Generic Properties of Datatypes

Roland Backhouse and Paul Hoogendijk

Generic Programming Summer School

Oxford, August 2002

Outline

- Theorems For Free
- Commuting Datatypes (“Zips”)
- Relators, Fans and Membership
- Properties of Zips
- Conclusion

Parametric Polymorphism

Summary: parametric polymorphism is a verifiable form of (type) genericity.

Common Type = Common Properties

$$\mathsf{length} : \langle \forall \alpha :: \mathbb{N} \leftarrow \mathsf{List}.\alpha \rangle$$

For all types A and B and all functions f of type $A \leftarrow B$,

$$\mathsf{length}_A \circ \mathsf{List}.f \;=\; \mathsf{length}_B \quad .$$

Let sq denote the function that squares a number.

$$sq \circ \mathsf{length} : \langle \forall \alpha :: \mathbb{N} \leftarrow \mathsf{List}.\alpha \rangle$$

$$(sq \circ \mathsf{length}_A) \circ \mathsf{List}.f \;=\; sq \circ \mathsf{length}_B \quad .$$

Suppose $copycat$ appends a copy of a list to itself.

$$\mathsf{length} \circ copycat : \langle \forall \alpha :: \mathbb{N} \leftarrow \mathsf{List}.\alpha \rangle$$

$$(\mathsf{length}_A \circ copycat_A) \circ \mathsf{List}.f \;=\; \mathsf{length}_B \circ copycat_B \quad .$$

Polymorphism

Consider the type expressions defined by the following grammar:

$$\mathsf{Exp} \quad ::= \quad \mathsf{Exp} \times \mathsf{Exp} \;\mid\; \mathsf{Exp} \leftarrow \mathsf{Exp} \;\mid\; \mathsf{Const} \;\mid\; \mathsf{Var} \quad .$$

Here, Const denotes a set of constant types, like $\mathbb{N}$ (the natural numbers) and $\mathbb{Z}$ (the integers). Var denotes a set of type variables. We use Greek letters to denote type variables.

A term t is said to have *polymorphic* type $\langle\forall\alpha :: \mathsf{T}.\alpha\rangle$, where T is a type expression parameterised by type variables α, if t assigns to each type A a value t_A of type $\mathsf{T}.A$.

Mapping Relations to Relations

Type expressions are extended to denote functions from relations to relations.

$$R\times S : A\times B \sim C\times D \quad \Leftarrow \quad R : A\sim C \ \wedge \ S : B\sim D$$

$$((a,b)\ ,\ (c,d)) \in R\times S \quad \equiv \quad (a,c)\in R \ \wedge \ (b,d)\in S \quad .$$

$$R\leftarrow S : (A\leftarrow B) \sim (C\leftarrow D) \quad \Leftarrow \quad R : A\sim C \ \wedge \ S : B\sim D$$

$$(f,g) \in R\leftarrow S \quad \equiv \quad \langle \forall\, b,d :: (f.b\ ,\ g.d)\in R \Leftarrow (b,d)\in S \rangle \quad .$$

The constant type A is read as the identity relation id_A on A.

$$(x\ ,\ y)\in A \quad \equiv \quad x=y \quad .$$

Example

$$R{\times}R{\leftarrow}R \;:\; (A{\times}A \leftarrow A) \sim (B{\times}B \leftarrow B) \quad \Leftarrow \quad R : A \sim B$$

$$
\begin{array}{ll}
 & (f, g) \in R{\times}R{\leftarrow}R \\
= & \{ \quad \text{definition of} \leftarrow \text{on relations} \quad \} \\
 & \langle \forall\, a,b :: (f.a\,,\, g.b) \in R{\times}R \;\Leftarrow\; (a, b) \in R \rangle \\
= & \{ \quad \text{definition of} \times \text{on relations} \quad \} \\
 & \langle \forall\, a,b :: \\
 & \qquad (\mathsf{fst}.(f.a)\,,\, \mathsf{fst}.(g.b)) \in R \;\wedge\; (\mathsf{snd}.(f.a)\,,\, \mathsf{snd}.(g.b)) \in R \\
 & \qquad \Leftarrow (a, b) \in R \\
 & \rangle
\end{array}
$$

Example

$$\mathsf{id}_{\mathrm{Bool}} \leftarrow R\times R \;:\; (\mathrm{Bool} \leftarrow A\times A) \sim (\mathrm{Bool} \leftarrow B\times B) \quad \Leftarrow \quad R : A \sim B$$

$$
\begin{array}{ll}
 & (f,g) \;\in\; \mathsf{id}_{\mathrm{Bool}} \leftarrow R\times R \\
= & \qquad \{ \qquad \text{definition of } \leftarrow \text{ and } \times \text{ on relations} \quad \} \\
 & \langle \forall\, a,a',b,b' :: (f.(a,a')\,,\, g.(b,b')) \in \mathsf{id}_{\mathrm{Bool}} \;\Leftarrow\; (a,b) \in R \wedge (a',b') \in R \rangle \\
= & \qquad \{ \qquad \text{definition of } \mathsf{id}_{\mathrm{Bool}} \quad \} \\
 & \langle \forall\, a,a',b,b' :: f.(a,a') = g.(b,b') \;\Leftarrow\; (a,b) \in R \wedge (a',b') \in R \rangle
\end{array}
$$

Parametric

A term t of polymorphic type $\langle\forall\alpha :: T.\alpha\rangle$ is said to be *parametrically polymorphic* if, for each instantiation of relations R to type variables, $(t_A\ ,\ t_B) \in T.R$, where R has type $A \sim B$.

$$\mathsf{fst} : \langle\forall\, \alpha,\beta :: \alpha \leftarrow \alpha\times\beta\rangle$$

Suppose $R : A \sim B$ and $S : C \sim D$.

$$(\mathsf{fst}_{A,C}\ ,\ \mathsf{fst}_{B,D}) \ \in\ R \leftarrow R\times S$$

$=$ { definition of $\leftarrow$ and $\times$ on relations }

$$\langle\forall\, a,b,c,d :: (\mathsf{fst}_{A,C}.(a,c)\ ,\ \mathsf{fst}_{B,D}.(b,d)) \in R \ \Leftarrow\ (a,b) \in R \wedge (c,d) \in S\rangle$$

$=$ { definition of fst }

$$\langle\forall\, a,b,c,d :: (a,b) \in R \ \Leftarrow\ (a,b) \in R \wedge (c,d) \in S\rangle$$

$=$ { calculus }

$$\mathsf{true}$$

Ad Hoc Polymorphism

Suppose "==" denotes a polymorphic "equality" operator. That is,

$$== \;:\; \langle\forall\alpha :: \mathsf{Bool} \leftarrow \alpha\times\alpha\rangle$$

$$\begin{array}{ll}
& == \text{ is parametric} \\
= & \{ \quad \text{definition } (R \text{ ranges over relations of type } A{\leftarrow}B) \quad \} \\
& \langle\forall R :: (==_A\,,\,==_B) \;\in\; \mathsf{id}_{\mathtt{Bool}} \leftarrow R\times R\rangle \\
= & \{ \quad \text{definition of } \leftarrow \text{ and } \times \text{ on relations, and of } \mathsf{id}_{\mathtt{Bool}} \quad \} \\
& \langle\forall R :: \langle\forall u,v,x,y :: (u ==_A v) = (x ==_B y) \;\Leftarrow\; (u,x)\in R \wedge (v,y)\in R\rangle\rangle \\
\Rightarrow & \{ \quad \text{take } R \text{ to be an arbitrary function } f \\
& \quad\; (\text{so } (u,x)\in R \equiv u = f.x \text{ and } (v,y)\in R \equiv v = f.y \quad \} \\
& \langle\forall f :: \langle\forall x,y :: (f.x ==_A f.y) = (x == y)\rangle\rangle
\end{array}$$

Conclusion: all functions in the language of terms are injective, or "equality" is not both real equality and parametric.

Commuting Datatypes

Roland Backhouse and Paul Hoogendijk

Generic Programming Summer School
Oxford, August 2002

Introductory Examples

Zip (of lists)

$$([a_1, a_2, \ldots, a_n], [b_1, b_2, \ldots, b_n]) \mapsto [(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)]$$

$$\mathsf{Pair} \cdot \mathsf{List} \qquad \mapsto \qquad \mathsf{List} \cdot \mathsf{Pair}$$

Matrix Transposition

$$\mathsf{List} \cdot \mathsf{List} \qquad \mapsto \qquad \mathsf{List} \cdot \mathsf{List}$$

Broadcast

$$(a, [b_1, b_2, \ldots, b_n]) \mapsto [(a, b_1), (a, b_2), \ldots, (a, b_n)]$$

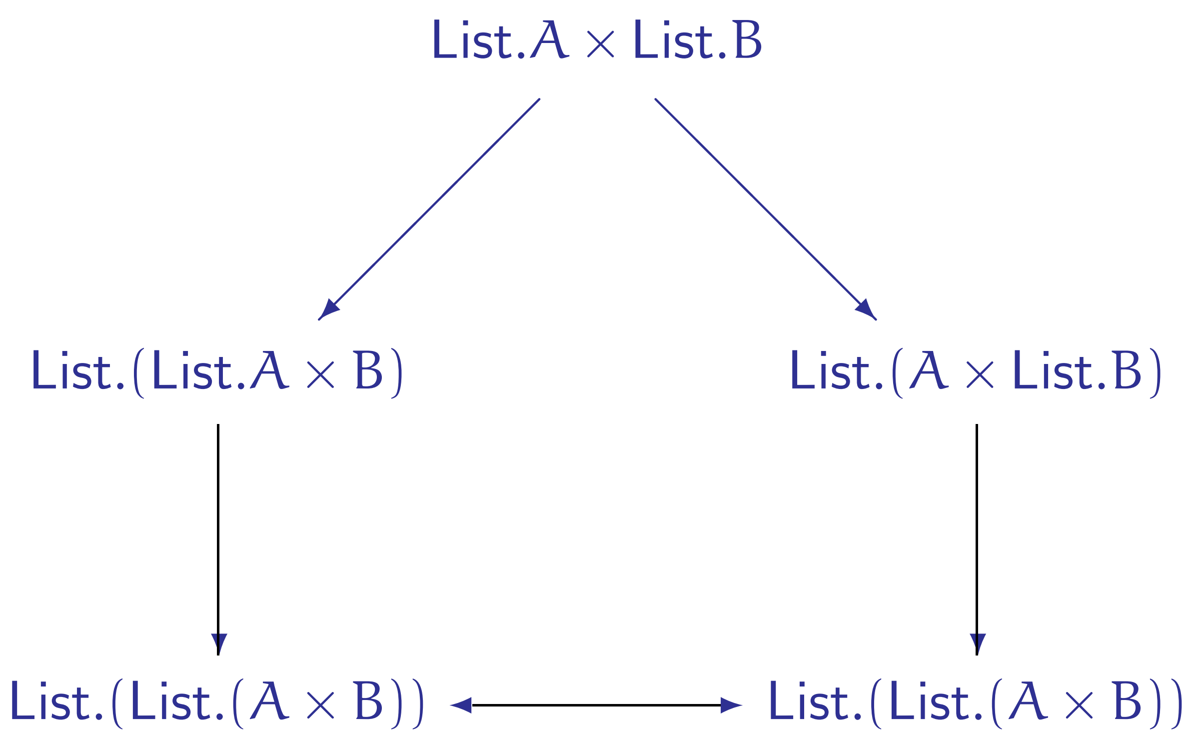
$$A\times \cdot \mathsf{List} \qquad \mapsto \qquad \mathsf{List} \cdot A\times$$

Primitive

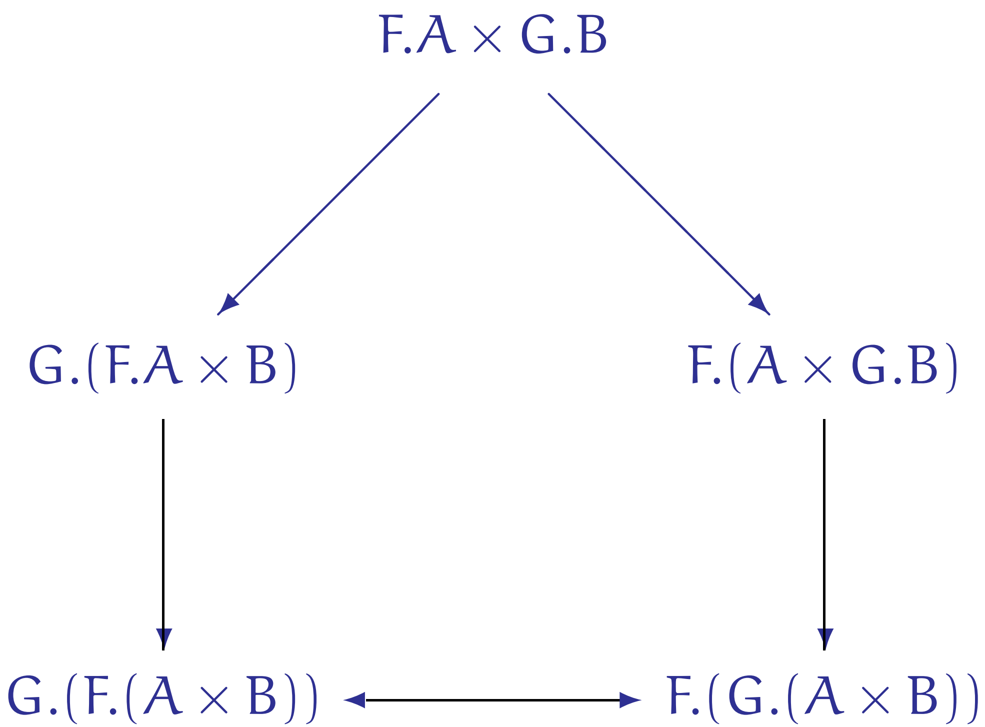
$$(A+B) \times (C+D) \mapsto (A\times C) + (B\times D)$$

$$\times \cdot + \qquad \mapsto \qquad + \cdot \times$$

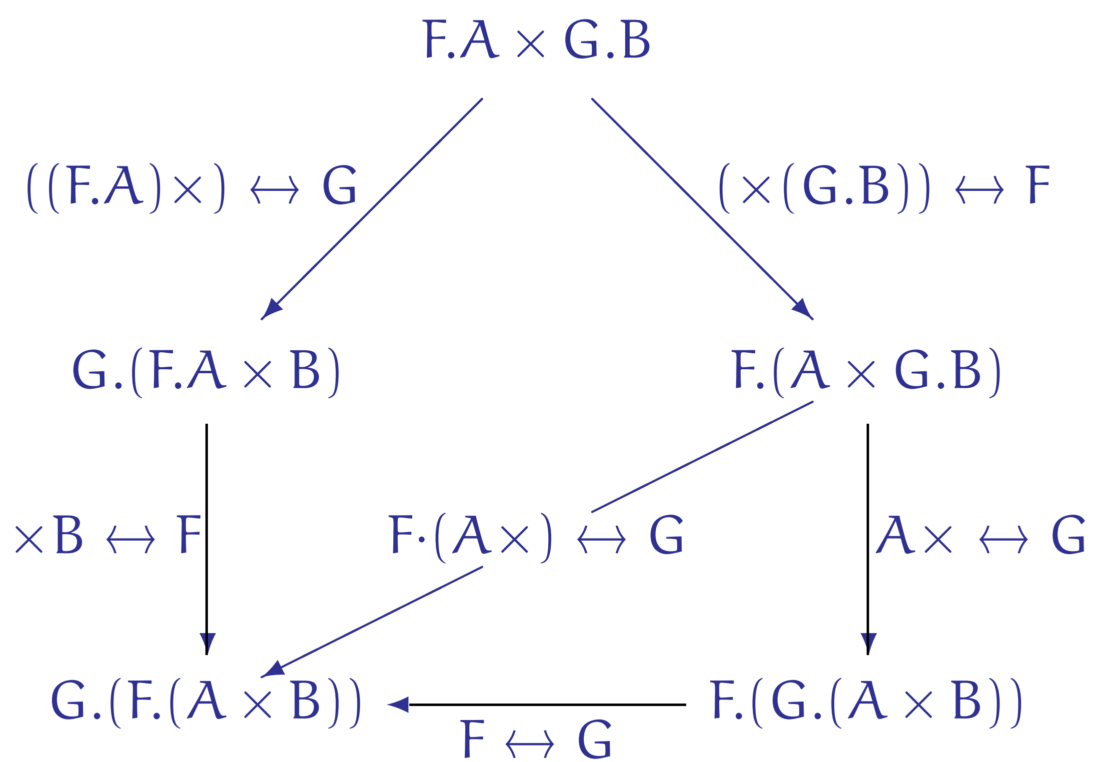
Structure Multiplication ...



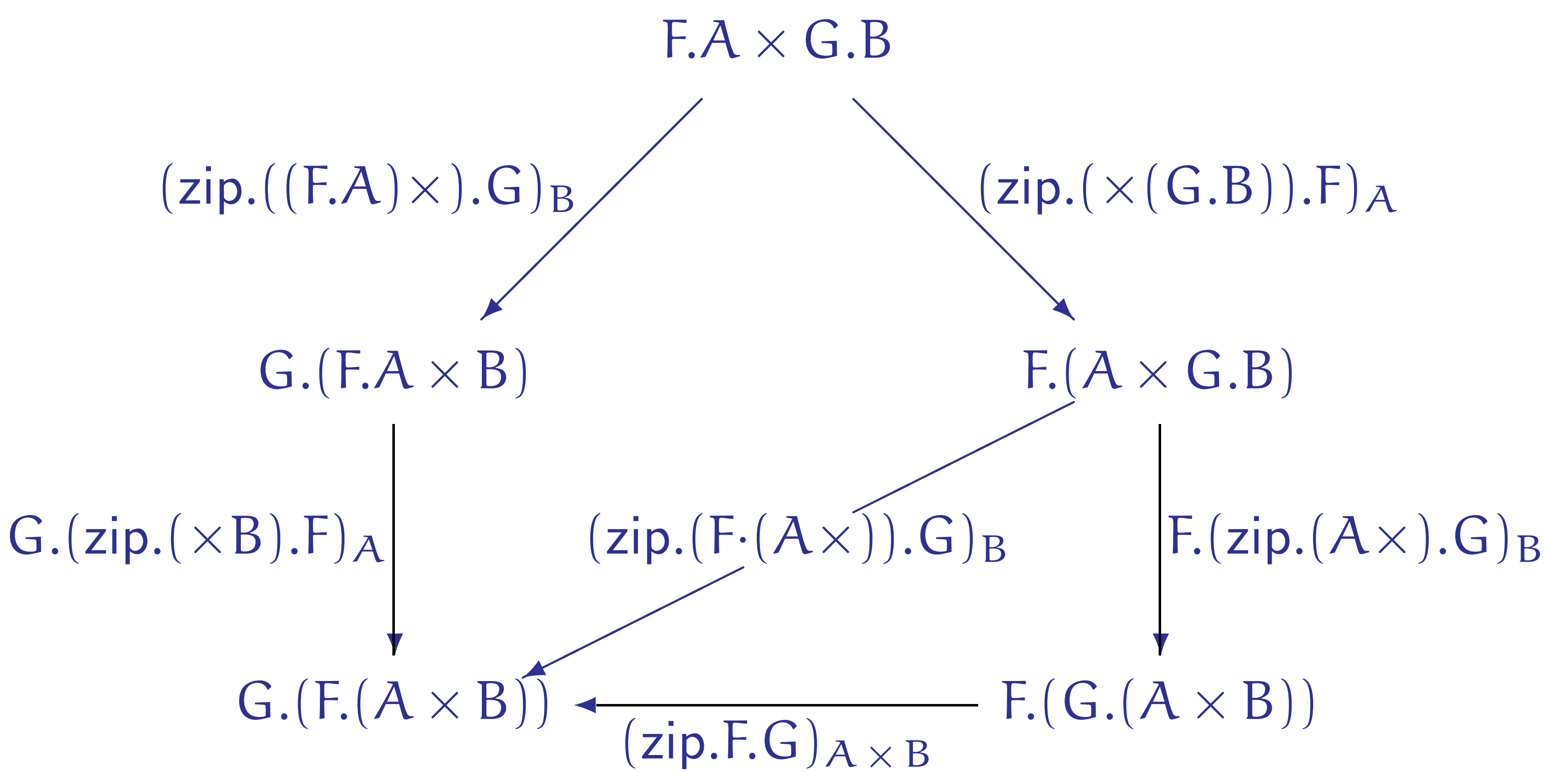
... Generalised ...



... Illustrates Generic Requirements



Multi-Coloured Zips



Broadcasts ...

A broadcast copies a given value across all storage locations of a datatype.

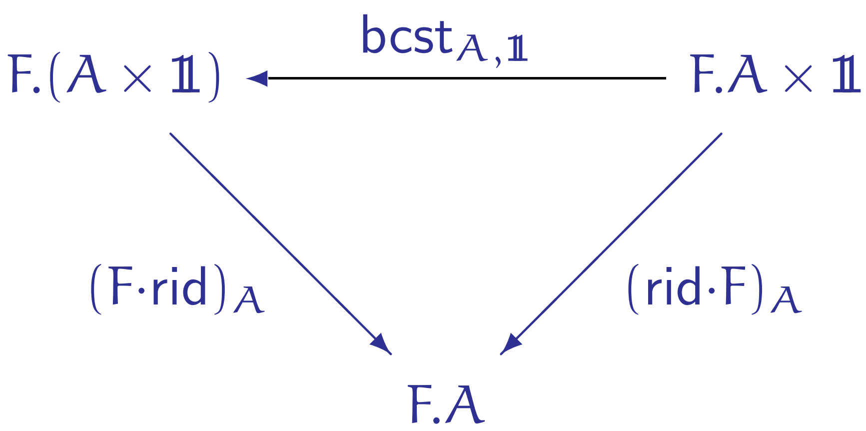
Formally, a family of functions bcst, where

$$\mathsf{bcst}_{A,B} \ : \ F.(A \times B) \leftarrow F.A \times B$$

is said to be a *broadcast* for datatype F iff it is parametrically polymorphic in the parameters A and B and $\mathsf{bcst}_{A,B}$ behaves coherently with respect to product in the following sense:

... Respect the Unit of Product ...

The following diagram



(where $\mathsf{rid}_A \ : \ A \leftarrow A \times \mathbb{1}$ is the obvious natural isomorphism)
commutes.

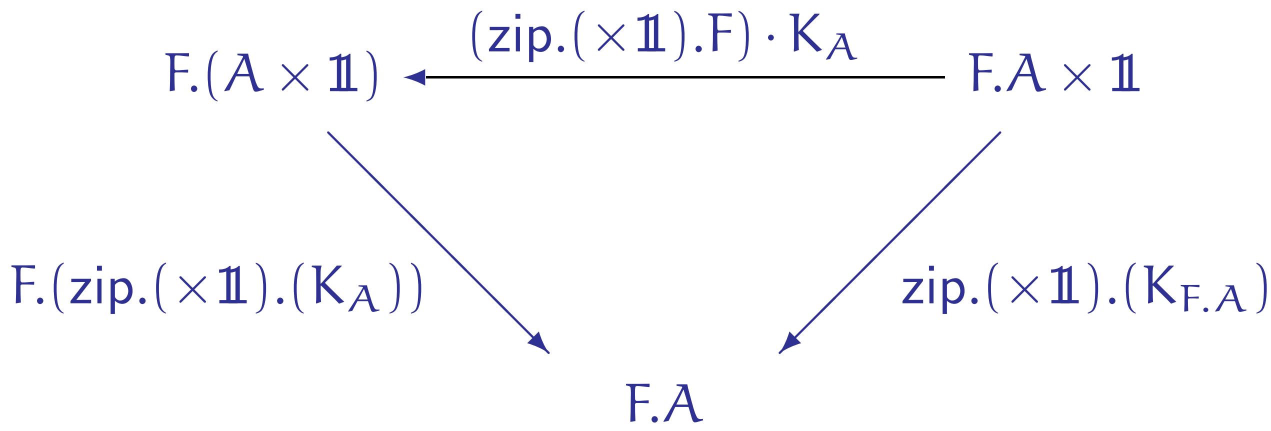
… and Associativity of Product

The following diagram

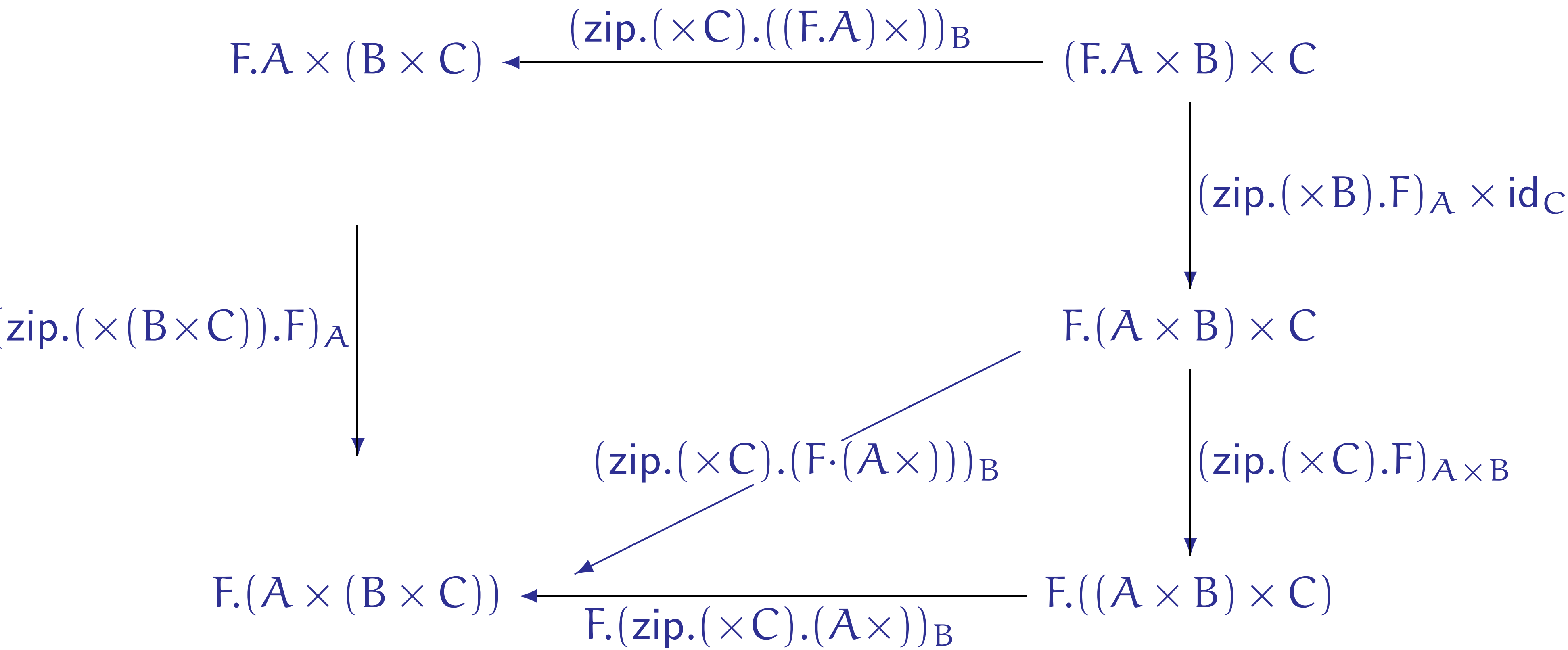
$$
\begin{array}{ccc}
F.A \times (B \times C) & \xleftarrow{\ \mathsf{ass}_{F.A,B,C}\ } & (F.A \times B) \times C \\
 & & \Big\downarrow \mathsf{bcst}_{A,B} \times \mathsf{id}_C \\
\mathsf{bcst}_{A\,,\,B\times C} \Big\downarrow & & F.(A \times B) \times C \\
 & & \Big\downarrow \mathsf{bcst}_{A\times B\,,\,C} \\
F.(A \times (B \times C)) & \xleftarrow[\ F\cdot\mathsf{ass}_{A,B,C}\ ]{} & F.((A \times B) \times C)
\end{array}
$$

(where $\mathsf{ass}_{A,B,C} : A \times (B \times C) \leftarrow (A \times B) \times C$ is the obvious natural isomorphism) commutes as well.

Unit of Product is a "zip"



Associativity of Product is a "Zip"



Conclusion

- Commuting Datatypes ("Zips") are everywhere!
- Generic specification and proof is (potentially) very effective.
- A relational framework is necessary.
- Challenge: give generic specification of "commuting datatypes" from which "zips" can be constructed calculationally.

Relators, Fans and Membership

Roland Backhouse and Paul Hoogendijk

Generic Programming Summer School

Oxford, August 2002

Allegories

Categorical formulation of (point-free) relation algebra.

Category (objects A, B, C, arrows —"relations"— R, S)

$$\mathsf{R}\circ\mathsf{S} : A \leftarrow \mathsf{B} \quad \Leftarrow \quad \mathsf{R} : A \leftarrow \mathsf{C} \wedge \mathsf{S} : \mathsf{C} \leftarrow \mathsf{B} \quad ,$$

$$\mathsf{id}_A : A \leftarrow A \quad .$$

Arrows of same type are partially ordered by $\subseteq$.

$$\mathsf{S}_1\circ\mathsf{T}_1 \subseteq \mathsf{S}_2\circ\mathsf{T}_2 \quad \Leftarrow \quad \mathsf{S}_1 \subseteq \mathsf{S}_2 \wedge \mathsf{T}_1 \subseteq \mathsf{T}_2 \quad .$$

$$\mathsf{X} \subseteq \mathsf{R} \wedge \mathsf{X} \subseteq \mathsf{S} \quad \equiv \quad \mathsf{X} \subseteq \mathsf{R}\cap\mathsf{S} \quad .$$

Converse

$$\mathsf{R}^{\cup} \subseteq \mathsf{S} \quad \equiv \quad \mathsf{R} \subseteq \mathsf{S}^{\cup} \quad ,$$

$$(\mathsf{R}\circ\mathsf{S})^{\cup} = \mathsf{S}^{\cup}\circ\mathsf{R}^{\cup} \quad ,$$

$$\mathsf{R}\circ\mathsf{S} \cap \mathsf{T} \subseteq (\mathsf{R} \cap \mathsf{T}\circ\mathsf{S}^{\cup})\circ\mathsf{S} \quad .$$

Relator

Relator: functor that is monotonic and respects converse.

Let $\mathcal{A}$ and $\mathcal{B}$ be allegories. A mapping F from objects of $\mathcal{A}$ to objects of $\mathcal{B}$ and arrows of $\mathcal{A}$ to arrows of $\mathcal{B}$ is a relator iff

$$\mathsf{F.R} : \mathsf{F.}A \leftarrow \mathsf{F.B} \quad \Leftarrow \quad \mathsf{R} : A \leftarrow \mathsf{B} \quad ,$$

$$\mathsf{F.R} \circ \mathsf{F.S} = \mathsf{F.(R \circ S)} \quad \text{for each } \mathsf{R} : A \leftarrow \mathsf{B} \text{ and } \mathsf{S} : \mathsf{B} \leftarrow \mathsf{C} \quad ,$$

$$\mathsf{F.id}_A = \mathsf{id}_{\mathsf{F.}A} \quad \text{for each object } A \quad ,$$

$$\mathsf{F.R} \subseteq \mathsf{F.S} \quad \Leftarrow \quad \mathsf{R} \subseteq \mathsf{S} \quad \text{for each } \mathsf{R} : A \leftarrow \mathsf{B} \text{ and } \mathsf{S} : A \leftarrow \mathsf{B} \quad ,$$

$$(\mathsf{F.R})^{\cup} = \mathsf{F.}(\mathsf{R}^{\cup}) \quad \text{for each } \mathsf{R} : A \leftarrow \mathsf{B} \quad .$$

Examples: List is an endorelator. $\times$ is a binary relator.

Functions

Relation $R : A \leftarrow B$ is *total* iff

$$\mathsf{id}_B \subseteq R^{\cup} \circ R \quad ,$$

and relation R is single-valued or *simple* iff

$$R \circ R^{\cup} \subseteq \mathsf{id}_A \quad .$$

A function is a relation that is total and simple.

Relators preserve totality

$$(F.R)^{\cup} \circ F.R$$
$$= \quad \{ \text{ relators respect converse } \}$$
$$F.(R^{\cup}) \circ F.R$$
$$= \quad \{ \text{ relators distribute through composition } \}$$
$$F.(R^{\cup} \circ R)$$
$$\supseteq \quad \{ \text{ assume } id_B \subseteq R^{\cup} \circ R \text{, relators are monotonic } \}$$
$$F.id_B$$
$$= \quad \{ \text{ relators preserve identities } \}$$
$$id_{F.B} \quad .$$

Similarly, relators preserve simplicity. Hence relators preserve functions.

Parametricity — point-free

Recall

$$(f,g) \in R \leftarrow S \quad \equiv \quad \langle \forall c,d :: (f.c\,,\,g.d) \in R \Leftarrow (c,d) \in S \rangle \quad .$$

Point-free:

$$(f,g) \in R \leftarrow S \;\equiv\; f^{\cup} \circ R \circ g \supseteq S \quad .$$

Equivalently, using *shunting* rule:

$$(f,g) \in R \leftarrow S \;\equiv\; R \circ g \supseteq f \circ S \quad .$$

Relators are Parametric

Type:

$$F.R : F.A \leftarrow F.B \quad \Leftarrow \quad R : A \leftarrow B \quad .$$

That is,

$$F \;:\; \langle \forall \alpha,\beta :: (F.\alpha \leftarrow F.\beta) \leftarrow (\alpha \leftarrow \beta) \rangle \quad .$$

F is parametric iff, for all relations R and S, and all functions f and g,

$$(F.f\,,\,F.g) \in F.R \leftarrow F.S \quad \Leftarrow \quad (f,g) \in R \leftarrow S \quad .$$

Exercise: verify that this is the case using point-free definition of $R \leftarrow S$.

Natural Transformations

Parametricity of reverse function, rev, on lists, and of fork:

$$\mathsf{List.R} \circ \mathsf{rev}_B \supseteq \mathsf{rev}_A \circ \mathsf{List.R}$$

$$\mathsf{R}{\times}\mathsf{R} \circ \mathsf{fork}_B \supseteq \mathsf{fork}_A \circ \mathsf{R}$$

In fact,

$$\mathsf{List.R} \circ \mathsf{rev}_B = \mathsf{rev}_A \circ \mathsf{List.R} \quad .$$

But, it is *not* the case that, for all R,

$$\mathsf{R}{\times}\mathsf{R} \circ \mathsf{fork}_B = \mathsf{fork}_A \circ \mathsf{R} \quad .$$

For example,

$$\{(0,0),(1,0)\} \times \{(0,0),(1,0)\} \circ \mathsf{fork}_B \neq \mathsf{fork}_A \circ \{(0,0),(1,0)\} \quad .$$

fork is a (lax) *natural transformation*, rev is a *proper* natural transformation.

Natural Transformations

$$\theta : F \hookleftarrow G \quad = \quad F.R \circ \theta_B \supseteq \theta_A \circ G.R \quad \text{for each } R : A \leftarrow B$$

$$\theta : F \hookrightarrow G \quad = \quad F.R \circ \theta_B \subseteq \theta_A \circ G.R \quad \text{for each } R : A \leftarrow B \quad .$$

Facts:

$$(F.f \circ \theta_B = \theta_A \circ G.f \quad \text{for each function } f : A \leftarrow B) \;\Leftarrow\; \theta : F \hookleftarrow G \quad .$$

In a “tabular allegory”,

$$\theta : F \hookleftarrow G \;\Leftarrow\; (F.f \circ \theta_B = \theta_A \circ G.f \quad \text{for each function } f : A \leftarrow B) \quad .$$

In words, $\theta : F \hookleftarrow G$ iff θ is a (categorical) natural transformation in the underlying category of maps.

Conclusion: we take $\theta : F \hookleftarrow G$ to be the definition of a *natural transformation* in an allegory.

Division

An allegory is *locally complete* if for each set $\mathcal{S}$ of relations of type $A \leftarrow B$, the union $\cup\mathcal{S} : A \leftarrow B$ exists and, furthermore, intersection and composition distribute over arbitrary unions.

$\perp\!\!\!\perp_{A,B}$ is the smallest relation of type $A \leftarrow B$ and $\top\!\!\!\top_{A,B}$ is the largest relation of the same type.

In a *division* allegory, composition distributes through union. That is, there are two *division* operators "\" and "/", such that, for all $R : A \leftarrow B$, $S : B \leftarrow C$ and $T : A \leftarrow C$,

$$R \circ S \subseteq T \quad \equiv \quad S \subseteq R \backslash T \quad ,$$

$$R \circ S \subseteq T \quad \equiv \quad R \subseteq T / S \quad ,$$

$$S \subseteq R \backslash T \quad \equiv \quad R \subseteq T / S \quad .$$

Domain and Range

The *range* of a relation R is the set of all x such that $(x,y) \in R$ for some y.

Formally, the range operator "$<$" is defined by, for all $R : A \leftarrow B$ and all $X \subseteq \mathsf{id}_A$,

$$R< \subseteq X \;\; \equiv \;\; R \subseteq X \circ \top\!\top_{A,B} \;\; .$$

The *domain* $R>$ is defined by

$$R> \;\; = \;\; (R\cup)< \;\; .$$

Membership

The membership relation of a relator F is a family of relations mem_A, indexed by objects A, such that

$$\mathsf{mem}_A \ : \ A \leftarrow \mathsf{F}.A \quad , \text{ and}$$

for all A, all $\mathsf{X} \subseteq \mathsf{id}_A$ and $\mathsf{Y} \subseteq \mathsf{id}_{\mathsf{F}.A}$,

$$\mathsf{F}.\mathsf{X} \supseteq \mathsf{Y} \ \equiv \ (\mathsf{mem}_A \circ \mathsf{Y})_< \subseteq \mathsf{X} \quad .$$

In words, $\mathsf{F}.\mathsf{X}$ is the largest subset Y of F-structures, each of type $\mathsf{F}.A$, such that the data stored in elements is in the set X.

Weakest Liberal Precondition

For all $X \subseteq id_A$ and $Y \subseteq id_{F.A}$,

$$
\begin{array}{ll}
 & (mem_A \circ Y)^{<} \ \subseteq \ X \\
= & \{ \quad \text{definition of range} \quad \} \\
 & mem_A \circ Y \subseteq X \circ \top\!\top \\
= & \{ \quad \text{division} \quad \} \\
 & Y \subseteq mem_A \backslash (X \circ \top\!\top) \\
= & \{ \quad Y \subseteq id_{F.A} \quad \} \\
 & Y \subseteq mem_A \backslash (X \circ \top\!\top) \cap id_{F.A} \quad .
\end{array}
$$

For those familiar with the wp calculus: $mem_A \backslash (X \circ \top\!\top) \cap id_{F.A}$ is the weakest liberal precondition guaranteeing a state satisfying X after "execution" of mem.

Properties of F structures

For all A, all $\mathsf{X} \subseteq \mathsf{id}_A$ and $\mathsf{Y} \subseteq \mathsf{id}_{\mathsf{F}.A}$,

$$\mathsf{F.X} \supseteq \mathsf{Y} \quad \equiv \quad \mathsf{mem}_A \backslash (\mathsf{X} \circ \top\!\!\top) \cap \mathsf{id}_{\mathsf{F}.A} \;\supseteq\; \mathsf{Y} \quad .$$

So,

$$\mathsf{F.X} \;=\; \mathsf{mem}_A \backslash (\mathsf{X} \circ \top\!\!\top) \cap \mathsf{id}_{\mathsf{F}.A} \quad .$$

Interpreting $\mathsf{X} \subseteq \mathsf{id}_A$ as a property of values of type A, $\mathsf{F.X}$ is a property of values of type $\mathsf{F}.A$. The identity says that a property of an F-structure is characterised by properties of the values stored in the structure (its "members").

Largest Natural Transformations

Recall: for each object A,

$$\mathsf{mem}_A \;:\; A \leftarrow \mathsf{F}.A \quad .$$

Membership is parametric: for all R,

$$\mathsf{R} \circ \mathsf{mem} \;\supseteq\; \mathsf{mem} \circ \mathsf{F.R} \quad .$$

Equivalently,

$$\mathsf{mem} : \mathrm{Id} \hookleftarrow \mathsf{F} \quad .$$

Also,

$$\mathsf{mem}\backslash\mathsf{id} : \mathsf{F} \hookleftarrow \mathrm{Id} \quad .$$

Theorem: The fan of relator F, $\mathsf{mem}\backslash\mathsf{id}$, is the largest natural transformation of type $\mathsf{F} \hookleftarrow \mathrm{Id}$. The membership of relator F is the largest natural transformation of type $\mathrm{Id} \hookleftarrow \mathsf{F}$.

Understanding Natural Transformations

Theorem: Suppose F and G are relators with memberships $\mathsf{mem.F}$ and $\mathsf{mem.G}$ respectively. Then the largest natural transformation of type $\mathsf{F} \hookleftarrow \mathsf{G}$ is $\mathsf{mem.F} \backslash \mathsf{mem.G}$.

Interpretation: A natural transformation of type $\mathsf{F} \hookleftarrow \mathsf{G}$ changes structure only. Stored values may be lost or duplicated, but no computation is performed on them.

A *proper* natural transformation to F from G changes the structure without loss or duplication of stored values.

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

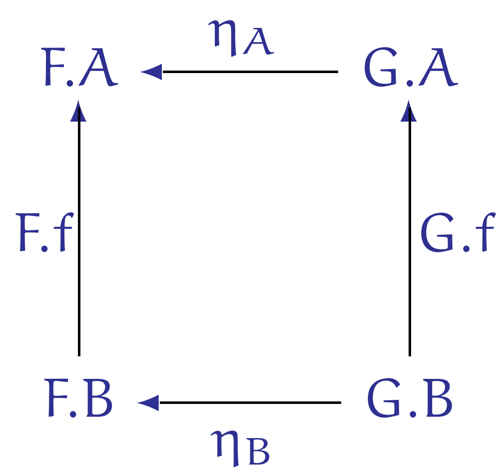
$$\mathsf{zip.F} \;:\; (\bullet\mathsf{F}) \longleftarrow (\mathsf{F}\bullet) \quad .$$

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 $f : A \leftarrow B$)



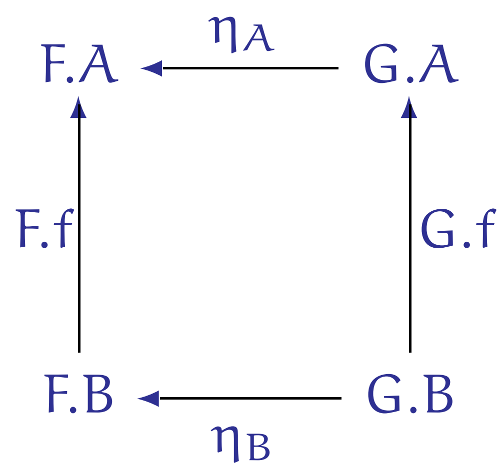
Now, if F is a functor, $(\bullet F)$ and $(F\bullet)$ are endofunctors on the functor category.

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

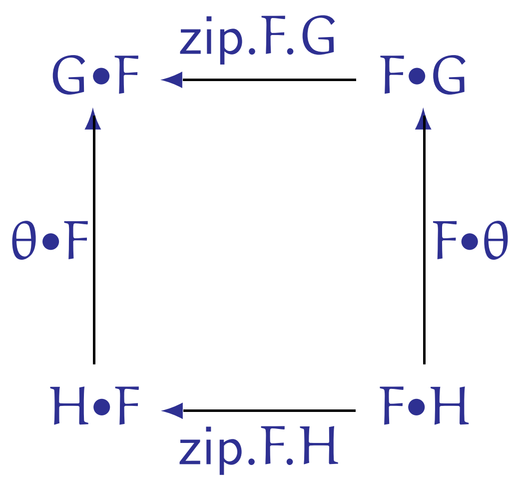
$(F\bullet)$ maps functor (object) G to $F\bullet G$ and natural transformation (arrow) η to $F\bullet\eta$, where $(F\bullet\eta)_A = F.(\eta_A)$.

Categorical NT Revision (Continued)

Diagram defining $\eta : F \leftarrow G$



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



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 $\langle\forall\alpha :: F.\alpha \leftarrow G.\alpha\rangle$, 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

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

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

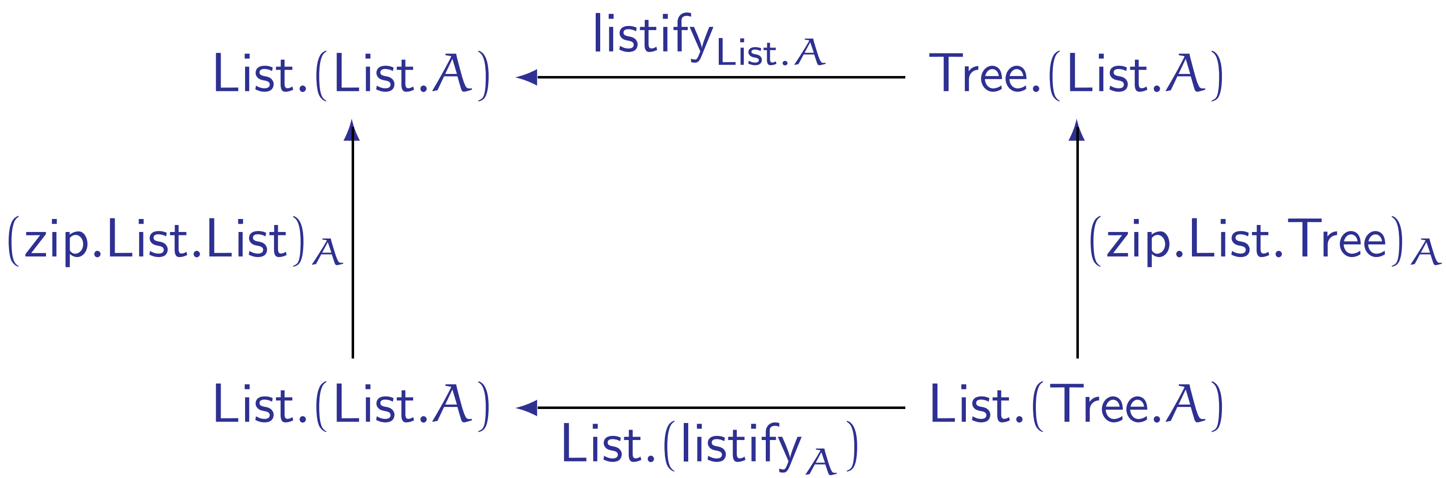
θ nondeterministically swaps the elements of a pair or not.

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



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

The lower path (via List.(List.A)) includes the upper path (via Tree.(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.

zip.F **is parametric.**

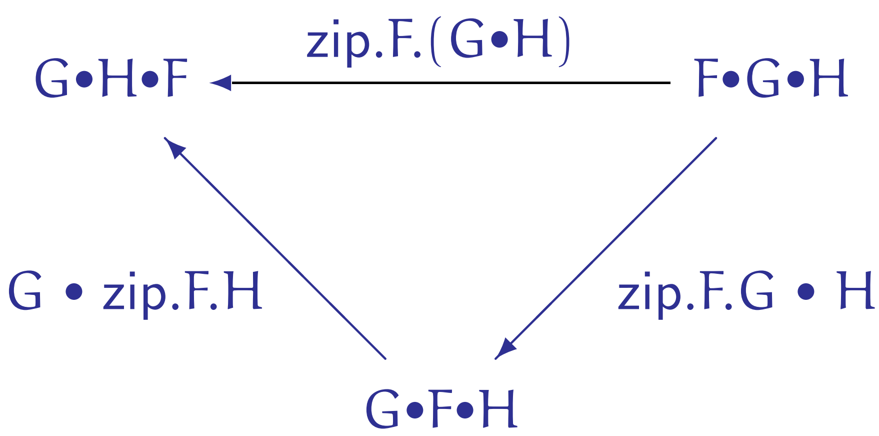
That is, for all $\theta : G \hookleftarrow H$,

$$(\theta \bullet F) \circ zip.F.H \quad \subseteq \quad zip.F.G \circ (F \bullet \theta) \quad .$$

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.)



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

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

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 $\otimes$.