What is a data type?

Paul Hoogendijk
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, NL

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

August 21, 1996

Abstract

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

1 Introduction

What is a data type? It is easy to list a number of examples: pairs, lists, bags, finite sets, possibly infinite sets, function spaces ... but such a list of examples hardly makes a definition. The obvious formalisation is a definition that builds up the class of data types inductively; such an inductive definition, however, leads to cumbersome proofs if we want to prove a property of all data types. Here we aim to give a non-inductive characterisation, defining a data 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 data types. 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 data types. This paper does not go into examples of polytypic program derivation, and the reader is referred to *e.g.* [2, 4, 6, 13, 16] for such examples and a more in-depth motivation of polytypism.

Because our interest is in a definition that is useful in specification, the class of data 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. 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 argue that data types are *functors*. Motivated by a need to deal with nondeterminism, we subsequently note that data 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 data 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 support our claim that data types can be defined as relators that have membership.

2 Specifications and 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 $id : A \leftarrow A$. 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.

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.

All our examples are drawn from *Fun* and *Rel*, but the definitions work in more general categories, including certain models of programming languages. For that reason, our abstract definitions refer to a base category C that is not further specified. We shall impose a number of conditions on C as the need arises.

3 Data types are functors

Throughout, F denotes a type constructor, and for a set A, $F\,A$ is the set of F-structures with elements drawn from A. For example, F could be the list forming operation *list*, where $list\,A$ is the set of lists over A:

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

Any type constructor F comes equipped with a *map* operation that allows one to apply a function to all elements of a data structure; we denote this map operation also by F, and when $h : A \leftarrow B$, we have $F\,h : F\,A \leftarrow F\,B$. Thus F consists in fact of two components: one for taking a type to a new type, and another component that consists of the map operation. In the example of lists, the map operation applies a function to all elements of a list:

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

The mapping F preserves the operations of a category: we have already stipulated that it preserves the typing of arrows. It also preserves identities and composition:

$$F\,id = id \quad \text{and} \quad F(h \cdot k) = F\,h \cdot F\,k.$$

Again illustrating with the example of lists, it is indeed the case that applying the identity function to all elements of a list leaves the list unchanged. Also, first applying k to all elements and then applying h has the same effect as applying $h \cdot k$ in one go. Category theorists summarise these properties by saying that F is a *functor*. We shall define data types as a special kind of functor. Below we consider three more data type constructors, and how they can be viewed as functors.

A rather trivial, but nevertheless important example of a functor is the *identity* functor id: it leaves both objects and arrows unchanged.

The *powerset* constructor takes a set and returns the set of all its subsets:

$$P\,A \quad = \quad \{x \mid x \subseteq A\}.$$

The corresponding action on arrows is the operator that applies a function to all elements of a set:

$$P\,f\,x \quad = \quad \{f\,a \mid a \in x\}.$$

The *exponential* data constructor returns the set of all functions that have a given domain A:

$$\mathit{From}_A B \quad = \quad \{f \,|\, f : B \leftarrow A\}.$$

Here the map operation may be slightly less obvious. The typing rule says that if $h : B \leftarrow C$, then $\mathit{From}_A h : \mathit{From}_A B \leftarrow \mathit{From}_A C$. From these type considerations, it becomes clear that the only possible definition is

$$\mathit{From}_A h \quad = \quad \lambda f : h \cdot f.$$

4 Data types are relators

It is worth emphasising once more that our purpose is a calculus for program synthesis where specifications and programs are treated on the same footing. In particular, we wish to reason about nondeterministic specifications where a program is specified as a relation between input and output. We already noted that sets and relations form a category. Of course there are many more operations possible on relations besides sequential composition. For instance, one can take the intersection $R \cap S$ of two relations R and S, and by flipping all the pairs in a relation $R : A \leftarrow B$ one obtains its converse $R^\circ : B \leftarrow A$. Functions can actually be defined as those relations f that satisfy the inequations

$$f \cdot f^\circ \subseteq id \quad \text{and} \quad f^\circ \cdot f \supseteq id.$$

More generally, for a so-called *regular* category C, it is possible to construct the category of relations $\mathit{Rel}(C)$, so that $\mathit{Rel}(C)$ satisfies all the usual axioms of relation algebra, and in particular, the arrows of C occur as *functions* in $\mathit{Rel}(C)$ [7]. It is our view that any model of a deterministic programming language ought to be embedded in a regular category (or in the category of partial functions over a regular category), so that one has a corresponding notion of nondeterministic specification.

Having decided to work with relations over our base category C, the next question to ask is how data types carry over from C to $\mathit{Rel}(C)$. It stands to reason that we consider them as functors of $\mathit{Rel}(C)$. A minimum healthiness condition on such functors is that they are monotonic with respect to inclusion of relations: we shall call monotonic functors of $\mathit{Rel}(C)$ *relators*. Relators are a special kind of functor on functions:

Fact 1 *([14]) Let F be an functor of a regular category C. There exists at most one relator of $\mathit{Rel}(C)$ that agrees with F on functions.*

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

$$
\begin{aligned}
& [a_1, a_2, \ldots, a_n](\mathit{list}\, R)[b_1, b_2, \ldots, b_n] \\
\equiv & \\
& n = m \wedge (\forall i : 0 \leq i \leq n : a_i R b_i).
\end{aligned}
$$

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

$$x(PR)y \quad \equiv \quad (\forall a \in x : (\exists b \in y : aRb)) \ \wedge \ (\forall b \in y : (\exists a \in x : aRb)).$$

This example 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.

There exist functors of *Fun* that do not have a generalisation to *Rel*; we claim that such functors cannot be considered data types. On the other hand, as we shall see below, there are also relators that do not have some essential characteristics of data types, so it is not correct to define data types as relators. Before continuing our refinement of the notion of data types, however, it will be expedient to introduce some more results about relators. First we note that any relator preserves converse:

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

When discussing data types, we need ways of systematically transforming one data type into another. Formally, this is captured by the notion of a *natural transformation*. Let F and G be functors of C. A natural transformation of type $F \leftarrow G$ is a collection of arrows $\phi : F\,A \leftarrow G\,A$ (one for each object A) so that

$$F\,h \cdot \phi \quad = \quad \phi \cdot G\,h,$$

for all arrows h of C. For example, consider the operation *setify* that turns a list of elements into the corresponding set. This is a natural transformation of type $P \leftarrow \mathit{list}$. More generally, all polymorphic operations are natural transformations. Strictly speaking we ought to index our natural transformations by objects of C, and write $\phi_A : F\,A \leftarrow G\,A$, but we shall not do so.

Note 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

$$PR \cdot setify \;\; \supseteq \;\; setify \cdot list\, R.$$

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

$$FR \cdot \phi \;\; \supseteq \;\; \phi \cdot GR,$$

for all R. In fact, writing J for the inclusion of C into $Rel(C)$, we have

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

Proofs of all facts cited in this section can be found in [3].

5 Data types have membership

The purpose of any data structure is to record the presence of elements. One might expect, therefore, that any data type F comes equipped with a collection of membership relations $\delta : A \leftarrow FA$, such that $a\delta x$ holds precisely when a is an element of data structure x. Because membership is polymorphic, it is a lax natural transformation of type $id \hookleftarrow F$. In a first attempt, we defined membership as the largest lax natural transformation of this type. Unfortunately, it turned out to be hard to prove interesting properties from that definition, and because a largest lax natural transformation exists for any relator, it does not further refine our definition of data types.

An alternative approach (which does not suffer these drawbacks) makes use of the *division* operator in relation algebra, also known as the *weakest prespecification* [9, 10]. It is defined by the equivalence

$$X \subseteq R\backslash S \;\; \equiv \;\; R \cdot X \subseteq S \quad \text{for all } X.$$

As a predicate, $R\backslash S$ can be written

$$x(R\backslash S)y \;\; \equiv \;\; (\forall a : aRx : aSy).$$

The weakest prespecification operator does not exist in every category of relations $Rel(C)$, but it does whenever the base category C is a so-called *logos* [7]. Armed with the weakest prespecification, we are in a position to give the official definition of membership.

A collection of arrows $\delta : A \leftarrow FA$ is a *membership relation* of F if for each R we have

$$\delta\backslash R \;\; \equiv \;\; FR \cdot \delta\backslash id.$$

In terms of predicates, this equation can be interpreted as follows: for all b and x

$$(\forall a : a\delta x : aRb) \equiv (\exists y : x(FR)y : (\forall b' : b'\delta y : b' = b))$$

Admittedly, this definition is highly non-obvious and, to add insult to injury, in certain categories the identity relator has multiple membership relations. To remedy that second problem, 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 partial functions p of type $A \leftarrow 1$.) An immediate consequence of identification is uniqueness of membership:

Fact 4 *Suppose that δ is a membership relation of F. Then δ is a lax natural transformation of type $id \hookleftarrow F$.*

The membership relation on lists is given by

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

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

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

There exist relators that do not have membership, and an example is detailed in subsection 5.1 below. Our definition of a data type has now been refined to the requirement that it is a relator that has membership.

The existence of membership gives a way of constructing largest lax natural transformations of any type:

Fact 5 *Let F and F' be relators with membership relations δ and δ' respectively. Then the largest lax natural transformation of type $F \hookleftarrow F'$ is $\delta\backslash\delta'$.*

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 $\phi \subseteq \delta\backslash\delta'$, with ϕ not a lax natural transformation.

In the subsection below proofs of the above two 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.

5.1 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 1 *A collection of arrows δ is a membership relation of F iff for all R and S we have*

$$\delta\backslash(R \cdot S) \quad = \quad F\,R \cdot \delta\backslash S.$$

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

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

□

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:

Lemma 2 *If δ is a membership relation of F, then $\delta : id \hookleftarrow F$.*

Proof.

$$\begin{array}{cl} & \delta \cdot F\,R \subseteq R \cdot \delta \\ \equiv & \quad \{\text{division}\} \\ & F\,R \subseteq \delta\backslash(R \cdot \delta) \\ \equiv & \quad \{\delta \text{ membership, Lemma 1}\} \\ & F\,R \subseteq F\,R \cdot \delta\backslash\delta \\ \Leftarrow & \quad \{\text{since } id \subseteq \delta\backslash\delta\} \\ & true \end{array}$$

□

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

Lemma 3 *Suppose that δ is a membership relation of F. Then $\delta\backslash id$ is a lax natural transformation of type $F \hookleftarrow id$.*

Proof.

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

□

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 'data type' to lead to intricate and cumbersome proofs.

Fact 6 *If δ is a membership relation of F, then δ is the largest lax natural transformation of type $id \hookleftarrow F$.*

Proof. Let δ be a membership relation of F. By Lemma 2, $\delta : id \hookleftarrow F$. Let ϕ be another lax natural transformation of type $id \hookleftarrow F$. Then

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

The claim is that $\phi \cdot \delta\backslash id \subseteq id$. 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 $\phi \cdot \delta\backslash id : id \hookleftarrow id$ because $\phi : id \hookleftarrow F$, and Lemma 3 says that $\delta : F \hookleftarrow id$.

□

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.

Fact 7 *Let F and F' be relators with membership relations δ and δ' respectively. Then $\delta\backslash\delta'$ is the largest lax natural transformation of type $F \hookleftarrow F'$.*

Proof. First note that $\delta\backslash\delta' : F \hookleftarrow F'$, for it is the composition of two lax natural transformations (Lemmas 2 and 3):

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

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

$$\begin{array}{ll} & \phi \subseteq \delta\backslash\delta' \\ \equiv & \quad \{\text{division}\} \\ & \delta \cdot \phi \subseteq \delta' \\ \Leftarrow & \quad \{\text{Lemma 6}\} \\ & \delta \cdot \phi : id \hookleftarrow F' \\ \equiv & \quad \{\text{since } \delta : id \hookleftarrow F \text{ and } \phi : F \hookleftarrow F'\} \\ & true \end{array}$$

□

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 of 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.

As a first step towards constructing such an example, observe that a data 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. A *subset* of an object A is a relation C that is contained in the identity $id : A \leftarrow A$. Such subsets are in one-to-one correspondence with arrows $R : A \leftarrow \mathbf{1}$; and these arrows R are called *conditions* (here $\mathbf{1}$ stands for some fixed one-element set called the *terminal object*). The bijection between subsets and conditions is given by the two mappings

$$Ran\, R = id \cap R \cdot R^{\circ} \quad \text{and} \quad Cond\, C = C \cdot \Pi,$$

where Π is the largest arrow of type $A \leftarrow \mathbf{1}$. Furthermore this bijection respects the ordering on subsets and conditions: we shall refer to these facts by the hint 'order-iso'. It is easily seen that any relator F preserves subsets (because relators are monotonic and preserve identities), but few relators preserve conditions.

Lemma 4 *Let F be a relator that has membership. Then F preserves arbitrary intersections of subsets.*

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 [8]. Below it is shown that F is a right adjoint: let C and D be subsets of an object A. We have

$$\begin{array}{ll} & D \subseteq FC \\ \equiv & \quad \{\text{order-iso}\} \\ & D \cdot \Pi \subseteq F\,C \cdot \Pi \\ \equiv & \quad \{\text{division: } \delta\backslash\Pi = \Pi\} \\ & D \cdot \Pi \subseteq F\,C \cdot \delta\backslash\Pi \\ \equiv & \quad \{\text{membership}\} \\ & D \cdot \Pi \subseteq \delta\backslash(C \cdot \Pi) \\ \equiv & \quad \{\text{division}\} \\ & \delta \cdot D \cdot \Pi \subseteq C \cdot \Pi \\ \equiv & \quad \{\text{range: } R \cdot \Pi = Ran(R) \cdot \Pi\text{, order-iso}\} \\ & Ran(\delta \cdot D) \subseteq C \end{array}$$

$\square$

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 subsets of A such that F does not preserve the intersection of X. Since any relator preserves finite intersections of subsets, 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 data 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 jargon has the advantage that it does not mention relations. Although we were able to phrase our requirements in this learned manner, 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 $G = From_N$, 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

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

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

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

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

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

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 $\bigcap\{FC \mid C \in X\}$ 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 data types.

6 Data 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 $\lambda R : (F\,R \cdot \phi)$ preserves finite intersections ($\Pi : A \leftarrow B$ is the largest relation of its type):

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

This definition of fans originated with [2], where they were called *generators*. Remarkably, membership guarantees the existence of unique fans:

Fact 8 *If F has membership, then $\phi = \delta \backslash id$ is the unique fan of F.*

The converse is not true: as we shall see later, 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 data 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.

6.1 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 $\delta \backslash id$ is the largest fan of F. Together these two lemmas give the desired result, namely that $\delta \backslash id$ is the unique fan of 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 5 *Let ϕ be a fan of F. Then for all R, S and T, we have*

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

Proof. In the proof below, we shall use the fact that for any relation R, there exist functions f and g so that $R = f \cdot g^\circ$.

Furthermore, we shall need the *range* operator, defined by

$$Ran\,X \quad = \quad id \cap X \cdot X^\circ .$$

It is a fact, which we shall not prove, that relators preserve range: $F(Ran\,X) = Ran(F\,X)$. Furthermore, the range of an intersection is given by

$$Ran(P \cap Q) \quad = \quad id \cap P \cdot Q^\circ .$$

Finally, we have $Ran(X \cdot Y) \subseteq Ran\,X$.

A consequence of the so-called *modular law* of relation algebra (also known as Dedekind's rule) is the *modular identity*

$$(h \cdot X \cdot k^\circ) \cap Y \quad = \quad h \cdot (X \cap (h^\circ \cdot Y \cdot k)) \cdot k^\circ .$$

With the above facts in hand, we calculate as follows:

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

$\square$

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

Lemma 6 *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

$$\begin{array}{cl}
 & \mu \\
= & \quad \{\text{intersection}\} \\
 & \mu \cap \Pi \\
= & \quad \{\text{since } \phi \text{ is a fan}\} \\
 & \mu \cap (F\Pi \cdot \phi) \\
\subseteq & \quad \{\text{modular law (also known as Dedekind's rule)}\} \\
 & (\mu \cdot \phi^\circ \ \cap \ F\Pi) \cdot \phi \\
\subseteq & \quad \{\text{given: } \phi \subseteq \mu\} \\
 & (\mu \cdot \mu^\circ \ \cap \ F\Pi) \cdot \phi \\
\subseteq & \quad \{\text{Lemma 5}\} \\
 & \phi
\end{array}$$

□

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 $\delta\backslash id$. Given the importance of this relation in the definition of membership, it should be no surprise that $\delta\backslash id$ is of independent interest.

Fact 9 *Let F be a relator whose membership relation is δ. Then $\phi = \delta\backslash id$ is a fan of F.*

The proof of this fact is trivial, for $\lambda X : Y\backslash X$ preserves intersections for all Y, in particular when $Y = \delta$. Finally, we can put all the above results together to obtain that a relator with membership has precisely one fan. That begs the question whether fans and membership are perhaps in some sense equivalent, but as we shall see in the next section, this is not the case.

Fact 10 *Let F be a relator that has membership. Then $\phi = \delta\backslash id$ is the unique fan of F.*

Proof. We have just proved that ϕ is indeed a fan. Furthermore, by Lemma 7, it is the largest lax natural transformation of its type. Therefore any other fan is included in it, and Lemma 6 gives the desired result.

□

7 Data types have strength

Above we already alluded to work of others which also (at least implicitly) aimed to pin down the notion of 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* [18, 5]. 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 data type, namely a relator that has membership, refines earlier attempts in the literature.

To present the definition of strength, we need the notion of *finite products*. We shall denote the terminal object (or empty product) by $\mathbf{1}$ and the binary product of A and B by $(outl : A \leftarrow A \times B, outr : A \leftarrow A \times B)$. Readers who are not familiar with the categorical vernacular can think of $\mathbf{1}$ as a one-element set (the empty product), and of $A \times B$ as the cartesian product of A and B. Note that we have isomorphisms

$$rid : A \leftarrow A \times \mathbf{1} \quad \text{and} \quad assl : (A \times B) \times C \leftarrow A \times (B \times C).$$

Both of these isomorphisms will play an important role below.

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

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

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

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

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

$$\theta(x, b) \;\; = \;\; [(a, b) \,|\, a \leftarrow x].$$

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 $c : F\,A \leftarrow F\,\mathbf{1} \times A$, defined by

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

where $lid : A \leftarrow (\mathbf{1} \times A)$ is the obvious isomorphism. The intention of this definition is that c takes a shape (a value of type $F\mathbf{1}$) 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 [12].

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 influential paper [15]. 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

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

Note that the inclusion $\supseteq$ is 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 11 *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 12 *Let F be a relator that has membership. Then F has a unique strength, and that strength is relational.*

We have not found a comparable result about uniqueness of strengths in the literature. It is unlikely that any interesting improvements can be made without assuming the identification axiom: 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 existence of membership is also necessary, 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 13 *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 is particularly relevant 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 data types. As in the case of fans, strength does not require further refinement of our definition of data 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 data 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.

7.1 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. Below we state this alternative definition; it makes use of the fact that for each object A, there is precisely one function $! : \mathbf{1} \leftarrow A$ (pronounced 'pling'), and furthermore that $!$ is the largest relation of type $\mathbf{1} \leftarrow A$. One may 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 [1]. To get some practice in pushing all these new operators around, and for future reference, we first prove two little lemmas before going on to the new, alternative definition of fans.

Lemma 7 *For all R and S, we have*

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

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

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

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,

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

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

□

Note the slightly curious structure of the above proof, where $F(R \times S)$ is first decomposed into $F(R \times id)$ and $F(id \times S)$. 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 8 *For all 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.

$$\begin{array}{ll} & \langle F\,outl, F\,outr\rangle \cdot F\langle R, S\rangle \\ = & \quad \{\text{product absorption: } \langle R, S\rangle = (R \times S) \cdot \langle id, id\rangle,\ F \text{ relator}\} \\ & \langle F\,outl, F\,outr\rangle \cdot F(R \times S) \cdot F\langle id, id\rangle \\ = & \quad \{\text{Lemma 7}\} \\ & (F\,R \times F\,S) \cdot \langle F\,outl, F\,outr\rangle \cdot F\langle id, id\rangle \\ = & \quad \{F \text{ relator, } outl \cdot \langle id, id\rangle = id = outr \cdot \langle id, id\rangle\} \\ & (F\,R \times F\,S) \cdot \langle id, id\rangle \\ = & \quad \{\text{product absorption}\} \\ & \langle F\,R, F\,S\rangle \end{array}$$

□

We are now ready to present the alternative definition of fans. Recall the original definition: a lax natural transformation $\phi : F \hookleftarrow id$ is a *fan* if the mapping $\lambda R : F\,R \cdot \phi$ preserves finite intersections. We have just explained the intimate connection between intersections and products, and the following fact will come as no surprise:

Fact 14 *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 of F iff*

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

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

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

For the second equation, we reason:

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

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:

$$
\begin{array}{ll}
& F\Pi \cdot \phi \\
= & \quad \{\text{since } \Pi = !^\circ \cdot !,\ F \text{ relator}\}
\end{array}
$$

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

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

$$
\begin{array}{ll}
& F(R \cap S) \cdot \phi = F\,R \cdot \phi \cap F\,S \cdot \phi \\
\equiv & \quad \{\text{since } R \cdot U \cap S \cdot V = \langle R^\circ, S^\circ\rangle^\circ \cdot \langle U, V\rangle, \ F \text{ relator}\} \\
& F\langle R^\circ, S^\circ\rangle^\circ \cdot F\langle id, id\rangle \cdot \phi = \langle F\,R^\circ, F\,S^\circ\rangle^\circ \cdot \langle \phi, \phi\rangle \\
\equiv & \quad \{\phi : F \hookleftarrow id, \text{ and } \langle id, id\rangle \text{ function, Lemma 8}\} \\
& F\langle R^\circ, S^\circ\rangle^\circ \cdot \phi \cdot \langle id, id\rangle = F\langle R^\circ, S^\circ\rangle^\circ \cdot z^\circ \cdot \langle \phi, \phi\rangle \\
\Leftarrow & \quad \{\text{since } \langle \phi, \phi\rangle = (\phi \times \phi) \cdot \langle id, id\rangle\} \\
& \phi = z^\circ \cdot \phi \times \phi \\
\equiv & \quad \{\phi = \mu^\circ, \text{ converse}\} \\
& \mu = \mu \times \mu \cdot z
\end{array}
$$

□

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 15 *Let F be a relator. Assume that ϕ is a fan of 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 7, and

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

The two equations

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

follow from Lemma 14; 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,

$$\begin{array}{cl}
 & \theta \cdot \theta^\circ \\
= & \quad \{\text{definitions of } \theta, \text{ split and intersection}\} \\
 & F(outl^\circ \cdot outl) \cap F\,outr^\circ \cdot \phi \cdot \phi^\circ \cdot F\,outr \\
\subseteq & \quad \{\text{Lemma 5}\} \\
 & F(outl^\circ \cdot outl \cap outr^\circ \cdot outr) \\
= & \quad \{\text{since } outl^\circ \cdot outl \cap outr^\circ \cdot outr = id\} \\
 & id
\end{array}$$

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

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

□

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:

Fact 16 *Let F be a relator. Then the relational strengths of F are in one-to-one correspondence with the fans of F.*

Together with an earlier Lemma, 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 δ. Call

$$\begin{array}{l} \quad \langle F\,outl, (\delta \backslash id)^\circ \cdot F\,outr\rangle^\circ \\ = \\ \quad F\,outl^\circ \cdot outl \;\cap\; \delta\backslash(outr^\circ \cdot outr). \end{array}$$

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. Clearly, to say that $\theta \subseteq \Theta$ is to say that

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

We shall prove these two equations as separate lemmas.

Lemma 9 *Let F be a functor with strength θ. Then $F\,outl \cdot \theta = outl$.*

Proof.

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

□

Lemma 10 *Let F be a relator with membership relation δ, and let θ be a strength of F. Then $outr \cdot \delta \cdot \theta \subseteq outr$.*

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 $outr \cdot \delta \cdot \theta$ 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:

Fact 17 *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

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

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

Lemma 11 *Let F and G be relators that have membership, and let α be a lax natural transformation of type $F \hookleftarrow G$. Then*

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

Proof.

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

□

Lemma 12 *Let F and G be relators that have membership, and let α be a lax natural transformation of type $F \hookleftarrow G$. Then*

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

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

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

or, equivalently,

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

This inequation can be proved as follows:

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

□

We can now conclude

Fact 18 *Let F and G be relators that have membership. Then any lax natural transformation $\alpha : G \hookleftarrow F$ is strong.*

As said before, 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 functor that has membership) is strong. Tuijnman [19] 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 in any detail.

8 Conclusions

To sum up, we propose the following

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

The validity of this definition hinges on the assumption that any calculus of specifications and programs can be embedded in a logos that satisfies the identification axiom. That it is a logos means that we can freely use first-order predicate calculus to reason about specifications; that it satisfies the identification axiom means that polymorphism is well-behaved (does a decent model of polymorphic λ-calculus allow total functions of type $\forall a : a \leftarrow a$ apart from the identity function?). We feel, therefore, that these conditions on the category are healthiness conditions on the semantic model of the calculus.

We note that synthetic domain theory is precisely concerned with the construction of models of programming languages that satisfy our assumptions [11]. In this context we should own up to an embarassment, namely that in certain 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 [17].

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 C, rather than just endofunctors of C (which is what we have done here). The details of such a treatment are however rather technical.

It is of course quite likely that we have missed out a number of operations that are common to all data 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.

An important task, which is currently under investigation at Chalmers, Eindhoven, Oxford, Sydney and Utrecht, is to identify large classes of data types that share common structure. For instance, many data types that occur in executable code (with the exception of exponentials) allow a 'flattening' operation that turns a data structure into a list. Presently that flattening operation (and others like it) are defined by induction over the construction of a data type. We hope to have argued convincingly in this paper that such inductive constructions can and should be avoided.

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.

References

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

[2] R. C. Backhouse, H. Doornbos, and P. Hoogendijk. A class of commuting relators. Available from URL `http://www.win.tue.nl/win/cs/wp/papers/papers.html`.

[3] R. S. Bird and O. De Moor. *Algebra of Programming*. International Series in Computer Science. Prentice Hall, 1996.

[4] R. S. Bird, P. F. Hoogendijk, and O. De Moor. Generic programming with relations and functors. *Journal of Functional Programming*, 6(1):1–28, 1996.

[5] J. R. B. Cockett and D. Spencer. Strong categorical datatypes I. In R. A. G. Seely, editor, *Category Theory 1991*, volume 13 of *CMS Conference Proceedings*, pages 141–169. Canadian Mathematical Society, 1992.

[6] H. Doornbos. *Reductivity Arguments and Program Construction*. Ph.D. thesis, Department of Computing Science, Eindhoven University of Technology, The Netherlands, 1996.

[7] P. J. Freyd and A. Ščedrov. *Categories, Allegories*, volume 39 of *Mathematical Library*. North-Holland, 1990.

[8] G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott. *A Compendium of Continuous Lattices*. Springer-Verlag, 1980.

[9] C. A. R. Hoare and J. He. The weakest prespecification, I. *Fundamenta Informaticae*, 9(1):51–84, 1986.

[10] C. A. R. Hoare and J. He. The weakest prespecification, II. *Fundamenta Informaticae*, 9(2):217–251, 1986.

[11] J. M. E. Hyland. First steps in synthetic domain theory. In *Category Theory '90*, volume 1144 of *Lecture Notes in Mathematics*. Springer-Verlag, 1992.

[12] C. B. Jay. Matrices, monads and the fast fourier transform. In *Proceedings of the Massey Functional Programming Workshop 1994*, pages 71–80, 1994.

[13] J. T. Jeuring. Polytypic pattern matching. In S. Peyton-Jones, editor, *Functional Programming Languages and Computer Architecture*, pages 238–248. Association for Computing Machinery, 1995.

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

[15] A. Kock. Strong functors and monoidal monads. *Archiv für Mathematik*, 23:113–120, 1972.

[16] L. Meertens. Calculate polytypically. In *Proceedings of the Eighth International Symposium on Programming Languages, Implementations, Logics and Programs*, Lecture Notes in Computer Science. Springer-Verlag, 1996.

[17] J. C. Mitchell and A. Ščedrov. Notes on sconing and relators. In E. Boerger, editor, *Computer Science Logic '92, Selected Papers*, volume 702 of *Lecture Notes in Computer Science*, pages 352–378, 1993.

[18] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

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