Indexed Containers

Tallinn

Peter Morris

pwm@cs.nott.ac.uk

University of Nottingham

What is a Container?

Barry Jay—Shapely Types

Joyal—Species

Hyland and Gambino—Polynomial Functors

Petersson and Synek—Tree Sets

All a means to have an general notion of data-type. In order to develop some theory of data.

All include some separation of the *shape* of data and the storage of information.

What is a Container? (2)

A container has a set of shapes S.

$$\underline{\text{data}}\ \frac{A : \star}{[A] : \star}\ \underline{\text{where}}\ \frac{}{[] : [A]}\quad \frac{a : A \quad as : [A]}{a :: as : [A]}$$

If we abstract away for the business of storing data, for instance by considering only the type [One], we get something suspiciously like the natural numbers.

The *shape* of a list, is given by its *length*.

What is a Container? (3)

If we know the shape of a list, we know how many pieces of data are stored within it. A list of length n contains n pieces of data.

More generally we'll want to construct the set of *positions* $P\ s$ where data is stored within a piece of data with the shape s.

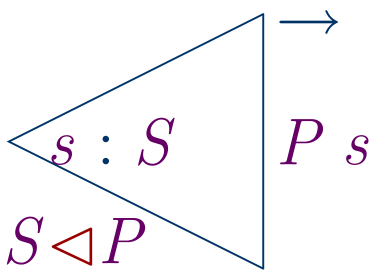
For lists the positions can be given by *finite sets*:

$$\underline{\text{data}}\ \frac{n : \mathsf{Nat}}{\mathsf{Fin}\ n : \star}\ \underline{\text{where}}\ \frac{}{\mathsf{fzero} : \mathsf{Fin}\ (1{+}\ n)}\ \frac{i : \mathsf{Fin}\ n}{\mathsf{fsucc}\ i : \mathsf{Fin}\ (1{+}\ n)}$$

Containers

$$\underline{\text{data}}\ \frac{}{\mathsf{Cont}\ :\ \star}\ \underline{\text{where}}\ \frac{S\ :\ \star\quad P\ :\ S \to \star}{S \lhd P\ :\ \mathsf{Cont}}$$

It's often useful to draw diagrams of these things:



Container Functors

Every container gives rise to a functor given by:

$$\begin{aligned}(S \triangleleft P)\ X &= \Sigma(s : S).(P\ s \to X)\\ (S \triangleleft P)\ f\ (s, g) &= (s, g \circ f)\end{aligned}$$

For the objects, we pick a shape s and then assign a piece of data to every position $p : P\ s$.

With lists, once we've fixed a length n, $\mathsf{Fin}\ n \to X$ is a collection of n Xs.

Container Morphisms

A container morphism is given by:

$$\underline{\text{data}}\ \frac{C, D\ :\ \mathsf{Cont}}{\mathsf{CMor}\ C\ D\ :\ \star}\ \underline{\text{where}}\ \frac{\begin{array}{l} sf\ :\ S \rightarrow T \\ pf\ :\ (s\ :\ S) \rightarrow Q\ (f\ s) \rightarrow P\ s \end{array}}{\mathsf{cmor}\ sf\ pf\ :\ \mathsf{CMor}\ (S \triangleleft P)\ (T \triangleleft Q)}$$

A function between shapes and a contravariant function on positions. We have to explain where the data in the new container came from in the old container.

This notion captures precisely the polymorphic functions between container functors.

Closure Properties

$$(\mathbf{K}\ T)\ X \simeq (T \lhd \lambda\_ \to \mathsf{Zero})\ X$$

$$\mathbf{Id}\ X \simeq (\mathsf{One} \lhd \lambda\_ \to \mathsf{One})\ X$$

$$(S \lhd P)\ X + (T \lhd Q)\ X \simeq (S + T \lhd [P, Q])\ X$$

$$(S \lhd P)\ X \times (T \lhd Q)\ X \simeq (S \times T \lhd \lambda(s, t) \to P\ s + Q\ t)\ X$$

Constructing Fixed Points

The University of Nottingham

Binary containers have 2 position sets.

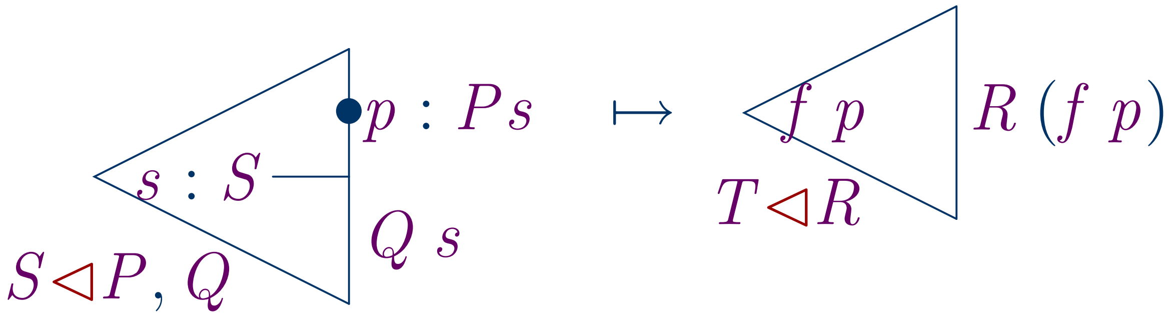
They allow us to encode composition:

$$\frac{F : \star \times \star \to \star \quad G : \star \to \star}{F[G] : \star \to \star} \; ; \quad F[G]\ X = F\ (X, (G\ X))$$

$$(S \lhd P, Q)[T \lhd R]\ X \simeq$$
$$((S \lhd Q)\ T) \lhd \lambda(s, f) \to P\ s + \Sigma(q : Q\ s).R\ (f\ s)$$

Constructing Fixed Points (2)

$(S \lhd P, Q)[T \lhd R]$ diagrammatically:



Now, we want to construct the least fixed point such that:

$$\mu\;(S \lhd P, Q)\;X \simeq (S \lhd P, Q)[\mu\;(S \lhd P, Q)]\;X$$

Constructing Fixed Points (3)

The shapes are given by the least fixed point of the equation:

$$X = (S \lhd Q)\ X$$

i.e. the lfp of a unary container. Type theoreticians know this as a W-Type:

$$\underline{\text{data}}\ \frac{A : \star \quad B : A \to \star}{\mathsf{W}\ A\ B : \star}\ \underline{\text{where}}\ \frac{a : A \quad f : B\ a \to \mathsf{W}\ A\ B}{\mathsf{sup}\ a\ f : \mathsf{W}\ A\ B}$$

The shapes of the lfp are trees of shapes branching over the sub-tree positions.

Constructing Fixed Points (4)

We can then define the positions as *paths* through such a tree to a payload position. This can be done recursively:

$$\mathbf{Path}\ P\ Q\ (s, f) = P\ s\ +\ \Sigma(q\ :\ Q\ s).\mathbf{Path}\ P\ Q\ (f\ q)$$

So, to interpret a parameterised least fixed point construction, we need the least fixed point of a unary container to exist (W-types). It can also be show that the parameterised greatest fixed point exists under the same conditions.

Advanced types

How should we go about modelling types like the vectors or the well scoped lambda terms?

$$\underline{\text{data}}\ \frac{n : \mathsf{Nat} \quad A : \star}{\mathsf{Vec}\ n\ A : \star}\ \underline{\text{where}}$$

$$\frac{}{[\,] : \mathsf{Vec}\ 0\ A} \quad \frac{a : A \quad as : \mathsf{Vec}\ n\ A}{a :: as : \mathsf{Vec}\ (1+\ n)\ A}$$

$$\underline{\text{data}}\ \frac{n : \mathsf{Nat}}{\mathsf{Lam}\ n : \star}\ \underline{\text{where}}$$

$$\frac{i : \mathsf{Fin}\ n}{\mathsf{var}\ i : \mathsf{Lam}\ n} \quad \frac{f, a : \mathsf{Lam}\ n}{f \$ a : \mathsf{Lam}\ n} \quad \frac{t : \mathsf{Lam}\ (1+\ n)}{\lambda\ t : \mathsf{Lam}\ n}$$

Or Fin, or W itself?

Indexed Containers

Firstly, we need to index the shapes:

$$\underline{\text{data}}\ \frac{O\ :\ \star}{\mathsf{Cont}\ O\ :\ \star}\ \underline{\text{where}}\ \frac{\begin{array}{l} S\ :\ O \to \star \\ P\ :\ (o\ :\ O) \to S\ o \to \star \end{array}}{S \lhd P\ :\ \mathsf{Cont}\ O}$$

We can now model the vectors by:

$$(\lambda n \to \mathsf{One}) \lhd \lambda n\ s \to \mathsf{Fin}\ n\ :\ \mathsf{Cont\ Nat}$$

Indexed Containers

For symmetry, though, we would like that the payload can also be indexed. The way we set it up here we have a function that assigns to every position, an index:

$$\underline{\text{data}}\ \ \frac{I, O\ :\ \star}{\mathsf{Cont}\ I\ \ O\ :\ \star}\ \ \underline{\text{where}}\ \ \frac{\begin{array}{l} S\ :\ O \to \star \\ P\ :\ (o\ :\ O) \to S\ o \to \star \\ \phi\ :\ (o\ :\ O) \to S\ o \to P\ s \to I \end{array}}{S \lhd P\ \phi\ :\ \mathsf{Cont}\ I\ \ O}$$

Indexed Functors

An indexed container in Cont $I\ O$ gives rise to a indexed functor in $(I \to \star) \to (O \to \star)$ given by:

$$
\begin{aligned}
(S \lhd P\ \phi)\ X\ o \ &= \ \Sigma(s : S\ o).(p : P\ o\ s) \to X\ (\phi\ o\ s\ p) \\
(S \lhd P\ \phi)\ f\ (s, g) &= (s, g \circ f)
\end{aligned}
$$

We exposed a polynomial structure on plain containers, but what structure is appropriate for indexed containers?

Constructing advanced types

We can re-index functors, which amounts to pre-composition:

$$\frac{F \;:\; (I \to \star) \to (O \to \star) \qquad f \;:\; O' \to O}{\mathbf{\Delta} F\ f \;:\; (I \to \star) \to (O' \to \star)}$$

$$\mathbf{\Delta} F\ f\ X\ o' = F\ (f\ o')$$

For containers that becomes:

$$\mathbf{\Delta} (S \lhd P\ \phi)\ f\ X\ o' \simeq (S\ (f\ o')) \lhd (P\ (f\ o'))\ (\phi\ (f\ o'))$$

Constructing advanced types

The University of Nottingham

$\mathbf{\Delta}$ has left and right adjoints, which relate to Sigma and Pi. Here $(S \lhd P\ \phi)\ :\ \mathsf{Cont}\ I\ O'$ and $f\ :\ O' \to O$:

$$\mathbf{\Sigma}(S \lhd P\ \phi)\ f\ X\ o \simeq$$
$$(\Sigma(o'\ :\ O').(f\ o' = p) \times S\ o') \lhd$$
$$(\lambda(o', eq, s) \to P\ o'\ s)\ (\lambda(o', eq, s)\ p \to \phi\ o'\ s\ p)$$

$$\mathbf{\Pi}(S \lhd P\ \phi)\ f\ X\ o' \simeq$$
$$((o'\ :\ O') \to (f\ o = o') \to S\ o')$$
$$\lhd(\lambda o'\ f \to \Sigma(o'\ :\ O'), (eq\ :\ f\ o = o').P\ o\ (f\ o\ eq))\ \cdots$$

Indexed fixed points

We want the least fixed point of a container in $\mathsf{Cont}\ (I + O)\ O$ to be in $\mathsf{Cont}\ I\ O$. We first construct *indexed* W-types:

$$\underline{\text{data}}\quad \frac{\begin{array}{c} A : X \to \star \quad x : X \quad B : (x : X) \to A\ x \to \star \\ \phi : (x : X) \to (a : A\ x) \to B\ x\ a \to X \end{array}}{\mathsf{W}_X\ A\ x\ B\ \phi : \star}\quad \underline{\text{where}}$$

$$\frac{a : A\ x \quad f : (b : B\ x\ a) \to \mathsf{W}_X\ A\ (\phi\ x\ a\ b)\ B}{\mathsf{supi}\ a\ f : \mathsf{W}_X\ A\ x\ B\ \phi}$$