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)

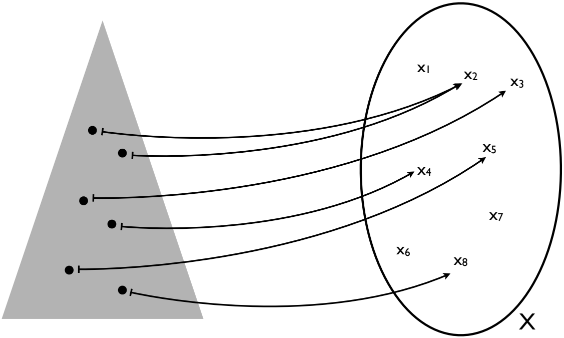
D. Ahman, T. Uustalu.
Taking Updates Seriously (BX'17)

Directed containers

(and directed polynomials)

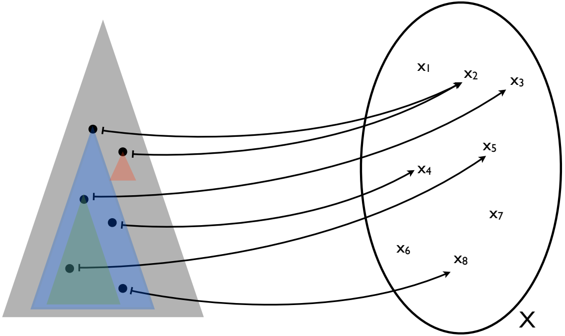
Container syntax of datatypes

- Many datatypes can be represented in terms of
  - shapes
  - 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 : **Set** *(shapes)*
  - $P : S \to$ **Set** *(positions)*

  and

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

  such that

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

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

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

  where

  - S : **Set** (or more generally, in suitable $\mathcal{C}$) *(shapes)*
  - $\overline{P}$: **Set** (or more generally, in suitable $\mathcal{C}$) *(total positions)*
  - $\overline{P} \cong \Sigma\, s\!:\!S.\, P\, s$

Directed polynomials

- A polynomial (in one variable) is given by

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

  where

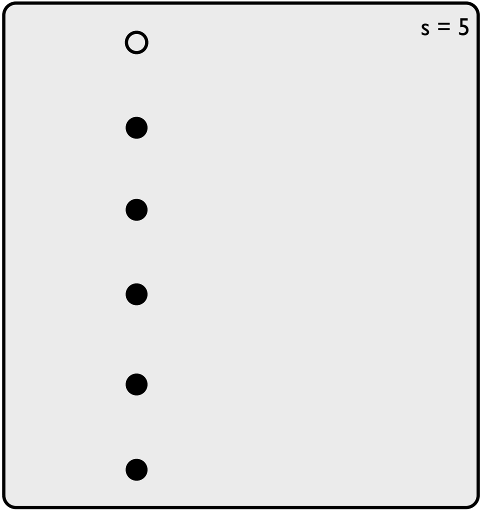
  - S : **Set** (or more generally, in suitable $\mathcal{C}$) *(shapes)*
  - $\overline{P}$: **Set** (or more generally, in suitable $\mathcal{C}$) *(total positions)*
  - $\overline{P} \cong \Sigma\, s : S.\, P\, s$

- 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)

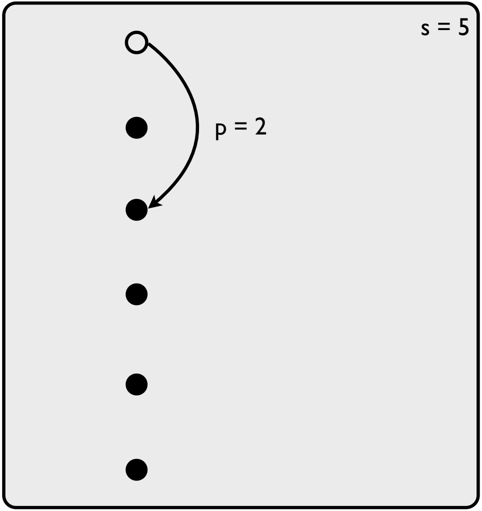
Non-empty lists and streams

- 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)*
  - $o_{\{s\}} \stackrel{\text{def}}{=} 0$ *(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”

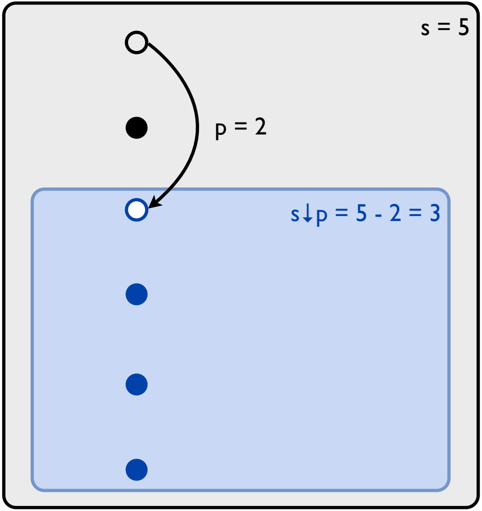
Non-empty lists illustrated



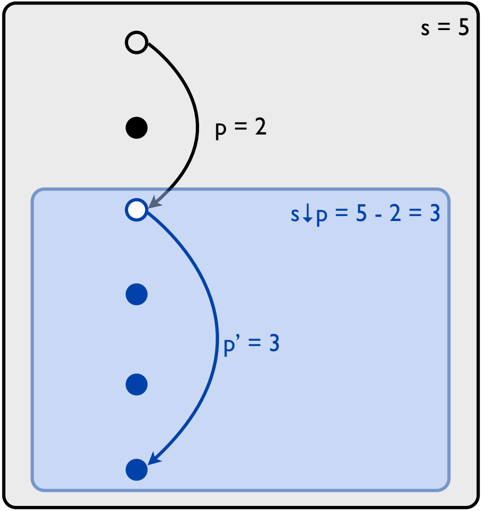
Non-empty lists illustrated



Non-empty lists illustrated

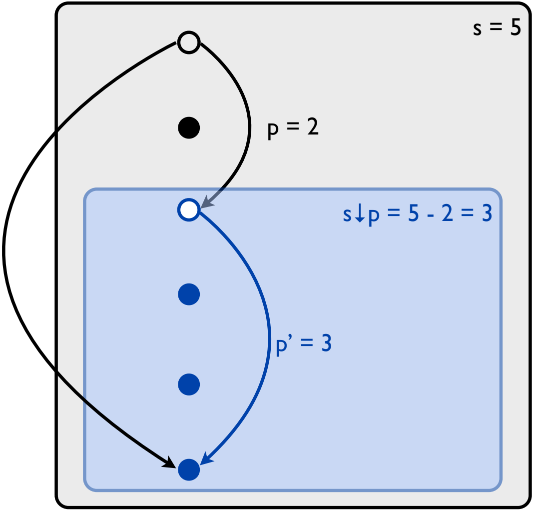


Non-empty lists illustrated



Non-empty lists illustrated

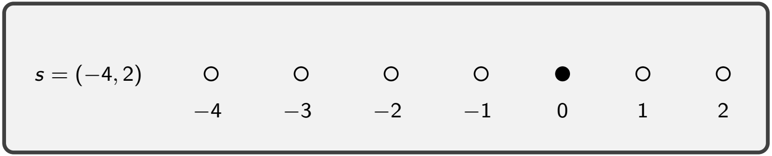
p ⊕ p'
=
2 + 3
=
5



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

Directed container morphisms

- A directed container morphism

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

  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 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~~

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

  where

  - $D : \mathbf{Set} \to \mathbf{Set}$ (or in any cat. with enough pullbacks)

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

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

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

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

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

Interpretation of directed containers

- Any directed container

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

defines a ~~functor~~/comonad

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

where

  - $D : \mathbf{Set} \to \mathbf{Set}$ (or in any cat. with enough pullbacks)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Interpretation is fully faithful

- Every natural transformation/~~comonad morphism~~

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

  defines a directed container morphism

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

  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

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, ε, δ), such that $D = [\![S \lhd P]\!]^{\mathrm{c}}$, determines

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

  where

$$s \downarrow p \stackrel{\mathsf{def}}{=} \mathsf{snd}\,(t^{\delta}\, s)\, p \qquad \mathsf{o}_{\{s\}} \stackrel{\mathsf{def}}{=} q^{\varepsilon}_{\{s\}} * \qquad p \oplus_{\{s\}} p' \stackrel{\mathsf{def}}{=} q^{\delta}_{\{s\}}(p, p')$$

- $\lceil - \rceil$ 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**:

$$\begin{tikzcd}
\mathbf{DCont} \arrow[r, "U"] \arrow[d, "{[\![-]\!]^{\mathrm{dc}}}"', "\text{f.f.}"] & \mathbf{Cont} \arrow[d, "{[\![-]\!]^{\mathrm{c}}}"', "\text{f.f.}"] \\
\mathbf{Comonads}(\mathbf{Set}) \arrow[r, "U"] & {[\mathbf{Set}, \mathbf{Set}]}
\end{tikzcd}$$

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
- Bidirected containers (dependently typed group structure)
  - $(-)^{-1} : \Pi\{s : S\}.\, \Pi p : P\, s.\, P(s \downarrow p)$ $\quad$ + two equations
  - Which comonads do these correspond to? Hopf algebra like?

Update monads

(update your state instead of 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

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

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

$$\begin{tikzcd}
\mathbf{DCont}^{\mathrm{op}} \arrow[r, "U"] \arrow[d, "\langle\!\langle - \rangle\!\rangle^{\mathrm{dc}}"'] & \mathbf{Cont} \arrow[d, "\langle\!\langle - \rangle\!\rangle^{\mathrm{c}}"] \\
\mathbf{Monads}(\mathbf{Set}) \arrow[r, "U"] & [\mathbf{Set}, \mathbf{Set}]
\end{tikzcd}$$

Dependently typed update monads

- In more detail, given a directed container $(S \lhd P, \downarrow, \mathrm{o}, \oplus)$ the corresponding dependently typed update monad is given by
  - $T : \mathbf{Set} \longrightarrow \mathbf{Set}$
    $T\,X \stackrel{\mathrm{def}}{=} \langle\!\langle S \lhd P, \downarrow, \mathrm{o}, \oplus \rangle\!\rangle^{\mathrm{dc}}\,X \;=\; \Pi s : S.\,(P\,s \times X)$
  - $\eta_X : X \longrightarrow T\,X$
    $\eta_X\,x \stackrel{\mathrm{def}}{=} \lambda s.\,(\mathrm{o}, x)$
  - $\mu_X : T\,T\,X \longrightarrow T\,X$
    $$\mu_X\,f \stackrel{\mathrm{def}}{=} \lambda s.\,\mathbf{let}\ (p, g) = f\,s\ \mathbf{in}\ \mathbf{let}\ (p', x) = g\,(s \downarrow p)\ \mathbf{in}\ (p \oplus p', x)$$
- Intuitively
  - S – set of states
  - $(P, \mathrm{o}, \oplus)$ – 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 : **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 : **Set**, then we get a simply typed update monad

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

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

- This monad is the compatible composition of the monads

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

- 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

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

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

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

and a lifting operation on morphisms

$$\begin{tikzcd}
s \arrow[rr, "(F_{t \lhd q})_1(s,p)"] & & \circledast & \text{in } \mathcal{C} \\
(F_{t \lhd q})_0(s) \arrow[rr, "p"'] & \arrow[u, mapsto] & s' & \text{in } \mathcal{D}
\end{tikzcd}$$

Constructions on directed containers

- On the one hand, we can relate existing constructions on directed containers to constructions (small) categories
- For example, the symmetry 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

- On the other hand, the (small) categories view also provides new constructions on directed containers and comonads
- For example, factorisation of dcontainer/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 it in **DCont** as

$$(S \lhd P, \downarrow, \mathsf{o}, \oplus) \xrightarrow{\mathrm{id}_s \lhd q} (S \lhd P' \circ t, \downarrow'', \mathsf{o}'', \oplus'') \xrightarrow{t \lhd \lambda s.\mathrm{id}_{P'(t\,s)}} (S' \lhd P', \downarrow', \mathsf{o}', \oplus')$$

  which can be characterised as the following pullback in **Cont**

$$\begin{tikzcd}
(S \lhd P) \arrow[r, "\mathrm{id}_S \lhd q"] & (S \lhd P' \circ t) \arrow[r, "t \lhd \lambda s.\mathrm{id}_{P'(t\,s)}"] \arrow[d, "\mathrm{id}_S \lhd \lambda_.?"'] \arrow[dr, phantom, "\mathrm{pb.}"] & (S' \lhd P') \arrow[d, "\mathrm{id}_{S'} \lhd \lambda_.?"] \\
 & (S \lhd \lambda_.\,0) \arrow[r, "t \lhd \lambda_\,\mathrm{id}_0"'] & (S' \lhd \lambda_.\,0)
\end{tikzcd}$$

  which corresponds to the full image factorisation of functors

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

Conclusion

- So, directed containers, what are they good for?
- They 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