J. Functional Programming **10** (2): 191–225, March 2000. Printed in the United Kingdom

Container types categorically

PAUL HOOGENDIJK

Philips Research Laboratories, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands

OEGE DE MOOR

Programming Research Group, Oxford University, Wolfson Building, Parks Road, Oxford OX1 3QD, UK

Abstract

A program derivation is said to be *polytypic* if some of its parameters are data types. Often these data types are container types, whose elements store data. Polytypic program derivations necessitate a general, non-inductive definition of 'container (data) type'. Here we propose such a definition: a container type is a relator that has membership. It is shown how this definition implies various other properties that are shared by all container types. In particular, all container types have a unique strength, and all natural transformations between container types are strong.

Capsule Review

Progress in a scientific dicipline is readily equated with an increase in the volume of knowledge, but the true milestones are formed by the introduction of solid, precise and usable definitions.

Here you will find the first generic ('polytypic') definition of the notion of 'container type', a definition that is remarkably simple and suitable for formal generic proofs (as is amply illustrated in the paper). Among the startling results is the proof that any lax natural transformation between two container types is strong.

1 Introduction

What is a container type? It is easy to list a number of examples: pairs, lists, bags, finite sets, possibly infinite sets..., but such a list of examples hardly makes a definition. The obvious formalisation is a definition that builds up the class of container types inductively; such an inductive definition, however, leads to cumbersome proofs if we want to prove a property of all container types. Here we aim to give a non-inductive characterisation, defining a container type as a mathematical object that has certain properties.

Why is such a definition desirable? In recent years it has become apparent that significant advances in formal program development are possible if both specifications and programs are parametrised by container types (and sometimes more general types as well). For example, one can reason about a program that finds a minimum element of a data structure, without actually committing oneself to lists, arrays, trees, bags or sets. Such type parametric programs and their derivations are said to be *polytypic*. To carry out polytypic program derivations we need to appeal to properties that are shared by all container types. This paper does not go into examples of polytypic program derivation, and the reader is referred elsewhere (Backhouse *et al.*, 1993; Bird *et al.*, 1996; Doornbos, 1996; Hoogendijk & Backhouse, 1997; Jeuring, 1995; Meertens, 1996) for such examples, and a more in-depth motivation of polytypism. In particular, Bird *et al.* (1996) and Bird & De Moor (1996) go into applications that motivated the theory presented here.

Because our interest is in a definition that is useful in specification, the class of container types considered here may be somewhat too liberal for those whose primary interest is in executable code. For example, possibly infinite sets are an essential ingredient of any specification formalism, but they rarely feature as a data type in executable programs.

The structure of the paper is as follows. First, we briefly introduce those elements of category theory that are necessary for our purposes, as well as the relational calculus. We shall occasionally make reference to more advanced aspects of category theory, but we have taken pains to make the paper as self-contained as possible. Next, we accumulate a number of informal requirements of container types: this culminates in a technical definition. Briefly, we shall argue that container types are *functors*. Motivated by a need to deal with nondeterminism, we subsequently note that container types are a special kind of functor called *relators*. We then examine the notion of *membership* tests; not all relators support this notion, and we define a container type as a relator that has membership. Of course one should not only be able to inspect data structures; one also needs ways of creating them. This leads to an investigation of *fans*, and it turns out that any relator with membership also has a unique fan. Next we compare our work with the definition of *strong* functors, which is the leading notion of what data types ‘really are’ among category theorists. We show that any relator that has membership is strong, and that much of the tedious conditions involved in reasoning about strength are vacuously satisfied. The results about fans and strength give some credence to our claim that container types can be defined as relators that have membership.

2 Preliminaries

2.1 Categories

A *category* is a universe of typed specifications: it consists of objects (types) and arrows (specifications or programs). Each arrow h has a target type A and a source type B. We write $h : A \leftarrow B$ to indicate this type information. For each object A there is a distinguished arrow $id_A : A \leftarrow A$. Arrows can be composed, subject to some typing rules. That is, when $f : A \leftarrow B$ and $g : B \leftarrow C$, their composite is $f \cdot g : A \leftarrow C$. Composition is associative and has identity element id. We shall use $\mathscr{C}$ and $\mathscr{D}$ to denote arbitrary categories.

The canonical example of a category is *Fun*, where the objects are sets and the arrows are total functions. Another example is *Rel*, where the objects are also sets, but where the arrows are binary relations, i.e. sets of pairs, composed in the usual manner. The category *Fun* is often named *Set* in the literature.

All our examples are drawn from *Fun* and *Rel*, but the definitions work in more general categories, including certain models of programming languages. For those with a background in category theory, we remark that all our proofs go through in a logos that satisfies the axiom of subextensionality.

An *isomorphism* is an arrow $i : A \leftarrow B$ that has an inverse $i^{-1} : B \leftarrow A$ such that $i \cdot i^{-1} = id_A$ and $i^{-1} \cdot i = id_B$. We say that the objects A and B are isomorphic. In *Fun*, the isomorphisms are bijective functions, and these are also the isomorphisms in *Rel*.

2.2 *Functors and natural transformations*

Functors. A *functor* is a structure-preserving mapping between categories. That is, a functor to $\mathscr{C}$ from $\mathscr{D}$ is a mapping F that maps objects of $\mathscr{D}$ to objects of $\mathscr{C}$ and arrows of $\mathscr{D}$ to arrows of $\mathscr{C}$, preserving the type information. If $h : A \leftarrow B$, then $F\,h : F\,A \leftarrow F\,B$. Furthermore, it is required that functors preserve identities and composition:

$$F\,id_A = id_{FA} \qquad \text{and} \qquad F\,(h \cdot k) = F\,h \cdot F\,k.$$

We write $F : \mathscr{C} \leftarrow \mathscr{D}$ to indicate that F is a functor to $\mathscr{C}$ from $\mathscr{D}$.

An example of a functor $Fun \leftarrow Fun$ is *list*. It maps a set A to the set *list* A is the set of lists over A:

$$\mathit{list}\,A \quad = \quad \{[a_1, a_2, \ldots, a_n] \mid a_i \in A\}.$$

On arrows, *list* applies a function to all elements of a list:

$$\mathit{list}\,h\,[a_1, a_2, \ldots, a_n] = [h\,a_1, h\,a_2, \ldots, h\,a_n].$$

The reader may wish to check for herself that *list* does indeed preserve identities and composition.

Another example of a functor is $J : Rel \leftarrow Fun$. It leaves objects unchanged, and it maps each function to the corresponding set of pairs.

In the opposite direction of J, we have the *existential image* functor $E : Fun \leftarrow Rel$. It takes a set to the collection of all its subsets, and a relation is mapped to its existential image:

$$E\,R\,x \quad = \quad \{a \mid \exists b \in x : aRb\}.$$

Functors can be composed by defining $(F \cdot G)h = F(G\,h)$. For instance, the composite $P = E \cdot J : Fun \leftarrow Fun$ is the *powerset* functor that applies a function to all elements of a set.

Natural transformations. A *natural transformation* is a mapping between functors. That is, given two functors F and G of type $\mathscr{C} \leftarrow \mathscr{D}$, a natural transformation $\phi : F \leftarrow G$ is a collection of arrows

$$\phi_A : F\,A \leftarrow G\,A \quad \text{for each object } A \text{ of } \mathscr{D}.$$

Furthermore, this collection of arrows should satisfy the equation

$$F\,h \cdot \phi_B \quad = \quad \phi_A \cdot G\,h \quad \text{for each } h : A \leftarrow B \text{ in } \mathscr{D}.$$

We shall usually omit the index of the components of a natural transformation.

For example, consider the operation *setify* that turns a list of elements into the corresponding set. This is a natural transformation $P \leftarrow \mathit{list}$.

2.3 *Finite products*

A *terminal* object of a category $\mathscr{C}$ is an object 1 such that for each object A there exists exactly one arrow $1 \leftarrow A$. That unique arrow is written $!_A$, and it is pronounced 'pling'. The index of $!_A$ is omitted whenever it is clear from the context. The definition of ! can be stated as the equivalence:

$$h = ! \quad \equiv \quad h : 1 \leftarrow A, \qquad \text{for all } h.$$

In *Fun*, any singleton set is a terminal object. More generally, it can be shown that any two terminal objects are isomorphic. When we speak of *the* terminal object, we mean that a canonical representative has been chosen from the class of all terminal objects. The empty set is the terminal object of *Rel*.

Given two objects A and B, their *product* is an object $A \times B$ together with two arrows $\mathit{outl} : A \leftarrow A \times B$ and $\mathit{outr} : B \leftarrow A \times B$. The product is characterised by the following property: for each pair of arrows $f : A \leftarrow C$ and $g : B \leftarrow C$ there exists an arrow $\langle f, g \rangle : A \times B \leftarrow C$ such that

$$h = \langle f, g \rangle \quad \equiv \quad \mathit{outl} \cdot h = f \quad \text{and} \quad \mathit{outr} \cdot h = g,$$

for all $h : A \times B \leftarrow C$. Again, this defines products up to isomorphism.

In *Fun*, the set $A \times B$ is the cartesian product of A and B, and $\langle f, g \rangle\, a = (f\,a, g\,a)$. The same construction does not define a product in *Rel* because $\langle \emptyset, R \rangle = \emptyset$ for any R. Products satisfy the expected isomorphisms, such as

$$\mathit{rid} \quad : \quad A \leftarrow A \times \mathbf{1}$$

and associativity

$$\mathit{assl} \quad : \quad (A \times B) \times C \leftarrow A \times (B \times C).$$

2.4 *Allegories and relation algebra*

While *Rel* is in many respects similar to *Fun*, it has some additional structure that is useful to our purposes. In particular, relations of the same type can be compared by inclusion, $R \subseteq S$, and composition is monotonic with respect to inclusion.

The largest relation of type $A \leftarrow B$ is written $\Pi : A \leftarrow B$. The smallest relation of type $A \leftarrow B$ is the empty set, written $\emptyset : A \leftarrow B$.

The intersection of two relations R and S of the same type is denoted $R \cap S$. Note that composition does *not* distribute over intersection; in general, we only have the inclusion

$$(R \cap S) \cdot T \quad \subseteq \quad (R \cdot T) \cap (S \cdot T).$$

Every relation $R : A \leftarrow B$ has a *converse* $R^\circ : B \leftarrow A$, obtained by flipping the pairs in R. Note that the converse operation leaves identities unchanged, but that it reverses composition:

$$id^\circ = id \qquad \text{and} \qquad (R \cdot S)^\circ = S^\circ \cdot R^\circ.$$

A useful fact relating composition and converse is the so-called *modular law*

$$(R \cdot S) \cap T \quad \subseteq \quad (R \cap (T \cdot S^\circ)) \cdot S,$$

which, in predicate calculus, shows how intersection distributes over existential quantification:

$$(\exists b : aRb \wedge bSc) \wedge aTc \quad \Rightarrow \quad \exists b : (aRb \wedge (\exists c' : aTc' \wedge bSc') \wedge bSc).$$

Note that we obtain the symmetric law

$$(R \cdot S) \cap T \quad \subseteq \quad R \cdot (S \cap R^\circ \cdot T)$$

by applying converse to the formulation above.

Given two relations that share the same target, say $R : A \leftarrow B$ and $S : A \leftarrow C$, the *division* of R by S is a relation of type $B \leftarrow C$, characterised by the equivalence

$$X \subseteq R\backslash S \quad \equiv \quad R \cdot X \subseteq S, \quad \text{for all } X : B \leftarrow C.$$

In words, $R\backslash S$ is the largest relation X such that $R \cdot X \subseteq S$. For this reason, $R\backslash S$ is sometimes called the *weakest prespecification* (Hoare & He, 1986a, 1986b). As a predicate, $R\backslash S$ can be written

$$b(R\backslash S)c \quad \equiv \quad (\forall a : aRb : aSc).$$

Here we use the quantifier notation $(\forall a : aRb : aSc)$ as an alternative to the more conventional $(\forall a : aRb \Rightarrow aSc)$. Some useful facts about division that we shall use in the sequel are

$$\begin{aligned} S &\subseteq R\backslash(R \cdot S) \\ R\backslash\Pi &= \Pi \\ (R\backslash S) \cdot T &\subseteq R\backslash(S \cdot T). \end{aligned}$$

2.5 *Special relations*

A relation $R : A \leftarrow B$ is said to be *total* if each element in the source is related to at least one element in the target. In a formula:

$$R^\circ \cdot R \quad \supseteq \quad id.$$

A relation $R : A \leftarrow B$ is said to be *single-valued* if each element in the source is related to at most one element in the target. In a formula:

$$R \cdot R^\circ \quad \subseteq \quad id.$$

A *function* is a relation that is both total and single-valued. Functions are important because they satisfy all sorts of useful identities and equivalences that are not generally valid. We shall denote functions by lower case identifiers so these equivalences are easy to spot. For example, here is the so-called *shunting* rule for functions:

$$f \cdot R \subseteq S \quad \equiv \quad R \subseteq f^\circ \cdot S.$$

Inclusion of functions is the same as equality, that is

$$f \subseteq g \quad \equiv \quad f = g.$$

Every arrow in *Rel* can be factored in terms of functions. That is, for each arrow $R : A \leftarrow B$ there exist $f : A \leftarrow C$ and $g : B \leftarrow C$ such that $R = f \cdot g^\circ$. This fact is very useful in generalising operations on functions to relations. For example, to generalise the functor *list* : *Fun* ← *Fun* to a functor *Rel* ← *Rel*, we can define $F\,R = F\,f \cdot (F\,g)^\circ$. We shall have more to say about generalising functors of *Fun* to functors of *Rel* below.

In the special case of functions, the modular law that we quoted above can be strengthened from an inclusion to an equation:

$$(h \cdot R \cdot k^\circ) \cap S \quad = \quad h \cdot (R \cap (h^\circ \cdot S \cdot k)) \cdot k^\circ.$$

Below we shall refer to this fact as the *modular identity*.

2.6 Derived operators on relations

A *coreflexive* relation is a subset of an identity arrow. The domain of a relation $R : A \leftarrow B$ is the subset of B on which R is defined, represented as a coreflexive $B \leftarrow B$:

$$\mathit{Dom}\,R \quad = \quad \mathit{id} \cap (R^\circ \cdot R).$$

It is easily shown that R is total if and only if $\mathit{Dom}\,R = \mathit{id}$. The domain of the intersection of two relations is given by

$$\mathit{Dom}\,(R \cap S) \quad = \quad \mathit{id} \cap (R^\circ \cdot S).$$

Similarly, the domain of a composite relation is always smaller than the domain of the right-hand component:

$$\mathit{Dom}\,(R \cdot S) \quad \subseteq \quad \mathit{Dom}\,S.$$

The *range* of a relation R is the domain of the converse of R:

$$\mathit{Ran}\,R \quad = \quad \mathit{Dom}\,R^\circ.$$

Each property of *Dom* has an equivalent for *Ran*.

Besides coreflexives, one can also represent ranges as so-called *left-conditions*. A relation C is said to be a left-condition if

$$C : A \leftarrow 1$$

Any relation can be mapped to a left-condition through

$$\mathit{Cond}\,R \quad = \quad R \cdot !^\circ.$$

The connection with range is made explicit in the equations

$$\mathit{Ran}(\mathit{Cond}\,R) = \mathit{Ran}\,R \qquad \text{and} \qquad \mathit{Cond}(\mathit{Ran}\,R) = \mathit{Cond}\,R.$$

2.7 From functions to relations and back

As we have seen, functions can be characterised as a special kind of relations. Furthermore, every relation can be factored as a pair of functions.

Relators. Having identified this fundamental way of constructing relations from functions and vice versa, the next question to ask is how functors of *Fun* generalise to *Rel*. A minimum healthiness condition on functors of *Rel* is that they are monotonic with respect to inclusion of relations: we shall call monotonic functors of *Rel* *relators*. For each functor of *Fun*, there is a unique generalisation to relations, and each relator of *Rel* can be viewed as a functor of *Fun*:

Fact 1
(Kawahara, 1973a) Let F be a functor of *Fun*. There exists at most one relator F' of *Rel* that agrees with F on functions in the sense that $F' \cdot J = J \cdot F$. Conversely, any monotonic functor F' of *Rel* preserves functions.

We shall consequently use the same identifier for a functor of *Fun* and its generalisation to *Rel*. As an example of a functor that is a relator, consider the list functor. Its generalisation to relations is given by

$$\begin{array}{l} \quad [a_1, a_2, \ldots, a_n](\mathit{list}\, R)[b_1, b_2, \ldots, b_m] \\ \equiv \\ \quad n = m \wedge (\forall i : 0 \leqslant i \leqslant n : a_i R b_i). \end{array}$$

The powerset functor is also a relator, and its action on relations is well-known from the Plotkin powerdomain:

$$x(P\,R)y \quad \equiv \quad (\forall a \in x : (\exists b \in y : aRb)) \;\wedge\; (\forall b \in y : (\exists a \in x : aRb)).$$

Again, we use the quantifier notation $(\forall a : P : Q)$ as an alternative to the more conventional $(\forall a : P \Rightarrow Q)$, and $(\exists a : P : Q)$ in lieu of $(\exists a : P \wedge Q)$. The example of the powerset functor is instructive because the relator P does *not* preserve intersection of relations.

Finally, the exponential functor is a relator:

$$f(\mathit{From}_A R)g \quad \equiv \quad \forall a : (f\,a)R(g\,a).$$

To prove that this definition actually preserves composition of relations, one needs the axiom of choice. This is an indication that our definition of relators is somewhat too strong. A much weaker definition is considered in Mitchell & Ščedrov (1993).

There exist functors of *Fun* that do not have a generalisation to *Rel*. An example is the following:

$$F\,A \quad = \quad \begin{cases} \emptyset, & \text{if } A = \emptyset \\ \{0\}, & \text{otherwise} \end{cases}$$

On arrows, F sends arrows whose source is empty, and whose target is non-empty to the unique arrow $\{0\} \leftarrow \emptyset$; all other arrows are mapped to the identity on $\{0\}$.

The notion of a relator was first introduced by Kawahara (1973a); the concept then went unnoticed for a long time, until it was reinvented in Carboni *et al.* (1991).

Almost simultaneously, Backhouse and his colleagues started to write a series of papers that demonstrate the relevance of relators to computing (Backhouse *et al.*, 1991). Backhouse has the additional requirement that a relator preserves converse; this does in fact follow from monotonicity:

Fact 2
If F is a relator, then $F(R^\circ) = (F\,R)^\circ$.

Lax natural transformations. What happens to natural transformations of type $F \leftarrow G$ if F and G are generalised from $Fun \leftarrow Fun$ to $Rel \leftarrow Rel$? As an example, consider again the operation *setify* that turns a list into a set. Above we saw that this is a natural transformation $P \leftarrow list$ when P and $list$ are read as functors $Fun \leftarrow Fun$:

$$P\,h \cdot setify \quad = \quad setify \cdot list\,h.$$

Note, however, that the above equation for *setify* is not true when the function h is replaced by an arbitrary relation R: we only have the inclusion

$$P\,R \cdot setify \quad \supseteq \quad setify \cdot list\,R.$$

So while *setify* is a natural transformation in *Fun*, it is not natural when considered as a collection of arrows in *Rel*. This is a very common phenomenon, and we shall say that ϕ is a *lax natural transformation* of type $F \hookleftarrow G$ when

$$F\,R \cdot \phi \quad \supseteq \quad \phi \cdot G\,R,$$

for all R. In fact, writing J for the inclusion of *Fun* into *Rel*, we have

Fact 3
For any collection ϕ of arrows $\phi : F\,A \leftarrow G\,A$, we have $\phi : F \cdot J \leftarrow G \cdot J$ if and only if $\phi : F \hookleftarrow G$.

2.8 *Lifting products from* Fun *to* Rel

Let us now return to the discussion of finite products. As we have already remarked, the constructions in *Fun* do *not* give rise to categorical products in *Rel*. However, we can of course define them as products in *Fun*, and then investigate their properties in *Rel*. This is the topic of the present subsection. The key idea is that ! is very closely related to Π, and $\langle \_, \_ \rangle$ is very closely related to intersection.

Recall that there is precisely one function $! : 1 \leftarrow A$ (pronounced 'pling'). It follows that this function is the largest relation of its type. For let $R : 1 \leftarrow A$. This relation R can be factorised as a pair of functions: $R = \,! \cdot h^\circ$. Now

$$R \;=\, ! \cdot h^\circ \;=\, ! \cdot h \cdot h^\circ \;\subseteq\; !.$$

By taking $R =\, ! \cdot \Pi$ and using the shunting rule, we conclude that

$$\Pi \quad = \quad !^\circ \cdot !.$$

This already hints at the possibility that a condition in terms of Π might also be phrased in terms of !.

We shall also need a number of further operations for manipulating binary products. The most important of these is the *split* operation, defined by

$$\langle R, S\rangle \quad = \quad (outl^\circ \cdot R) \cap (outr^\circ \cdot S).$$

An important fact about split, which we shall use repeatedly, is

$$\langle R, S\rangle^\circ \cdot \langle U, V\rangle \quad = \quad (R^\circ \cdot U) \cap (S^\circ \cdot V).$$

This hints at the possibility that a property in terms of intersection might also be phrased in terms of split.

The split $\langle R, S\rangle$ is a function whenever R and S are functions. The *product relator* is defined by

$$R \times S \quad = \quad \langle R \cdot outl, S \cdot outr\rangle,$$

and we have, for example, the *product absorption* rule

$$(R \times S) \cdot \langle U, V\rangle \quad = \quad \langle R \cdot U, S \cdot V\rangle,$$

as well as the naturality properties

$$outl \cdot (R \times id) = R \cdot outl \quad \text{and} \quad outr \cdot (id \times S) = S \cdot outr.$$

If one is willing to accept inclusions instead of equalities, the above two equations can be generalised to

$$outl \cdot (R \times S) \subseteq R \cdot outl \quad \text{and} \quad outr \cdot (R \times S) \subseteq S \cdot outr.$$

The properties of products in a relational setting have been thoroughly explored by numerous researchers; the most comprehensive account we know of can be found in Aarts *et al.* (1992). To get some practice in pushing all these new operators around, and for future reference, we first prove two little lemmas.

Lemma 4
For any relator F, and relations R and S, we have

$$(FR \times FS) \cdot \langle F\,outl, F\,outr\rangle \quad = \quad \langle F\,outl, F\,outr\rangle \cdot F(R \times S).$$

Proof
The containment ($\supseteq$) is easy, and details are omitted. For the other inclusion, the proof is in two stages. First,

$$\begin{array}{ll}
 & (FR \times id) \cdot \langle F\,outl, F\,outr\rangle \\
= & \quad \{\text{product absorption}\} \\
 & \langle FR \cdot F\,outl, F\,outr\rangle \\
= & \quad \{F \text{ relator, naturality of } outl\} \\
 & \langle F\,outl \cdot F(R \times id), F\,outr\rangle \\
\subseteq & \quad \{\text{modular law: } \langle R \cdot S, T\rangle \subseteq \langle R, T \cdot S^\circ\rangle \cdot S\} \\
 & \langle F\,outl, F\,outr \cdot F(R^\circ \times id)\rangle \cdot F(R \times id) \\
\subseteq & \quad \{F \text{ relator, naturality of } outr\} \\
 & \langle F\,outl, F\,outr\rangle \cdot F(R \times id)
\end{array}$$

By symmetry, we also have

$$(id \times F\,S) \cdot \langle F\,outl, F\,outr \rangle \quad \subseteq \quad \langle F\,outl, F\,outr \rangle \cdot F(id \times S).$$

Therefore,

$$\begin{array}{cl}
 & (F\,R \times F\,S) \cdot \langle F\,outl, F\,outr \rangle \\
= & \quad \{\text{product relator}\} \\
 & (id \times F\,S) \cdot (F\,R \times id) \cdot \langle F\,outl, F\,outr \rangle \\
\subseteq & \quad \{\text{above}\} \\
 & (id \times F\,S) \cdot \langle F\,outl, F\,outr \rangle \cdot F(R \times id) \\
\subseteq & \quad \{\text{above}\} \\
 & \langle F\,outl, F\,outr \rangle \cdot F(id \times S) \cdot F(R \times id) \\
= & \quad \{F \text{ and product relators}\} \\
 & \langle F\,outl, F\,outr \rangle \cdot F(R \times S)
\end{array}$$

□

Note the slightly curious structure of the above proof, where $(F\,R \times F\,S)$ is first decomposed into $id \times F\,S$ and $F\,R \times id$. We have no good heuristic for justifying this decision at present; the unexpected nature of the proof does however explain why we originally thought this fact needed the side condition that F preserves intersections of binary relations. It was a pleasant discovery that no such side condition is necessary. An important consequence is

Lemma 5
For any relator F, and any relations R and S, we have

$$\langle F\,R, F\,S \rangle \quad = \quad \langle F\,outl, F\,outr \rangle \cdot F\langle R, S \rangle.$$

Proof

$$\begin{array}{cl}
 & \langle F\,outl, F\,outr \rangle \cdot F\langle R, S \rangle \\
= & \quad \{\text{product absorption: } \langle R, S \rangle = (R \times S) \cdot \langle id, id \rangle,\ F \text{ relator}\} \\
 & \langle F\,outl, F\,outr \rangle \cdot F(R \times S) \cdot F\langle id, id \rangle \\
= & \quad \{\text{Lemma 4}\} \\
 & (F\,R \times F\,S) \cdot \langle F\,outl, F\,outr \rangle \cdot F\langle id, id \rangle \\
= & \quad \{F \text{ relator, } outl \cdot \langle id, id \rangle = id = outr \cdot \langle id, id \rangle\} \\
 & (F\,R \times F\,S) \cdot \langle id, id \rangle \\
= & \quad \{\text{product absorption}\} \\
 & \langle F\,R, F\,S \rangle
\end{array}$$

□

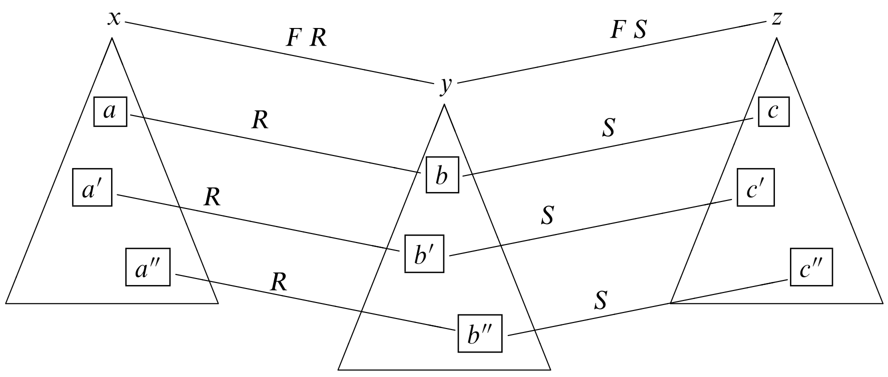
Proofs of other facts cited in this section are all elementary, and can be found in Bird & De Moor (1996).

3 A definition of container types

We now proceed to present our definition of container types. If F is a container type, then an F-structure $x : F\,A$, for some type A, can be thought of as having 'slots' which are 'filled' with values of type A, and we say then that x 'contains' these values, or that these values are 'members' of x. The following questions are meaningful. Does x contain a member having some given property? In particular: does some given $a : A$ occur as a member of x? Do all members of x have some given property?

Some container types are 'shapely' (Jay, 1995b). For such types it is meaningful to ask whether $x : F\,A$ and $y : F\,B$ have the same 'shape', and if they do there is a canonical one-to-one correspondence (bijection) between their slots. If x, y and z have the same shape, the correspondence between x and z is given by composing the correspondences between x and y, and y and z. If x and y have the same shape and all pairs of members in corresponding slots are related by a given relation $R : A \leftarrow B$, we say that x and y are $F\,R$-related.

It is (informally) clear that F preserves identities and composition, commutes with converse, and is monotonic with respect to relation inclusion (see the picture below):



We conclude that the container type F is in fact a relator. Note that this conclusion does not refer to 'shapes' and 'corresponding slots'. The approach, now, is to do the same more generally: *gather properties which hold for all shapely types, but which can be formulated without reference to shapes and correspondences (notions not meaningful for, for example, the power set type), and then postulate them for all container types, shapely or not.*

Here are two such properties:

- If x is $F\,R$-related to y then each member of x is R-related to some member of y (for shapely types, the corresponding one).
- If each member of x is R-related to at least one value, then x is $F\,R$-related to some value (for shapely types, the value y of the same shape as x, such that each slot of y is filled with a value b such that aRb, where a is the contents of the corresponding slot of x).

To formalise the first item, write $a\varepsilon x$ for "a is a member of the F-structure x". We then have

$$x(F\,R)y \quad \Rightarrow \quad (\forall a : a\varepsilon x : (\exists b : b\varepsilon y : aRb)), \tag{1}$$

or, abstracting from x and y,

$$F\,R \quad \subseteq \quad \varepsilon\backslash(R \cdot \varepsilon).$$

This holds for all R, which (by the definitions of $(\backslash)$ and $(\hookleftarrow)$) means that

$$\varepsilon : id \hookleftarrow F.$$

Using $(F\,R)^\circ = F(R^\circ)$, we also get from (1):

$$x(F\,R)y \quad \Rightarrow \quad (\forall b : b\varepsilon y : (\exists a : a\varepsilon x : aRb)). \tag{2}$$

Formalising the second item in the two bullet points above gives:

$$(\forall a : a\varepsilon x : (\exists b : aRb)) \quad \Rightarrow \quad (\exists y : x(F\,R)y). \tag{3}$$

We now explore the formal consequences of this formalisation. Define $aR_b b' \equiv aRb \wedge b' = b$. Then

$$\begin{array}{ll}
 & (\forall a : a\varepsilon x : aRb) \\
\equiv & \quad \{\text{since } aRb \equiv (\exists b' : aR_b b')\} \\
 & (\forall a : a\varepsilon x : (\exists b' : aR_b b')) \\
\Rightarrow & \quad \{\text{Implication (3)}\} \\
 & (\exists y : x(F\,R_b)y : x(F\,R_b)y) \\
\Rightarrow & \quad \{\text{since } R_b \subseteq R, \text{ Implication (2)}\} \\
 & (\exists y : x(F\,R)y : (\forall b' : b'\varepsilon y : (\exists a : a\varepsilon x : aR_b b'))) \\
\Rightarrow & \quad \{\text{since } (\exists a : a\varepsilon x : aR_b b') \Rightarrow (\exists a : a\varepsilon x : b = b') \Rightarrow b = b'\} \\
 & (\exists y : x(F\,R)y : (\forall b' : b'\varepsilon y : b' = b)) \\
\Rightarrow & \quad \{\text{Implication (1)}\} \\
 & (\exists y : (\forall a : a\varepsilon x : (\exists b' : b'\varepsilon y : aRb')) \wedge (\forall b' : b'\varepsilon y : b' = b)) \\
\Rightarrow & \quad \{\text{predicate calculus}\} \\
 & (\exists y : (\forall a : a\varepsilon x : (\exists b' : b'\varepsilon y : aRb' \wedge b' = b))) \\
\Rightarrow & \quad \{\text{predicate calculus}\} \\
 & (\exists y : (\forall a : a\varepsilon x : aRb)) \\
\equiv & \quad \{\text{Heidegger}\} \\
 & (\forall a : a\varepsilon x : aRb)
\end{array}$$

This proves by circular implication that

$$(\forall a : a\varepsilon x : aRb)) \quad \equiv \quad (\exists y : x(F\,R)y : (\forall b' : b'\varepsilon y : b' = b)),$$

or, abstracting from x and b,

$$\varepsilon\backslash R \quad = \quad F\,R \cdot \varepsilon\backslash id. \tag{4}$$

We take this equation as a *definition*: a collection of arrows ε is said to be a *membership relation* for F if it satisfies the above equation for all R. Below it is proven that there exists at most one such ε. (The symbol ε was chosen on account of its mild similarity to $\in$). A relator F is said to *have membership* if there exists a membership relation for F.

Summarising the above discussion, we propose the following definition of a container type:

A type constructor F is a container type if it is a relator that has membership.

In the remainder of this paper we explore the consequences of that definition. As an aside, we remark that the definition of membership implies the two properties that were used to motivate it.

4 Elementary properties of membership

To prove uniqueness of membership, we assume the *identification axiom*, which says that the largest lax natural transformation of type $id \hookleftarrow id$ is id itself. The identification axiom is satisfied in any reasonable model of a programming language, as it follows from subextensionality. (Subextensionality says that two functions f and g are equal if $f \cdot p = g \cdot p$ for all single-valued relations p of type $A \leftarrow 1$.) We do not know whether the identification axiom is equivalent to subextensionality.

Fact 6
Suppose that ε is a membership relation for F. Then ε is a lax natural transformation of type $id \hookleftarrow F$.

We shall defer detailed proofs of the facts stated here until section 4.1. The membership relation on lists is given by

$$a\varepsilon[a_0, a_1, \ldots, a_n] \quad \equiv \quad (\exists i : a = a_i).$$

The membership relation for the powerset functor is simply set membership $\in$. The membership relation for the exponential functor $From_B$ tests for existence in the range of a function

$$a\varepsilon f \quad \equiv \quad (\exists b : a = f\,b).$$

One might wonder whether every relator has membership, and indeed we originally believed that the membership relation could be constructed as the largest lax natural transformation of type $id \hookleftarrow F$. However, Peter Freyd provided us with an example that this construction does not necessarily satisfy the required property:

Fact 7
(Freyd) There exists a relator that does not have membership.

It is, however, true that if membership exists, it is the largest lax natural transformation of its type. More generally,

Fact 8
Let F and F' be relators with membership relations ε and ε' respectively. Then the largest lax natural transformation of type $F \hookleftarrow F'$ is $\varepsilon\backslash\varepsilon'$.

It is interesting to interpret this result in set theory. It says that a lax natural transformation $\phi : F \hookleftarrow G$ can never invent new values: if $x\phi y$, the set of elements of x is a subset of the elements of y. This captures one aspect of what it means for an operator to be polymorphic, but the converse is not true: one can have $\phi \subseteq \varepsilon\backslash\varepsilon'$, with ϕ not a lax natural transformation.

Finally, we remark that the class of relators that have membership is closed under composition of functors. In particular, all *regular* functors (built from products, sums, composition, and least fixed points) have membership, and are thus container types.

In the subsection below, proofs of the above three facts are spelled out. Readers who first wish to get a general overview of our results can skip to the next section without loss of continuity.

4.1 Proofs

We start by giving an equivalent definition of membership which is sometimes more convenient in proofs than the official version given above. It does, however, contain another bound variable, and therefore the official definition is perhaps easier to digest on first encounter.

Lemma 9
A collection of arrows ε is a membership relation for F iff for all R and S we have

$$\varepsilon\backslash(R \cdot S) \quad = \quad FR \cdot \varepsilon\backslash S.$$

Proof
The *follows-from* direction is trivial: take $S = id$. For *implies*, we argue

$$\begin{array}{cl} & \varepsilon\backslash(R \cdot S) \\ = & \quad \{\varepsilon \text{ membership}\} \\ & F(R \cdot S) \cdot \varepsilon\backslash id \\ = & \quad \{F \text{ relator}\} \\ & FR \cdot FS \cdot \varepsilon\backslash id \\ = & \quad \{\varepsilon \text{ membership}\} \\ & FR \cdot \varepsilon\backslash S \end{array}$$

□

Note that neither the original definition of membership nor the preceding lemma make reference to the fact that membership is a lax natural transformation. The reason is, of course, that naturality can be deduced from the definition of membership:

Restatement of Fact 6. If ε is a membership relation for F, then $\varepsilon : id \hookleftarrow F$.

Proof

$$\begin{array}{ll} & \varepsilon \cdot F R \subseteq R \cdot \varepsilon \\ \equiv & \quad \{\text{division}\} \\ & F R \subseteq \varepsilon \backslash (R \cdot \varepsilon) \\ \equiv & \quad \{\varepsilon \text{ membership, Lemma 9}\} \\ & F R \subseteq F R \cdot \varepsilon \backslash \varepsilon \\ \equiv & \quad \{\text{since } id \subseteq \varepsilon \backslash \varepsilon\} \\ & true \end{array}$$

□

While exploring naturality of membership, we might as well mention that the relation $\varepsilon \backslash id$ in the original definition of membership is also natural, in the opposite direction of ε itself. Although the proof of this fact is nearly trivial, it is still worthwhile to record it separately for future reference.

Lemma 10
Suppose that ε is a membership relation for F. Then $\varepsilon \backslash id$ is a lax natural transformation of type $F \hookleftarrow id$.

Proof

$$\begin{array}{ll} & \varepsilon \backslash id \cdot R \\ \subseteq & \quad \{\text{division}\} \\ & \varepsilon \backslash R \\ = & \quad \{\text{membership}\} \\ & F R \cdot \varepsilon \backslash id \end{array}$$

□

Now we are in a position to prove the fundamental result that asserts uniqueness of membership. We take the elegance of the proof as evidence that the definitions given here are right: one certainly would not wish the definition of 'container type' to lead to intricate and cumbersome proofs.

Fact 11
If ε is a membership relation for F, then ε is the largest lax natural transformation of type $id \hookleftarrow F$.

Proof
Let ε be a membership relation for F. By Lemma 6, $\varepsilon : id \hookleftarrow F$. Let ϕ be another lax natural transformation of type $id \hookleftarrow F$. Then

$$\begin{array}{ll} & \phi \\ \subseteq & \quad \{\text{division}\} \\ & \phi \cdot \varepsilon \backslash \varepsilon \\ = & \quad \{\text{membership}\} \end{array}$$

$$\begin{array}{ll} & \phi \cdot F\varepsilon \cdot \varepsilon\backslash id \\ \subseteq & \quad \{\text{since } \phi : id \hookleftarrow F\} \\ & \varepsilon \cdot \phi \cdot \varepsilon\backslash id \\ \subseteq & \quad \{\text{claim: see below}\} \\ & \varepsilon \end{array}$$

The claim is that $\phi \cdot \varepsilon\backslash id \subseteq id$. This does in fact follow from the identification axiom, which says that id is the largest lax natural transformation of type $id \hookleftarrow id$. We have $\phi \cdot \varepsilon\backslash id : id \hookleftarrow id$ because $\phi : id \hookleftarrow F$, and Lemma 10 says that $\varepsilon\backslash id : F \hookleftarrow id$.
□

Finally, as an application of the little theory developed above, we prove a result about largest lax natural transformations between an arbitrary pair of relators that have membership. It also happens that a special case of this result is useful in the section on fans below.

Restatement of Fact 8. Let F and F' be relators with membership relations ε and ε' respectively. Then $\varepsilon\backslash\varepsilon'$ is the largest lax natural transformation of type $F \hookleftarrow F'$.

Proof
First note that $\varepsilon\backslash\varepsilon' : F \hookleftarrow F'$, for it is the composition of two lax natural transformations (Fact 6 and Lemma 10):

$$\varepsilon\backslash\varepsilon' \quad = \quad F\varepsilon' \cdot \varepsilon\backslash id.$$

To prove that $\varepsilon\backslash\varepsilon'$ contains any other lax natural transformation of type $F \hookleftarrow F'$, let $\phi : F \hookleftarrow F'$. We have

$$\begin{array}{ll} & \phi \subseteq \varepsilon\backslash\varepsilon' \\ \equiv & \quad \{\text{division}\} \\ & \varepsilon \cdot \phi \subseteq \varepsilon' \\ \Leftarrow & \quad \{\text{Lemma 11}\} \\ & \varepsilon \cdot \phi : id \hookleftarrow F' \\ \equiv & \quad \{\text{since } \varepsilon : id \hookleftarrow F \text{ and } \phi : F \hookleftarrow F'\} \\ & \mathit{true} \end{array}$$

□

While these initial results are encouraging, there remains the question whether we could not have avoided the peculiar definition of membership, by simply defining the membership relation for F as the largest lax natural transformation $id \hookleftarrow F$. It is easily checked that in *Rel* such a largest lax natural transformation exists for any F, since the union of a collection of lax natural transformations is again lax natural. We aim to show, therefore, that there exist relators that do not have membership in our sense, hence we show that the largest lax natural transformation $id \hookleftarrow F$ is not necessarily a membership relation.

Restatement of Fact 7. There exists a relator that does not have membership.

As a first step towards constructing such an example, observe that a container type constructor ought to preserve intersection of subsets: for any collection X of subsets of a set C, the set of F-structures over $\bigcap X$ should be precisely the intersection $\bigcap\{F A' \mid A' \in X\}$. Indeed, it is easily verified that this condition holds for the examples (lists, powersets and exponentials) considered so far.

We can formalise the above intuition as follows (the restriction to *Rel* is necessary to guarantee existence of arbitrary intersections):

Lemma 12
Let F be a relator of *Rel* that has membership. Then F preserves arbitrary intersections of coreflexives.

Proof
It is possible to give a direct proof of this lemma, but such a proof is clumsy. Instead, we prefer to use the fact that the right (or upper) adjoint in a Galois connection preserves intersections; readers who are not familiar with Galois connections are referred to Gierz *et al.* (1980). Below it is shown that F is a right adjoint: let C and D be coreflexives of an object A. We have

$$\begin{array}{ll}
 & D \subseteq FC \\
\equiv & \quad \{\text{order-isomorphism between coreflexives and left-conditions}\} \\
 & D \cdot \Pi \subseteq FC \cdot \Pi \\
\equiv & \quad \{\text{division: } \varepsilon \backslash \Pi = \Pi\} \\
 & D \cdot \Pi \subseteq FC \cdot \varepsilon \backslash \Pi \\
\equiv & \quad \{\text{membership}\} \\
 & D \cdot \Pi \subseteq \varepsilon \backslash (C \cdot \Pi) \\
\equiv & \quad \{\text{division}\} \\
 & \varepsilon \cdot D \cdot \Pi \subseteq C \cdot \Pi \\
\equiv & \quad \{\text{range: } R \cdot \Pi = \mathit{Ran}(R) \cdot \Pi\text{, order-isomorphism}\} \\
 & \mathit{Ran}(\varepsilon \cdot D) \subseteq C
\end{array}$$

□

This last result suggests a strategy for proving that not every relator has a membership relation. It suffices to find a relator F, an object A and a collection X of coreflexives of A such that F does not preserve the intersection of X. Since any relator preserves finite intersections of coreflexives, the collection X will have to be infinite.

A truly convincing example would satisfy a number of additional requirements. First, F should be a functor of *Fun*, for that is the model of primary interest. Second, F should preserve binary intersections of relations: this is a property of many container types, albeit not of the power set relator. Readers who are intimately familiar with category theory will realise that these requirements can be stated a bit more concisely: we want a functor F on *Fun* that preserves pullbacks, plus an object A and a collection X of subobjects of A such that F does not preserve the intersection of X. This formulation in categorical terms has the advantage that it does not mention relations. Although we were able to phrase our requirements in this style, at the time we were unable to construct an example ourselves. The question was finally answered by Peter Freyd, and we now proceed to sketch his construction.

First consider the functor $G = \mathit{From}_N$, where N is the set of natural numbers. For any set A, one can think of GA as the set of infinite sequences over A. As we have already seen, G is a relator. For any A, we can define an equivalence relation R on GA by

$$sRt \quad \equiv \quad (\exists m : (\forall i : m \leqslant i : s\,i = t\,i)).$$

In words, two sequences s and t are related by R when they are 'eventually equal'. It is easily checked that R is indeed an equivalence relation. Define FA to be the set of equivalence classes in GA, and let $q_A : FA \leftarrow GA$ be the function that sends a sequence to the corresponding equivalence class.

We can make F into a functor by defining its action on functions by

$$Fh \quad = \quad q_A \cdot Gh \cdot q_B{}^{\circ}, \qquad \text{for } h : A \leftarrow B.$$

There are quite a number of things to verify now: we should check that Fh is indeed a function, that F preserves composition and identities, and that F is a relator which preserves binary intersections. These verifications are however rather tedious, and we omit details.

It remains to construct an object A, together with a collection X of subsets of A so that F does not preserve the intersection of X. We take for A the set of natural numbers itself, and X is defined by

$$X \quad = \quad \{\{i \,|\, m < i\} \,|\, m \in N\}.$$

Technically speaking the components of X should be subsets of the identity relation, but we sweep the distinction between such relational subsets and ordinary subsets under the carpet. Note that the intersection of X is empty, and that $F(\bigcap X)$ is also the empty set. However, for each $C \in X$, FC contains $q_N\,id$. So the intersection $\bigcap\{FC \,|\, C \in X\}$ is nonempty. Freyd's counterexample is therefore complete. As we shall see below, his construction is useful in refuting other conjectures in the theory of container types.

5 Container types have fans

Not only do we wish to inspect the contents of data structures; we should also be able to create them. Therefore, any type constructor F comes equipped with a family of relations that captures the idea of creating F-structures. There are various formalisations of such families in the literature, notably *strengths* and *copy maps*. We shall consider these in the next section, but first we explore an intuitively simpler notion of data structure creation.

A *fan* is a nondeterministic mapping that, when given a seed value a, creates an F-structure all of whose elements are equal to a. Formally, a fan is a lax natural transformation of type $\phi : F \hookleftarrow id$ such that the function $\lambda R : (FR \cdot \phi)$ preserves finite intersections ($\Pi : A \leftarrow B$ is the largest relation of its type):

$$F\Pi \cdot \phi = \Pi \quad \text{and} \quad F(R \cap S) \cdot \phi = (FR \cdot \phi) \cap (FS \cdot \phi).$$

This definition of fans originated with Backhouse *et al.* (1993), where they were called *generators*. Remarkably, membership guarantees the existence of unique fans:

Fact 13
If F has membership, then $\varepsilon \backslash id$ is the unique fan for F.

The converse is not true: as Freyd's example shows, it is possible for a relator to have a fan, but no membership. It follows that there is no need to revise our current definition of a container type: we stick to the view that it is a relator that has membership. Again, the proofs in the next subsection can be skipped without loss of continuity.

5.1 *Proof*

The proof strategy will be as follows. First, we show that any two fans are either incomparable (under $\subseteq$), or they are equal. Then we note that $\varepsilon \backslash id$ is the largest fan for F. Together these two lemmas give the desired result, namely that $\varepsilon \backslash id$ is the unique fan for F.

To prove the first lemma, we shall need an auxiliary technical result, which states that two F-structures of the same shape, both of which have been created with the same fan, are equal. (To be precise, this is what the lemma below says when we take $R = \Pi$, and $S = T = id$.) The proof is admittedly somewhat unattractive, but we see no other way.

Lemma 14
Let ϕ be a fan for F. Then, for all R, S and T, we have

$$(FR) \cap (FS \cdot \phi \cdot \phi^\circ \cdot FT) \quad \subseteq \quad F(R \cap (S \cdot T)).$$

Proof
In the proof below, we shall use the fact that for any relation R, there exist functions f and g so that $R = f \cdot g^\circ$. We shall also need the properties of the range operator Ran, as discussed in section 2.6. With these facts in hand, we calculate as follows:

$$\begin{array}{ll}
 & F(f \cdot g^\circ) \ \cap \ (FS \cdot \phi \cdot \phi^\circ \cdot FT) \\
= & \quad \{\text{modular identity, } F \text{ relator}\} \\
 & Ff \cdot (id \cap (F(f^\circ \cdot S) \cdot \phi \cdot \phi^\circ \cdot F(T \cdot g))) \cdot Fg^\circ \\
= & \quad \{\text{range of intersection}\} \\
 & Ff \cdot Ran(F(f^\circ \cdot S) \cdot \phi \ \cap \ F(T \cdot g)^\circ \cdot \phi) \cdot Fg^\circ \\
= & \quad \{\phi \text{ is a fan}\} \\
 & Ff \cdot Ran(F((f^\circ \cdot S) \cap (T \cdot g)^\circ) \cdot \phi) \cdot Fg^\circ
\end{array}$$

$$
\begin{array}{ll}
\subseteq & \{\text{since } \mathit{Ran}(X \cdot Y) \subseteq \mathit{Ran}\, X\} \\
& Ff \cdot \mathit{Ran}(F((f^\circ \cdot S) \cap (T \cdot g)^\circ)) \cdot Fg^\circ \\
= & \{\text{relators preserve range}\} \\
& Ff \cdot F(\mathit{Ran}((f^\circ \cdot S) \cap (T \cdot g)^\circ)) \cdot Fg^\circ \\
= & \{\text{range of intersection}\} \\
& Ff \cdot F(id \cap (f^\circ \cdot S \cdot T \cdot g)) \cdot Fg^\circ \\
= & \{\text{modular identity, } F \text{ relator}\} \\
& F((f \cdot g^\circ) \cap (S \cdot T))
\end{array}
$$

□

The above result, though somewhat tricky to prove itself, facilitates a nice proof that fans are either incomparable or equal:

Lemma 15
Let μ and ϕ both be fans of F. If $\phi \subseteq \mu$, then $\phi = \mu$.

Proof
In the proof below, Π stands for the largest relation of appropriate type. We argue

$$
\begin{array}{ll}
& \mu \\
= & \{\text{intersection}\} \\
& \mu \cap \Pi \\
= & \{\text{since } \phi \text{ is a fan}\} \\
& \mu \cap (F\Pi \cdot \phi) \\
\subseteq & \{\text{modular law (also known as Dedekind's rule)}\} \\
& (\mu \cdot \phi^\circ \;\cap\; F\Pi) \cdot \phi \\
\subseteq & \{\text{given: } \phi \subseteq \mu\} \\
& (\mu \cdot \mu^\circ \;\cap\; F\Pi) \cdot \phi \\
\subseteq & \{\text{Lemma 14}\} \\
& \phi
\end{array}
$$

□

The above results do not make use of membership. When a relator has membership, there is an obvious candidate for the fan, namely the relation $\varepsilon\backslash id$. Given the importance of this relation in the definition of membership, it should be no surprise that $\varepsilon\backslash id$ is of independent interest.

Fact 16
Let F be a relator whose membership relation is ε. Then $\varepsilon\backslash id$ is a fan for F.

The proof of this fact is trivial, for $\lambda X : Y\backslash X$ preserves intersections for all Y, in particular when $Y = \varepsilon$. Finally, we can put all the above results together to obtain that a relator with membership has precisely one fan.

Restatement of Fact 13. Let F be a relator that has membership. Then $\varepsilon\backslash id$ is the unique fan for F.

Proof
We have just proved that $\varepsilon\backslash id$ is indeed a fan. Furthermore, by Fact 8, it is the largest lax natural transformation of its type. Therefore any other fan is included in it, and Lemma 15 gives the desired result. $\square$

6 Container types have strength

Above we already alluded to work of others which also (at least implicitly) aimed to pin down the notion of container types (as well as more general data types). Because the details are rather more technical than those in preceding sections, we first outline the main ideas and results.

The related work concentrates on ways of creating data structures; it is not concerned with relators or membership. Functors that have a certain data structure creation mechanism are said to be *strong* (Moggi, 1991; Cockett & Spencer, 1992). Because little interesting can be said about strong functors in connection to arbitrary natural transformations, natural transformations are required to satisfy an additional condition; such natural transformations are also called *strong*. The main result of this section says that relators that have membership are necessarily strong. Furthermore, any natural transformation (between relators that have membership) is strong. So all conditions related to strength come for free if we have a relator that has membership; but there are strong relators that do not have membership. The conclusion is that our current proposal for the definition of a container type, namely a relator that has membership, refines earlier attempts in the literature. Readers who are not familiar with category theory may wish to review our introduction to finite products in section 2.8 before proceeding.

Let F be a functor. A *strength* of F is a collection of functions $\theta : F(A \times B) \leftarrow FA \times B$ that is natural in the following sense:

$$F(h \times k) \cdot \theta \quad = \quad \theta \cdot (Fh \times k) \quad \text{for all functions } h \text{ and } k.$$

Furthermore, there are two conditions to ensure that θ interacts properly with products:

$$F\,rid \cdot \theta = rid \qquad \text{and} \qquad F\,assl \cdot \theta = \theta \cdot (\theta \times id) \cdot assl.$$

A functor that has a strength is said to be *strong*. It is possible for a functor to have several different strengths; every functor of *Fun* is strong.

It is worth thinking about the strengths of our three example relators. Using a so-called *list comprehension* (a common device in functional programming) the strength of the list functor is given by

$$\theta(x,b) \quad = \quad [(a,b) \mid a \leftarrow x].$$

Similarly, the strength of the powerset functor is

$$\theta(x,b) \quad = \quad \{(a,b) \mid a \in x\}.$$

Finally the strength of the exponential functor is

$$\theta(f,c) \quad = \quad \lambda b : (f\,b,c).$$

Some readers may be puzzled by our introduction of strength as a data structure creation mechanism, as it may not be immediate what is being created here. Perhaps the terminology becomes more perspicuous when one programs the so-called *copy map* in terms of strength. For a functor F, the copy map is a collection of arrows $c : F\,A \leftarrow F\,1 \times A$, defined by

$$c \quad = \quad F\,lid \cdot \theta,$$

where $lid : A \leftarrow (1 \times A)$ is the obvious isomorphism. The intention of this definition is that c takes a shape (a value of type $F\,1$) and a seed value (of type A) and that it returns an F-structure of the same shape all of whose elements are equal to the seed value. Clearly copy maps are closely related to the notion of fans; we shall however not further elaborate the connection. Copy maps are equivalent to strengths for a certain class of functors (to the categorically wise: functors that preserve pullbacks): one can define copy maps independently, and then prove that there exists a one-to-one correspondence between copy maps and strengths. The interested reader is referred to Jay (1994).

Another useful operation that can be programmed in terms of strength is the *map transformation*. Let F be a functor of *Fun* that has strength θ. We can then define $map : (F\,A \leftarrow F\,B) \leftarrow (A \leftarrow B)$ by

$$map\,f\,x \quad = \quad (F\,app)\,(\theta(x,f)),$$

where $app(a,f) = f\,a$. One could say that *map* internalises the action of F as a collection of arrows within *Fun*. The above construction of *map* generalises to any category that has exponentials. Again *map* can be defined as an alternative to strengths: the strengths of F are in one-to-one correspondence with its map transformations. Details can be found in Kock's (1972) influential paper. Using Kock's correspondence we get immediately that all functors of *Fun* are strong. In particular, the functor F in Freyd's counterexample is strong, and thus we have an example of a strong functor that does not have membership.

The correspondence between strengths, copy maps, and map transformations hopefully convinces the reader of the importance of its rather technical definition. We now proceed to detail the connection with fans. As we shall see, there is a one-to-one correspondence between fans and a special kind of strength.

Let F be a relator that has strength θ. The strength θ is said to be *relational* if it satisfies the naturality condition

$$F(R \times id) \cdot \theta \quad = \quad \theta \cdot (F\,R \times id).$$

Note that the inclusion $\supseteq$ is almost immediate from the definition of θ; the additional requirement is therefore that we also have $\subseteq$. It can be shown that if F preserves binary intersections of relations, then any strength of F is relational. We have been unable to prove that for arbitrary F, but we have also been unable to find a strength that is not relational. This is disappointing, and the matter obviously needs to be resolved, but in any case we can make progress:

Fact 17
Let F be a relator. Then the relational strengths of F are in one-to-one correspondence with the fans of F.

This result does not use division or the identification axiom. In particular, it does not require that F has membership. However, if one does make these additional assumptions, one obtains

Fact 18
Let F be a relator that has membership. Then F has a unique strength, and that strength is relational.

As an aside, we note that using the axiom of extensionality (and some weakenings of that axiom), one can also establish uniqueness of strength (Moggi, 1991). It is unlikely that any interesting improvements can be made without assuming the identification axiom (or subextensionality, which implies identification): without it, even the identity functor need not have a unique strength! (To the categorically wise: consider the topos of G-sets Set^G, where G is a non-trivial abelian group.) The condition that F has membership cannot be omitted in the above result, for there are relators that have neither membership nor strength: an example is $F(R,S) = (S,R)$ on Rel^2.

As indicated in the brief overview at the beginning of this section, the theory of strong functors requires that natural transformations behave consistently with respect to strength. Let F and G be relators with strengths θ and κ respectively. A lax natural transformation $\alpha : F \hookleftarrow G$ is said to be *strong* if

$$\theta \cdot (\alpha \times id) \quad = \quad \alpha \cdot \kappa.$$

For the unique strength constructed from membership, this condition is always satisfied:

Fact 19
Let F and G be relators that have membership. Then any lax natural transformation $F \hookleftarrow G$ is strong.

This saves considerable proof effort when working with strength, and the result might have applications in the work on computational monads, which has recently attracted a lot of attention in the functional programming community. It is precisely this type of saving in tedious calculations that we hope to gain by identifying properties that are common to all container types. As in the case of fans, strength does not require further refinement of our definition of container types as relators that have membership.

The proofs of the results in this subsection require some excruciating symbol manipulation: such intricate yet tedious proofs are a common feature of arguments involving strength. As our results show, most of these manipulations can be entirely avoided when our definition of container types is adopted. Readers who have a taste for symbol pushing are invited to read the next subsection, or indeed to reconstruct it for themselves; others might wish to jump to the conclusions.

6.1 Proofs

To establish the connection between fans, strength, and membership we shall need an alternative (but equivalent) definition of fans that is in terms of products instead of intersections. As indicated in section 2.8, we can do this by exploiting the close relationship between Π and $!$, and between $\langle \_ , \_ \rangle$ and intersection.

Recall the original definition of fans: a lax natural transformation $\phi : F \hookleftarrow id$ is a *fan* if the mapping $\lambda R : FR \cdot \phi$ preserves finite intersections. Given the intimate connection between intersections and products, the following fact will come as no surprise:

Fact 20
Let F be a relator, and let ϕ be a lax natural transformation of type $id \hookleftarrow F$. Define $\mu = \phi^\circ$. Then ϕ is a fan for F iff

$$\begin{array}{rcl} (\mu : 1 \leftarrow F1) & = & ! \ , \quad \text{and} \\ (\mu : A \times B \leftarrow F(A \times B)) & = & \langle \mu \cdot F outl, \mu \cdot F outr \rangle . \end{array}$$

Proof
First assume that ϕ is a fan for F. We aim to show that μ satisfies the two above equations:

$$\begin{array}{ll} & \mu : 1 \leftarrow F1 \\ = & \quad \{\text{identity arrow}\} \\ & \mu \cdot F id \\ = & \quad \{\text{since } (id : 1 \leftarrow 1) = \Pi\} \\ & \mu \cdot F\Pi \\ = & \quad \{\text{converse, definition of } \mu\} \\ & (F\Pi \cdot \phi)^\circ \\ = & \quad \{\text{since } \phi \text{ is a fan}\} \\ & \Pi^\circ \\ = & \quad \{\text{since } \Pi^\circ = \Pi = !\} \\ & ! \end{array}$$

For the second equation, we reason:

$$\begin{array}{ll} & \langle \mu \cdot F outl, \mu \cdot F outr \rangle \\ = & \quad \{\text{definition of split}\} \\ & outl^\circ \cdot \mu \cdot F outl \ \cap \ outr^\circ \cdot \mu \cdot F outr \\ = & \quad \{\text{definition of } \mu\text{, converse}\} \\ & (\phi \cdot outl)^\circ \cdot F outl \ \cap \ (\phi \cdot outr)^\circ \cdot F outr \\ = & \quad \{\text{naturality of } \phi\} \\ & (F outl \cdot \phi)^\circ \cdot F outl \ \cap \ (F outr \cdot \phi)^\circ \cdot F outr \end{array}$$

$$
\begin{array}{ll}
= & \quad \{\text{converse, } F \text{ relator}\} \\
 & (F(outl^\circ \cdot outl) \cdot \phi \ \cap \ F(outr^\circ \cdot outr) \cdot \phi)^\circ \\
= & \quad \{\phi \text{ is a fan}\} \\
 & (F(outl^\circ \cdot outl \cap outr^\circ \cdot outr) \cdot \phi)^\circ \\
= & \quad \{\text{since } outl^\circ \cdot outl \cap outr^\circ \cdot outr = id\} \\
 & \phi^\circ \\
= & \quad \{\text{definition of } \mu\} \\
 & \mu
\end{array}
$$

This completes the proof that the old definition of fan implies the new one. Now assume that μ satisfies the two equations given above. It is our task to show that ϕ is a fan. For the first equation:

$$
\begin{array}{ll}
 & F\Pi \cdot \phi \\
= & \quad \{\text{since } \Pi = !^\circ \cdot !, F \text{ relator}\} \\
 & F!^\circ \cdot F! \cdot \phi \\
= & \quad \{\phi : F \hookleftarrow id\} \\
 & F!^\circ \cdot \phi \cdot ! \\
= & \quad \{\text{converse, } F \text{ relator, definition of } \mu\} \\
 & (\mu \cdot F!)^\circ \cdot ! \\
= & \quad \{\text{given: } \mu \cdot F! = !\} \\
 & !^\circ \cdot ! \\
= & \quad \{\text{since } \Pi = !^\circ \cdot !\} \\
 & \Pi
\end{array}
$$

The proof that $\lambda R : FR \cdot \phi$ also preserves binary intersections goes as follows: (abbreviate $z = \langle F\,outl, F\,outr\rangle$)

$$
\begin{array}{ll}
 & F(R \cap S) \cdot \phi = FR \cdot \phi \cap FS \cdot \phi \\
\equiv & \quad \{\text{since } R \cdot U \cap S \cdot V = \langle R^\circ, S^\circ\rangle^\circ \cdot \langle U, V\rangle, F \text{ relator}\} \\
 & F\langle R^\circ, S^\circ\rangle^\circ \cdot F\langle id, id\rangle \cdot \phi = \langle FR^\circ, FS^\circ\rangle^\circ \cdot \langle \phi, \phi\rangle \\
\equiv & \quad \{\phi : F \hookleftarrow id, \text{ and } \langle id, id\rangle \text{ function, Lemma 5}\} \\
 & F\langle R^\circ, S^\circ\rangle^\circ \cdot \phi \cdot \langle id, id\rangle = F\langle R^\circ, S^\circ\rangle^\circ \cdot z^\circ \cdot \langle \phi, \phi\rangle \\
\Leftarrow & \quad \{\text{since } \langle \phi, \phi\rangle = (\phi \times \phi) \cdot \langle id, id\rangle\} \\
 & \phi = z^\circ \cdot \phi \times \phi \\
\equiv & \quad \{\phi = \mu^\circ, \text{ converse}\} \\
 & \mu = \mu \times \mu \cdot z \\
\equiv & \quad \{\text{meaning of } z, \text{ product absorption, properties of } \mu\} \\
 & true
\end{array}
$$

□

It is now fairly easy to set up the correspondence between fans and relational strengths, as there is almost a one-to-one correspondence between the required properties. First, we show how to construct a strength from a fan.

Fact 21
Let F be a relator. Assume that ϕ is a fan for F. Define $\mu = \phi^\circ$. Then

$$\theta \quad = \quad \langle F\,outl, \mu \cdot F\,outr\rangle^\circ$$

is a relational strength of F. Furthermore, we have

$$\phi \quad = \quad F\,lid \cdot \theta \cdot outr^\circ,$$

where $lid : A \leftarrow (1 \times A)$ is the obvious isomorphism.

Proof
That θ is lax natural and satisfies the additional naturality property of a *relational* strength follows from Lemma 4, and

$$\theta \quad = \quad \langle F\,outl, F\,outr\rangle^\circ \cdot (id \times \phi).$$

The two equations

$$F\,rid \cdot \theta = rid \qquad \text{and} \qquad F\,assl \cdot \theta = \theta \cdot (\theta \times id) \cdot assl$$

follow from Fact 20; details are omitted. To show that θ is a function, we need to show both totality and single-valuedness. For totality, we shall show that $Dom\,\theta = id$, where $Dom\,X = id \cap X^\circ \cdot X = Ran(X^\circ)$:

$$\begin{array}{cl}
 & Dom\,\theta \\
= & \quad \{\text{definition of } \theta,\ Dom(R) = Ran(R^\circ),\ \text{range of split}\} \\
 & id \cap outl^\circ \cdot F(outl \cdot outr^\circ) \cdot \phi \cdot outr \\
= & \quad \{\text{since } \Pi = outl \cdot outr^\circ\} \\
 & id \cap outl^\circ \cdot F\Pi \cdot \phi \cdot outr \\
= & \quad \{\phi \text{ fan}\} \\
 & id \cap outl^\circ \cdot \Pi \cdot outr \\
= & \quad \{\text{both } outl \text{ and } outr \text{ are total}\} \\
 & id \cap \Pi \\
= & \quad \{\text{intersection}\} \\
 & id
\end{array}$$

To prove that θ is single-valued, we reason as follows,

$$\begin{array}{cl}
 & \theta \cdot \theta^\circ \\
= & \quad \{\text{definitions of } \theta,\ \text{split and intersection}\} \\
 & F(outl^\circ \cdot outl) \cap F\,outr^\circ \cdot \phi \cdot \phi^\circ \cdot F\,outr \\
\subseteq & \quad \{\text{Lemma 14}\} \\
 & F(outl^\circ \cdot outl \cap outr^\circ \cdot outr) \\
= & \quad \{\text{since } outl^\circ \cdot outl \cap outr^\circ \cdot outr = id\} \\
 & id
\end{array}$$

It remains to show that ϕ can be recovered from θ:

$$
\begin{array}{ll}
 & F\,lid \cdot \theta \cdot outr^{\circ} \\
= & \quad \{\text{definition of } \theta\} \\
 & F\,lid \cdot \langle F\,outl, \mu \cdot F\,outr\rangle^{\circ} \cdot outr^{\circ} \\
= & \quad \{\text{converse, split cancellation } (F\,outl \text{ total})\} \\
 & F\,lid \cdot F\,outr^{\circ} \cdot \mu^{\circ} \\
= & \quad \{F \text{ relator, } lid \cdot outr^{\circ} = id,\ \mu^{\circ} = \phi\ \} \\
 & \phi
\end{array}
$$

□

Fortunately, the transition from relational strengths to fans is much easier to verify, because the definition of fans is simpler, and so there is less to check. In fact, we omit the proof, because it is a mostly mechanical exercise. We have thus completed our discussion of the correspondence between fans and strengths:

Restatement of Fact 17. Let F be a relator. Then the relational strengths of F are in one-to-one correspondence with the fans of F.

Together with Fact 13, this result proves that a relator with membership has a unique relational strength, but we can in fact do slightly better than that. Let F be a relator with membership relation ε. Call

$$
\begin{array}{ll}
 & \langle F\,outl, (\varepsilon\backslash id)^{\circ} \cdot F\,outr\rangle^{\circ} \\
= & \\
 & F\,outl^{\circ} \cdot outl \ \cap\ \varepsilon\backslash(outr^{\circ} \cdot outr).
\end{array}
$$

the *canonical strength* Θ of F. To prove that the canonical strength is the only strength, it suffices to prove that $\theta \subseteq \Theta$ for all strengths θ of F since strengths are functions. By shunting of functions, it follows that $\theta \subseteq \Theta$ is equivalent to the conjunction

$$F\,outl \cdot \theta = outl \quad \text{and} \quad outr \cdot \varepsilon \cdot \theta \subseteq outr.$$

We shall prove these two equations as separate lemmas.

Lemma 22
Let F be a functor with strength θ. Then $F\,outl \cdot \theta = outl$.

Proof

$$
\begin{array}{ll}
 & F\,outl \cdot \theta \\
= & \quad \{\text{naturality of outl}\} \\
 & F\,outl \cdot F(id \times !) \cdot \theta \\
= & \quad \{\text{naturality of } \theta\} \\
 & F\,outl \cdot \theta \cdot (id \times !) \\
= & \quad \{\text{since } (outl\ : A \leftarrow A \times 1) = rid, \theta \text{ strength}\}
\end{array}
$$

$$
\begin{array}{ll}
 & outl \cdot (id \times !) \\
= & \quad \{\text{naturality of } outl\} \\
 & outl
\end{array}
$$

□

Lemma 23
Let F be a relator with membership relation ε, and let θ be a strength of F. Then $outr \cdot \varepsilon \cdot \theta \subseteq outr$.

Proof
Define $G\,R = id \times R$. Then $outr$ is the membership relation of G, and therefore the largest lax natural transformation $id \hookleftarrow G$. The composition $outr \cdot \varepsilon \cdot \theta$ is a lax natural transformation of the same type. □

It seems fitting that we end our exploration of uniqueness of strength with such a delightful little proof. We believe that the theory of largest lax natural transformations put forward here simplifies many polytypic arguments, especially ones that would otherwise require an appeal to extensionality (pointwise reasoning). Indeed, we first tried to prove the results of this paper by pointwise means, and although we mostly succeeded at the time, the proofs were rather impenetrable. The achievements so far are summed up in:

Restatement of Fact 18. If F is a relator that has membership, then it has a unique strength, and that strength is relational.

In what follows, we shall denote the unique strength of F by θ_F, and its unique fan by ϕ_F. Now it only remains to show that all lax natural transformations are strong, in the sense that

$$\theta_F \cdot (\alpha \times id) \quad = \quad \alpha \cdot \theta_G,$$

for each $\alpha : F \hookleftarrow G$. We do so by proving an inclusion for each direction of the equation:

Lemma 24
Let F and G be relators that have membership, and let α be a lax natural transformation of type $F \hookleftarrow G$. Then

$$\theta_F \cdot (\alpha \times id) \quad \supseteq \quad \alpha \cdot \theta_G.$$

Proof

$$
\begin{array}{ll}
 & \alpha \cdot \theta_G \\
= & \quad \{\text{canonical strength of } G\} \\
 & \alpha \cdot \langle G\,outl, \phi_G{}^\circ \cdot G\,outr\rangle^\circ \\
\subseteq & \quad \{\text{converse, split: } \langle X, Y\rangle \cdot Z \subseteq \langle X \cdot Z, Y \cdot Z\rangle\} \\
 & \langle G\,outl \cdot \alpha^\circ, \phi_G{}^\circ \cdot G\,outr \cdot \alpha^\circ\rangle^\circ
\end{array}
$$

$$
\begin{array}{ll}
\subseteq & \{\text{converse, } \alpha : F \hookleftarrow G\} \\
& \langle \alpha^{\circ} \cdot F\,outl, {\phi_G}^{\circ} \cdot \alpha^{\circ} \cdot F\,outr\rangle^{\circ} \\
\subseteq & \{\alpha \cdot \phi_G : F \hookleftarrow id \text{, and } \phi_F \text{ largest of this type}\} \\
& \langle \alpha^{\circ} \cdot F\,outl, {\phi_F}^{\circ} \cdot F\,outr\rangle^{\circ} \\
= & \{\text{converse, product absorption}\} \\
& \langle F\,outl, {\phi_F}^{\circ} \cdot F\,outr\rangle^{\circ} \cdot (\alpha \times id) \\
= & \{\text{canonical strength of } F\} \\
& \theta_F \cdot (\alpha \times id)
\end{array}
$$

□

Lemma 25
Let F and G be relators that have membership, and let α be a lax natural transformation of type $F \hookleftarrow G$. Then

$$\theta_F \cdot (\alpha \times id) \quad \subseteq \quad \alpha \cdot \theta_G .$$

Proof
First we note that, because θ is a function, the proof obligation is equivalent to

$$(\alpha \times id) \cdot {\theta_G}^{\circ} \quad \subseteq \quad {\theta_F}^{\circ} \cdot \alpha$$

or, equivalently,

$$\theta_G \cdot (\alpha^{\circ} \times id) \quad \subseteq \quad \alpha^{\circ} \cdot \theta_F .$$

This inequation can be proved as follows:

$$
\begin{array}{ll}
& \theta_G \cdot (\alpha^{\circ} \times id) \\
= & \{\text{canonical strength of } G\} \\
& \langle G\,outl, {\phi_G}^{\circ} \cdot G\,outr\rangle^{\circ} \cdot (\alpha^{\circ} \times id) \\
= & \{\text{converse, product absorption}\} \\
& \langle \alpha \cdot G\,outl, {\phi_G}^{\circ} \cdot G\,outr\rangle^{\circ} \\
\subseteq & \{\alpha : F \hookleftarrow G\} \\
& \langle F\,outl \cdot \alpha, {\phi_G}^{\circ} \cdot G\,outr\rangle^{\circ} \\
\subseteq & \{\text{converse, modular law: } \langle R \cdot S, T\rangle \subseteq \langle R, T \cdot S^{\circ}\rangle \cdot S\} \\
& \alpha^{\circ} \cdot \langle F\,outl, {\phi_G}^{\circ} \cdot G\,outr \cdot \alpha^{\circ}\rangle^{\circ} \\
\subseteq & \{\text{converse, } \alpha : F \hookleftarrow G\} \\
& \alpha^{\circ} \cdot \langle F\,outl, {\phi_G}^{\circ} \cdot \alpha^{\circ} \cdot F\,outr\rangle^{\circ} \\
\subseteq & \{\alpha \cdot \phi_G : F \hookleftarrow id \text{, and } \phi_F \text{ largest of this type}\} \\
& \alpha^{\circ} \cdot \langle F\,outl, {\phi_F}^{\circ} \cdot F\,outr\rangle^{\circ} \\
= & \{\text{canonical strength of } F\} \\
& \alpha \cdot \theta_F
\end{array}
$$

□

We can now conclude

Restatement of Fact 19. Let F and G be relators that have membership. Then any lax natural transformation $F \hookleftarrow G$ is strong.

We find this fact quite remarkable, as it shows that all conditions regarding strong functors in the literature are vacuously satisfied. In particular, we obtain that any monad (on a on a container type) is strong.

7 Related work

Categories of relations Of course all the machinery we have used in this paper is well-established in the category theory community. The calculus of relations itself has a rich history, going back to De Morgan (1860), Peirce (1870) and Schröder (1895). The subject as we know it today was mostly shaped by Tarski and his students in a series of articles, starting with Tarski (1941). An overview of the origins of the relational calculus can be found in Maddux (1991) and Pratt (1992). During the 1960s, several authors started to explore relations in a categorical setting (Brinkmann, 1969; Mac Lane, 1961; Puppe, 1962). This resulted in a concensus that regular categories are the appropriate setting for studying relations in general (Grillet, 1970; Kawahara, 1973b). The latter paper also has the result about relators that forms the lynchpin of the present paper. The study of categories of relations has since received much more attention (Carboni *et al.*, 1984; Carboni & Street, 1986; Carboni & Walters, 1987; Carboni *et al.*, 1991). The definitive introduction to this area of category theory is the text book by Freyd and Ščedrov (1990). In the terminology of that text, all the proofs in this paper go through in an arbitrary logos $\mathscr{C}$, that satisfies the identification axiom.

The main advantage of the categorical viewpoint of relations is that one can freely move between functions and relations, choosing whichever is most convenient for the proof in hand. We have not capitalised much on this advantage, because it requires the introduction of some extra categorical machinery (pullbacks, image factorisation, strong epimorphisms), making the paper less self-contained. As Freyd's example illustrates, however, the freedom afforded by such extra machinery can lead to new results that are very hard to uncover otherwise.

Program derivation At the start of this paper we mentioned that this theory was developed for the purpose of program specification and derivation: detailed examples of its use can be found in earlier publications by ourselves (in particular Chapters 6 and 7 of Bird *et al.* (1996), as well as Bird & De Moor (1996)). Tuijnman (1996) studies strength in the context of program derivation. It seems likely that at least some of his proofs can be simplified using the results presented here, but we have not investigated this

Programming language design There are other approaches to the generic treatment of data types that are more geared towards programming language design. Drawing on the categorical view of types as functors, Cocket and Fukushima designed the programming language *Charity* (Cockett & Fukushima, 1991). This was followed by the design of an extension of Haskell, called *PolyP* (Jeuring & Jansson, 1996). The designers of PolyP explicitly set out to implement the programs we derived through the theory of membership in Bird & De Moor (1996). A yet more recent development is the design of *Functorial ML* (Bellé *et al.*, 1996). The work on Functorial ML has a sophisticated categorical semantics based on the idea of a *shapely functor*. Shapely functors (Jay, 1995b) give equal weight to the contents and the structure of a composite value. As we have pointed out, not every container type in our sense is shapely. A rather nice reworking of these results has recently been proposed by Hinze (1999). A formal connection between the calculus of relational specifications, and implementations in Functorial ML would clearly be of great value.

Many of the above references define properties of data types by induction on a class of functors. It is a natural question whether our definition of 'container type' is closed under the constructors of these classes. This is indeed the case. Write (μF) for the initial algebra of F, if it exists. The class of *Kleene functors* is inductively defined as follows:

- The identity functor $Fun \leftarrow Fun$ is a Kleene functor.
- For any set A, the constant functor $K\ A$ (sending every arrow to the identity on A) is a Kleene functor.
- If F and G are Kleene functors, so are their pointwise sum and product $(\lambda A : F\ A + G\ A)$ and $(\lambda A : F\ A \times G\ A)$.
- If F and G are Kleene functors, so is $F \cdot G$.
- Let $(\otimes)$ is a binary functor so that for each set A, $(\lambda X\ : A \otimes X)$ is a Kleene functor. Then the functor $(\lambda A : \mu(\lambda X\ : A \otimes X))$ is a Kleene functor.

The terminology derives from the fact that the combining operations are akin to those of a Kleene algebra. It is well-known (Manes & Arbib, 1986) that the class of Kleene functors is well-defined, in the sense that the initial algebras referred to in the last clause exist.

Fact 26
Kleene functors are container types.

The proof of this fact is easy, except perhaps for finding the membership of initial algebras. Let $(\otimes)$ be a binary functor as in the last clause of the definition of Kleene functors. Let $\alpha_A : T\ A \leftarrow (A \otimes T\ A)$ be the initial algebra of $(\lambda X\ : A \otimes X)$. It is our aim to show that T has membership. Assume (inductively) that $\lambda X\ : X \otimes B$ has membership relation ε_1 and $(\lambda X\ : A \otimes X)$ has membership relation ε_2. Then the membership relation of T is given by

$$\varepsilon_1 \cdot \alpha^\circ \cdot (\varepsilon_2 \cdot \alpha^\circ)^*$$

Here R^* denotes the reflexive transitive closure of R. Intuitively, we can read this result as follows. The relation $(\varepsilon_2 \cdot \alpha^\circ)$ can be viewed a non-deterministic mapping that given a tree will return one of its immediate subtrees. Its reflexive transitive closure thus returns *any* subtree. The relation $(\varepsilon_1 \cdot \alpha^\circ)$ can be viewed as a nondeterministic mapping that takes a tree and returns any element that occurs as a label at the root of the tree. Hence the above formula states that to find an element in a tree, one nondeterministically selects an arbitrary subtree, and then returns a label at the root of the selected subtree. A formal proof can be found in Hoogendijk's thesis, who in fact proves similar results for a wider class than unary Kleene functors (Hoogendijk, 1996).

We conclude our discussion of related work with a remark about Jay (1995b). Jay convincingly argues that an appropriate notion of *data type* for programming languages is that of a *shapely functor.* A functor F is shapely (over lists) if it preserves pullbacks, and if there exists a natural transformation $l : \mathit{list} \leftarrow F$, satisfying some further conditions. A shapely functor of *Fun* is a relator (all functors of *Fun* that preserve pullbacks are). Furthermore, such a functor F has a membership relation, namely $\delta \cdot l$, where δ is the membership relation of the list functor. To see this, note that for any R,

$$(\delta \cdot l)\backslash R \;=\; l^{\circ} \cdot (\delta\backslash R) \;=\; l^{\circ} \cdot \delta\backslash id \cdot F\,R \;=\; (\delta \cdot l)\backslash id \cdot F\,R.$$

In the first and last step, we used the fact that l is a function.

8 Conclusions

To sum up, we propose the following

Definition
A *container type* is a relator that has membership.

We have argued the validity of this definition by deriving a large number of mathematical properties of container types. One might object that many of these properties (perhaps all) seem to be shared by all data types in the categories *Fun* and *Rel*, not just those that contain data. This seeming generality is however an artefact of our exposition in terms of these two specific categories. In certain other categories (topoi), the exponential functor is a relator only if the internal axiom of choice is satisfied. This indicates that our definition of relator may need modification for those who accept only constructivist reasoning about their specifications. A weaker definition of relators may be found in Mitchell & Ščedrov (1993). It is mainly because of this problem with the exponentional functor that our definition works for container types only, and not for arbitrary data types.

Another shortcoming of this paper is that we have dealt only with data types that have a single kind of element: to deal with more general data types one needs to consider functors between powers of $\mathscr{C}$, rather than just endofunctors of $\mathscr{C}$ (which is what we have done here). The details of such a treatment are, however, rather technical (Hoogendijk, 1996).

It is of course quite likely that we have missed out a number of operations that are common to all container types; it remains to be seen whether these can be coded in terms of membership. The results on fans and strength give us some confidence that this is indeed the case. We only became aware of Manes (1998) when making a final revision to this paper, and connections between the work of Manes and ours remain to be explored.

Acknowledgements

We would like to thank Peter Freyd for supplying the example of a functor of *Fun* that does not have membership, and for general discussion and encouragement. Roland Backhouse suggested that we should take a closer look at the notion of fans, and he invited Oege de Moor for one month to Eindhoven to get the collaboration started. Oege de Moor would also like to thank Fujitsu for supporting this research during his stay at Tokyo University in 1993/94. This paper was revised while Oege de Moor was on sabbatical leave at Microsoft Research, Cambridge. The five anonymous referees went over and above the call of duty, providing extremely helpful suggestions for the revision of this paper. We are grateful to them all.

References

Aarts, C. J., Backhouse, R. C., Hoogendijk, P. F., Voermans, E. & Van der Woude, J. C. S. P. (1992) *A relational theory of datatypes*. Available from URL `http://www.win.tue.nl/win/cs/wp/papers/papers.html`.

Backhouse, R. C., De Bruin, P., Malcolm, G., Voermans, T. S. & Van der Woude, J. C. S. P. (1991) Relational catamorphisms. In: Möller, B. (ed.), *Constructing Programs from Specifications*, pp. 287–318. Elsevier.

Backhouse, R. C., Doornbos, H. & Hoogendijk, P. (1993) *A class of commuting relators*. Available from URL `http://www.win.tue.nl/win/cs/wp/papers/papers.html`.

Bellé, G., Jay, C. B. & Moggi, E. (1996) Functorial ML. *Proceedings of PLILP '96: Lecture Notes in Computer Science 1140*, pp. 32–46. Springer-Verlag.

Bird, R. S. & De Moor, O. (1996) *Algebra of Programming*. International Series in Computer Science. Prentice Hall.

Bird, R. S., Hoogendijk, P. F. & De Moor, O. (1996) Generic programming with relations and functors. *J. Functional Programming*, **6**(1), 1–28.

Brinkmann, H. B. (1969) Relations for exact categories. *J. Algebra*, **13**, 465–480.

Carboni, A. & Street, R. (1986) Order ideals in categories. *Pacific J Math.*, **124**(2), 275–288.

Carboni, A. & Walters, R. F. C. (1987) Cartesian bicategories I. *J. Pure & Appl. Algebra*, **49**(1–2), 11–32.

Carboni, A., Kasangian, S. & Street, R. (1984) Bicategories of spans and relations. *J. Pure & Appl. Algebra*, **33**(3), 259–267.

Carboni, A., Kelly, G. M. & Wood, R. J. (1991) A 2-categorical approach to geometric morphisms I. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, **32**(1), 47–95.

Cockett, J. R. B. & Fukushima, T. (1991) *About Charity*. Technical Report 92/480/18, Department of Computer Science, University of Calgary, Canada. (Available from URL `http://www.cpsc.ucalgary.ca/projects/charity/home.html`.)

Cockett, J. R. B. & Spencer, D. (1992) Strong categorical datatypes I. In: Seely, R. A. G. (ed.), *Category Theory 1991: CMS Conference Proceedings*, **13**, 141–169. Canadian Mathematical Society.

De Morgan, A. (1860) On the syllogism, no. IV, and on the logic of relations. *Trans. Cambridge Philosophical Soc.*, **10**, 331–358.

De Morgan, A. (1966) *"On the Syllogism" and other Logical Writings*. Yale University Press.

Doornbos, H. (1996) *Reductivity Arguments and Program Construction*. PhD thesis, Department of Computing Science, Eindhoven University of Technology, The Netherlands.

Freyd, P. J. & Ščedrov, A. (1990) *Categories, Allegories*. Mathematical Library, vol. 39. North-Holland.

Gierz, G., Hofmann, K. H., Keimel, K., Lawson, J. D., Mislove, M. & Scott, D. S. (1980) *A Compendium of Continuous Lattices*. Springer-Verlag.

Grillet, P. A. (1970) Regular categories. In: Barr, M., Grillet, P. A. & Van Osdol, D. H. (eds.), *Exact Categories and Categories of Sheaves: Lecture Notes in Mathematics 236*, pp. 121–222. Springer-Verlag.

Hinze, R. (1999) *Polytypic programming with ease*. Technical Report IAI-TR-99-2, Institut für Informatik III, Universität Bonn.

Hoare, C. A. R. & He, J. (1986a) The weakest prespecification, I. *Fundamenta Informaticae*, **9**(1), 51–84.

Hoare, C. A. R. & He, J. (1986b) The weakest prespecification, II. *Fundamenta Informaticae*, **9**(2), 217–251.

Hoogendijk, P. F. (1996) *A generic theory of datatypes*. PhD thesis, Department of Computing Science, Eindhoven University of Technology, The Netherlands.

Hoogendijk, P. F. & Backhouse, R. C. (1997) When do datatypes commute? In: *Category Theory and Computer Science: Lecture Notes in Computer Science 1290*, pp. 242–260. Springer-Verlag.

Jay, C. B. (1994) Matrices, monads and the fast fourier transform. *Proc. Massey Functional Programming Workshop*, pp. 71–80.

Jay, C. B. (1995a) Polynomial polymorphism. In: Kotagiri, R. (ed.), *Proc. 18th Australasian Computer Science Conference*, pp. 237–243. Glenelg, South Australia.

Jay, C. B. (1995b) A semantics for shape. *Science of Computer Programming*, **25**, 251–283.

Jeuring, J. & Jansson, P. (1996) Polytypic programming. In: Launchbury, J., Meijer, E. & Sheard, T. (eds.), *Advanced Functional Programming, Second International School: Lecture Notes in Computer Science 1129*, pp. 68–114. Springer-Verlag.

Jeuring, J. T. (1995) Polytypic pattern matching. In: Peyton-Jones, S. (ed.), *Functional Programming Languages and Computer Architecture*, pp. 238–248. ACM.

Kawahara, Y. (1973a) Notes on the universality of relational functors. *Memoirs of the Faculty of Science, Kyushu University, Series A, Mathematics*, **27**(3), 275–289.

Kawahara, Y. (1973b) Relations in categories with pullbacks. *Memoirs of the Faculty of Science, Kyushu University, Series A, Mathematics*, **27**(1), 149–173.

Kock, A. (1972) Strong functors and monoidal monads. *Archiv für Mathematik*, **23**, 113–120.

Mac Lane, S. (1961) An algebra of additive relations. *Proc. Nat. Academy Sci.*, **47**, 1043–1051.

Maddux, R. D. (1991) The origin of relation algebras in the development and axiomatization of the calculus of relations. *Studia Logica*, **50**(3–4), 421–455.

Manes, E. G. (1998) Implementing collection classes with monads. *Math. Structures in Computer Sci.*, **8**(3), 231–276.

Manes, E. G. & Arbib, M. A. (1986) *Algebraic approaches to program semantics*. Texts and Monographs in Computer Science. Springer-Verlag.

Meertens, L. (1996) Calculate polytypically! *Proc. 8th Int. Symposium on Programming Languages, Implementations, Logics and Programs: Lecture Notes in Computer Science*, pp. 1–16. Springer-Verlag.

Mitchell, J. C. & Ščedrov, A. (1993) Notes on sconing and relators. In: Boerger, E. (ed.), *Computer Science Logic '92, Selected Papers: Lecture Notes in Computer Science 702*, pp. 352–378. Springer-Verlag.

Moggi, E. (1991) Notions of computation and monads. *Inform. & Computation*, **93**(1), 55–92.

Peirce, C. S. (1870) Description of a notation for the logic of relatives, resulting from an amplification of the conceptions of Boole's calculus of logic. *Memoirs of the American Academy of Sciences*, **9**, 317–378.

Peirce, C. S. (1933) *Collected Papers*. Harvard University Press.

Pratt, V. R. (1992) Origins of the calculus of binary relations. *Logic in Computer Science*, pp. 248–254. IEEE Press.

Puppe, D. (1962) Korrespondenzen in Abelschen Kategorien. *Mathematische Annalen*, **148**, 1–30.

Schröder, E. (1895) *Vorlesungen über die Algebra der Logik (exakte logik). Dritter band: Algebra und Logik der Relative*. Teubner, Leipzig. (Reprinted by Chelsea Publishing Co., New York, 1966.)

Tarski, A. (1941) On the calculus of relations. *J. Symbolic Logic*, **6**(3), 73–89.

Tuijnman, D. (1996) *A Categorical Approach to Functional Programming*. PhD thesis, Department of Computer Science, University of Ulm, Germany.