Update Monads: Cointerpreting Directed Containers

Danel Ahman[1] and Tarmo Uustalu[2]

[1] Laboratory for Foundations of Computer Science, University of Edinburgh,
10 Crichton Street, Edinburgh EH8 9LE, United Kingdom; d.ahman@ed.ac.uk
[2] Institute of Cybernetics, Tallinn University of Technology,
Akadeemia tee 21, 12618 Tallinn, Estonia; tarmo@cs.ioc.ee

Containers are a neat representation of a wide class of set functors. We have previously [1] introduced directed containers as a concise representation of comonad structures on such functors. Here we examine interpreting the *opposite* categories of containers and directed containers. We arrive at a new view of a different (considerably narrower) class of set functors and monads on them, which we call update monads.[1]

A *container* is given by a set S (of shapes) and an S-indexed family of sets P (of positions). Containers form a category **Cont** with a (composition) monoidal structure. Containers interpet into set functors by

$$[\![S, P]\!]^{\mathrm{c}}\, X = \Sigma s : S.\, P\, s \to X$$

The functor $[\![-]\!]^{\mathrm{c}} : \mathbf{Cont} \to [\mathbf{Set}, \mathbf{Set}]$ is monoidal and fully faithful.

A *directed container* is a container (S, P) together with operations

$\downarrow : \Pi s : S. P\, s \to S$ (subshapes)
$\mathsf{o} : \Pi\{s : S\}.\, P\, s$ (the root)
$\oplus : \Pi\{s : S\}.\, \Pi p : P\, s.\, P\,(s \downarrow p) \to P\, s$ (subshape positions as positions in the global shape)

satisfying the laws

$$\begin{array}{l}
\forall\{s\}.\, s \downarrow \mathsf{o} = s \\
\forall\{s, p, p'\}.\, s \downarrow (p \oplus p') = (s \downarrow p) \downarrow p' \\
\forall\{s, p\}.\, p \oplus \{s\}\, \mathsf{o} = p \\
\forall\{s, p\}.\, \mathsf{o}\, \{s\} \oplus p = p \\
\forall\{s, p, p', p''\}.\, (p \oplus \{s\}\, p') \oplus p'' = p \oplus (p' \oplus p'')
\end{array}$$

so $(P, \mathsf{o}, \oplus)$ is a bit like a monoid (but dependently typed) and $(S, \downarrow)$ like its action on a set. Directed containers are the same as comonoids in the category of containers: $\mathbf{DCont} \cong \mathbf{Comonoids}(\mathbf{Cont})$. The interpretation of containers into set functors extends into an interpretation of directed containers into comonads via

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

The functor $[\![-]\!]^{\mathrm{dc}} : \mathbf{DCont} \to \mathbf{Comonads}(\mathbf{Set})$ is the pullback of the functor $[\![-]\!]^{\mathrm{c}} : \mathbf{Cont} \to [\mathbf{Set}, \mathbf{Set}]$ along $U : \mathbf{Comonads}(\mathbf{Set}) \to [\mathbf{Set}, \mathbf{Set}]$, meaning that directed containers are the same as containers whose interpretation carries a comonad structure.

[1]This is not the same as having containers with suitable additional structure interpret into monads under the standard interpretation of containers into set functors.

Here we are interested in the "cointerpretation" of containers given by

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

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

It is straightforward that $\mathbf{DCont}^{\mathrm{op}} \cong (\mathbf{Comonoids}(\mathbf{Cont}))^{\mathrm{op}} \cong \mathbf{Monoids}(\mathbf{Cont}^{\mathrm{op}})$. It follows therefore that directed containers cointerpret to monads via

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

i.e., $\langle\!\langle -\rangle\!\rangle^{\mathrm{c}}$ extends to a functor $\langle\!\langle -\rangle\!\rangle^{\mathrm{dc}} : \mathbf{DCont}^{\mathrm{op}} \to \mathbf{Monads}(\mathbf{Set})$. We do not get that the functor $\langle\!\langle -\rangle\!\rangle^{\mathrm{dc}}$ is the pullback of $\langle\!\langle -\rangle\!\rangle^{\mathrm{c}}$ along $U : \mathbf{Monads}(\mathbf{Set}) \to [\mathbf{Set}, \mathbf{Set}]$.

$\langle\!\langle -\rangle\!\rangle^{\mathrm{dc}}$ describes the free models of the (generally non-finitary) Lawvere theory given by one operation $\mathsf{act} : S \to \Pi s : S.\, P\, s$ and two equations

$$\begin{tikzcd}
1 \arrow[r, "\lambda s.\, *"] \arrow[d, equal] & S \arrow[d, "\mathsf{act}"] \\
1 & \Pi s : S.\, P\, s \arrow[l, "\lambda *.\lambda s.\, \mathsf{o}\,\{s\}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
S \times S \arrow[r, "\lambda(s,f).\,(s,f\ s)"] \arrow[d, "S \times \mathsf{act}"] & S \times (S \to S) \arrow[ddd, "\mathsf{act} \times (S \to S)"] \\
S \times \Pi s' : S.\, P\, s' \arrow[d, "\lambda(s,f).\,(s,f\ s)"] & \\
S \times (S \to \Pi s' : S.\, P\, s') \arrow[d, "\mathsf{act} \times (S \to \Pi s' : S.\ P\ s')"] & \\
(\Pi s : S.\, P\, s) \times (S \to \Pi s' : S.\, P\, s') & (\Pi s : S.\, P\, s) \times (S \to S) \arrow[l, "\lambda(f,g).\,(\lambda s.\ f\ s \oplus g\ s\ (s \downarrow f\ s), \lambda s.\ s \downarrow f\ s)"]
\end{tikzcd}$$

For cointerpretation, it is useful to think of elements of S as states, those of $P\, s$ as updates applicable to a state s, $s \downarrow p$ as the result of applying an update p to the state s, $\mathsf{o}\{s\}$ as the nil update, $p \oplus p'$ composition of two updates. Monads induced by directed containers generalize the state monad much in the spirit of the generalizations considered by Kammar [2], but still a bit differently. The state monad $TX = S \to S \times X$ is recovered by taking $S = S$, $P\, s = S$, $s \downarrow p = p$, $\mathsf{o}\,\{s\} = s$, $p \oplus p' = p'$. The directed container for the nonempty list comonad, $S = \mathsf{Nat}$, $P\, s = [0..s]$, $s \downarrow p = s - p$, $\mathsf{o} = 0$, $p \oplus p' = p + p'$ gives us a monad on the functor $TX = \Pi s : \mathsf{Nat}.\, [0..s] \times X$. The states are natural numbers; the updates applicable to a state s are numbers not greater than s; applying an update means decrementing the state.

Acknowledgements We thank Ohad Kammar for discussions. This ongoing work is being supported by the University of Edinburgh Principal's Career Development PhD Scholarship, the ERDF funded Estonian Centre of Excellence in Computer Science, EXCS, and the Estonian Research Council grant no. 9475.

References

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

[2] O. Kammar. Take action for your state: effective conservative restrictions. Slides from Scottish Programming Language Seminar, Strathclyde, 2010.