Container combinatorics: 
Monads and more

Tarmo Uustalu, Tallinn University of Technology

SYCO 1, Birmingham, 20–21 September 2018
Containers?

- **Containers** (Abbott, Altenkirch, Ghani; cf polynomials, Gambino, Hyland, Kock) are an elegant “syntax” in terms of shapes and positions for a wide class of set functors.

- In particular, they are good for enumerative combinatorics, for enumerating structures of a given type on a functor.

- Prior work: **Directed containers** (Ahman, Chapman, Uustalu) as containers with additional structure denoting comonads.

- This talk: **Further specializations of containers** corresponding to monads, lax monoidal functors (aka idioms) and more.
Containers

- A container is given by
  - a set $S$ (of shapes)
  - and a $S$-indexed family $P$ of sets (of positions in each shape)

- A container $(S, P)$ interprets into a set functor $\mathbb{[S, P]}^c = F$ where
  - $F X = \Sigma s : S. P s \to X$
  - $F f = \lambda(s, v). (s, f \circ v)$
Lists container

- Let
  - \( S = \mathbb{N} \)
  - \( P s = [0..s) \)
- The container \((S, P)\) represents the list datatype, as
  - \( [S, P]^c X = \Sigma s : \mathbb{N}. [0..s) \rightarrow X \)
  - \( \cong \) List \( X \).
A container morphism between \((S, P)\) and \((S', P')\) is given by operations

- \(t : S \rightarrow S'\) (the shape map)
- and \(q : \prod_{s:S} P' (t s) \rightarrow P s\) (the position map)

A container morphism \((t, q)\) between \((S, P)\) and \((S', P')\) interprets into a natural transformation \([t, q]^c = \tau\) between \([S, P]^c\) and \([S', P']^c\) where

\[
\tau_X : [S, P]^c X \rightarrow [S', P']^c X
\]

\[
(\Sigma s : S. P s \rightarrow X) \rightarrow (\Sigma s' : S'. P' s' \rightarrow X)
\]

\[
\tau(s, v) = (t s, v \circ q_s)
\]
Some lists container endomorphisms

- Let $S = \mathbb{N}$, $P s = [0..s)$ as before.
- We can define a container endomorphism $(t, q)$ on $(S, P)$ for example by
  - $t s = s$
  - $q_s p = s - p$
  This denotes the list reversal function.
- But setting
  - $t s = s + s$
  - $q_s p = p \mod s$
  we get a representation of the list self-append function.
The category of containers

- Identity on \((S, P)\) is \((\text{id}_S, \lambda_s. \text{id}_{P_s})\).
- Composition of \((t, q) : (S, P) \rightarrow (S', P')\) and \((t', q') : (S', P') \rightarrow (S'', P'')\) is \((t' \circ t, \lambda_s. q_s \circ q'_{t_s})\).

Containers form a category \textbf{Cont}.

\([-\]\)\(^c\) makes a fully-faithful functor from \textbf{Cont} to \([\text{Set}, \text{Set}]\).
Two monoidal structures

- The identity container is $\text{Id}^c = (1, \lambda \ast. 1)$.
- Composition of $(S, P)$ and $(S', P')$ is $(S, P) \cdot^c (S', P') = (\Sigma s : S. P s \rightarrow S', \lambda(s, v). \Sigma p : P s . P'(v p))$.
- $(\text{Cont}, \text{Id}^c, \cdot^c)$ is a monoidal category and $[-]^c$ a monoidal functor to $([\text{Set}, \text{Set}], \text{Id}, \cdot)$.

- Day convolution of $(S, P)$ and $(S', P')$ is $(S, P) \otimes^c (S', P') = (S \times S', \lambda(s, s'). P s \times P' s)$.
- $(\text{Cont}, \text{Id}^c, \otimes^c)$ is a symmetric monoidal category and $[-]^c$ a symmetric monoidal functor to $([\text{Set}, \text{Set}], \text{Id}, \otimes)$.

- For any $(S, P), (S', P')$, there is a container morphism from $(S, P) \otimes^c (S', P') \rightarrow (S, P) \cdot^c (S', P')$.
- This makes $\text{Id}_{\text{Cont}}$ a lax monoidal functor from $(\text{Cont}, \text{Id}^c, \cdot^c)$ to $(\text{Cont}, \text{Id}^c, \otimes^c)$.
Mnd-containers

- Call an \textit{mnd-container} a container \((S, P)\) with operations
  - \(e : S\)
  - \(\bullet : \prod s : S. (P \ s \to S) \to S\)
  - \(q_0 : \prod s : S. \prod v : P \ s \to S. P (s \bullet v) \to P \ s\)
  - \(q_1 : \prod s : S. \prod v : P \ s \to S. \prod p : P (s \bullet v). P (v (v \ \triangleleft_s p))\)

where we write

- \(q_0 \ s \ v \ p\) as \(v \ \triangleleft_s p\) and
- \(q_1 \ s \ v \ p\) as \(p \ \triangleright_v s\)

satisfying

- \(s = s \bullet (\lambda_. \ e)\)
- \(e \bullet (\lambda_. \ s) = s\)
- \((s \bullet v) \bullet (\lambda p''. w (v \ \triangleleft_s p'') (p'' \ \triangleright_v s)) = s \bullet (\lambda p'. v p' \bullet w p')\)

and \ldots\
... and

- \( p = (\lambda_. e) \downarrow_s p \)
- \( p \uparrow_{\lambda_. e} = p \)
- \( \nu \downarrow_s ((\lambda p''. w (\nu \downarrow_s p'') (p'' \uparrow_v s)) \downarrow_{s \bullet v} p) = (\lambda p'. v p' \bullet w p') \downarrow_s p \)
- \( ((\lambda p''. w (\nu \downarrow_s p'') (p'' \uparrow_v s)) \downarrow_{s \bullet v} p) \uparrow_v s = \)
  - let \( u p' \leftarrow v p' \bullet w p' \) in \( w (u \downarrow s p) \uparrow_v (u \downarrow_{s p}) (p \uparrow u s) \)
- \( p \uparrow_{\lambda p''. w (\nu \downarrow_s p'')} (p'' \uparrow_v s) (s \bullet v) = \)
  - let \( u p' \leftarrow v p' \bullet w p' \) in \( (p \uparrow u s) \uparrow_w (u \downarrow_{s p}) v (u \downarrow s p) \)
Mnd-containers ctd

- An mnd-container \((S, P, e, \bullet, \land, \lor)\) interprets into a monad \(\llbracket S, P, e, \bullet, \land, \lor \rrbracket^{mc} = (T, \eta, \mu)\) where
  - \(T = \llbracket S, P \rrbracket^c\)
  - \(\eta_X : X \rightarrow TX\)
    \[X \rightarrow \sum s : S. Ps \rightarrow X\]
    \(\eta x = (e, \lambda x)\)
  - \(\mu_X : T(TX) \rightarrow TX\)
    \[(\sum s : S. Ps \rightarrow \sum s' : S. Ps' \rightarrow X) \rightarrow (\sum s : S. Ps \rightarrow X)\]
    \(\mu (s, v) =\)
    let \((v_0 p, v_1 p) \leftarrow v p \text{ in } (s \bullet v_0, \lambda p. v_1 (v_0 \land s p) (p \lor v_0 s))\)
The category of mnd-containers

- Mnd-containers form a category $\text{MCont}$, with identities and composition inherited from $\text{Cont}$.

- Mnd-container interpretation $\llbracket \_ \rrbracket^{mc}$ makes a fully-faithful functor between $\text{MCont}$ and $\text{Monad}(\text{Set})$.

\[
\begin{align*}
\text{MCont} & \quad \cong \text{Monoid}(\text{Cont}, \text{id}^C, \cdot^C) \\
\text{Monad}(\text{Set}) & \quad \cong \text{Monoid}([\text{Set}, \text{Set}], \text{id}, \cdot)
\end{align*}
\]
Exception container

- Let \( S = 1 + E \) for some set \( E \) and \( P(\text{inl} \,* ) = 1, \ P(\text{inr} \, _) = 0 \).
- Then \( T X = \sum s : 1 + E. \left( \begin{array}{c}
\text{case } s \text{ of } \\
\text{inl} \,* & \mapsto 1 \\
\text{inr} \, _{\text{ }} & \mapsto 0
\end{array} \right) \rightarrow X \cong X + E \).
- If, in a hypothetical mnd-container structure on \((S, P)\), \( e = \text{inr} \, e_0 \) for some \( e_0 : E \), then \( P\, e = 0 \) and therefore \( \text{inl} \,* = e \bullet (\lambda \cdot \text{inl} \,* ) = e \bullet (\lambda \cdot \text{inr} \, e_0 ) = \text{inr} \, e_0 \), which is absurd.
- If \( e = \text{inl} \,* \), then necessarily \( \text{inl} \,* \bullet v = e \bullet (\lambda \,* . \, v \,* ) = v \,* \) and \( \text{inr} \, e \bullet v = \text{inr} \, e \bullet (\lambda _{\text{ }} . \, e) = \text{inr} \, e \).
- This choice of \( e \) and \( \bullet \) satisfies the conditions of an mnd-container.
- So there is exactly one mnd-container structure on \((S, P)\) and exactly one monad structure on \( T \).
Lists container

- Let $S = \mathbb{N}$, $P_s = [0..s)$.
- Then $T X = \sum s : \mathbb{N}. [0..s) \rightarrow X \cong \text{List } X$.

The following is an mnd-container structure:

- $e = 1$
- $s \cdot v = \sum_{p:[0..s)} v p$
- $v \uparrow_s p = \text{greatest } p_0 : [0..s) \text{ st } \sum_{p' : [0..p_0)} v p' \leq p$
- $p \uparrow_v s = p - \sum_{p' : [0..v \uparrow_s p)} v p'$

The corresponding monad structure is

$\eta_X x = [x], \mu_X xss = \text{concat } xss$.

But these are not the only mnd-container structure on $(S, P)$ and not the only monad structure on $T$. 
Mnd-containers as generalized operads

- The (standard) lists mnd-container generalizes for non-symmetric operads.
- Given an operad, i.e., a set $O$ (of operations) and functions $\# : O \rightarrow \mathbb{N}$ (fixing the arities) and $\text{id} : O$ (the identity) and $\circ : \prod \circ : O.$ ($\# \circ \rightarrow O) \rightarrow O$ (composition) satisfying $\# \text{id} = 1$ and $\# (o \circ v) = \sum_{i: [0, \# o)} \# (v i)$ and a number of further equations.
- We can take $S = O$, $P o = [0..\# o)$, $e = \text{id}$, $s \bullet v = s \circ v$ and $\left, \right$ as in the lists mnd-container.
- The lists mnd-container corresponds to the operad Assoc with exactly one operation of every arity.
- General mnd-containers are like operads, but arities may be infinite, identification of the arguments of an operation is nominal, and the arguments of a composition may be used non-linearly by the operations involved (as specified by $\left, \right$).
Altenkirch, Pinyo have observed that an mnd-container defines a “lax” \((1, \Sigma)\)-universe.

- \(S\) is the set of “(codes for) types”.
- \(P\ s\) is the “denotation” of \(s\),
- \(e\) is the type 1,
- \(\bullet\) is the \(\Sigma\)-type former,
- \(\land, \lor\) are projections from denotations of \(\Sigma\)-types

The laxity is that 1 need not really denote the singleton set and \(\Sigma\)-types need not really denote dependent products, we only have functions \(P\ e \rightarrow 1\) and \(P\ (s\ \bullet\ v) \rightarrow \Sigma p : P\ s.\ P\ (v\ p)\), not isomorphisms.
Lmf-containers

- Call an *lmf-container* a container \((S, P)\) with operations
  - \(e : S\)
  - \(\bullet : S \rightarrow S \rightarrow S\)
  - \(q_0 : \Pi s : S. \Pi s' : S. P (s \bullet s') \rightarrow P s\)
  - \(q_1 : \Pi s : S. \Pi s' : S. P (s \bullet s') \rightarrow P s'\)

where we write
  - \(q_0 s s' p\) as \(s' \triangleleft_s p\) and \(q_1 s s' p\) as \(p \triangleright_{s'} s\)

satisfying
  - \(e \bullet s = s\)
  - \(s = s \bullet e\)
  - \((s \bullet s') \bullet s'' = s \bullet (s' \bullet s'')\)
  - \(e \triangleleft_s p = p\)
  - \(p \triangleright_{s} e = p\)
  - \(s' \triangleleft_s (s'' \triangleleft_{s \bullet s'} p) = (s' \bullet s'') \triangleleft_s p\)
  - \((s'' \triangleleft_{s \bullet s'} p) \triangleright_{s'} s = s'' \triangleleft_{s'} (p \triangleright_{s' \bullet s''} s)\)
  - \(p \triangleright_{s''} (s \bullet s') = (p \triangleright_{s' \bullet s''} s) \triangleright_{s''} s'\)
Lmf-containers ctd

An lmf-container \((S, P, e, \bullet, \wedge, \vee)\) interprets into a lax monoidal functor \(\lbrack S, P, e, \bullet, \wedge, \vee \rbrack^1 = (F, m^0, m)\) where

- \(F = \lbrack S, P \rbrack^c\)
- \(m^0 : 1 \to T 1\)
  
  
  \[
  1 \to (\Sigma s : S. Ps \to 1) \]

  \[
  m^0 \ast = (e, \lambda. \ast)
  \]

- \(m_{X, Y} : T X \times T Y \to T (X \times Y)\)

  
  \[
  (\Sigma s : S. Ps \to X) \times (\Sigma s : S. Ps \to Y) \to (\Sigma s : S. Ps \to X \times Y)
  \]

- \(m_{X, Y} ((s, v), (s', v')) = (s \bullet s', \lambda p. (v (s' \wedge_s p), v' (p \vee_{s'} s)))\)
The category of lmf-containers

- Lmf-containers form a category \( \text{LCont} \), with identities and composition inherited from \( \text{Cont} \).

- \( [-]^{lc} \) is a fully-faithful functor between \( \text{LCont} \) and \( \text{LMF}(\text{Set}) \).
Mnd-containers vs Lmf-containers

- Any mnd-container \((S, P, e, \bullet, \land, \lor)\) defines an Lmf-container \((S, P, e, \bullet', \land', \lor')\) by \(s \bullet' s' = s \bullet (\lambda_. s')\).

- Any mnd-container morphism is an Lmf-container morphism.

- This gives a faithful functor from \(\textbf{MCont}\) to \(\textbf{LCont}\). This is the functor induced by the lax monoidal functor \(\text{Id}_{\text{Cont}} : (\textbf{Cont}, \text{Id}^c, \cdot^c) \rightarrow (\textbf{Cont}, \text{Id}^c, \otimes^c)\).
**Exception container**

- Let $S = 1 + E$ for some set $E$ and $P(\text{inl } \ast) = 1$, $P(\text{inr } _) = 0$.
- Then $T X \cong X + E$.

If, in an lmf-container structure on $(S, P)$, we had $e = \text{inr } e_0$ for some $e_0 : E$, then $\text{inr } e_0 \bullet \text{inl } \ast = \text{inl } \ast$. But then $q_0 (\text{inr } e_0) (\text{inl } \ast) : 1 \to 0$, which cannot be.

If $e = \text{inl } \ast$, then $\text{inl } \ast \bullet s = s$, $\text{inr } e \bullet \text{inl } \ast = \text{inr } e$, $\text{inr } e \bullet \text{inr } e' = e \otimes e'$ where $\otimes$ must be some semigroup structure on $E$.

The unique mnd-container structure on $(S, P)$ corresponds to the particular case of the left zero semigroup, i.e., the semigroup where $e \otimes e' = e$.  

Lists container

- Let $S = \mathbb{N}$, $P s = [0..s)$. Then $TX \cong \text{List } X$.

- The standard mnd-container structure on $(S, P)$ gives this lmf-container structure:
  
  - $e = 1$
  - $s \bullet s' = s \ast s'$
  - $s' \downarrow_s p = p \text{ div } s'$, $p \uparrow_{s'} s = p \text{ mod } s'$

- The corresponding lax monoidal functor structure on $T$ is
  
  $m^0 \ast = [*], m_{X, Y} (xs, ys) = [(x, y) \mid x \leftarrow xs, y \leftarrow ys]$.

- But we also have, eg, this lmf-container structure:
  
  - $e = 1$
  - $s \bullet s' = s \min s'$
  - $s' \downarrow_s p = p, p \uparrow_{s'} s = p$

- The corresponding lax monoidal functor structure is
  
  $m^0 \ast = [*], m_{X, Y} (xs, ys) = \text{zip}(xs, ys)$. 

Similarly to the mnd-containers case, the list container example can be generalized.

The appropriate generalization is a relaxation of non-symmetric operads where parallel composition is only defined when the given $n$ operations composed with the given $n$-ary operation are all the same, ie, we have $\circ : O \to O \to O$ and $\# (o \circ o') = \# o \ast \# o'$. 
Lmf-containers as lax \((1, \times)\)-universes

- While an mnd-container defines a lax \((1, \Sigma)\)-universe, an Lmf-container defines a lax \((1, \times)\)-universe.

- \(\bullet\) is the \(\times\)-type former.
Containers ∩ commutative monads

- The monad interpreting an mnd-container is commutative (which reduces to the corresponding lax monoidal functor being symmetric) iff
  - $s \bullet (\lambda_. s') = s' \bullet (\lambda_. s)$
  - $(\lambda_. s') \triangleleft_s p = p \trianglerightequation{\lambda_. s} s'$
Containers ∩ Cartesian monads

- The monad interpreting an mnd-container is Cartesian (in the sense that all naturality squares of \( \eta, \mu \) are pullbacks) iff
  - the function \( \lambda_\cdot \ast : P e \to 1 \) is an isomorphism,
  - for any \( s : S, \nu : P s \to S \), the function
    \[
    \lambda p. (\nu \downarrow_s p, p \uparrow_{\nu} s) : P (s \bullet \nu) \to \Sigma p : P s. P (\nu p)
    \]
    is an isomorphism.

- Such mnd-containers are proper \((1, \Sigma)\)-universes.

- With Veltri, we also analyzed a number of other specializations of monads—copy monads, equational lifting monads etc.
Takeaway

- Containers whose interpretation carries a monad or a lax monoidal functor structure admit insightful explicit characterizations as mnd-containers and lmf-containers.

- These explain why set monads and lax monoidal endofunctors have very similar properties (the former also being a special case of the latter).

- Mnd-containers generalize operads, lmf-containers operads with restricted composition.

- Mnd-containers are lax \((1, \Sigma)\) universes, lmf-containers are lax \((1, \times)\) universes.