# 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. Containers** (TYPES'13)

D. Ahman, T. Uustalu.
**Coalgebraic Update Lenses** (MFPS'14)

D. Ahman, T. Uustalu.
**Directed Containers as Categories** (MSFP'16)

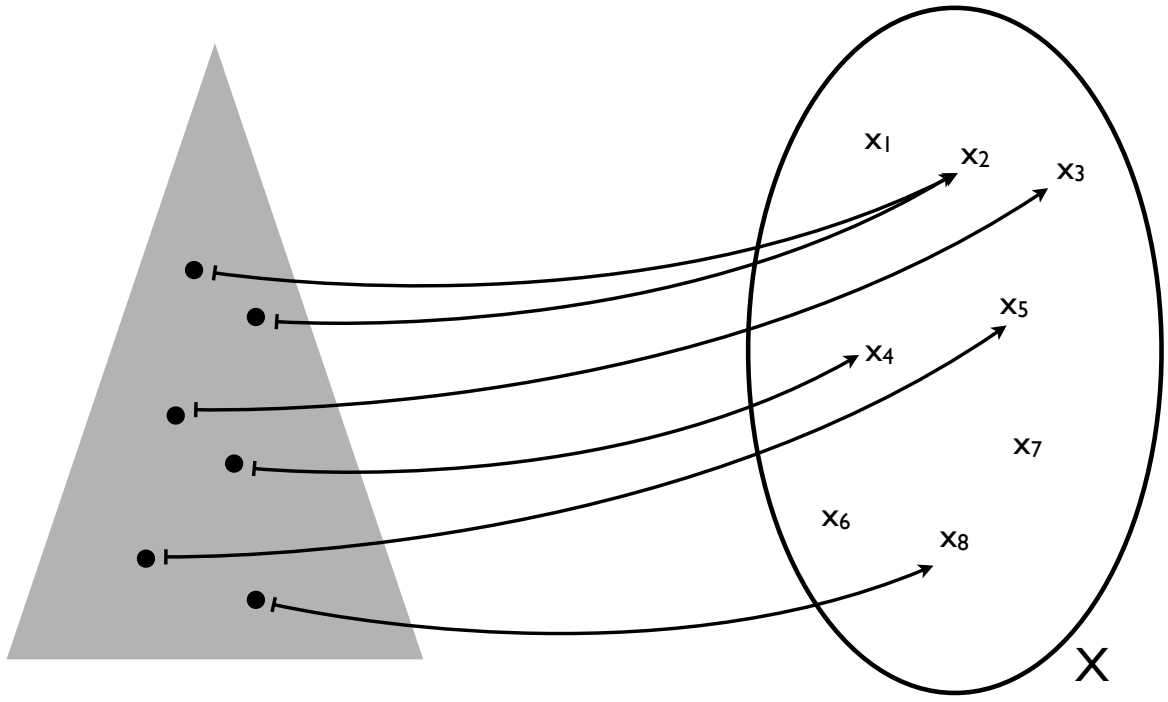
D. Ahman, T. Uustalu.
**Taking Updates Seriously** (BX'17)

# Directed containers

## (and directed polynomials)

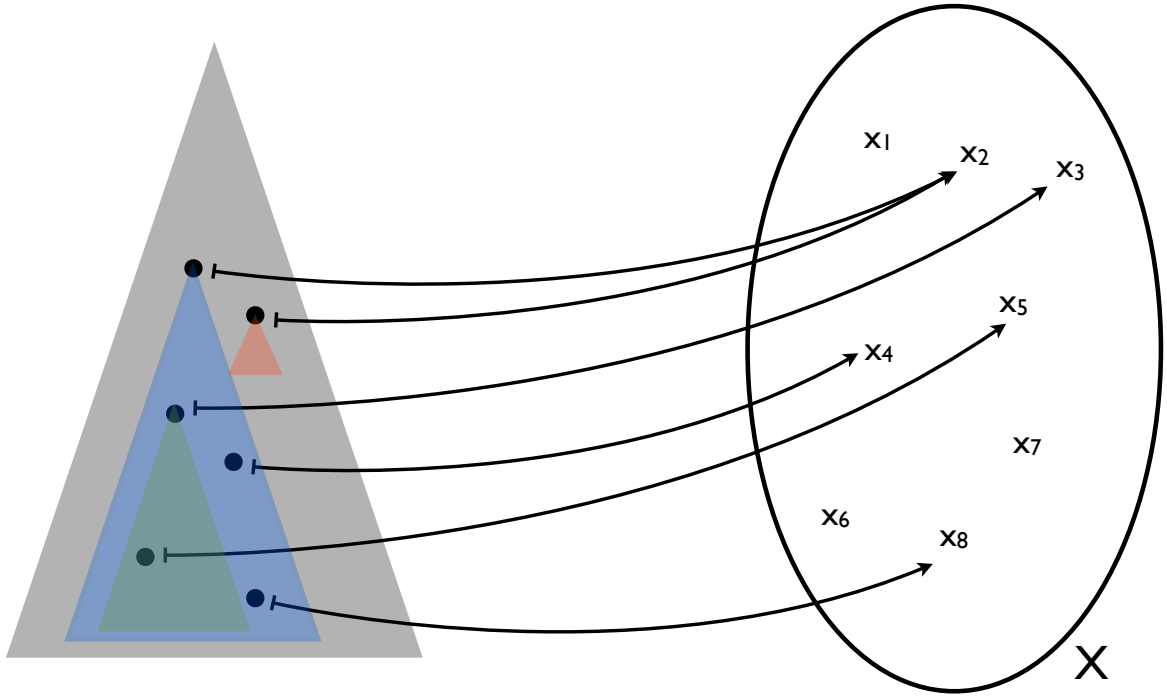
# Container syntax of datatypes

- Many datatypes can be represented in terms of
  - shapes of structured data, and
  - 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)*
  - $P : S \to \mathbf{Set}$ *(positions)*

  and
  - $\downarrow : \Pi s : S.\, P\, s \to S$ *(subshape)*
  - $\mathsf{o} : \Pi\{s : S\}.\, P\, s$ *(root position)*
  - $\oplus : \Pi\{s : S\}.\, \Pi p : P\, s.\, P\,(s \downarrow p) \to P\, s$ *(subshape positions)*

  such that
  - $s \downarrow \mathsf{o} = s$
  - $s \downarrow (p \oplus p') = (s \downarrow p) \downarrow p'$
  - $p \oplus\{s\}\ \mathsf{o} = p$
  - $\mathsf{o}\{s\} \oplus p = p$
  - $(p \oplus\{s\}\ p') \oplus p'' = p \oplus (p' \oplus p'')$

# Directed containers

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

  and
  - $\downarrow : \Pi s : S.\, P\, s \to S$ *(subshape)*
  - $\mathsf{o} : \Pi\{s : S\}.\, P\, s$ *(root position)*
  - $\oplus : \Pi\{s : S\}.\, \Pi p : P\, s.\, P\,(s \downarrow p) \to P\, s$ *(subshape positions)*

  such that
  - $s \downarrow \mathsf{o} = s$
  - $s \downarrow (p \oplus p') = (s \downarrow p) \downarrow p'$
  - $p \oplus_{\{s\}} \mathsf{o} = p$
  - $\mathsf{o}_{\{s\}} \oplus p = p$
  - $(p \oplus_{\{s\}} p') \oplus p'' = p \oplus (p' \oplus p'')$

# Directed polynomials

- A polynomial (in one variable) is given by


  where

  - $S$ : **Set** *(shapes)*
  - $\overline{P}$ : **Set** *(total positions)*

- Polynomials correspond to containers via $\overline{P} \cong \Sigma\, s : S.\, P\, s$

# Directed polynomials

- A polynomial (in one variable) is given by

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

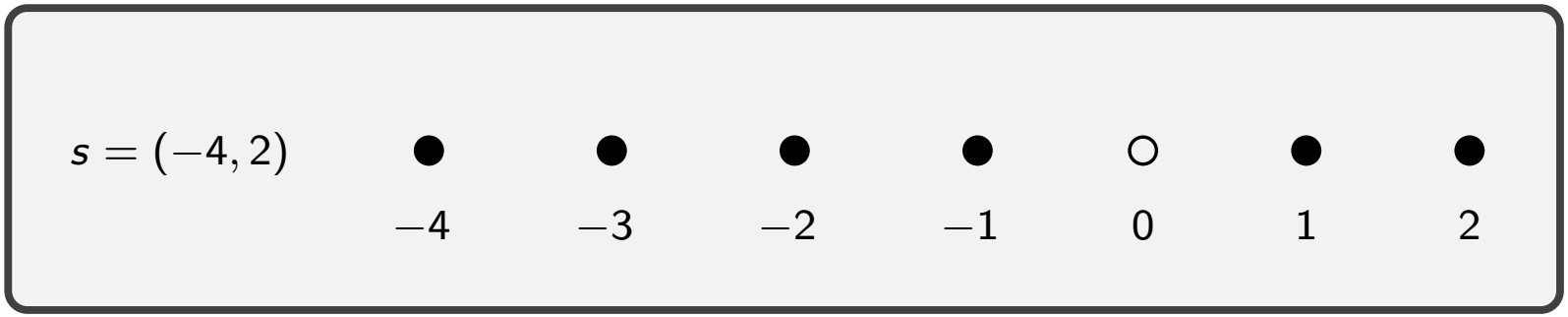
  where
  - $S$ : **Set** *(shapes)*
  - $\overline{P}$ : **Set** *(total positions)*
- Polynomials correspond to containers via $\overline{P} \cong \Sigma\, s : S.\, P\, s$
- A directed polynomial is given by
  - $\mathsf{s} : \overline{P} \longrightarrow S$ (a polynomial)
  - $\downarrow : \overline{P} \longrightarrow S$
  - $\mathsf{o} : S \longrightarrow \overline{P}$ s.t. $\mathsf{s} \circ \mathsf{o} = \mathrm{id}_S$ and $\downarrow \circ\, \mathsf{o} = \mathrm{id}_S$
  - ...
  - def. is remarkably symmetric in s and $\downarrow$ (more on this later)

# Examples: 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 non-empty lists with cyclic shifts

# Examples: non-empty lists with a focus

- Zippers – 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)*
  - $\text{o}_{\{s_0,s_1\}} \stackrel{\text{def}}{=} 0$ *(root)*
  - $p \oplus_{\{s_0,s_1\}} p' \stackrel{\text{def}}{=} p + p'$ *(subshape positions)*

# Directed container morphisms

- A directed container morphism

$$t \lhd q : (S \lhd P, \downarrow, \mathrm{o}, \oplus) \longrightarrow (S' \lhd P', \downarrow', \mathrm{o}', \oplus')$$

  is given by
  - $t : S \to S'$
  - $q : \Pi\{s : S\}.\, P'\,(t\,s) \to P\,s$

  such that
  - $t\,(s \downarrow q\,p) = t\,s \downarrow' p$
  - $\mathrm{o}_{\{s\}} = q\,(\mathrm{o}'_{\{t\,s\}})$
  - $q\,p \oplus_{\{s\}} q\,p' = q\,(p \oplus'_{\{t\,s\}} p')$

- Identities and composition are defined component-wise
- Directed containers form a category **DCont**

# Directed container morphisms

- A directed container morphism

$$t \lhd q : (S \lhd P, \downarrow, \mathsf{o}, \oplus) \longrightarrow (S' \lhd P', \downarrow', \mathsf{o}', \oplus')$$

  is given by

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

  such that

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

- 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

$$(S \lhd P, \downarrow, \mathsf{o}, \oplus)$$

defines a functor/comonad

$$[\![S \lhd P, \downarrow, \mathsf{o}, \oplus]\!]^{\mathrm{dc}} \stackrel{\text{def}}{=} (D, \varepsilon, \delta)$$

where

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

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

- $\varepsilon_X : D\,X \longrightarrow X$

  $\varepsilon_X\,(s, v) \stackrel{\text{def}}{=} v\,(\mathsf{o}_{\{s\}})$

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

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

# Interpretation of directed containers

- Any directed container

$$(S \lhd P, \downarrow, \mathsf{o}, \oplus)$$

defines a ~~functor~~ comonad

$$[\![S \lhd P, \downarrow, \mathsf{o}, \oplus]\!]^{\mathrm{dc}} \stackrel{\text{def}}{=} (D, \varepsilon, \delta)$$

where

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

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

- $\varepsilon_X : D\,X \longrightarrow X$

  $\varepsilon_X\,(s, v) \stackrel{\text{def}}{=} v\,(\mathsf{o}_{\{s\}})$

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

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

# Interpretation of dir. cont. morphisms

- Any directed container morphism

$$t \lhd q : (S \lhd P, \downarrow, \mathrm{o}, \oplus) \longrightarrow (S' \lhd P', \downarrow', \mathrm{o}', \oplus')$$

defines a natural transformation / comonad morphism

$$[\![t \lhd q]\!]^{\mathrm{dc}} : [\![S \lhd P, \downarrow, \mathrm{o}, \oplus]\!]^{\mathrm{dc}} \longrightarrow [\![S' \lhd P', \downarrow', \mathrm{o}', \oplus']\!]^{\mathrm{dc}}$$

by

- $[\![t \lhd q]\!]^{\mathrm{dc}}_X : \Sigma s : S.\,(P\,s \to X) \longrightarrow \Sigma s' : S'.\,(P'\,s' \to X)$

  $[\![t \lhd q]\!]^{\mathrm{dc}}_X\,(s, v) \stackrel{\mathrm{def}}{=} (t\,s, v \circ q_{\{s\}})$

- $[\![-]\!]^{\mathrm{dc}}$ preserves the identities and composition

- $[\![-]\!]^{\mathrm{dc}}$ is a functor from $\mathbf{Dir}_{\mathbf{Cont}}$ to $[\mathbf{Set}, \mathbf{Set}]$ / $\mathbf{Comonads}(\mathbf{Set})$

# Interpretation of dir. cont. morphisms

- Any directed container morphism

$$t \lhd q : (S \lhd P, \downarrow, \mathsf{o}, \oplus) \longrightarrow (S' \lhd P', \downarrow', \mathsf{o}', \oplus')$$

defines a natural transformation / comonad morphism

$$[\![t \lhd q]\!]^{\mathrm{dc}} : [\![S \lhd P, \downarrow, \mathsf{o}, \oplus]\!]^{\mathrm{dc}} \longrightarrow [\![S' \lhd P', \downarrow', \mathsf{o}', \oplus']\!]^{\mathrm{dc}}$$

by

  - $[\![t \lhd q]\!]^{\mathrm{dc}}_X : \Sigma s : S.\, (P\, s \to X) \longrightarrow \Sigma s' : S'.\, (P'\, s' \to X)$

    $[\![t \lhd q]\!]^{\mathrm{dc}}_X\, (s, v) \stackrel{\mathrm{def}}{=} (t\, s, v \circ q_{\{s\}})$

- $[\![-]\!]^{\mathrm{dc}}$ preserves the identities and composition

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

# Interpretation is fully faithful

- Every natural transformation / comonad morphism

$$\tau : [\![ S \lhd P, \downarrow, \mathrm{o}, \oplus ]\!]^{\mathrm{c}} \longrightarrow [\![ S' \lhd P', \downarrow', \mathrm{o}', \oplus' ]\!]^{\mathrm{c}}$$

defines a directed container morphism

$$\ulcorner \tau \urcorner^{\mathrm{c}} : (S \lhd P, \downarrow, \mathrm{o}, \oplus) \longrightarrow (S' \lhd P', \downarrow', \mathrm{o}', \oplus')$$

satisfying

  - $\ulcorner [\![ t \lhd q ]\!]^{\mathrm{c}} \urcorner^{\mathrm{c}} = t \lhd q$
  - $[\![ \ulcorner \tau \urcorner^{\mathrm{c}} ]\!]^{\mathrm{c}} = \tau$

- $[\![ - ]\!]^{\mathrm{c}}$ is a fully faithful functor

# Interpretation is fully faithful

- Every ~~natural transformation~~ / comonad morphism

$$\tau : [\![ S \lhd P, \downarrow, \mathsf{o}, \oplus ]\!]^{\mathrm{dc}} \longrightarrow [\![ S' \lhd P', \downarrow', \mathsf{o}', \oplus' ]\!]^{\mathrm{dc}}$$

defines a directed container morphism

$$\ulcorner \tau \urcorner^{\mathrm{dc}} : (S \lhd P, \downarrow, \mathsf{o}, \oplus) \longrightarrow (S' \lhd P', \downarrow', \mathsf{o}', \oplus')$$

satisfying

  - $\ulcorner [\![ t \lhd q ]\!]^{\mathrm{dc}} \urcorner^{\mathrm{dc}} = t \lhd q$
  - $[\![ \ulcorner \tau \urcorner^{\mathrm{dc}} ]\!]^{\mathrm{dc}} = \tau$

- $[\![ - ]\!]^{\mathrm{dc}}$ is a fully faithful functor

# Directed containers = cons. $\cap$ cmnds.

- Any comonad $(D, \varepsilon, \delta)$, such that $D = [\![S \lhd P]\!]^{\mathrm{c}}$, determines

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

- $\lceil - \rceil$ satisfies

$$[\![\lceil (D, \varepsilon, \delta), S \lhd P \rceil]\!]^{\mathrm{dc}} = (D, \varepsilon, \delta)$$

$$\lceil [\![S \lhd P, \downarrow, \mathsf{o}, \oplus]\!]^{\mathrm{dc}}, S \lhd P \rceil = (S \lhd P, \downarrow, \mathsf{o}, \oplus)$$

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

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

# 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)
  - $(-)^{-1} : \Pi\{s : S\}.\, \Pi p : P\, s.\, P(s \downarrow p)$ + 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

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

one can also define a cointerpretation functor

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

given by

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

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

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

# Dependently typed update monads

- In more detail, given a directed container $(S \lhd P, \downarrow, \mathsf{o}, \oplus)$ the corresponding dependently typed update monad is given by
  - $T : \mathbf{Set} \longrightarrow \mathbf{Set}$
    $T\,X \stackrel{\text{def}}{=} \langle\!\langle S \lhd P \rangle\!\rangle^{\mathrm{c}}\,X = \Pi s : S.\,(P\,s \times X)$
  - $\eta_X : X \longrightarrow T\,X$
    $\eta_X\,x \stackrel{\text{def}}{=} \lambda s.\,(\mathsf{o}_{\{s\}}, x)$
  - $\mu_X : T\,T\,X \longrightarrow T\,X$
    $\mu_X\,f \stackrel{\text{def}}{=} \lambda s.\,\mathbf{let}\ (p, g) = f\,s\ \mathbf{in}$
    $\mathbf{let}\ (p', x) = g\,(s \downarrow p)\ \mathbf{in}\ (p \oplus_{\{s\}} p', x)$
- Intuitively
  - $S$ – set of states
  - $(P, \mathsf{o}, \oplus)$ – dependently typed monoid of updates
- Use cases: non-overflowing buffers, non-underflowing stacks

# Dependently typed update monads

- The dependently typed update monad

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

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

$$\mathsf{lkp} : (S \to M) \longrightarrow M \qquad \mathsf{upd} : (\Pi s : S.\,P\,s) \times M \longrightarrow M$$

subject to three natural equations

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

# Simply typed update monads

- If $P : \mathbf{Set}$, then we get a simply typed update monad

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

- In this case,
  - $(P, \mathsf{o}, \oplus)$ is a monoid in the standard sense
  - $\downarrow : S \times P \longrightarrow S$ is an action of $(P, \mathsf{o}, \oplus)$ on $S$

- This monad is the compatible composition of the monads

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

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

# Update lenses

## (the dual of update monads)

# Update lenses

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

$$D\,X \stackrel{\text{def}}{=} [\![S \lhd P, \downarrow, \mathsf{o}, \oplus]\!]^{\mathrm{dc}}\,X = \Sigma s : S.\,(P\,s \to X)$$

  that is, a carrier $M$ : **Set** and operations

$$\mathsf{lkp} : M \longrightarrow S \qquad \mathsf{upd} : (\Pi s : S.\,P\,s) \times M \longrightarrow M$$

  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
  - $(P, \mathsf{o}, \oplus)$ – dependently typed monoid of source updates

# Directed containers as (small) categories

# Directed containers as (small) categories

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

# Directed containers as (small) categories

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

# Directed container morphisms as cofunctors

- Given a directed container morphism

$$t \lhd q : (S \lhd P, \downarrow, \mathsf{o}, \oplus) \longrightarrow (S' \lhd P', \downarrow', \mathsf{o}', \oplus')$$

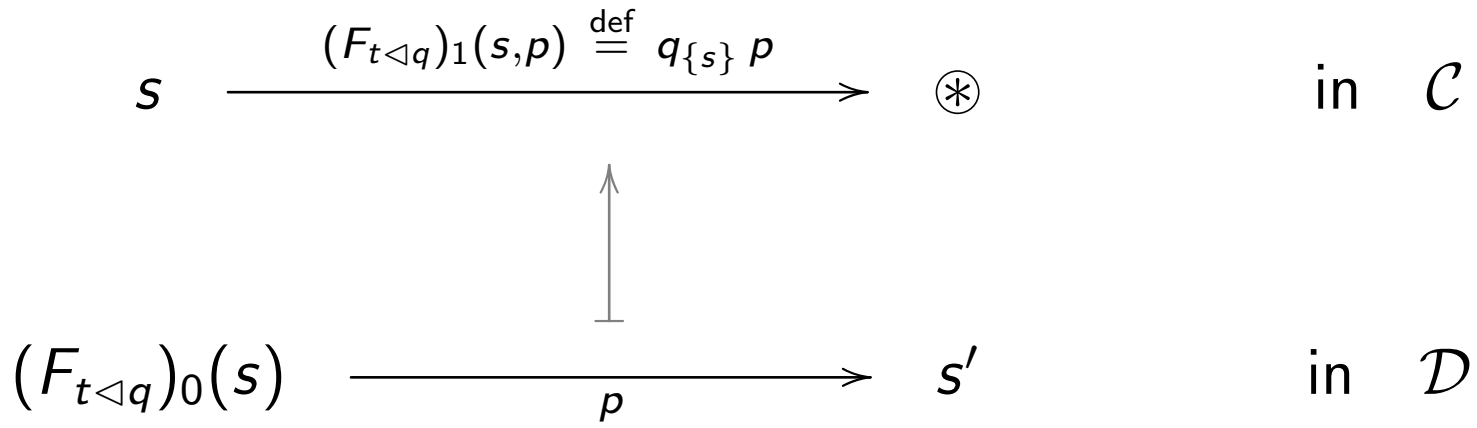
we do not get a functor, but instead a cofunctor [Aguiar'97]

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

given by a mapping of objects

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

and a lifting operation on morphisms



# Constructions on dir. containers 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

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

    manifests as every category having an opposite category
  - bidirected containers with $(-)^{-1}$ correspond to groupoids

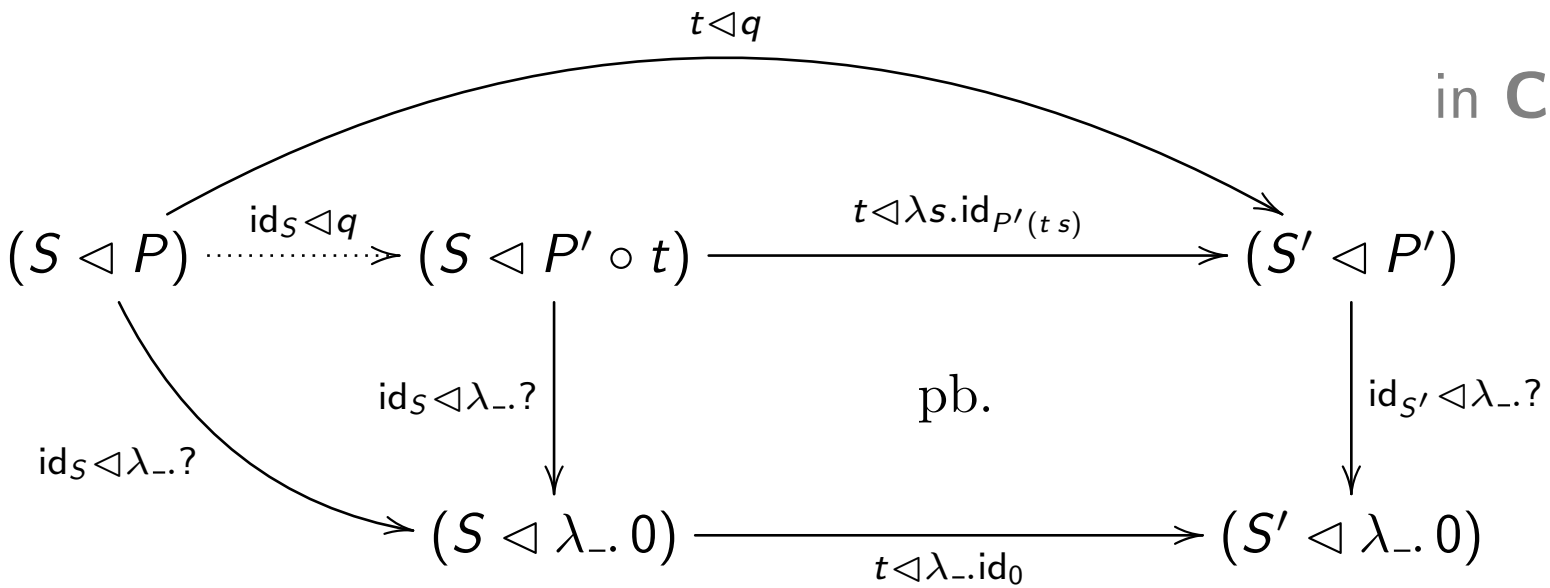
- On the other hand, the (small) categories view also provides new constructions on directed containers and comonads, e.g.,
  - factorisation of directed container/comonad morphisms

# Factorisation of morphisms

- Given a directed container morphism

$$t \lhd q : (S \lhd P, \downarrow, \mathsf{o}, \oplus) \longrightarrow (S' \lhd P', \downarrow', \mathsf{o}', \oplus')$$

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




inspired by the full image factorisation of ordinary 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