad \{\text{definition of } \oplus\} \\ & (s, v) \downarrow (\mathsf{inr}\,(p, p' \oplus p'')) \\ = & \quad \{\text{definition of } \downarrow\} \\ & v\,p \downarrow (p' \oplus p'') \\ = & \quad \{\text{inductive hypothesis for } p'\} \\ & (v\,p \downarrow p') \downarrow p'' \\ = & \quad \{\text{definition of } \downarrow\} \\ & ((s, v) \downarrow \mathsf{inr}\,(p, p')) \downarrow p'' \end{array}$$

Directed container law 3 by induction on p. Case $p = \mathsf{inl}\,*$:

$$\begin{array}{ll} & \mathsf{inl}\,* \oplus \mathsf{o}\,\{(s, v) \downarrow \mathsf{inl}\,*\} \\ = & \quad \{\text{definitions of } \oplus, \downarrow\} \\ & \mathsf{o}\,\{s, v\} \\ = & \quad \{\text{definition of } \mathsf{o}\} \\ & \mathsf{inl}\,* \end{array}$$

Case $p = \mathsf{inr}\,(p, p')$:

$$\begin{array}{ll} & \mathsf{inr}\,(p, p') \oplus \mathsf{o}\,\{(s, v) \downarrow \mathsf{inr}\,(p, p')\} \\ = & \quad \{\text{definitions of } \oplus, \downarrow\} \\ & \mathsf{inr}\,(p, p' \oplus \mathsf{o}\,\{v\,p \downarrow p'\}) \\ = & \quad \{\text{inductive hypothesis for } p'\} \\ & \mathsf{inr}\,(p, p') \end{array}$$

Directed container law 4.

$$\begin{array}{ll} & \mathsf{o}\,\{s, v\} \oplus p \\ = & \quad \{\text{definition of } \mathsf{o}\} \\ & \mathsf{inl}\,* \oplus p \\ = & \quad \{\text{definition of } \oplus\} \\ & p \end{array}$$

Directed container law 5 by induction on p. Case $p = \mathsf{inl}\,*$:

$$\begin{array}{ll} & (\mathsf{inl}\,* \oplus p') \oplus p'' \\ = & \quad \{\text{definition of } \oplus\} \\ & p' \oplus p'' \\ = & \quad \{\text{definition of } \oplus\} \\ & \mathsf{inl}\,* \oplus (p' \oplus p'') \end{array}$$

Case $p = \mathsf{inr}\,(p, p')$:

$$\begin{array}{ll} & (\mathsf{inr}\,(p, p') \oplus p'') \oplus p''' \\ = & \quad \{\text{definition of } \oplus\} \\ & \mathsf{inr}\,(p, p' \oplus p'') \oplus p''' \\ = & \quad \{\text{definition of } \oplus\} \\ & \mathsf{inr}\,(p, (p' \oplus p'') \oplus p''') \\ = & \quad \{\text{inductive hypothesis for } p'\} \\ & \mathsf{inr}\,(p, p' \oplus (p'' \oplus p''')) \\ = & \quad \{\text{definition of } \oplus\} \\ & \mathsf{inr}\,(p, p') \oplus (p'' \oplus p''') \end{array}$$

□

That the directed container $E = (S \lhd P, \downarrow, \mathsf{o}, \oplus)$ is cofree on the container $C_0 = S_0 \lhd P_0$ can be shown either directly or by proving that it interprets into a cofree comonad on $[\![C_0]\!]^{\mathrm{c}}$. In the following, we illustrate the first route. This involves a fair amount of straightforward, but tedious inductive and coinductive reasoning in the lemmas below.

For the directed container E to be cofree on the container C_0, there must be a container morphism $\pi = t^\pi \lhd q^\pi : S \lhd P \to S_0 \lhd P_0$. This is defined by

- $t^\pi : S \to S_0$
  $t^\pi\,(s, v) = s$
- $q^\pi : \Pi\{(s, v) : S\}.\, P_0\, s \to P\,(s, v)$
  $q^\pi\, p = \mathsf{inr}\,(p, \mathsf{inl}\,*)$

The universal property of cofreeness states that, for any other directed container $E' = (S' \lhd P', \downarrow', \mathsf{o}', \oplus')$ and container morphism $f_0 = t^{f_0} \lhd q^{f_0} : S' \lhd P' \to S_0 \lhd P_0$, there must exist a unique directed container morphism $f = t^f \lhd q^f : E' \to E$ such that the following triangle commutes:

$$\begin{array}{ccc} S' \lhd P' & & \\ \Big\downarrow{\scriptstyle f} & \searrow^{f_0} & \\ S \lhd P & \xrightarrow{\pi} & S_0 \lhd P_0 \end{array} \qquad (\ddagger)$$

We claim that this directed container morphism f is given by

- $t^f : S' \to S$
  (by corecursion)
  $t^f\, s = (t^{f_0}\, s, \lambda p.\, t^f\,(s \downarrow' q^{f_0}\, p))$

- $q^f : \Pi\{s : S'\}.\, P\,(t^{f_0}\, s, \lambda p.\, t^f\,(s \downarrow' q^{f_0}\, p)) \to P'\, s$
  (by recursion)
  $q^f\,(\mathsf{inl}\, *) = \mathsf{o}'$
  $q^f\,(\mathsf{inr}\,(p, p')) = q^{f_0} p \oplus' q^f\, p$

and prove it with the lemmas below.

Lemma B.7. *The container morphism f is a directed container morphism.*

Proof. Directed container morphism law 1 by induction on p. Case $p = \mathsf{inl}\, *$:

$$\begin{array}{ll} & t^f\,(s \downarrow' q^f\,(\mathsf{inl}\, *)) \\ = & \quad \{\text{definition of } q^f\} \\ & t^f\,(s \downarrow' \mathsf{o}') \\ = & \quad \{\text{directed container law 1}\} \\ & t^f\, s \\ = & \quad \{\text{definition of } \downarrow\} \\ & t^f\, s \downarrow \mathsf{inl}\, * \end{array}$$

Case $p = \mathsf{inr}\,(p, p')$:

$$\begin{array}{ll} & t^f\,(s \downarrow' q^f\,(\mathsf{inr}\,(p, p'))) \\ = & \quad \{\text{definition of } q^f\} \\ & t^f\,(s \downarrow' (q^{f_0}\, p \oplus' q^f\, p)) \\ = & \quad \{\text{directed container law 2}\} \\ & t^f\,((s \downarrow' q^{f_0}\, p) \downarrow' q^f\, p') \\ = & \quad \{\text{inductive hypothesis for } p'\} \\ & t^f\,(s \downarrow' q^{f_0}\, p) \downarrow p' \\ = & \quad \{\text{definition of } \downarrow\} \\ & (t^{f_0}\, s, \lambda p.\, t^f\,(s \downarrow' q^{f_0}\, p)) \downarrow \mathsf{inr}\,(p, p') \\ = & \quad \{\text{definition of } t^f\} \\ & t^f\, s \downarrow \mathsf{inr}\,(p, p') \end{array}$$

Directed container morphism law 2.

$$\begin{array}{ll} & q^f\, \mathsf{o} \\ = & \quad \{\text{definition of } \mathsf{o}\} \\ & q^f\,(\mathsf{inl}\, *) \\ = & \quad \{\text{definition of } q^f\} \\ & \mathsf{o}' \end{array}$$

Directed container morphism law 3 by induction on p. Case $p = \mathsf{inl}\, *$:

$$\begin{array}{ll} & q^f\,(\mathsf{inl}\, * \oplus p') \\ = & \quad \{\text{definition of } \oplus\} \\ & q^f\, p' \\ = & \quad \{\text{directed container law 4}\} \\ & \mathsf{o}' \oplus' q^f\, p' \\ = & \quad \{\text{definition of } q^f\} \\ & q^f\,(\mathsf{inl}\, *) \oplus' q^f\, p' \end{array}$$

Case $p = \mathsf{inr}\,(p, p')$:

$$\begin{array}{ll} & q^f\,(\mathsf{inr}\,(p,p') \oplus p'') \\ = & \quad \{\text{definition of } \oplus\} \\ & q^f\,(\mathsf{inr}\,(p, p' \oplus' p'')) \\ = & \quad \{\text{definition of } q^f\} \\ & q^{f_0}\,p \oplus' q^f\,(p' \oplus' p'') \\ = & \quad \{\text{inductive hypothesis for } p'\} \\ & q^{f_0}\,p \oplus' (q^f\,p' \oplus' q^f\,p'') \\ = & \quad \{\text{directed container law 5}\} \\ & (q^{f_0}\,p \oplus' q^f\,p') \oplus' q^f\,p'' \\ = & \quad \{\text{definition of } q^f\} \\ & q^f\,(\mathsf{inr}\,(p,p')) \oplus' q^f\,p'' \end{array}$$

□

Lemma B.8. *The cofreeness triangle* (‡) *commutes, i.e.,* $\pi \circ f = f_0$.

Proof. Statement for shapes, i.e., $t^\pi \circ t^f = t^{f_0}$:

$$\begin{array}{ll} & t^\pi\,(t^f\,s) \\ = & \quad \{\text{definition of } t^f\} \\ & t^\pi\,(t^{f_0}\,s, \lambda p.\, t^f\,(s \downarrow' q^{f_0}\,p)) \\ = & \quad \{\text{definition of } t^\pi\} \\ & t^{f_0}\,s \end{array}$$

Statement for positions, i.e., $q^f \circ q^\pi = q^{f_0}$:

$$\begin{array}{ll} & q^f\,(q^\pi\,p) \\ = & \quad \{\text{definition of } q^\pi\} \\ & q^f\,(\mathsf{inr}\,(p, \mathsf{inl}\,*)) \\ = & \quad \{\text{definition of } q^f\} \\ & q^{f_0}\,p \oplus' q^f\,(\mathsf{inl}\,*) \\ = & \quad \{\text{definition of } q^f\} \\ & q^{f_0}\,p \oplus' \mathsf{o}' \\ = & \quad \{\text{directed container law 3}\} \\ & q^{f_0}\,p \end{array}$$

□

Lemma B.9. *The directed container morphism f is unique, i.e., if there is a directed container morphism $h = t^h \lhd q^h : E' \to E$ such that $\pi \circ h = f_0$, then $f = h$.*

Proof. Statement for shapes, i.e., $t^f = t^h$, by coinduction.

$$
\begin{array}{ll}
 & t^f\ s \\
= & \quad \{\text{definition of } t^f\} \\
 & (t^{f_0}\ s, \lambda p.\, t^f\ (s \downarrow' q^{f_0}\ p)) \\
= & \quad \{\text{coinductive hypothesis}\} \\
 & (t^{f_0}\ s, \lambda p.\, t^h\ (s \downarrow' q^{f_0}\ p)) \\
= & \quad \{\text{assumption, i.e., } t^\pi \circ t^h = t^{f_0} \text{ and } q^h \circ q^\pi = q^{f_0}\} \\
 & (t^\pi\ (t^h\ s), \lambda p.\, t^h\ (s \downarrow' q^h\ (q^\pi\ p))) \\
= & \quad \{\text{directed container morphism law 1}\} \\
 & (t^\pi\ (t^h\ s), \lambda p.\, t^h\ s \downarrow q^\pi p) \\
= & \quad \{\text{definitions of } t^\pi,\ q^\pi\} \\
 & (\mathsf{fst}\ (t^h\ s), \lambda p.\, t^h\ s \downarrow \mathsf{inr}\ (p, \mathsf{inl}\ *)) \\
= & \quad \{\text{definition of } \downarrow\} \\
 & (\mathsf{fst}\ (t^h\ s), \lambda p.\, \mathsf{snd}\ (t^h\ s)\ p \downarrow \mathsf{inl}\ *) \\
= & \quad \{\text{definition of } \downarrow\} \\
 & t^h\ s
\end{array}
$$

Statement for positions, i.e., $q^f = q^h$, by induction on position p. Case $p = \mathsf{inl}\ *$:

$$
\begin{array}{ll}
 & q^f\ (\mathsf{inl}\ *) \\
= & \quad \{\text{definition of } q^f\} \\
 & \mathsf{o}' \\
= & \quad \{\text{directed container morphism law 2}\} \\
 & q^h\ \mathsf{o} \\
= & \quad \{\text{definition of } \mathsf{o}\} \\
 & q^h\ (\mathsf{inl}\ *)
\end{array}
$$

Case $p = \mathsf{inr}\ (p, p')$:

$$
\begin{array}{ll}
 & q^f\ (\mathsf{inr}\ (p, p')) \\
= & \quad \{\text{definition of } q^f\} \\
 & q^{f_0}\ p \oplus' q^f\ p' \\
= & \quad \{\text{inductive hypothesis for } p'\} \\
 & q^{f_0}\ p \oplus' q^h\ p' \\
= & \quad \{\text{assumption for positions, i.e., } q^h \circ q^\pi = q^{f_0}\} \\
 & q^h\ (q^\pi\ p) \oplus' q^h\ p' \\
= & \quad \{\text{directed container morphism law 3}\} \\
 & q^h\ (q^\pi\ p \oplus p') \\
= & \quad \{\text{definition of } q^\pi\} \\
 & q^h\ (\mathsf{inr}\ (p, \mathsf{inl}\ *) \oplus p') \\
= & \quad \{\text{definition of } \oplus\} \\
 & q^h\ (\mathsf{inr}\ (p, \mathsf{inl}\ * \oplus p') \\
= & \quad \{\text{definition of } \oplus\} \\
 & q^h\ (\mathsf{inr}\ (p, p'))
\end{array}
$$

□