Monoid Structures on Indexed Containers

17 Apr 2025

Michele De Pascalis (me), Tarmo Uustalu & Niccolò Veltri

Containers and their extent

A *container* $S \triangleleft P$ is given by:

- A set S of *shapes*;
- For every shape $s : S$, a set P_s of *positions*.

For every such container, its *extent* $[\![S \triangleleft P]\!]$ is a set endofunctor, taking:

- Every set X to the set $\sum_{s:S} P_s \to X$;
- Every function $X \to Y$ to the function

$$f' : \left(\sum_{s:S} P_s \to X\right) \to \left(\sum_{s:S} P_s \to Y\right)$$
$$f' \ (s, v) := (s, v \ ; f)$$

Containers arrange in a category Cont, whose objects are containers, and morphisms are given by $\mathsf{Cont}(S \triangleleft P, S' \triangleleft P') := \prod_{s:S} [\![S' \triangleleft P']\!] \ P_s$. The extent is now a full and faithful functor $[\![-]\!]$ in $[\mathsf{Cont}, \mathsf{Endo}(\mathsf{Set})]$.

Monad containers

Given a container $S \triangleleft P$, how many monad instances feature $[\![S \triangleleft P]\!]$?

T. Uustalu [1] gave a combinatorial answer: a monad instance boils down to:

- A unit shape $e : S$;
- A multiplication of shapes $- \bullet - : \prod_{s:S}(P_s \to S) \to S$
- Position projections: for every $s : S, v : P_s \to S, p : P_{s \bullet v}$ there shall be defined:
  - $v \nwarrow_s p : P_s$
  - $p \nearrow_v s : P_{v\,(v \nwarrow_s p)}$

These functions must satisfy some equations as well:

- $e \bullet \lambda\_.s = s \bullet \lambda\_.e = s$;
- $(s \bullet v) \bullet (\lambda p''.w\,(v \nwarrow_s p'')\,(p'' \nearrow_v s)) = s \bullet (\lambda p'.v\,p' \bullet w\,p')$
- $(\lambda\_.e) \nwarrow_s p = p \nearrow_{\lambda\_.s} e = p$
- $v \nwarrow_s ((\lambda p''.w\,(v \nwarrow_s p'')\,(p'' \nearrow_v s)) \nwarrow_{s \bullet v} p) = (\lambda p'.v\,p' \bullet w\,p') \nwarrow_s p$
- … And two more.

Indexed Containers

Fix set I, J. An *indexed container* $S \triangleleft P$ is given by:

- For every $i : I$, a set S_i of *shapes*;
- For every $i : I$, $s : S_i$, $j : J$, a set $P_{s,j}$ of *positions*.

As before, its *extent* $[\![S \triangleleft P]\!]$ is a functor in $[\mathsf{Set}^J, \mathsf{Set}^I]$, taking:

- Every J-indexed set X to the I-indexed set $i \mapsto \sum_{s:S_i} \prod_{j:J} P_{s,j} \to X_j$;
- Every indexed function $\prod_{j:J} X_j \to Y_j$ to the indexed function

$$f' : \prod_{i:I} \left(\sum_{s:S_i} \prod_{j:J} P_{s,j} \to X_j \right) \to \left(\sum_{s:S_i} \prod_{j:J} P_{s,j} \to Y_j \right)$$

$$f' i \,(s, v) := (s, \lambda j.v\, j \,;\, f\, j)$$

We shall omit indices when they can be inferred from the context.

Once again, these arrange in a category $\mathsf{ICont}_{I,J}$, and $[\![-]\!]$ extends to a full and faithful functor.

Monoidal Structure

When $I = J$, extents become endofunctors on Set^I, with the well known strict monoidal structure given by identity and composition. These are in turn isomorphic to extents of indexed containers, and in fact they are reflected by a lax-monoidal structure.

$$\mathsf{I} := (\lambda\_.\ \mathsf{Unit}) \lhd (\lambda i\ \_\ j.i \equiv j)$$

$$S \lhd P \otimes S' \lhd P' := (\llbracket S' \lhd P' \rrbracket(S)) \lhd \left(\lambda(s, v)\ k. \sum_{j:I} \sum_{p:P'_{s,j}} P_{v\,p,k} \right)$$

A few unsurprising lemmas

Lemma *(unsurprising)* $[\![-]\!] : (\mathsf{ICont}_{I,I}, \mathsf{I}, \otimes) \to (\mathsf{Endo}(\mathsf{Set}^I), \mathrm{id}, ;)$ is strong monoidal.

Lemma *(also unsurprising)* Full, faithful and strong monoidal functors reflect monoids.

And since we know that:

Definition *(meme)* A monad is just a monoid in the category of endofunctors.

We can conclude that monads on extents of indexed containers are in bijection with monoids in $(\mathsf{ICont}_{I,I}, \mathsf{I}, \otimes)$.

Indexed monad containers

Analogously to monad containers, they comprise:

- A family of unit shapes $e : \prod_{i:I} S_i$;
  - such that $P_{e_i,j}$ is only (possibly) inhabited if $i \equiv j$;
- A multiplication of shapes $- \bullet - : \prod_{i:I} \prod_{s:S_i} \left(\prod_{j:I} P_{s,j} \to S_j\right) \to S_i$
- Position projections: for every $i : I$, $s : S_i$, $v : \prod_{j:I} P_{s,j} \to S_j$, $j : I$, $p : P_{s \bullet v,j}$ there shall be defined:
  - $v \uparrow p : I$;
  - $v \nwarrow p : P_{s,v\uparrow p}$
  - $v \nearrow p : P_{v\,(v \nwarrow p),j}$

They have to satisfy similar equations to the non-indexed ones, plus:

- $(\lambda q.w\,(v \nwarrow q)(v \nearrow q)) \uparrow p \equiv w\,((\lambda q.v\,q \bullet w\,q) \nwarrow p) \uparrow ((\lambda q.v\,q \bullet w\,q) \nearrow p)$;
- $v \uparrow ((\lambda q.w\,(v \nwarrow q)(v \nearrow q)) \nwarrow p) \equiv (\lambda q.v\,q \bullet w\,q) \uparrow p$;

Example: Indexed Writer

Given a Set-monoid $(W, \varepsilon, \cdot)$, and a W-action $(- \blacktriangleright -)$ on I, we can define a Set^I endofunctor as follows:

$$\mathsf{Wr}^{\blacktriangleright} X i := \sum_{w:W} X_{w \blacktriangleright i}$$

This is a generalization of the well known writer monad, isomorphic to the extent of the container $(\lambda\_.W) \triangleleft (\lambda i\ w\ j.w \blacktriangleright i \equiv j)$. An appropriate monad structure is described by:

$$e_i := \varepsilon \qquad w \bullet w' := w \cdot w'$$

$$\_ \uparrow_{i,w,j} \_ := w \blacktriangleright i \qquad \_ \nwarrow_{i,w,j} \_ := \mathsf{refl}$$

$$\_ \nearrow_{i,w,j} \_ := (\ \blacktriangleright \text{ preserves } \cdot\ )$$

The constraint on $P_{e_i,j}$ is granted by $\blacktriangleright$ respecting ε, while the other constraints are either trivial or granted by the monoid structure on W.

About the formalization

Probably the most relevant slide for everyone here. This is all formalized in https://github.com/mikidep/indexed-containers (contains Unicode crimes).

Several subtleties of working formally with indexed containers in Cubical Agda emerged. Let's discuss after the talk if you're into this sort of stuff.

Conclusions and future works

- We should soon be able to describe cartesian monads and monad morphisms in this framework.
- We would like to use this to rule out monad candidates (ideas?)
- The free monad on the extent of an indexed container is represented by an indexed container as well.
- There are another couple instances that we expect to see (not proven yet).
- We plan to figure out groupoid containers and extend this approach there (cf. Philipp's talk).

Thank you!

References

[1] T. Uustalu, "Container Combinatorics: Monads and Lax Monoidal Functors," in *Lecture Notes in Computer Science*, Springer International Publishing, 2017, pp. 91–105. doi: 10.1007/978-3-319-68953-1_8.