The Specification of a Generic Zip

Roland Backhouse and Paul Hoogendijk

Generic Programming Summer School

Oxford, August 2002

(Lower Order) Naturality

$$\mathsf{zip.F.G} \quad : \quad \mathsf{G \bullet F} \leftarrow \mathsf{F \bullet G} \quad .$$

A zip is a *proper* natural transformation.

A zip transforms one structure to another without loss or duplication of values.

(Higher Order) Naturality

$$\mathsf{zip.F} \;:\; (\bullet\mathsf{F}) \leftarrow (\mathsf{F}\bullet) \quad .$$

Categorical Nat Trans (Revision)

A natural transformation is an arrow in the functor category. I.e.,

$$\eta : F \leftarrow G$$

means that the following diagram commutes (for all A, B and $f : A \leftarrow B$)

$$\begin{tikzcd}
F.A & G.A \arrow[l, "\eta_A"'] \\
F.B \arrow[u, "F.f"] & G.B \arrow[l, "\eta_B"] \arrow[u, "G.f"']
\end{tikzcd}$$

Now, if F is a functor, $(\bullet F)$ and $(F\bullet)$ are endofunctors on the functor category.

$(\bullet F)$ maps functor (object) G to $G \bullet F$ and natural transformation (arrow) η to $\eta \bullet F$, where $(\eta \bullet F)_A = \eta_{F.A}$.

$(F\bullet)$ maps functor (object) G to $F \bullet G$ and natural transformation (arrow) η to $F \bullet \eta$, where $(F \bullet \eta)_A = F.(\eta_A)$.

Categorical NT Revision (Continued)

Diagram defining $\eta : F \leftarrow G$

$$\begin{tikzcd} F.A & G.A \arrow[l, "\eta_A"'] \\ F.B \arrow[u, "F.f"] & G.B \arrow[l, "\eta_B"] \arrow[u, "G.f"'] \end{tikzcd}$$

instantiated for $\mathsf{zip}.F : (\bullet F) \leftarrow (F\bullet)$

$$\begin{tikzcd} G\bullet F & F\bullet G \arrow[l, "\mathsf{zip}.F.G"'] \\ H\bullet F \arrow[u, "\theta\bullet F"] & F\bullet H \arrow[l, "\mathsf{zip}.F.H"] \arrow[u, "F\bullet\theta"'] \end{tikzcd}$$

where $\theta : G \leftarrow H$ is a natural transformation.

Allegorical Naturality

Recall that parametricity was defined in terms of *relations*.

Recall also that, in the particular case that t has type $\langle\forall\alpha :: \mathsf{F}.\alpha \leftarrow \mathsf{G}.\alpha\rangle$, t is parametric is equivalent to t is a natural transformation (in the underlying category of maps).

This is a stroke of luck for functional programmers, BUT their luck has run out!

The equality in

$$(\theta\bullet\mathsf{F})\circ\mathsf{zip.F.H} \quad = \quad \mathsf{zip.F.G}\circ(\mathsf{F}\bullet\theta)$$

is too severe — because

- θ may be nondeterministic.
- Zips are partial.

Nondeterminism

Take $\mathsf{F} := \mathsf{List}$ and $\mathsf{G} = \mathsf{H} := \times$.

$\mathsf{zip.F.H}$ and $\mathsf{zip.F.G}$ are both the inverse of conventional zips. They unzip a list of pairs to a pair of lists.

Take $\theta := \mathsf{id} \cup \mathsf{swap}$.

θ nondeterministically swaps the elements of a pair or not.

$(\theta \bullet \mathsf{F}) \circ \mathsf{zip.F.H}$ unzips a list of pairs into a pair of lists and swaps the lists or not.

$\mathsf{zip.F.G} \circ (\mathsf{F} \bullet \theta)$ first swaps some of the elements of a list of pairs and then unzips it into a pair of lists.

$$(\theta \bullet \mathsf{F}) \circ \mathsf{zip.F.H} \quad \subset \quad \mathsf{zip.F.G} \circ (\mathsf{F} \bullet \theta) \quad .$$

Partiality

$$\begin{tikzcd}
\mathsf{List}.(\mathsf{List}.A) & \mathsf{Tree}.(\mathsf{List}.A) \arrow[l, "\mathsf{listify}_{\mathsf{List}.A}"'] \\
\mathsf{List}.(\mathsf{List}.A) \arrow[u, "(\mathsf{zip}.\mathsf{List}.\mathsf{List})_A"] & \mathsf{List}.(\mathsf{Tree}.A) \arrow[l, "\mathsf{List}.(\mathsf{listify}_A)"] \arrow[u, "(\mathsf{zip}.\mathsf{List}.\mathsf{Tree})_A"']
\end{tikzcd}$$

View both paths through the diagram as partial relations of type $\mathsf{List}.(\mathsf{List}.A) \leftarrow \mathsf{List}.(\mathsf{Tree}.A)$.

The lower path (via $\mathsf{List}.(\mathsf{List}.A)$) includes the upper path (via $\mathsf{Tree}.(\mathsf{List}.A)$).

Reason: for the lower path, the sizes of the trees must be the same; for the upper path, the trees must have the same shape.

$\mathsf{zip.F}$ is parametric.

That is, for all $\theta : \mathsf{G} \hookleftarrow \mathsf{H}$,

$$(\theta{\bullet}\mathsf{F}) \circ \mathsf{zip.F.H} \;\subseteq\; \mathsf{zip.F.G} \circ (\mathsf{F}{\bullet}\theta) \quad .$$

Compositionality

Informally, zip.F is a monoid homomorphism.

(Note: more than this: zip.F should respect pointwise extension of relators. For full discussion see Hoogendijk's thesis.)

$$
\begin{tikzcd}
\mathsf{G{\bullet}H{\bullet}F} & & \mathsf{F{\bullet}G{\bullet}H} \arrow[ll, "\mathsf{zip.F.(G{\bullet}H)}"'] \arrow[dl, "\mathsf{zip.F.G} \bullet \mathsf{H}"] \\
& \mathsf{G{\bullet}F{\bullet}H} \arrow[ul, "\mathsf{G} \bullet \mathsf{zip.F.H}"] &
\end{tikzcd}
$$

$$\mathsf{zip.F.(G{\bullet}H)} \quad = \quad (\mathsf{G} \bullet \mathsf{zip.F.H}) \circ (\mathsf{zip.F.G} \bullet \mathsf{H}) \quad .$$

$$\mathsf{zip.F.Id} \quad = \quad \mathsf{id{\bullet}F} \quad .$$

Zips

Definition 1 (Half Zip) Consider a fixed relator F and a pointwise closed class of relators $\mathcal{G}$. Then the members of the collection $\mathsf{zip.F.G}$, where G ranges over $\mathcal{G}$, are called *half-zips* iff

(a) $\mathsf{zip.F.G} : \mathsf{G}\bullet\mathsf{F} \leftarrow \mathsf{F}\bullet\mathsf{G}$, for each G in $\mathcal{G}$,

(b) $(\theta\bullet\mathsf{F})\circ\mathsf{zip.F.H} \subseteq \mathsf{zip.F.G}\circ(\mathsf{F}\bullet\theta)$ for each $\theta : \mathsf{G} \hookleftarrow \mathsf{H}$,

(c) $\mathsf{zip.F.(G\bullet H)} = (\mathsf{G}\bullet\mathsf{zip.F.H})\circ(\mathsf{zip.F.G}\bullet\mathsf{H})$ for all G and H,

(d) $\mathsf{zip.F.Id} = \mathsf{id}\bullet\mathsf{F}$.

□

Definition 2 (Commuting Relators) The half-zip $\mathsf{zip.F.G}$ is said to be a *zip* of (F, G) if there exists a half-zip $\mathsf{zip.G.F}$ such that

$$\mathsf{zip.F.G} = (\mathsf{zip.G.F})^{\cup}$$

We say that datatypes F and G *commute* if there exists a zip for (F, G).

□

Constructing Zips

See Hoogendijk's thesis for how these are calculated:

$$
\begin{aligned}
\mathsf{zip}.\mathsf{K}_A.\mathsf{G} &= \mathsf{fan}.\mathsf{G} \bullet \mathsf{K}_A \quad , \\
\mathsf{zip}.+.\mathsf{G} &= \mathsf{G}.\mathsf{inl} \triangledown \mathsf{G}.\mathsf{inr} \quad , \\
\mathsf{zip}.\times.\mathsf{G} &= (\mathsf{G}.\mathsf{outl} \vartriangle \mathsf{G}.\mathsf{outr})^{\cup} \quad , \\
\mathsf{zip}.\mathsf{T}.\mathsf{G} &= (\!|\mathsf{id}_{\mathsf{G}} \otimes ;\ \mathsf{G}.\mathsf{in} \circ (\mathsf{zip}.\otimes.\mathsf{G} \bullet \mathsf{Id} \Delta \mathsf{T})|\!) \quad .
\end{aligned}
$$

where T is the tree relator with pattern relator $\otimes$.

$$
\begin{aligned}
\mathsf{fan}.\mathsf{K}_A &= \top\!\!\top_{A,\_} \\
\mathsf{fan}.+ &= (\mathsf{id} \triangledown \mathsf{id})^{\cup} \\
\mathsf{fan}.\times &= \mathsf{id} \vartriangle \mathsf{id} \\
\mathsf{fan}.\mathsf{T} &= (\!|\mathsf{id} \otimes ;\ (\mathsf{fan}.\otimes)^{\cup}|\!)^{\cup}
\end{aligned}
$$

where T is the tree relator with pattern relator $\otimes$.