

Directed Containers

Danel Ahman

(based on joint work with James Chapman and Tarmo Uustalu)



Ljubljana, 11 October 2018

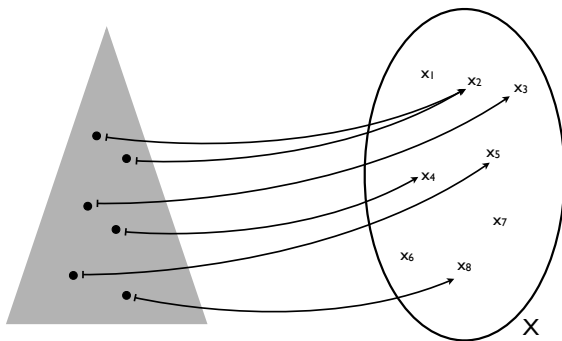
Today's plan

- Directed containers
 - type-theoretic and polynomial presentations
 - their use in functional programming
 - why are they canonical such structure?
- Some constructions on directed containers (see more in papers)
 - coproducts of directed containers
 - strict directed containers and their products
 - focussing a container
- Directed containers and computational effects
- Directed containers and \mathbf{BX}
- Directed containers and categories

Prelude

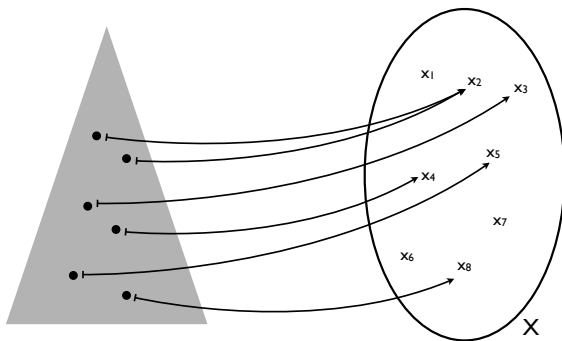
Container syntax of datatypes

- Many **datatypes** can be represented in terms of
 - **shapes** and
 - **positions** in shapes



Container syntax of datatypes

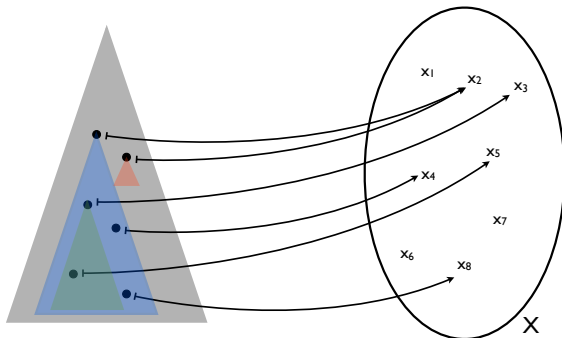
- Many **datatypes** can be represented in terms of
 - **shapes** and
 - **positions** in shapes



- **Examples:** lists, streams, trees, zippers, ...
- **Containers** provide us with a handy **syntax** to analyse them

Directing containers?

- Containers often exhibit a natural notion of **subshape**



- Natural questions arise:
 - What is the appropriate **specialisation** of containers?
 - Does this admit a nice **categorical** theory?
 - What else is this structure **useful** for?

Directed containers

Directed containers

- A directed container is given by

- $S : \mathbf{Set}$ *(shapes)*
- $P : S \rightarrow \mathbf{Set}$ *(positions)*

and

- $\downarrow : \prod s : S. P s \rightarrow S$ *(subshape)*
- $\circ : \prod \{s : S\}. P s$ *(root position)*
- $\oplus : \prod \{s : S\}. \prod p : P s. P (s \downarrow p) \rightarrow P s$ *(subshape positions)*

such that

- $s \downarrow \circ = s$
- $s \downarrow (p \oplus p') = (s \downarrow p) \downarrow p'$
- $p \oplus \{s\} \circ = p$
- $\circ \{s\} \oplus p = p$
- $(p \oplus \{s\} p') \oplus p'' = p \oplus (p' \oplus p'')$

Directed containers

- A **directed container** is given by
 - $S : \mathbf{Set}$ *(shapes)*
 - $P : S \rightarrow \mathbf{Set}$ *(positions)*

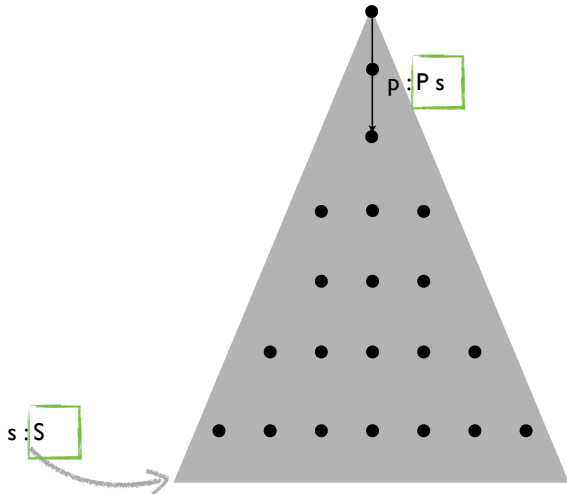
and

- $\downarrow : \prod s : S. P s \rightarrow S$ *(subshape)*
- $\circ : \prod \{s : S\}. P s$ *(root position)*
- $\oplus : \prod \{s : S\}. \prod p : P s. P (s \downarrow p) \rightarrow P s$ *(subshape positions)*

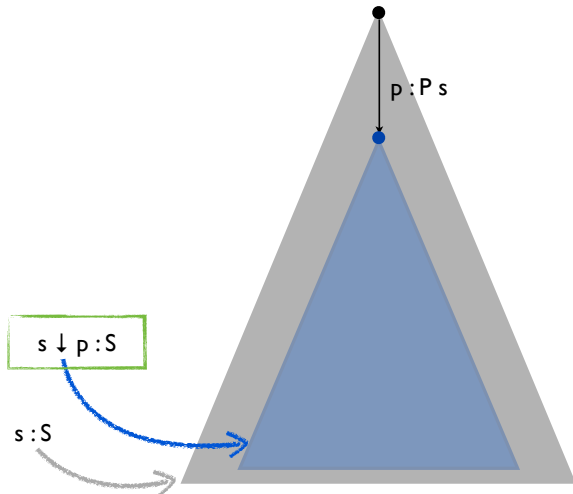
such that

- $s \downarrow \circ = s$
- $s \downarrow (p \oplus p') = (s \downarrow p) \downarrow p'$
- $p \oplus_{\{s\}} \circ = p$
- $\circ_{\{s\}} \oplus p = p$
- $(p \oplus_{\{s\}} p') \oplus p'' = p \oplus (p' \oplus p'')$

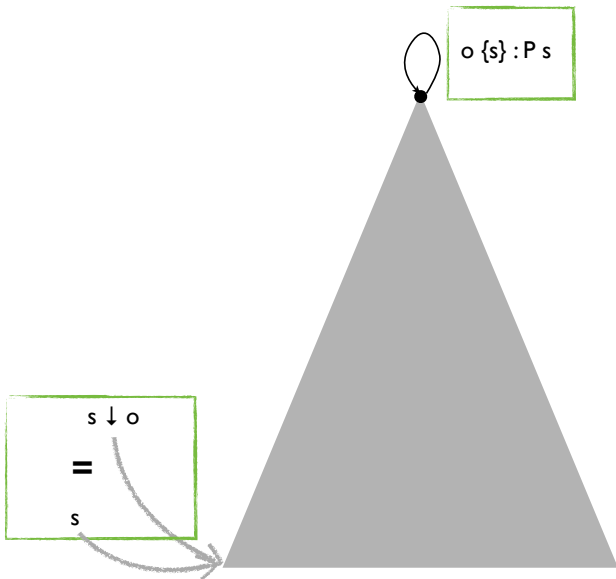
Directed containers illustrated



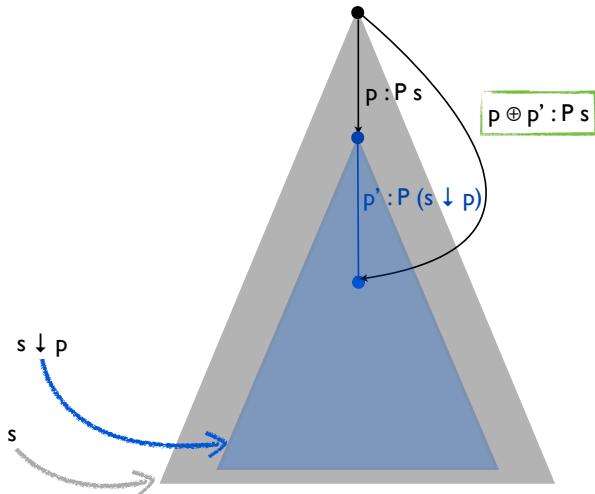
Directed containers illustrated



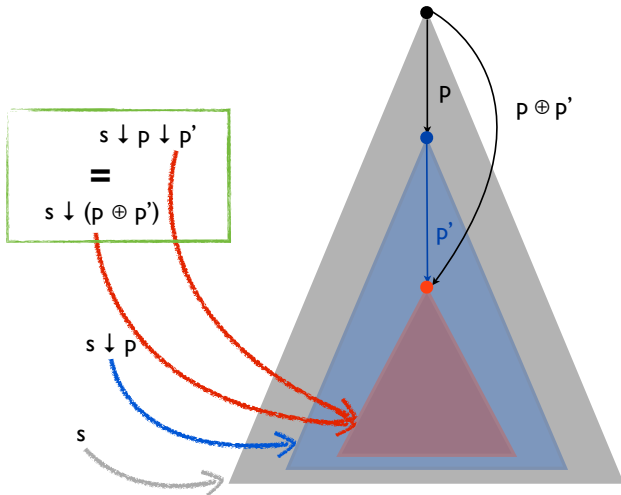
Directed containers illustrated



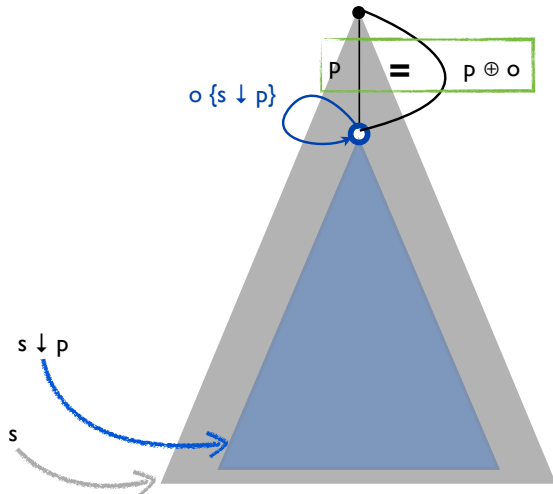
Directed containers illustrated



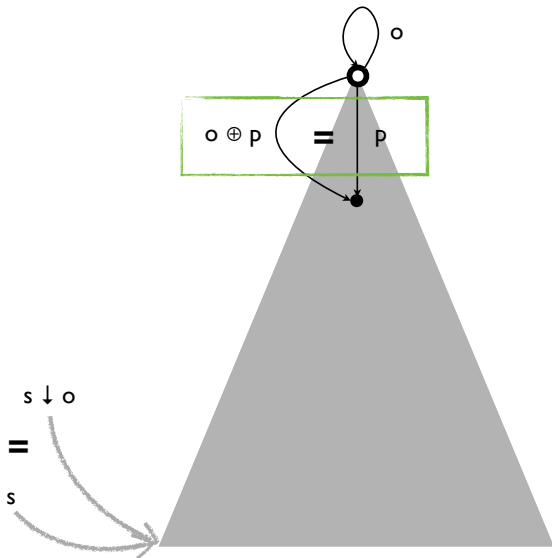
Directed containers illustrated



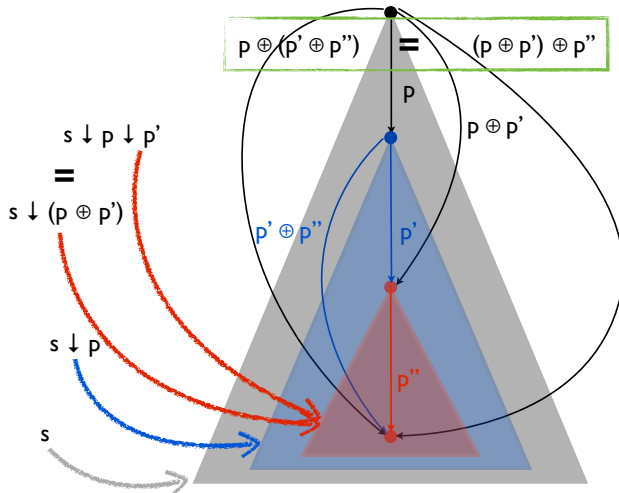
Directed containers illustrated



Directed containers illustrated



Directed containers illustrated



Directed containers (recap)

- A **directed container** is given by
 - $S : \mathbf{Set}$ *(shapes)*
 - $P : S \rightarrow \mathbf{Set}$ *(positions)*

and

- $\downarrow : \prod s : S. P s \rightarrow S$ *(subshape)*
- $\circ : \prod \{s : S\}. P s$ *(root position)*
- $\oplus : \prod \{s : S\}. \prod p : P s. P (s \downarrow p) \rightarrow P s$ *(subshape positions)*

such that

- $s \downarrow \circ = s$
- $s \downarrow (p \oplus p') = (s \downarrow p) \downarrow p'$
- $p \oplus_{\{s\}} \circ = p$
- $\circ_{\{s\}} \oplus p = p$
- $(p \oplus_{\{s\}} p') \oplus p'' = p \oplus (p' \oplus p'')$

Examples: non-empty lists and streams

- Non-empty lists are represented as
 - $S \stackrel{\text{def}}{=} \text{Nat}$ *(shapes)*
 - $P\ n \stackrel{\text{def}}{=} \text{Fin}(n+1) = \{0, \dots, n\}$ *(positions)*
 - $n \downarrow m \stackrel{\text{def}}{=} n - m$ *(subshapes)*
 - $\text{o}_{\{n\}} \stackrel{\text{def}}{=} 0$ *(root position)*
 - $m \oplus_{\{n\}} m' \stackrel{\text{def}}{=} m + m'$ *(subshape positions)*

Examples: non-empty lists and streams

- Non-empty lists are represented as
 - $S \stackrel{\text{def}}{=} \text{Nat}$ *(shapes)*
 - $P n \stackrel{\text{def}}{=} \text{Fin}(n + 1) = \{0, \dots, n\}$ *(positions)*
 - $n \downarrow m \stackrel{\text{def}}{=} n - m$ *(subshapes)*
 - $\text{o}_{\{n\}} \stackrel{\text{def}}{=} 0$ *(root position)*
 - $m \oplus_{\{n\}} m' \stackrel{\text{def}}{=} m + m'$ *(subshape positions)*
- Another example is non-empty lists with **cyclic shifts**

Examples: non-empty lists and streams

- Non-empty lists are represented as
 - $S \stackrel{\text{def}}{=} \text{Nat}$ *(shapes)*
 - $P\ n \stackrel{\text{def}}{=} \text{Fin}(n+1) = \{0, \dots, n\}$ *(positions)*
 - $n \downarrow m \stackrel{\text{def}}{=} n - m$ *(subshapes)*
 - $\text{o}_{\{n\}} \stackrel{\text{def}}{=} 0$ *(root position)*
 - $m \oplus_{\{n\}} m' \stackrel{\text{def}}{=} m + m'$ *(subshape positions)*
- Another example is non-empty lists with **cyclic shifts**
- **Streams** are represented similarly
 - $S \stackrel{\text{def}}{=} 1$ *(shapes)*
 - $P \ast \stackrel{\text{def}}{=} \text{Nat}$ *(positions)*
 - ...

Examples: non-empty lists with a focus

- Zippers – tree-like data-structures consisting of
 - a context and a focal subtree

Examples: non-empty lists with a focus

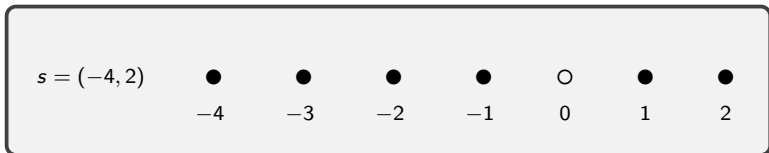
- **Zipper** – tree-like data-structures consisting of

- a **context** and a **focal subtree**

- Non-empty lists with a **focus**

- $S \stackrel{\text{def}}{=} \text{Nat} \times \text{Nat}$ *(shapes)*

- $P(n_0, n_1) \stackrel{\text{def}}{=} \{-n_0, \dots, n_1\} = \{-n_0, \dots, -1\} \cup \{0, \dots, n_1\}$ *(pos.)*



- $(n_0, n_1) \downarrow m \stackrel{\text{def}}{=} (n_0 + m, n_1 - m)$ *(subshapes)*

- $\circ_{\{n_0, n_1\}} \stackrel{\text{def}}{=} 0$ *(root)*

- $m \oplus_{\{n_0, n_1\}} m' \stackrel{\text{def}}{=} m + m'$ *(subshape positions)*

Directed polynomials

Directed **polynomials**

- A **polynomial** (in one variable) is given by

$$1 \xleftarrow{!} \overline{P} \xrightarrow{s} S \xrightarrow{!} 1$$

where

- $S : \mathbf{Set}$ *(shapes)*
- $\overline{P} : \mathbf{Set}$ *(total positions)*
- Polynomials correspond to **containers** via $\overline{P} \cong \sum s : S. P s$

Directed polynomials

- A **polynomial** (in one variable) is given by

$$1 \xleftarrow{!} \overline{P} \xrightarrow{s} S \xrightarrow{!} 1$$

where

- $S : \mathbf{Set}$ *(shapes)*
- $\overline{P} : \mathbf{Set}$ *(total positions)*
- Polynomials correspond to **containers** via $\overline{P} \cong \Sigma s : S. P s$
- A **directed polynomial** is given by
 - $s : \overline{P} \rightarrow S$ *(a polynomial)*
 - $\downarrow : \overline{P} \rightarrow S$
 - $o : S \rightarrow \overline{P}$ s.t. $s \circ o = \text{id}_S$ and $\downarrow \circ o = \text{id}_S$
 - ...

Directed polynomials

- A **polynomial** (in one variable) is given by

$$1 \xleftarrow{!} \overline{P} \xrightarrow{s} S \xrightarrow{!} 1$$

where

- $S : \mathbf{Set}$ *(shapes)*
- $\overline{P} : \mathbf{Set}$ *(total positions)*
- Polynomials correspond to **containers** via $\overline{P} \cong \Sigma s : S. P s$
- A **directed polynomial** is given by
 - $s : \overline{P} \rightarrow S$ *(a polynomial)*
 - $\downarrow : \overline{P} \rightarrow S$
 - $o : S \rightarrow \overline{P}$ s.t. $s \circ o = \text{id}_S$ and $\downarrow \circ o = \text{id}_S$
 - ...
 - this definition is remarkably **symmetric in s and \downarrow**

Directed container morphisms

- A directed container morphism

$$t \triangleleft q : (S \triangleleft P, \downarrow, o, \oplus) \longrightarrow (S' \triangleleft P', \downarrow', o', \oplus')$$

is given by

- $t : S \rightarrow S'$
- $q : \prod\{s : S\}. P' (t s) \rightarrow P s$ (note the direction!)

such that

- $t (s \downarrow q p) = t s \downarrow' p$
- $o_{\{s\}} = q (o'_{\{t s\}})$
- $q p \oplus_{\{s\}} q p' = q (p \oplus'_{\{t s\}} p')$
- Identities and composition are defined component-wise
- Directed containers form a category **DCont**

Directed container morphisms

- A directed container morphism

$$t \triangleleft q : (S \triangleleft P, \downarrow, o, \oplus) \longrightarrow (S' \triangleleft P', \downarrow', o', \oplus')$$

is given by

- $t : S \rightarrow S'$
- $q : \Pi\{s : S\}. P' (t s) \rightarrow P s$

such that

- $t (s \downarrow q p) = t s \downarrow' p$
 - $o_{\{s\}} = q (o'_{\{t s\}})$
 - $q p \oplus_{\{s\}} q p' = q (p \oplus'_{\{t s\}} p')$
- **Identities** and **composition** are defined component-wise
 - Directed containers form a **category** **DCont**

Directed containers
=
containers \cap comonads

Interpretation of directed containers

- Any directed container

$$(S \triangleleft P, \downarrow, \circ, \oplus)$$

defines a functor/comonad

$$[[S \triangleleft P, \downarrow, \circ, \oplus]]^{\text{dc}} \stackrel{\text{def}}{=} (D, \varepsilon, \delta)$$

where

- $D : \mathbf{Set} \longrightarrow \mathbf{Set}$

$$D X \stackrel{\text{def}}{=} \Sigma s : S. (P s \rightarrow X)$$

- $\varepsilon_X : D X \longrightarrow X$

$$\varepsilon_X(s, v) \stackrel{\text{def}}{=} v(o_{\{s\}})$$

- $\delta_X : D X \longrightarrow D D X$

$$\delta_X(s, v) \stackrel{\text{def}}{=} (s, \lambda p. (s \downarrow p, \lambda p'. v(p \oplus_{\{s\}} p')))$$

Interpretation of directed containers

- Any directed container

$$(S \triangleleft P, \downarrow, \circ, \oplus)$$

defines a **functor**/comonad

$$\llbracket S \triangleleft P, \downarrow, \circ, \oplus \rrbracket^{\text{dc}} \stackrel{\text{def}}{=} (D, \varepsilon, \delta)$$

where

- $D : \mathbf{Set} \longrightarrow \mathbf{Set}$

$$D X \stackrel{\text{def}}{=} \Sigma s : S. (P s \rightarrow X)$$

- $\varepsilon_X : D X \longrightarrow X$

$$\varepsilon_X(s, v) \stackrel{\text{def}}{=} v(o_{\{s\}})$$

- $\delta_X : D X \longrightarrow D D X$

$$\delta_X(s, v) \stackrel{\text{def}}{=} (s, \lambda p. (s \downarrow p, \lambda p'. v(p \oplus_{\{s\}} p')))$$

Interpretation of dcon. morphisms

- Any directed container morphism

$$t \triangleleft q : (S \triangleleft P, \downarrow, \circ, \oplus) \longrightarrow (S' \triangleleft P', \downarrow', \circ', \oplus')$$

defines a natural transformation / comonad morphism

$$\llbracket t \triangleleft q \rrbracket^c : \llbracket S \triangleleft P, \downarrow, \circ, \oplus \rrbracket^c \longrightarrow \llbracket S' \triangleleft P', \downarrow', \circ', \oplus' \rrbracket^c$$

by

- $\llbracket t \triangleleft q \rrbracket_X^c : \Sigma s : S. (P s \rightarrow X) \longrightarrow \Sigma s' : S'. (P' s' \rightarrow X)$

$$\llbracket t \triangleleft q \rrbracket_X^c (s, v) \stackrel{\text{def}}{=} (t s, v \circ q_{\{s\}})$$

- $\llbracket - \rrbracket^c$ preserves the identities and composition
- $\llbracket - \rrbracket^c$ is a functor from **DCont** to **[Set, Set]** / *Comonads(Set)*

Interpretation of dcon. morphisms

- Any directed container morphism

$$t \triangleleft q : (S \triangleleft P, \downarrow, o, \oplus) \longrightarrow (S' \triangleleft P', \downarrow', o', \oplus')$$

defines a ~~natural transformation~~/comonad morphism

$$\llbracket t \triangleleft q \rrbracket^{\text{dc}} : \llbracket S \triangleleft P, \downarrow, o, \oplus \rrbracket^{\text{dc}} \longrightarrow \llbracket S' \triangleleft P', \downarrow', o', \oplus' \rrbracket^{\text{dc}}$$

by

- $\llbracket t \triangleleft q \rrbracket_X^{\text{dc}} : \Sigma s : S. (P s \rightarrow X) \longrightarrow \Sigma s' : S'. (P' s' \rightarrow X)$

$$\llbracket t \triangleleft q \rrbracket_X^{\text{dc}}(s, v) \stackrel{\text{def}}{=} (t s, v \circ q_{\{s\}})$$

- $\llbracket - \rrbracket^{\text{dc}}$ preserves the identities and composition
- $\llbracket - \rrbracket^{\text{dc}}$ is a functor from **DCont** to ~~[Set, Set]~~ **Comonads(Set)**

Interpretation is fully faithful

- Every natural transformation / comonad morphism

$$\tau : \llbracket S \triangleleft P, \downarrow, \circ, \oplus \rrbracket^{\text{dc}} \longrightarrow \llbracket S' \triangleleft P', \downarrow', \circ', \oplus' \rrbracket^{\text{c}}$$

defines a directed container morphism

$$\lceil \tau \rceil^{\text{dc}} : (S \triangleleft P, \downarrow, \circ, \oplus) \longrightarrow (S' \triangleleft P', \downarrow', \circ', \oplus')$$

satisfying

- $\lceil \llbracket t \triangleleft q \rrbracket^{\text{c}} \rceil^{\text{dc}} = t \triangleleft q$
 - $\llbracket \lceil \tau \rceil^{\text{c}} \rrbracket^{\text{dc}} = \tau$
-
- $\llbracket - \rrbracket^{\text{c}}$ is a fully faithful functor

Interpretation is fully faithful

- Every ~~natural transformation~~/comonad morphism

$$\tau : \llbracket S \triangleleft P, \downarrow, \circ, \oplus \rrbracket^{\text{dc}} \longrightarrow \llbracket S' \triangleleft P', \downarrow', \circ', \oplus' \rrbracket^{\text{dc}}$$

defines a directed container morphism

$$\ulcorner \tau \urcorner^{\text{dc}} : (S \triangleleft P, \downarrow, \circ, \oplus) \longrightarrow (S' \triangleleft P', \downarrow', \circ', \oplus')$$

satisfying

- $\ulcorner \llbracket t \triangleleft q \rrbracket^{\text{dc}} \urcorner^{\text{dc}} = t \triangleleft q$
 - $\llbracket \ulcorner \tau \urcorner^{\text{dc}} \rrbracket^{\text{dc}} = \tau$
-
- $\llbracket - \rrbracket^{\text{dc}}$ is a fully faithful functor

Directed containers = cons. \cap cmnds.

- Any **comonad** (D, ε, δ) , such that $D = \llbracket S \triangleleft P \rrbracket^c$, determines

$$\lceil (D, \varepsilon, \delta), S \triangleleft P \rceil \stackrel{\text{def}}{=} (S \triangleleft P, \downarrow, \circ, \oplus)$$

- $\lceil - \rceil$ satisfies

$$\llbracket \lceil (D, \varepsilon, \delta), S \triangleleft P \rceil \rrbracket^{\text{dc}} = (D, \varepsilon, \delta)$$

$$\llbracket \llbracket S \triangleleft P, \downarrow, \circ, \oplus \rrbracket^{\text{dc}}, S \triangleleft P \rceil = (S \triangleleft P, \downarrow, \circ, \oplus)$$

Directed containers = cons. \cap cmnds.

- Any **comonad** (D, ε, δ) , such that $D = \llbracket S \triangleleft P \rrbracket^c$, determines

$$\llbracket (D, \varepsilon, \delta), S \triangleleft P \rrbracket \stackrel{\text{def}}{=} (S \triangleleft P, \downarrow, \circ, \oplus)$$

- $\llbracket - \rrbracket$ satisfies

$$\llbracket \llbracket (D, \varepsilon, \delta), S \triangleleft P \rrbracket \rrbracket^{\text{dc}} = (D, \varepsilon, \delta)$$

$$\llbracket \llbracket S \triangleleft P, \downarrow, \circ, \oplus \rrbracket^{\text{dc}}, S \triangleleft P \rrbracket = (S \triangleleft P, \downarrow, \circ, \oplus)$$

- The following is a **pullback** in **CAT**:

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

Some constructions on directed containers

Coproducts of directed containers

- Given $(S_0 \triangleleft P_0, \downarrow_0, o_0, \oplus_0)$ and $(S_1 \triangleleft P_1, \downarrow_1, o_1, \oplus_1)$, their **coproduct** is $(S \triangleleft P, \downarrow, o, \oplus)$ where
 - $S \triangleleft P \stackrel{\text{def}}{=} (S_0 \triangleleft P_0) + (S_1 \triangleleft P_1) = (S_0 + S_1 \triangleleft [\lambda s. P_0 s, \lambda s. P_1 s])$
 - $\text{inl } s \downarrow p \stackrel{\text{def}}{=} \text{inl } (s \downarrow_0 p)$
 $\text{inr } s \downarrow p \stackrel{\text{def}}{=} \text{inr } (s \downarrow_1 p)$
 - $o_{\{\text{inl } s\}} \stackrel{\text{def}}{=} o_0\{s\}$
 $o_{\{\text{inr } s\}} \stackrel{\text{def}}{=} o_1\{s\}$
 - $p \oplus_{\{\text{inl } s\}} p' \stackrel{\text{def}}{=} p \oplus_0\{s\} p'$
 $p \oplus_{\{\text{inr } s\}} p' \stackrel{\text{def}}{=} p \oplus_1\{s\} p'$

Coproducts of directed containers

- Given $(S_0 \triangleleft P_0, \downarrow_0, \mathbf{o}_0, \oplus_0)$ and $(S_1 \triangleleft P_1, \downarrow_1, \mathbf{o}_1, \oplus_1)$, their **coproduct** is $(S \triangleleft P, \downarrow, \mathbf{o}, \oplus)$ where
 - $S \triangleleft P \stackrel{\text{def}}{=} (S_0 \triangleleft P_0) + (S_1 \triangleleft P_1) = (S_0 + S_1 \triangleleft [\lambda s. P_0 s, \lambda s. P_1 s])$
 - $\text{inl } s \downarrow p \stackrel{\text{def}}{=} \text{inl } (s \downarrow_0 p)$
 $\text{inr } s \downarrow p \stackrel{\text{def}}{=} \text{inr } (s \downarrow_1 p)$
 - $\mathbf{o}_{\{\text{inl } s\}} \stackrel{\text{def}}{=} \mathbf{o}_0\{s\}$
 $\mathbf{o}_{\{\text{inr } s\}} \stackrel{\text{def}}{=} \mathbf{o}_1\{s\}$
 - $p \oplus_{\{\text{inl } s\}} p' \stackrel{\text{def}}{=} p \oplus_0\{s\} p'$
 $p \oplus_{\{\text{inr } s\}} p' \stackrel{\text{def}}{=} p \oplus_1\{s\} p'$
- It interprets as $\llbracket S_0 \triangleleft P_0, \downarrow_0, \mathbf{o}_0, \oplus_0 \rrbracket^{\text{dc}} + \llbracket S_1 \triangleleft P_1, \downarrow_1, \mathbf{o}_1, \oplus_1 \rrbracket^{\text{dc}}$

Products of strict directed containers

- Given $(S_0 \triangleleft P_0, \downarrow_0, o_0, \oplus_0)$ and $(S_1 \triangleleft P_1, \downarrow_1, o_1, \oplus_1)$, there is no general way to endow $(S_0 \triangleleft P_0) \times (S_1 \triangleleft P_1)$ with dcon. struct.

Products of strict directed containers

- Given $(S_0 \triangleleft P_0, \downarrow_0, o_0, \oplus_0)$ and $(S_1 \triangleleft P_1, \downarrow_1, o_1, \oplus_1)$, there is no general way to endow $(S_0 \triangleleft P_0) \times (S_1 \triangleleft P_1)$ with dcon. struct.
- But analogously to (ideal) monads, the product exists for strict directed containers/coideal comonads:
 - $S : \mathbf{Set}$
 - $P^+ : S \rightarrow \mathbf{Set}$
 - $\downarrow^+ : \prod s : S. P^+ s \rightarrow S$
 - $\oplus^+ : \prod \{s : S\}. \prod p : P^+ s. P^+ (s \downarrow^+ p) \rightarrow P^+ s$
 - satisfying two laws (omitted)
- The directed container determined by a strict dcon. has
 - $P s \stackrel{\text{def}}{=} 1 + P^+ s$
 - ...

Products of strict directed containers ctd.

- Now, given $(S_0 \triangleleft P_0^+, \downarrow_0^+, \oplus_0^+)$ and $(S_1 \triangleleft P_1^+, \downarrow_1^+, \oplus_1^+)$, we can define $(S \triangleleft P^+, \downarrow^+, \oplus^+)$ where

- $S \stackrel{\text{def}}{=} \overline{S_0} \times \overline{S_1}$

with

$$(\overline{S_0}, \overline{S_1}) \stackrel{\text{def}}{=} \nu(Z_0, Z_1). (\Sigma s_0 : S_0. P_0^+ s_0 \rightarrow Z_1, \Sigma s_1 : S_1. P_1^+ s_1 \rightarrow Z_0)$$

- $P^+(s_0, s_1) \stackrel{\text{def}}{=} \overline{P_0^+ s_0} + \overline{P_1^+ s_1}$

with

$$\begin{aligned} (\overline{P_0^+ s_0}, \overline{P_1^+ s_1}) \stackrel{\text{def}}{=} \mu(Z_0, Z_1). & (\lambda(s_0, v_0). \Sigma p_0 : P_0^+ s_0. 1 + Z_1 (v_0 p_0), \\ & \lambda(s_1, v_1). \Sigma p_1 : P_1^+ s_1. 1 + Z_0 (v_1 p_1)) \end{aligned}$$

- ...

Products of strict directed containers ctd.

- Now, given $(S_0 \triangleleft P_0^+, \downarrow_0^+, \oplus_0^+)$ and $(S_1 \triangleleft P_1^+, \downarrow_1^+, \oplus_1^+)$, we can define $(S \triangleleft P^+, \downarrow^+, \oplus^+)$ where
 - $S \stackrel{\text{def}}{=} \overline{S_0} \times \overline{S_1}$
with
 $(\overline{S_0}, \overline{S_1}) \stackrel{\text{def}}{=} \nu(Z_0, Z_1). (\Sigma s_0 : S_0. P_0^+ s_0 \rightarrow Z_1, \Sigma s_1 : S_1. P_1^+ s_1 \rightarrow Z_0)$
 - $P^+(s_0, s_1) \stackrel{\text{def}}{=} \overline{P_0^+ s_0} + \overline{P_1^+ s_1}$
with
 $(\overline{P_0^+ s_0}, \overline{P_1^+ s_1}) \stackrel{\text{def}}{=} \mu(Z_0, Z_1). (\lambda(s_0, v_0). \Sigma p_0 : P_0^+ s_0. 1 + Z_1 (v_0 p_0), \lambda(s_1, v_1). \Sigma p_1 : P_1^+ s_1. 1 + Z_0 (v_1 p_1))$
 - ...
- This gives the **product** of the given strict dcons. in **DCont**
- It interprets as the **product** of the corresponding coideal cmds.

Focussing a container

- Given any container $S_0 \triangleleft P_0$, we get $(S \triangleleft P, \downarrow, o, \oplus)$ where
 - $S \stackrel{\text{def}}{=} \Sigma s : S_0.P_0 s$
 - $P(s, p) \stackrel{\text{def}}{=} P_0 s$
 - $(s, p) \downarrow p' \stackrel{\text{def}}{=} (s, p')$
 - $o_{\{s, p\}} \stackrel{\text{def}}{=} p$
 - $p' \oplus_{\{s, p\}} p'' \stackrel{\text{def}}{=} p''$

Focussing a container

- Given any container $S_0 \triangleleft P_0$, we get $(S \triangleleft P, \downarrow, o, \oplus)$ where
 - $S \stackrel{\text{def}}{=} \Sigma s : S_0.P_0 s$
 - $P(s, p) \stackrel{\text{def}}{=} P_0 s$
 - $(s, p) \downarrow p' \stackrel{\text{def}}{=} (s, p')$
 - $o_{\{s, p\}} \stackrel{\text{def}}{=} p$
 - $p' \oplus_{\{s, p\}} p'' \stackrel{\text{def}}{=} p''$
- When positions in P_0 are decidable, then $\llbracket S \triangleleft P, \downarrow, o, \oplus \rrbracket^{\text{dc}}$ is isomorphic to the comonad structure on $\partial \llbracket S_0 \triangleleft P_0 \rrbracket^c \times \text{Id}$
- Focussing forms a functor from **Con**_{cart} to **DCon**

Other constructions on directed containers

- Cofree and cofree recursive directed containers

Other constructions on directed containers

- Cofree and cofree recursive directed containers
- Distributive laws between directed containers
 - $t^\theta \triangleleft q^\theta : (S_0 \triangleleft P_0) \circ^c (S_1 \triangleleft P_1) \longrightarrow (S_1 \triangleleft P_1) \circ^c (S_0 \triangleleft P_0)$
satisfying 11 laws (and with $t_0^\theta(s, v) \stackrel{\text{def}}{=} v(o_{0\{s\}})$ forced)

Other constructions on directed containers

- Cofree and cofree recursive directed containers
- Distributive laws between directed containers
 - $t^\theta \triangleleft q^\theta : (S_0 \triangleleft P_0) \circ^c (S_1 \triangleleft P_1) \longrightarrow (S_1 \triangleleft P_1) \circ^c (S_0 \triangleleft P_0)$
satisfying 11 laws (and with $t_0^\theta(s, v) \stackrel{\text{def}}{=} v(o_{0\{s\}})$ forced)
 - A dep. typed version of the Zappa–Szép product, i.e., of:
 - Given monoid actions $\alpha : N \times M \rightarrow M$ and $\beta : N \times M \rightarrow N$
satisfying two compat. laws, we get a monoid on $M \times N$ with
 $(m_0, n_0) \oplus (m_1, n_1) \stackrel{\text{def}}{=} (m_0 \oplus_M \alpha(n_0, m_1), \beta(n_0, m_1) \oplus_N n_1)$

Other constructions on directed containers

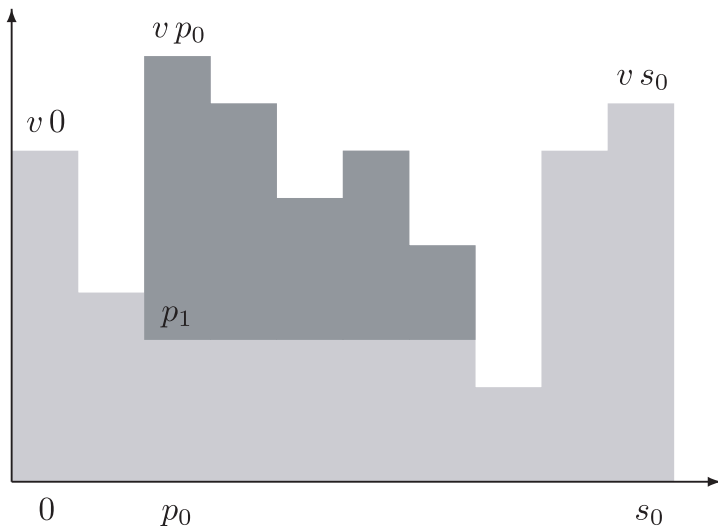
- Cofree and cofree recursive directed containers
- Distributive laws between directed containers
 - $t^\theta \triangleleft q^\theta : (S_0 \triangleleft P_0) \circ^c (S_1 \triangleleft P_1) \longrightarrow (S_1 \triangleleft P_1) \circ^c (S_0 \triangleleft P_0)$
satisfying 11 laws (and with $t_0^\theta(s, v) \stackrel{\text{def}}{=} v(o_{0\{s\}})$ forced)
 - A dep. typed version of the Zappa–Szép product, i.e., of:
 - Given monoid actions $\alpha : N \times M \rightarrow M$ and $\beta : N \times M \rightarrow N$ satisfying two compat. laws, we get a monoid on $M \times N$ with
$$(m_0, n_0) \oplus (m_1, n_1) \stackrel{\text{def}}{=} (m_0 \oplus_M \alpha(n_0, m_1), \beta(n_0, m_1) \oplus_N n_1)$$
- Composition of directed containers
 - Dist. laws give dcon. struct. on the comp. of underlying cons.
 - **Examples:** ne. lists over ne. lists; *streams over streams*, ...

Other constructions on directed containers

- Cofree and cofree recursive directed containers
- Distributive laws between directed containers
 - $t^\theta \triangleleft q^\theta : (S_0 \triangleleft P_0) \circ^c (S_1 \triangleleft P_1) \longrightarrow (S_1 \triangleleft P_1) \circ^c (S_0 \triangleleft P_0)$
satisfying 11 laws (and with $t_0^\theta(s, v) \stackrel{\text{def}}{=} v(o_{0\{s\}})$ forced)
 - A dep. typed version of the Zappa–Szép product, i.e., of:
 - Given monoid actions $\alpha : N \times M \rightarrow M$ and $\beta : N \times M \rightarrow N$ satisfying two compat. laws, we get a monoid on $M \times N$ with
$$(m_0, n_0) \oplus (m_1, n_1) \stackrel{\text{def}}{=} (m_0 \oplus_M \alpha(n_0, m_1), \beta(n_0, m_1) \oplus_N n_1)$$
- Composition of directed containers
 - Dist. laws give dcon. struct. on the comp. of underlying cons.
 - **Examples:** ne. lists over ne. lists; *streams over streams*, ...

“This should be called an aqueduct” —A.M.Pitts

Non-empty lists over non-empty lists



Directed containers and effects

Directed containers and effects

- **Recall:** Given a directed container $(S \triangleleft P, \downarrow, \circ, \oplus)$, we get a **comonad** on $DX = \Sigma s : S. (P s \rightarrow X)$

Directed containers and effects

- **Recall:** Given a directed container $(S \triangleleft P, \downarrow, \circ, \oplus)$, we get a **comonad** on $DX = \Sigma s : S. (P s \rightarrow X)$
- If you stare long enough at it (and it's quite late), then

Directed containers and effects

- **Recall:** Given a directed container $(S \triangleleft P, \downarrow, o, \oplus)$, we get a **comonad** on $DX = \sum s : S. (P s \rightarrow X)$
- If you stare long enough at it (and it's quite late), then
 - it looks a lot like the **costate comonad** on $S \times (S \rightarrow -)$

Directed containers and effects

- **Recall:** Given a directed container $(S \triangleleft P, \downarrow, \circ, \oplus)$, we get a **comonad** on $D X = \Sigma s : S. (P s \rightarrow X)$
- If you stare long enough at it (and it's quite late), then
 - it looks a lot like the **costate comonad** on $S \times (S \rightarrow -)$
 - what happens if instead we take $T X = \Pi s : S. (P s \times X)$?

Directed containers and effects

- **Recall:** Given a directed container $(S \triangleleft P, \downarrow, o, \oplus)$, we get a **comonad** on $DX = \Sigma s : S. (Ps \rightarrow X)$
- If you stare long enough at it (and it's quite late), then
 - it looks a lot like the **costate comonad** on $S \times (S \rightarrow -)$
 - what happens if instead we take $TX = \Pi s : S. (Ps \times X)$?
 - it looks suspiciously like the **state monad** $S \rightarrow (S \times -)$

Cointerpretation of (directed) containers

- In addition to the **interpretation** functor

$$\llbracket - \rrbracket^c : \mathbf{Cont} \longrightarrow [\mathbf{Set}, \mathbf{Set}]$$

one can also define a **cointerpretation** functor

$$\langle\langle - \rangle\rangle^c : \mathbf{Cont}^{\mathrm{op}} \longrightarrow [\mathbf{Set}, \mathbf{Set}]$$

given by

$$\langle\langle S \triangleleft P \rangle\rangle^c X \stackrel{\mathrm{def}}{=} \prod_{s : S} (P_s \times X)$$

which **lifts** to $\langle\langle - \rangle\rangle^{\mathrm{dc}}$, making the following a **pullback** in **CAT**

$$\begin{array}{ccc} \mathbf{DCont}^{\mathrm{op}} & \xrightarrow{U} & \mathbf{Cont}^{\mathrm{op}} \\ \langle\langle - \rangle\rangle^{\mathrm{dc}} \downarrow & & \downarrow \langle\langle - \rangle\rangle^c \\ \mathbf{Monads}(\mathbf{Set}) & \xrightarrow{U} & [\mathbf{Set}, \mathbf{Set}] \end{array}$$

Dependently typed update monads

- In more detail, given a **directed container** $(S \triangleleft P, \downarrow, \circ, \oplus)$, the corresponding **dependently typed update monad** is given by
 - $T : \mathbf{Set} \longrightarrow \mathbf{Set}$
$$T X \stackrel{\text{def}}{=} \llbracket S \triangleleft P \rrbracket^c X = \prod_{s : S} (P s \times X)$$
 - $\eta_X : X \longrightarrow T X$
$$\eta_X x \stackrel{\text{def}}{=} \lambda s. (\circ_{\{s\}}, x)$$
 - $\mu_X : T T X \longrightarrow T X$
$$\mu_X f \stackrel{\text{def}}{=} \lambda s. \mathbf{let} (p, g) = f s \mathbf{ in}$$
$$\mathbf{let} (p', x) = g (s \downarrow p) \mathbf{ in} (p \oplus_{\{s\}} p', x)$$
- Intuitively
 - S – set/type of **states**
 - (P, \circ, \oplus) – dependently typed monoid of **state updates**

Dependently typed update monads ctd.

- The dependently typed update monad

$$T X \stackrel{\text{def}}{=} \prod s : S. (P s \times X)$$

arises as the free-model monad for a (large) **Lawvere theory**, whose **models** are given by a **carrier** $M : \mathbf{Set}$ and two **operations**

$$\text{lkp} : (S \rightarrow M) \longrightarrow M \qquad \text{upd} : (\prod s : S. P s) \times M \longrightarrow M$$

subject to three natural **equations**

- $\text{lkp} (\lambda s. \text{upd}_{\lambda s. o_{\{s\}}} (m)) = m$
- $\text{lkp} (\lambda s. \text{upd}_f (\text{lkp} (\lambda s'. m s')))) = \text{lkp} (\lambda s. \text{upd}_f (m (s \downarrow (f s))))$
- $\text{upd}_f (\text{upd}_g (m)) = \text{upd}_{\lambda s. (f s) \oplus (g (s \downarrow f s))} (m)$

Examples of dep. typed update monads

- Global state
 - $S : \mathbf{Set}$
 - $P_S \stackrel{\text{def}}{=} S$
 - $s \downarrow s' \stackrel{\text{def}}{=} s'$
 - $O_{\{s\}} \stackrel{\text{def}}{=} s$
 - $s' \oplus_{\{s\}} s'' \stackrel{\text{def}}{=} s''$
 - $TX \stackrel{\text{def}}{=} S \rightarrow (S \times X)$

Examples of dep. typed update monads ctd.

- Monotonic state as in F^*

- $S : \mathbf{Set}$

- $P s \stackrel{\text{def}}{=} \{s' : S \mid s \mathcal{R} s'\}$

where \mathcal{R} is some fixed **preorder** on S , e.g.,

- \leq when $S \stackrel{\text{def}}{=} \mathbf{Nat}$ and modelling **monotonic counters**
- transition relation of some **state machine** (with states in S)
- subset relation for **references** when $S \stackrel{\text{def}}{=} \mathbf{heap}$

- $s \downarrow s' \stackrel{\text{def}}{=} s'$

- $O_{\{s\}} \stackrel{\text{def}}{=} s$

- $s' \oplus_{\{s\}} s'' \stackrel{\text{def}}{=} s''$

- $TX \stackrel{\text{def}}{=} \prod s : S. (\{s' : S \mid s \mathcal{R} s'\} \times X)$

- In F^* it is combined with a modal logic based **Hoare logic**

Examples of dep. typed update monads ctd.

- A non-overflowing (non-removal) buffer

- fixed size buffer of length n
- storing values of some type A

- $S \stackrel{\text{def}}{=} A^{\leq n}$

- $P\ as \stackrel{\text{def}}{=} A^{\leq n - \text{len } as}$

- $as \downarrow as' \stackrel{\text{def}}{=} as \uplus as'$

- $O_{\{as\}} \stackrel{\text{def}}{=} []$

- $as' \oplus_{\{as\}} as'' \stackrel{\text{def}}{=} as' \uplus as''$

- $TX \stackrel{\text{def}}{=} \prod as : A^{\leq n}. (A^{\leq n - \text{len } as} \times X)$

Examples of dep. typed update monads ctd.

- A non-underflowing (unbounded) stack
 - $S = A^*$
 - $P\ as = \{ps : (1 + A)^* \mid \text{removes } ps \leq \text{len } as\}$
where
 - $\text{removes } [] = 0$
 - $\text{removes } (\text{inl } * :: ps) = \text{removes } ps + 1$
 - $\text{removes } (\text{inr } a :: ps) = \text{removes } ps \div 1$
 - $as \downarrow [] = as$
 $as \downarrow (\text{inl } * :: ps) = as/1 \downarrow ps$
 $as \downarrow (\text{inr } a :: ps) = (as \uparrow [a]) \downarrow ps$
 - $o_{\{as\}} = []$
 - $as' \oplus_{\{as\}} as'' = as' \uparrow as''$

Simply typed update monads

- If P constant, then we get a simply typed update monad

$$T X \stackrel{\text{def}}{=} S \rightarrow (P \times X)$$

- In this case,
 - (P, o, \oplus) is a monoid in the standard sense
 - $\downarrow : S \times P \longrightarrow S$ is an action of (P, o, \oplus) on S
- This monad is the compatible composition of the monads

$$T_{\text{reader}} X \stackrel{\text{def}}{=} S \rightarrow X \qquad T_{\text{writer}} X \stackrel{\text{def}}{=} P \times X$$

- There is a one-to-one correspondence between
 - monoid actions $\downarrow : S \times P \longrightarrow S$
 - distributive laws $\theta : T_{\text{writer}} \circ T_{\text{reader}} \longrightarrow T_{\text{reader}} \circ T_{\text{writer}}$

Directed containers and BX

Directed containers and BX

- An **asymmetric lens** is a **comodel** for the th. of global state, i.e.,
 - $X : \mathbf{Set}$ (the database)
 - $\text{get} : X \longrightarrow S$ (computing the view)
 - $\text{put} : X \times S \longrightarrow X$ (updating the database)
 - satisfying natural laws relating get and put
- Equivalently a **coalgebra** for the costate comonad $S \times (S \rightarrow -)$

Directed containers and BX

- An **asymmetric lens** is a **comodel** for the th. of global state, i.e.,
 - $X : \mathbf{Set}$ (the database)
 - $\text{get} : X \longrightarrow S$ (computing the view)
 - $\text{put} : X \times S \longrightarrow X$ (updating the database)
 - satisfying natural laws relating get and put
- Equivalently a **coalgebra** for the costate comonad $S \times (S \rightarrow -)$
- Given a simply typed dcon. $(S \triangleleft P, \downarrow, \circ, \oplus)$, i.e., where $P : \mathbf{Set}$, we define a **simply typed update lens** to be given by
 - $X : \mathbf{Set}$
 - $\text{lkp} : X \longrightarrow S$
 - $\text{upd} : X \times P \longrightarrow X$
 - satisfying natural laws relating lkp and upd
- Equivalently a **coalgebra** for $\llbracket S \triangleleft P, \downarrow, \circ, \oplus \rrbracket^{\text{dc}}$

Directed containers and BX ctd.

- Analogously, given a general dcon. $(S \triangleleft P, \downarrow, \circ, \oplus)$, we can define a **dependently typed update lens** to be given by
 - $X : \mathbf{Set}$
 - $\text{lkp} : X \longrightarrow S$
 - $\text{upd} : (\Sigma x : X. P(\text{lkp } x)) \longrightarrow X$
 - satisfying natural laws relating lkp and upd

Directed containers and BX ctd.

- Analogously, given a general dcon. $(S \triangleleft P, \downarrow, o, \oplus)$, we can define a **dependently typed update lens** to be given by
 - $X : \mathbf{Set}$
 - $\text{lkp} : X \longrightarrow S$
 - $\text{upd} : (\sum x : X. P(\text{lkp } x)) \longrightarrow X$
 - satisfying natural laws relating lkp and upd
- Previous examples were about **asymmetric update lenses**, but it is also possible to do a **more symmetric** variant with dcons.:
 - $\text{fwd} \triangleleft \text{bwd} : (S_{\text{db}} \triangleleft P_{\text{db}}, \downarrow_{\text{db}}, o_{\text{db}}, \oplus_{\text{db}})$
 \longrightarrow
 $(S_{\text{view}} \triangleleft P_{\text{view}}, \downarrow_{\text{view}}, o_{\text{view}}, \oplus_{\text{view}})$
 - now both the database and the view have their own updates

Directed containers and (small) categories

Directed containers and (small) categories

- Given a **directed container** $(S \triangleleft P, \downarrow, o, \oplus)$ we get a corresponding **small category** $\mathcal{C}_{(S \triangleleft P, \downarrow, o, \oplus)}$ as follows
 - $\text{ob}(\mathcal{C}) \stackrel{\text{def}}{=} S$
 - $\mathcal{C}(s, s') \stackrel{\text{def}}{=} \Sigma p : P s. (s \downarrow p = s')$
 - **identities** are given using o
 - **composition** is given using \oplus
- And vice versa, every **small category** \mathcal{C} gives us a corresponding **directed container** $(S_{\mathcal{C}} \triangleleft P_{\mathcal{C}}, \downarrow_{\mathcal{C}}, o_{\mathcal{C}}, \oplus_{\mathcal{C}})$
- But then, is it simply the case that **Cat** \cong **DCont**?

Directed containers and (small) categories

- Given a **directed container** $(S \triangleleft P, \downarrow, o, \oplus)$ we get a corresponding **small category** $\mathcal{C}_{(S \triangleleft P, \downarrow, o, \oplus)}$ as follows
 - $\text{ob}(\mathcal{C}) \stackrel{\text{def}}{=} S$
 - $\mathcal{C}(s, s') \stackrel{\text{def}}{=} \Sigma p : P s. (s \downarrow p = s')$
 - **identities** are given using o
 - **composition** is given using \oplus
- And vice versa, every **small category** \mathcal{C} gives us a corresponding **directed container** $(S_{\mathcal{C}} \triangleleft P_{\mathcal{C}}, \downarrow_{\mathcal{C}}, o_{\mathcal{C}}, \oplus_{\mathcal{C}})$
- But then, is it simply the case that **Cat** \cong **DCont**? **NO!**

Directed container morphisms as cofunctors

- Given a directed container morphism

$$t \triangleleft q : (S \triangleleft P, \downarrow, o, \oplus) \longrightarrow (S' \triangleleft P', \downarrow', o', \oplus')$$

we do not get a functor, but instead a cofunctor [Aguiar'97]

$$F_{t \triangleleft q} : \mathcal{C}_{(S \triangleleft P, \downarrow, o, \oplus)} \longrightarrow \mathcal{D}_{(S' \triangleleft P', \downarrow', o', \oplus')}$$

given by a mapping of objects

$$(F_{t \triangleleft q})_0 \stackrel{\text{def}}{=} t : \text{ob}(\mathcal{C}) \longrightarrow \text{ob}(\mathcal{D})$$

and a lifting operation on morphisms (pre-opcleavage)

$$\begin{array}{ccc}
 s & \xrightarrow{(F_{t \triangleleft q})_1(s, p) \stackrel{\text{def}}{=} q_{\{s\}} p} & \circledast & \text{in } \mathcal{C} \\
 & \uparrow & & \\
 (F_{t \triangleleft q})_0(s) & \xrightarrow[p]{} & s' & \text{in } \mathcal{D}
 \end{array}$$

Constructions on dcons. revisited

- On the one hand, we can relate **existing constructions** on directed containers to constructions (small) categories, e.g.,
 - the **symmetry** of the definition of **directed polynomials** in

$$s : \overline{P} \longrightarrow S \quad \text{and} \quad \downarrow : \overline{P} \longrightarrow S$$

manifests as every category having an **opposite category**

Constructions on dcons. revisited

- On the one hand, we can relate **existing constructions** on directed containers to constructions (small) categories, e.g.,
 - the **symmetry** of the definition of **directed polynomials** in

$$s : \overline{P} \longrightarrow S \quad \text{and} \quad \downarrow : \overline{P} \longrightarrow S$$

manifests as every category having an **opposite category**

- **bidirected containers** (ongoing, a bit of a conundrum) with

$$(-)^{-1} : \prod \{s : S\}. \prod p : P \ s. P(s \downarrow p)$$

then correspond to **groupoids**

Constructions on dcons. revisited

- On the one hand, we can relate **existing constructions** on directed containers to constructions (small) categories, e.g.,

- the **symmetry** of the definition of **directed polynomials** in

$$s : \overline{P} \longrightarrow S \quad \text{and} \quad \downarrow : \overline{P} \longrightarrow S$$

manifests as every category having an **opposite category**

- bidirected containers** (ongoing, a bit of a conundrum) with

$$(-)^{-1} : \prod\{s : S\}. \prod p : P \rightarrow S. P (s \downarrow p)$$

then correspond to **groupoids**

- On the other hand, the (small) categories view also provides **new insights** into directed containers and comonads, e.g.,

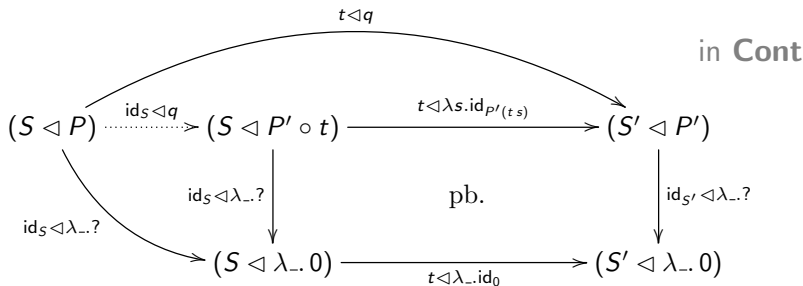
- factorisation** of directed container/comonad morphisms

Factorisation of morphisms

- Given a directed container morphism

$$t \triangleleft q : (S \triangleleft P, \downarrow, o, \oplus) \longrightarrow (S' \triangleleft P', \downarrow', o', \oplus')$$

we can factorise $(t \triangleleft q)$ as $(t \triangleleft \lambda s. \text{id}_{P'(t s)}) \circ (\text{id}_S \triangleleft q)$ where



inspired by the full image factorisation of ordinary functors

- Notably, this works for all pullback-preserving comonads

Conclusions

- Directed containers
 - type-theoretic and polynomial presentations
 - their use in functional programming
 - why are they canonical such structure?
- Some constructions on directed containers
 - coproducts of directed containers
 - strict directed containers and their products
 - focussing a container
 - ...
- Directed containers and computational effects
- Directed containers and BX
- Directed containers and categories