

# Directed containers, what are they good for?

Danel Ahman, Inria Paris

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



Edinburgh, 20 November 2017

# Outline

D. Ahman, J. Chapman, T. Uustalu.

**When is a Container a Comonad?** (FoSSaCS'12, LMCS 2014)

D. Ahman, T. Uustalu.

**Distributive laws of directed containers** (Progress in Inf. 2013)

D. Ahman, T. Uustalu.

**Update Monads: Cointerpreting Dir. Cons.** (TYPES'13)

D. Ahman, T. Uustalu.

**Coalgebraic update lenses** (MFPS'14)

D. Ahman, T. Uustalu.

**Directed containers as categories** (MSFP'16)

D. Ahman, T. Uustalu.

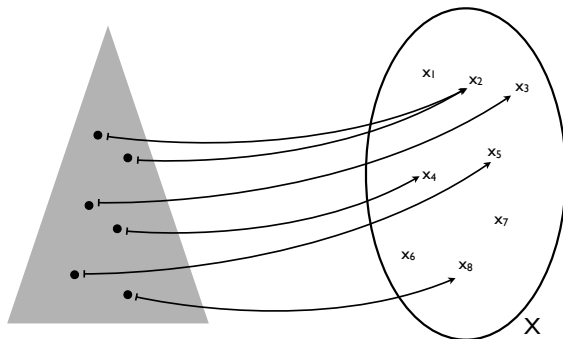
**Taking Updates Seriously** (BX'17)

# Directed containers

(and directed polynomials)

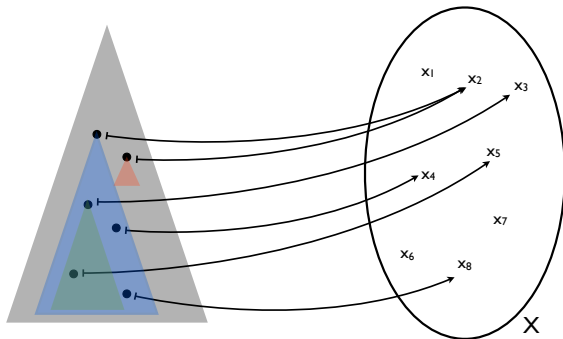
# Container syntax of datatypes

- Many **datatypes** can be represented in terms of
  - **shapes** of structured data
  - **positions** in shapes
- **Containers** provide us with a handy **syntax** to analyse them
- **Examples:** lists, streams, colists, trees, zippers, etc.



# 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

- 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 **polynomials**

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

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

where

- $S : \mathbf{Set}$  (or more generally, in suitable  $\mathcal{C}$ ) (*shapes*)
- $\overline{P} : \mathbf{Set}$  (or more generally, in suitable  $\mathcal{C}$ ) (*total positions*)
- $\overline{P} \cong \sum_{s:S} S \cdot P_s$



# Directed polynomials

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

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

where

- $S : \mathbf{Set}$  (or more generally, in suitable  $\mathcal{C}$ ) *(shapes)*
- $\overline{P} : \mathbf{Set}$  (or more generally, in suitable  $\mathcal{C}$ ) *(total positions)*
- $\overline{P} \cong \sum s : S. P_s$
- 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  $\downarrow \circ o = \text{id}_S$
  - ...
  - def. is remarkably **symmetric in  $s$  and  $\downarrow$**  (more on this later)

# Non-empty lists and streams

- Non-empty lists are represented as

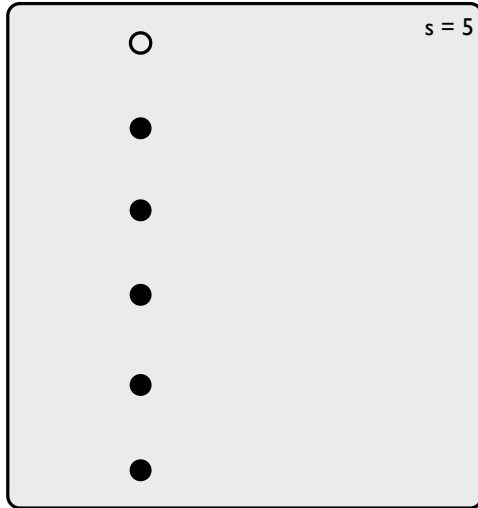
- $S \stackrel{\text{def}}{=} \text{Nat}$  *(shapes)*
- $P\ s \stackrel{\text{def}}{=} [0..s]$  *(positions)*
- $s \downarrow p \stackrel{\text{def}}{=} s - p$  *(subshapes)*
- $\text{o}_{\{s\}} \stackrel{\text{def}}{=} 0$  *(root position)*
- $p \oplus_{\{s\}} p' \stackrel{\text{def}}{=} p + p'$  *(subshape positions)*

- Streams are represented similarly

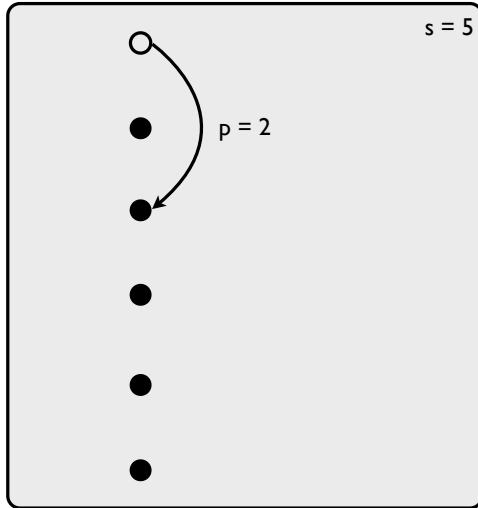
- $S \stackrel{\text{def}}{=} 1$  *(shapes)*
- $P\ * \stackrel{\text{def}}{=} \text{Nat}$  *(positions)*
- ...

- Another example is lists with **cyclic shifts** as “sublists”

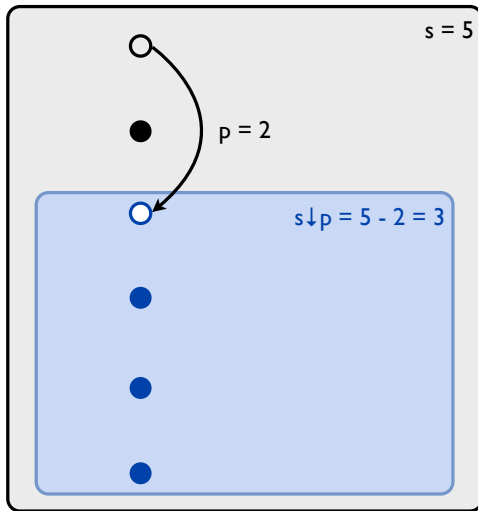
## Non-empty lists illustrated



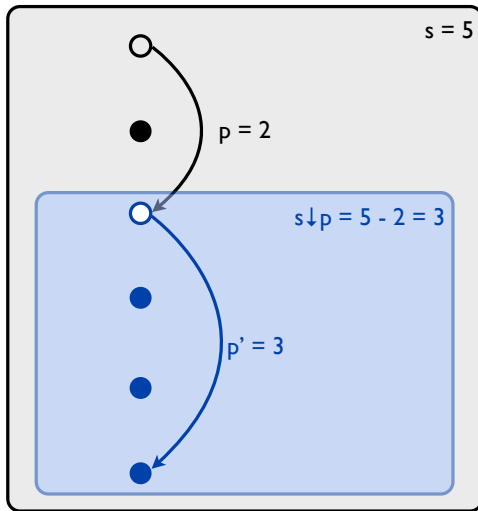
## Non-empty lists illustrated



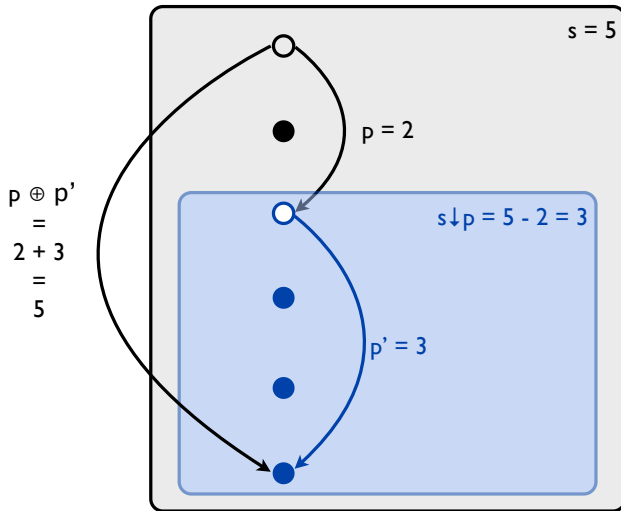
# Non-empty lists illustrated



# Non-empty lists illustrated



# Non-empty lists illustrated



# Non-empty lists with a focus

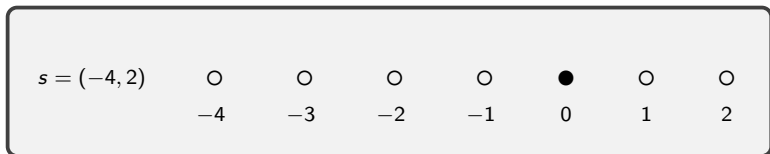
- **Zipper** – tree-like data-structures consisting of

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

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

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

- $P(s_0, s_1) \stackrel{\text{def}}{=} [-s_0..s_1] = [-s_0..-1] \cup [0..s_1]$  *(positions)*



- $(s_0, s_1) \downarrow p \stackrel{\text{def}}{=} (s_0 + p, s_1 - p)$  *(subshapes)*

- $o_{\{s_0, s_1\}} \stackrel{\text{def}}{=} 0$  *(root)*

- $p \oplus_{\{s_0, s_1\}} p' \stackrel{\text{def}}{=} p + p'$  *(subshape positions)*



# 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

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

such that

- $t(s \downarrow q p) = t s \downarrow' p$
- $o_{\{s\}} = q(o'_{\{ts\}})$
- $q p \oplus_{\{s\}} q p' = q(p \oplus'_{\{ts\}} p')$
- 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

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

such that

- $t(s \downarrow q p) = t s \downarrow' p$
- $o_{\{s\}} = q(o'_{\{ts\}})$
- $q p \oplus_{\{s\}} q p' = q(p \oplus'_{\{ts\}} p')$
- 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

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

defines a functor/comonad

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

where

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

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

- $\varepsilon_X : D X \rightarrow X$

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

- $\delta_X : D X \rightarrow D D X$

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

# 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

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

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

- $\varepsilon_X : D X \rightarrow X$

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

- $\delta_X : D X \rightarrow D D X$

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

# Interpretation of **dir. cont. 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

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

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

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

# Interpretation of dir. cont. 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

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

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

- $\llbracket - \rrbracket^{\text{dc}}$  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. $\cap$ cmdns.

- Any **comonad**  $(D, \varepsilon, \delta)$ , 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**:

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

## **Constructions on directed containers**

# Constructions on directed containers

- Coproduct of directed containers
- Cofree directed containers
- Focussing of a container
- Strict directed containers and their categorical product
- Distributive laws between directed containers
- Composition of directed containers
- **Ongoing:** Bidirected containers ([dep. typed group structure](#))
  - $(-)^{-1} : \prod\{s : S\}. \prod p : P s. P(s \downarrow p)$  + two equations
  - Which comonads do these correspond to? Hopf algebra like?

# **Update monads**

**(update the state instead of simply overwriting it!)**

# 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**

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

# 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
  - $T : \mathbf{Set} \longrightarrow \mathbf{Set}$   
$$T X \stackrel{\text{def}}{=} \llbracket S \triangleleft P, \downarrow, \circ, \oplus \rrbracket^{\text{dc}} X = \prod s : S. (P s \times X)$$
  - $\eta_X : X \longrightarrow T X$   
$$\eta_X x \stackrel{\text{def}}{=} \lambda s. (\circ, x)$$
  - $\mu_X : T T X \longrightarrow T X$   
$$\mu_X f \stackrel{\text{def}}{=} \lambda s. \mathbf{let} (p, g) = f s \mathbf{in}$$
$$\mathbf{let} (p', x) = g (s \downarrow p) \mathbf{in} (p \oplus p', x)$$
- Intuitively
  - $S$  – set of **states**
  - $(P, \circ, \oplus)$  – dependently typed monoid of **updates**
- Use **cases**: non-overflowing buffers, non-underflowing stacks

# Dependently typed update monads

- 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 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

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



# Simply typed update monads

- If  $P : \mathbf{Set}$ , 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  $\theta : T_{\text{writer}} \circ T_{\text{reader}} \longrightarrow T_{\text{reader}} \circ T_{\text{writer}}$

# Update lenses

(the dual of update monads)

# Update lenses

- A **dependently typed update lens** is a coalgebra for the comonad

$$DX \stackrel{\text{def}}{=} \llbracket S \triangleleft P, \downarrow, \circ, \oplus \rrbracket^{\text{dc}} X = \Sigma s : S. (P s \rightarrow X)$$

that is, a **carrier**  $M : \mathbf{Set}$  and **operations**

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

satisfying natural **equations** relating lkp and upd

- Equivalently, they are **comodels** for the Law. th. shown earlier
- Intuitively
  - $M$  – set of **sources**, i.e., the **database**
  - $S$  – set of **views**
  - $(P, \circ, \oplus)$  – dependently typed monoid of source **updates**

**Directed containers as (small) categories**

# Directed containers as (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 as (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 on objects

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

and a lifting operation on morphisms

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

# Constructions on directed containers

- 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** with  $(-)^{-1}$  correspond to **groupoids**
- On the other hand, the (small) categories view also provides **new constructions** on directed containers and comonads, e.g.,
  - **factorisation** of directed container/comonad morphisms

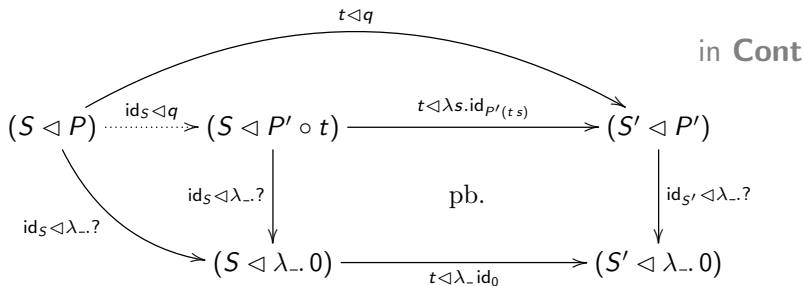


# Factorisation of morphisms

- 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



which corresponds to the full image factorisation of functors

- Notably, this works for all comonads that preserve pullbacks!

# Conclusion

- So, directed containers, what are they good for?
- Well, directed containers and their morphisms
  - describe datastructures with a notion of **subshape**
  - characterise containers that carry a **comonad** structure
  - admit a variety of natural **constructions**
  - give a natural updates-based refinement of the **state** monad
  - give a natural updates-based refinement of asymmetric **lenses**
  - provide a type-theoretic syntax for **categories** and **cofunctors**