

Nondeterministic Folds

C. E. Martin and S. A. Curtis

Oxford Brookes University, UK

Abstract. The *map* and *fold* operators are both key elements of every functional programmer’s toolkit. In this paper we examine the corresponding concepts in the domain of multirelations, which can be used to model both angelic and demonic nondeterminism.

1 Introduction

The *map* and *fold* operators of functional programming are both standard tools for capturing common patterns of recursion among list processing functions, and their relational equivalents are now widely used for the same purpose in specifications. A less well-known fact is that both these operators also have predicate transformer equivalents which have not previously been used for program derivation. One reason for this lack of attention is that the operators do not take on a very familiar form in this context. Another reason is that the class of problems that require the expressive power of predicate transformers, as opposed to relations, is quite restricted. A third factor is that, apart from a few exceptions such as [W94], predicate transformers have traditionally been associated with the derivation of imperative, rather than functional programs. The first contribution of this paper is to show how, when viewed in the equivalent model of multirelations, the *map* and *fold* operators do look familiar, and this is illustrated by a number of concrete examples. The second contribution is to show how the laws associated with such operators can be used to transform specifications at this level.

Multirelations were introduced in [Rew03] as an alternative to predicate transformers for reasoning about specifications that contain both angelic and demonic nondeterminism. Angelic nondeterminism occurs when the choice is made by an ‘angel’: it is assumed that the angel will choose the best possible outcome. Demonic nondeterminism occurs when the choice is made by a ‘demon’: no assumption can be made about the choice made by the demon, so we must be prepared for the worst possible outcome. Ordinary relations can only be used to describe one of these kinds of nondeterminism at a time, but several convincing examples in [MCR04] show how well-suited multirelations are for expressing both. In our view, the primary advantage of multirelations over predicate transformers is that they model programs forwards rather than backwards. So, instead of mapping postconditions to preconditions, they can be used to relate inputs, or initial states, to outputs, or final states.

The close relationship between multirelations and predicate transformers can be expressed as an isomorphism of categories. This is useful for translating some

well-established properties of the category of predicate transformers to multirelations. In particular, we will show that functors like *map* have a unique extension from relations to multirelations, as do initial algebras. It is this property of initial algebras that yields the definition of *foldr* on multirelations, together with its associated fusion law. The datatype of lists will be used to illustrate these concepts, but the principle can be applied to any inductive datatype.

The structure of the paper is as follows. Section 2 contains an introduction to multirelations, summarizing the main concepts and definitions from [MCR04]. Section 3 introduces some operations on multirelations, including *map*, *foldr* and *foldl*, and includes several illustrative examples. Section 4 describes one application in depth: that of a voting system, Section 5 gives the theory that underpins the definitions of Section 3, and Section 6 is the conclusion.

2 Multirelations

Multirelations are to relations what multifunctions (relations) are to functions: that is to say, they are relations whose target type is a powerset type. A multirelation represents a specification if and only if it is up-closed:

Definition 2.1 (up-closed multirelation). *An up-closed multirelation M with source A and target B is a subset of the cartesian product $A \times \mathbb{P} B$ such that for all $x \in A$ and $X, Y \subseteq B$,*

$$x M X \wedge X \subseteq Y \Rightarrow x M Y$$

The types of all relations and up-closed multirelations with source A and target B are denoted by $A \leftrightarrow B$ and $A \rightrightarrows B$ respectively. Note that we will abbreviate “up-closed multirelation” to “multirelation”.

Multirelations model program behaviour in a different way from ordinary relations. A relation R relates two values x and y , written $x R y$, if and only if the corresponding program can terminate with output y given input x . However, a multirelation M relates two values x and *post*, written $x M \textit{post}$, if and only if the corresponding program can terminate with an output value that satisfies the predicate *post* given input x . In addition, if *post'* is another postcondition such that $\textit{post} \subseteq \textit{post}'$, then clearly any output value that satisfies *post* must also satisfy *post'*, which is reflected by the up-closure property.

The choice between various postconditions is interpreted as angelic: the angel chooses which predicate to guarantee, and the demon chooses how to satisfy that predicate.

Example 2.1. Let *Int* denote the type of all integers, $M : \textit{Int} \rightrightarrows \textit{Int}$, and suppose that for all $x : \textit{Int}$ and $X : \mathbb{P} \textit{Int}$

$$x M X \Leftrightarrow (\{x\} \subseteq X \vee \{x-1, x+1\} \subseteq X)$$

then two of the postconditions that the angel can choose between are $\{x\}$ and $\{x-1, x+1\}$. If he chooses the latter, then the choice of output value is determined by the demon. \square

2.1 Strongest Postconditions

In the above example, the two postconditions $\{x\}$ and $\{x - 1, x + 1\}$ are the strongest ones that the angel can choose between, in the following sense:

Definition 2.2 (strongest postcondition). *Let $M : A \rightrightarrows B$, $x \in A$ and $post \subseteq B$. Then $post$ is a strongest postcondition of M with respect to x if and only if*

1. $x M post$
2. $(\forall post' : x M post' : post' \not\subseteq post)$

In general, we shall refer to the set of all strongest postconditions of a multirelation M with respect to an initial state x by $sp(x, M)$. The following examples illustrate the concept.

Example 2.2. Each type A has identity $\in_A : A \rightrightarrows A$, where \in_A (sometimes denoted by id_A) represents the set membership relation on subsets of A . So, given an input value $x \in A$, the only strongest postcondition it can achieve is $\{x\}$, which means that the value x is output. This specification is guaranteed to establish all postconditions that are satisfied by x itself. \square

Example 2.3. Consider the behaviour of a single reel of a fruit machine, represented by $reel : Fruit \rightrightarrows Fruit$ where $Fruit$ is the set of all the different kinds of fruit that can appear on a reel. The player of the machine (considered here to be the angel) can choose either to hold the current position or let the demon spin the wheel to a random new position. This choice is shown by the following definition of $reel$: for all $x : Fruit, X : \mathbb{P} Fruit$

$$x reel X \Leftrightarrow x \in X \vee X = Fruit$$

This definition is misleading, since for each input value x , there is only one strongest postcondition $\{x\}$, so $reel$ is really just the identity. This apparent paradox illustrates the fact that multirelations cannot model all the nuances of a system's behaviour: they only capture the absolute guarantees. In this case, the angel has no guarantee unless he holds the reel. It is also clear from this example that one multirelation may model several different kinds of behaviour. \square

Example 2.4. For each pair of types A, B , and each value $b \in B$, the constant specification $const\ b : A \rightrightarrows B$ is defined by

$$const\ b = A \times \uparrow \{b\}$$

Here, the strongest postcondition that is satisfied for any input value is $\{b\}$, ensuring that the value b is output. \square

Example 2.5. Let $M : Int \rightrightarrows Int$ be defined for all $x : Int$, $X : \mathbb{P} Int$, by

$$x M X \Leftrightarrow (\exists y : x < y : up\ y \subseteq X)$$

where $up\ y = \{n : Int \mid y \leq n\}$. This multirelation can model a game where the angel chooses a number greater than the input value x and the demon then outputs any number that is at least as large as the one chosen by the angel. The choices offered to the angel are among an infinite sequence $up(x+1)$, $up(x+2)$, \dots of postconditions, each stronger than its predecessor. This multirelation has no strongest postconditions because it is always possible to find a postcondition that is stronger than any other given postcondition. \square

The final example above demonstrates one of the dangers of relying solely on strongest postconditions for intuition about arbitrary multirelations, but there is a large class for which it is safe to do so, as characterised by the following definition:

Definition 2.3. *Let $M : A \rightrightarrows B$. Then M is representable if, for all $x : A$, $X : \mathbb{P} B$*

$$x M X \Leftrightarrow (\exists Y : Y \in sp(x, M) : Y \subseteq X)$$

All of the examples above, apart from the last, are representable in this sense.

3 Operations

We will now introduce some operations on multirelations, some of which will be familiar to readers of [MCR04], and the rest of which are new.

3.1 Composition

Multirelations cannot be composed using ordinary relational composition for obvious type reasons. Instead, composition is defined as follows:

Definition 3.1 (composition). *The composition of two multirelations $M : A \rightrightarrows B$, $N : B \rightrightarrows C$ is denoted by $M \circledast N : A \rightrightarrows C$ where for all $x : A$, $X : \mathbb{P} C$*

$$x (M \circledast N) X \Leftrightarrow (\exists Y : x M Y : (\forall y : y \in Y : y N X))$$

So, given input value x , the angel can only guarantee that $M \circledast N$ will output a value that satisfies X if he can ensure that M will establish some intermediate postcondition Y and if he can also guarantee that N will establish X given any value in Y .

The composition operator is associative, with identity id_A for each A , and preserves up-closure. An alternative formulation of its definition that can be useful in calculations involving representable multirelations is given below.

Lemma 3.1. *Let $M : A \rightrightarrows B$ be representable, and let $N : B \rightrightarrows C$, then for all $x : A, X : \mathbb{P} C$*

$$x (M \circledast N) X \Leftrightarrow (\exists Z : Z \in \text{sp}(x, M) : (\forall z : z \in Z : z N X))$$

The proof of this lemma is omitted.

When working with folds, it is frequently useful to be able to insert an additional value when composing one multirelation with another. This composition-like operator captures that idea:

Definition 3.2. *For any $z : C$ and multirelations $M : A \rightrightarrows B$ and $N : (C \times B) \rightrightarrows D$, the composition with z of M and N is denoted by $M \circledast_z N : A \rightrightarrows D$ where for all $x : A, X : \mathbb{P} D$*

$$x (M \circledast_z N) X \Leftrightarrow (\exists Y : x M Y : (\forall y : y \in Y : (z, y) N X))$$

The operator \circledast_z behaves very much like \circledast does: it has associative properties which include

$$\begin{aligned} M \circledast_z (N \circledast S) &= (M \circledast_z N) \circledast S \\ M \circledast (N \circledast_z S) &= (M \circledast N) \circledast_z S \\ M \circledast_y (N \circledast_z S) &= (M \circledast_y N) \circledast_z S \end{aligned}$$

for all suitably-typed values and multirelations.

3.2 Lifting

Another pair of operators that are useful in program specifications are the following well-known functions for lifting relations to multirelations [BvW98]:

Definition 3.3. *For any relation $R : A \leftrightarrow B$, its angelic lifting $\langle R \rangle : A \rightrightarrows B$ is defined for all $x : A, X : \mathbb{P} B$ by*

$$x \langle R \rangle X \Leftrightarrow (\exists y : x R y : y \in X)$$

Definition 3.4. *For any relation $R : A \leftrightarrow B$, its demonic lifting $[R] : A \rightrightarrows B$ is defined for all $x : A, X : \mathbb{P} B$ by*

$$x [R] X \Leftrightarrow (\forall y : x R y : y \in X)$$

Both lifting operators distribute through composition and if R is a total function, then the two liftings coincide. A multirelation is said to be *angelic* (*demonic*) if it is the angelic (demonic) lifting of some relation. Notice that a demonic multirelation has only one strongest postcondition for any initial state, which is to be expected since it contains no angelic choice. In contrast, an angelic one may have many strongest postconditions, but each one must be a singleton set since it contains no demonic choice. One property of angelic multirelations that will be required in Section 5 is given by the following lemma, where $;$ denotes composition of ordinary relations.

Lemma 3.2. *Let $R : A \leftrightarrow B$ and $M : B \rightrightarrows C$, then*

$$\langle R \rangle \wp M = R ; M$$

Proof

Let $x \in A$ and $X \in \mathbb{P} C$, then

$$\begin{aligned} & x (\langle R \rangle \wp M) X \\ \Leftrightarrow & \quad \{\text{Definition of } \wp\} \\ & (\exists Y : x \langle R \rangle Y : (\forall y \in Y : y M X)) \\ \Leftrightarrow & \quad \{\text{Definition of } \langle \rangle \text{ and logic}\} \\ & (\exists y : x R y : y M X) \\ \Leftrightarrow & \quad \{\text{Definition of } ;\} \\ & x (R ; M) X \end{aligned}$$

□

3.3 Sums and Products

Sums are defined in exactly the same way for relations and multirelations:

Definition 3.5 (sum). *The sum of $M : A \rightrightarrows C$ and $N : B \rightrightarrows D$ is defined for all $x : A$, $y : B$ and $X + Y : \mathbb{P}(C + D)$ by*

$$\begin{aligned} (x, 0) (M \uplus N) (X + Y) &\Leftrightarrow x M X \\ (y, 1) (M \uplus N) (X + Y) &\Leftrightarrow y N Y \end{aligned}$$

where $C + D$ denotes the disjoint sum $(C \times \{0\}) \cup (D \times \{1\})$ of sets C and D .

So this specification simply tests the tag component of its input and then behaves like M or N accordingly. Once again, the strongest postconditions of $M \uplus N$ are closely related to those of M and N :

$$\begin{aligned} sp((x, 0), M \uplus N) &= \{X \times \{0\} \mid X \in sp(x, M)\} \\ sp((y, 1), M \uplus N) &= \{Y \times \{1\} \mid Y \in sp(y, N)\} \end{aligned}$$

It follows that the sum of two representable multirelations is also representable.

The *product* operator can be used to model the simultaneous execution of programs [BB94]. It is not exactly like that for relations, and instead has the following definition:

Definition 3.6 (product). The product of two multirelations $M : A \rightrightarrows C$ and $N : B \rightrightarrows D$ is defined for all $x : A$, $y : B$ and $Z : \mathbb{P}(C \times D)$ by

$$(x, y) (M \# N) Z \Leftrightarrow (\exists X, Y : x M X \wedge y N Y \wedge X \times Y \subseteq Z)$$

where $C \times D$ denotes the cartesian product of the sets C and D .

Not surprisingly, there is a very simple relationship between the strongest postcondition of $M \# N$ and those of M and N individually:

$$sp((x, y), M \# N) = \{X \times Y \mid X \in sp(x, M) \wedge Y \in sp(y, N)\}$$

It follows that the product of two representable multirelations is itself representable.

The following example illustrates how this operator can be used.

Example 3.1. In the well-known game of rock, paper, scissors, two players simultaneously offer up their hand in one of three ways symbolising the three choices. The rules stipulate that rock beats scissors, scissors beat paper and paper beats rock. The possibilities for each player can be represented by a set $A = \{\text{rock}, \text{paper}, \text{scissors}\}$, and so we represent a player's move by the relation $\text{move} = \mathbf{1} \times A$, where $\mathbf{1}$ denotes any singleton set. Thus a game between two opposing players can be modelled by

$$RPS = \langle \text{move} \rangle \# [\text{move}]$$

Obviously this multirelation allows the angel to make no guarantees about the demon's move, and so this game has no strategy apart from that recommended by game theorists of choosing rock, paper and scissors at random. \square

3.4 Map

The *map* operator of functional programming, which applies a function to each element of a list has the following analogue for multirelations:

Definition 3.7 (map). Let $M : A \rightrightarrows B$, then *map* M is defined (using the Haskell syntax for lists) for all $x : A$, $xs : [A]$ and $Z : \mathbb{P}[B]$ by

$$[] (map M) Z \Leftrightarrow [] \in Z$$

$$(x : xs) (map M) Z \Leftrightarrow (\exists Y, YS : x M Y \wedge xs (map M) YS \wedge (\forall y, ys : y \in Y \wedge ys \in YS \Rightarrow (y : ys) \in Z))$$

So *map* M behaves like the identity when its input is the empty list, but its behaviour on non-empty lists is more easily understood by considering strongest postconditions: for all $x : A$, $xs : [A]$,

$$sp(x : xs, map M) = \{consall(Y, YS) \mid Y \in sp(x, M) \wedge YS \in sp(xs, map M)\}$$

where $consall(Y, YS) = \{y : ys \mid y \in Y \wedge ys \in YS\}$.

The following example shows how this operator can be used.

Example 3.2. Consider an elected body of representatives (each member belonging to one of two political parties) participating in a legislative debate. After the debate, each representative must vote on the proposed legislation, and suppose the type $Ballot = \{For, Against, Abstain\}$ represents all possible choices of votes. Each representative's voting intentions can be described as a multirelation $castvote : Representative \rightrightarrows Ballot$.

If the disciplinarians in the angelic party have perfect control over their representatives, then any result can be guaranteed, in other words for any angelic representative a and $X : \mathbb{P} Ballot$,

$$a \text{ castvote } X \Leftrightarrow X \neq \emptyset$$

Here, the requirement that $X \neq \emptyset$ reflects the compulsion on representatives to vote. This specification has a number of strongest postconditions that a can choose between: $\{For\}$, $\{Against\}$ and $\{Abstain\}$.

Now suppose voter d is from the opposition, then we have no control over his ballot: for all $X : \mathbb{P} Ballot$,

$$d \text{ castvote } X \Leftrightarrow X = Ballot$$

This specification has a single strongest postcondition, namely the set of all valid ballots.

If all the representatives are represented by a list, their combined behaviour can be specified by

$$\begin{aligned} allvotes & : [Representative] \rightrightarrows [Ballot] \\ allvotes & = \text{map } castvote \end{aligned}$$

Then the result of the debate is specified by:

$$\begin{aligned} debateresult & : [Representative] \rightrightarrows \{For, Against, Draw\} \\ debateresult & = allvotes \circledast \langle (remove \text{ Abstain}); majority \rangle \end{aligned}$$

where $remove$ is the function that removes all of the specified element from a list, and $majority$ is the function that returns the majority of elements in a list, if one, and otherwise returns a draw. \square

3.5 Folds

The *fold* operators of functional programming, which capture a more general pattern of recursion than *map*, have analogues for multirelations. Here is the multirelational *foldr*:

Definition 3.8 (foldr). For all $M : A \times B \rightrightarrows B$ and $N : \mathbf{1} \rightrightarrows B$, we have that $foldr \ M \ N : [A] \rightrightarrows B$, and for all $X : \mathbb{P} B$,

$$\begin{aligned} [](foldr \ M \ N)X & \Leftrightarrow X \in \text{ran } N \\ (x : xs)(foldr \ M \ N)X & \Leftrightarrow xs((foldr \ M \ N) \circledast_x \ M)X \end{aligned}$$

So if the initial state is the empty list, the guarantees of $\text{foldr } M \ N$ are the same as those of N . If the initial state is of the form $x : xs$, then the guarantees are more complicated. Informally, it may help to remember that

$$[x_1, x_2, \dots, x_n](\text{foldr } M \ N)X \Leftrightarrow X \in \text{ran } (N \circ_{x_n} M \dots \circ_{x_2} M \circ_{x_1} M)$$

The strongest postconditions for a foldr are also defined recursively. For all $x : A$ and $xs : [A]$, we have that

$$\text{sp}(x : xs, \text{foldr } M \ N) = \{\cap\{sp((x, z), M \mid z \in Z) \mid Z \in \text{sp}(xs, \text{foldr } M \ N)\}$$

The following example illustrates the use of foldr :

Example 3.3. Consider a version of the game of Nim where there is a pile of matches and a stack of positively numbered cards. The top card is removed from the stack at the start of each round, and the number shown on the card is the number of matches then removed from the pile. The players then each remove either one or two matches in turn. The last round of the game occurs when the last card is removed from the stack. If either player removes the last match on this round, then that player loses.

Let the operation to represent the removal of matches in accordance with the number at the top of the stack be called cardremove , and let move represent a valid move for either player, then

$$\begin{aligned} \text{cardremove} &: \text{Nat} \times \text{Int} \rightarrow \text{Int} & \text{move} &: \text{Int} \leftrightarrow \text{Int} \\ \text{cardremove } (w, x) &= x - w & x \text{ move } y &\Leftrightarrow y = x - 1 \vee y = x - 2 \end{aligned}$$

Then, assuming that the angel goes first, the angelic and demonic liftings can be used to define a round as:

$$\begin{aligned} \text{round} &: \text{Nat} \times \text{Int} \rightrightarrows \text{Int} \\ \text{round} &= \langle \text{cardremove} \rangle \circ \langle \text{move} \rangle \circ [\text{move}] \end{aligned}$$

Now if we assume that the list of cards is consumed from right to left we can define the moves in the game of $\text{nim} : \text{Int} \rightarrow ([\text{Nat}] \rightrightarrows \text{Int})$ recursively as follows: let $x : \text{Int}$ and $(w : ws) : [\text{Nat}]$ denote the starting pile of matches and cards respectively, then for all $X : \mathbb{P}\text{Int}$

$$\begin{aligned} [] & \quad (\text{nim } x) X \Leftrightarrow x \in X \\ (w : ws) & \quad (\text{nim } x) X \Leftrightarrow \\ & \quad (\exists Y : ws (\text{nim } x) Y \wedge (\forall y : y \in Y : (w, y) \text{ round } X)) \end{aligned}$$

Equivalently, this definition can be written more concisely as a foldr :

$$\text{nim } x = \text{foldr } \text{round } (\text{const } x)$$

□

It is sometimes more convenient to model problems using non-empty lists. The foldr^+ is intended for just such situations, and it is defined in a very similar way to foldr :

Definition 3.9 (foldr⁺). For all $M : A \times B \rightrightarrows B$ and $N : A \rightrightarrows B$, we have that $\text{foldr}^+ M N : [A] \rightrightarrows B$, and for all $X : \mathbb{P} B$,

$$\begin{aligned} [x](\text{foldr}^+ M N)X &\Leftrightarrow x N X \\ (x : y : xs)(\text{foldr}^+ M N)X &\Leftrightarrow (y : xs)((\text{foldr}^+ M N) \circ_x M)X \end{aligned}$$

As in functional programming, there are also foldl and foldl^+ operators, which process elements of the list starting from the left-hand end:

Definition 3.10 (foldl). For all $M : A \times B \rightrightarrows B$ and $N : \mathbf{1} \rightrightarrows B$, we have that $\text{foldl} M N : [A] \rightrightarrows B$, and for all $X : \mathbb{P} B$,

$$\begin{aligned} [](\text{foldl} M N)X &\Leftrightarrow X \in \text{ran } N \\ (x : xs)(\text{foldl} M N)X &\Leftrightarrow xs((\text{foldl} M (N \circ_x M))X) \end{aligned}$$

Definition 3.11 (foldl⁺). For all $M : A \times B \rightrightarrows B$ and $N : A \rightrightarrows B$, we have that $\text{foldl}^+ M N : [A] \rightrightarrows B$, and for all $X : \mathbb{P} B$,

$$\begin{aligned} [x](\text{foldl}^+ M N)X &\Leftrightarrow x N X \\ (x : y : xs)(\text{foldl}^+ M N)X &\Leftrightarrow (y : xs)((\text{foldl}^+ M (N \circ_x M))X) \end{aligned}$$

Informally, the foldl operator can be thought of in the following way:

$$[x_1, x_2, \dots, x_n](\text{foldl} M N)X \Leftrightarrow X \in \text{ran } (N \circ_{x_1} M \circ_{x_2} \dots M \circ_{x_n} M)$$

Example 3.4. Consider a gambling opportunity where a customer (the angel) is presented with a series of monetary options $[x_1, x_2, \dots, x_n]$. The customer can either choose to accept the first amount offered (x_1), or can gamble by the second amount (x_2) to receive a payout of either $x_1 - x_2$ or $x_1 + x_2$, depending on what the demon chooses. He may then choose to stick, or to gamble by a further amount (x_3) to receive one of the payoffs $x_1 - x_2 - x_3$, $x_1 - x_2 + x_3$, $x_1 + x_2 - x_3$ or $x_1 + x_2 + x_3$, and so on. The multirelation $M : [Nat] \rightrightarrows Int$ that describes this situation can be written informally as:

$$\begin{aligned} [x_1, x_2, \dots, x_n] M X \\ \Leftrightarrow \\ \{x_1\} \subseteq X \vee \\ \{x_1 - x_2, x_1 + x_2\} \subseteq X \vee \\ \{x_1 - x_2 - x_3, x_1 + x_2 - x_3, x_1 - x_2 + x_3, x_1 + x_2 + x_3\} \subseteq X \vee \dots \end{aligned}$$

for all $X : \mathbb{P} Int$. More formally, M can be defined as a fold:

$$M = \text{foldl}^+ \text{gamble} \in$$

where $\text{gamble} : Int \times Int \rightrightarrows Int$ is defined for all $x, y : Int$, and $X : \mathbb{P} Int$ by

$$(y, x) \text{gamble } X \Leftrightarrow x \in X \vee \{x - y, x + y\} \subseteq X$$

□

In functional programming, there are duality theorems for fold operators, which state under which conditions a *foldl* may be expressed as a *foldr*. There is also a duality theorem for multirelations:

Theorem 3.1. *Let $M : A \times B \rightrightarrows B$ and $N : \mathbf{1} \rightrightarrows B$. If for all $Q : \mathbf{1} \rightrightarrows B$ and $x, y : A$ we have that $Q \circ_x M \circ_y M = Q \circ_y M \circ_x M$, then*

$$\text{foldl } M \ N = \text{foldr } M \ N$$

The proof of this theorem is omitted. It can be proved in the same way as the corresponding proof of the second duality theorem in [Bi98], and given this hint, the proof is straightforward.

4 Case Study: A Series of Elections

This example is inspired by voting systems.

There is to be an election. Two candidates (one from the angelic party, and one from the demonic) are standing, and only one is to be elected. Unlike some election systems which elect a candidate in a single mass voting event, the election of this candidate is split over several regions (so we will have a type *Region*). Each region has its own election on the election date for that region, thus declaring that region's preference, and the results from all the regional elections are combined in some way to select the eventual overall winner.

This being politics, money is heavily involved (you'd be cynical too if you did multirelational proofs). Each candidate has a certain amount of money to spend on the election overall: the angel has an amount a , and the demon has an amount d , for some $a, d : \text{Money}$, where $\text{Money} = \mathbb{R}^{\geq 0}$. In this system, it is assumed that the total number of votes a candidate obtains in a region is directly proportional to the money he spends in that region. Thus if the angelic candidate spends more than the demonic candidate in a region, the angelic candidate wins that region. (If you prefer to be less cynical about it, you may imagine that the two candidates can count on a roughly equal amount of voting support from their staunch supporters, and the money spent simply woos the floating voters, resulting in the same observation.)

We can represent a candidate's tally of how he is doing so far, by a pair of type $\{\text{Region}\} \times \text{Money}$, representing the set of regions the candidate has won so far, and the amount of money he has left. Considering this data for both candidates suggests a suitable type for representing the results so far:

$$\text{Results} = (\{\text{Region}\} \times \text{Money}) \times (\{\text{Region}\} \times \text{Money})$$

We will use the convention that the pair on the left represents the angelic candidate's results, and the pair on the right the demonic candidate's results.

Thus initially, before any of the regions have had their elections, the results so far are described by this multirelation:

$$\begin{aligned} \text{initially} & : \mathbf{1} \rightrightarrows \text{Results} \\ \text{initially} & = \text{const } ((\{\ }, a), (\{\ }, d)) \end{aligned}$$

A candidate makes preparations for a regional election by choosing how much money to spend in that region.

$$\begin{aligned} \text{Spend} & : \{Region\} \times Money \leftrightarrow (\{Region\} \times Money) \times Money \\ \forall m, s : Money, rs : \{Region\} & : (rs, m) \text{ Spend } ((rs, m), s) \Leftrightarrow 0 \leq s \leq m \end{aligned}$$

A region's preferred candidate is determined by seeing which candidate spent the most money:

$$\begin{aligned} \text{declare } (r, ((rs_a, m_a), s_a), ((rs_d, m_d), s_d)) \\ & = ((rs_a \cup \{r\}, m_a - s_a), (rs_d, m_d - s_d)), \text{ if } s_a > s_d \\ & = ((rs_a, m_a - s_a), (rs_d \cup \{r\}, m_d - s_d)), \text{ otherwise} \end{aligned}$$

Note that if the monies spent are equal, the region is given to the demonic candidate. This is because in that case, the vote will be so close that the angel certainly can't guarantee winning that region.

We are now in a position to define a multirelation *Election* representing an election for a single region:

$$\begin{aligned} \text{Election} & : (Region \times Results) \rightrightarrows Results \\ \text{Election} & = (id \ast (\langle \text{Spend} \rangle \ast [\text{Spend}])) \circledast \langle \text{declare} \rangle \end{aligned}$$

The $\langle \text{Spend} \rangle \ast [\text{Spend}]$ illustrates the spending decisions of the angel and demon being made in parallel, before the results are declared for that region and added to the tallies so far.

Assuming that the regions are listed in chronological order according to the date of their elections, then the whole series of elections can be suitably described by a *foldl*:

$$\begin{aligned} \text{ElectionSeries} & : [Region] \rightrightarrows Results \\ \text{ElectionSeries} & = \text{foldl } \text{Election } \text{initially} \end{aligned}$$

The angelic guarantees of the multirelation *ElectionSeries* thus represent possible predicates that the angel can ensure, given a list of regions, his own pot of money, and the demon's.

As for whether the angel actually wins the election or not, that all depends on how the results from all the elections are combined to produce an overall result. One way to do it would be to give a region a weighting (e.g. according to population size), and add up the weightings of the regions won by each candidate, and compare to see which candidate has the most. We will consider a simpler version, where the candidate who wins the most regions is also the overall winner.

4.1 “Will Sufficient Money Buy The Election?”

Although we are interested in knowing whether the angelic candidate wins a majority of the regions, it turns out to be more useful to consider a generalisation:

that of whether the angel can guarantee winning at least w regions, for various values of w . So we define, for $st : Results$

$$Wins_w(st) \Leftrightarrow \#angelwins(st) \geq w$$

$$\text{where } angelwins((rs_a, m_a), (rs_d, m_d)) = \#rs_a$$

and where $\#$ denotes the size or length of a list or set.

Thus we are interested for what values of w it is the case that

$$rs \text{ ElectionSeries } Wins_w$$

Experiments with assorted angelic and demonic pots of money along with small region lists suggest that whilst the angel can always guarantee winning at least 0 of the regions, for $w > 1$ the angel can guarantee winning w of the regions rs if the angel has more money than $(\#rs)d/(\#rs + 1 - w)$.

The intuition behind this suggests that if the demon wants to prevent the angel winning w regions, then the demon must win $\#rs + 1 - w$ regions. If the angel is unlucky then the demon will concentrate his money on the $\#rs + 1 - w$ regions on which the angel spends the least money, thus spending $d/(\#rs + 1 - w)$ on each of those regions. Thus the angel needs to maximise spending in the weakest regions, which can best be achieved by spreading money as evenly as possible across the regions, thus spending $a/(\#rs)$ in each region. Thus the angel will only have a guarantee if $a/(\#rs) > d/(\#rs + 1 - w)$. But can we prove it formally using multirelations?

The above suggests that what we should aim to prove is

$$(w \geq 0) \vee (1 \leq w \leq \#rs \wedge (\#rs + 1 - w)a > (\#rs)d) \quad (1)$$

$$\Rightarrow rs \text{ ElectionSeries } Wins_w \quad (2)$$

for all $a, d : Money$, $w : \mathbb{N}$ and $rs : [Region]$.

4.2 “Can We Prove It?”

When a *fold* is involved, an inductive proof is indicated. If the above suggestion is taken as an inductive hypothesis, this means the inductive hypothesis is in the form

$$P(w, rs) \Rightarrow rs(\text{foldl Election initially})Wins_w$$

and this is problematic, for the same reason that *foldl* proofs are awkward in standard functional programming: as soon as we start on the inductive case, we run into a problem:

$$(r : rs)(\text{foldl Election initially})Wins_w$$

$$\Leftrightarrow \{\text{definition of foldl}\}$$

$$rs(\text{foldl Election (initially } \mathfrak{g}_r \text{ Election)})Wins_w$$

As the inductive hypothesis does not involve *initially* \mathfrak{g}_r *Election*, it cannot be used.

Using *foldr* instead doesn't help, even though we can express *ElectionSeries* using *foldr* with the list of regions *rs* in reverse chronological order, and there is no difficulty manipulating the expression into a form where the inductive hypothesis can be used, like this:

$$\begin{aligned}
& (r : rs)(\text{foldr Election initially}) Wins_w \\
\Leftrightarrow & \quad \{\text{definition of foldr}\} \\
& rs((\text{foldr Election initially}) \mathfrak{g}_r \text{ Election}) Wins_w \\
\Leftrightarrow & \quad \{\text{definition of } \mathfrak{g}_r\} \\
& \exists Y : rs(\text{foldr Election initially}) Y : (\forall st : st \in Y : (r, st) \text{ Election Wins}_w)
\end{aligned}$$

Trying to go any further is difficult. It relies on demonstrating the existence of a predicate *Y* which is true after all regions but one have had their elections. It is difficult to see what *Y* could be, especially since it is not obvious how much money the demon has left by this stage and whether this is a sufficiently small amount to allow the angel to win the final election.

Going back to *foldl*, we need to use the standard functional programmer's trick for *foldl* inductions, and generalise the inductive hypothesis. We require something of the form

$$Sufficient(w, rs, Q) \Rightarrow rs(\text{foldl Election } Q) Wins_w \quad (3)$$

for $Q : 1 \Rightarrow Results$. We will impose conditions on *Q*: it must be representable and contain at least one non-empty predicate for the angel to choose from.

Note that the predicate *Sufficient* must involve the initialising multirelation *Q* in some way - after all, if *Q* initialised the angel with 7 regions to his credit before the elections in *rs* had even begun, this would severely affect the *w* for which the angel can guarantee *Wins_w*!

The predicate *Sufficient* must somehow express that the angel can guarantee that after performing *Q*, the angel has sufficient money relative to the demon. We give a definition of *Sufficient* in terms of the range of *Q* that illustrates a standard form of predicate to use when doing multirelational inductive proofs with *foldl*:

$$Sufficient(w, rs, Q) \Rightarrow MoneyEnough_{w,rs} \in \text{ran } Q$$

where

$$\begin{aligned}
MoneyEnough_{w,rs}(st) \Leftrightarrow & (w \leq \#aw(st) \vee \\
& (1 \leq w - \#aw(st) \leq \#rs \wedge \\
& (\#rs + 1 + \#aw(st) - w)am(st) > (\#rs)dm(st)))
\end{aligned}$$

where *aw*, *am* and *dm* are abbreviations for the obvious functions *angelwins*, *angelmoney* and *demonmoney* on type *Results*.

4.3 “Yes!”

We are now in a position to prove (3), and this will imply (2) by substituting *initially* for Q .

Proof

The full proof is too lengthy for this paper, but we will sketch the proof, which uses induction over rs with hypothesis (3).

Case ($rs = []$):

This is straightforward when considering the guarantees Q provides in terms of the strongest postconditions of Q .

Case ($r : rs$):

$$\begin{aligned}
& (r : rs)(\text{foldl Election } Q) \text{Wins}_w \\
\Leftrightarrow & \{ \text{definition of foldl} \} \\
& rs(\text{foldl Election } (Q \circ_{\mathfrak{g}_r} \text{Election})) \text{Wins}_w \\
\Leftarrow & \{ \text{inductive hypothesis} \} \\
& \text{Sufficient}(w, rs, Q \circ_{\mathfrak{g}_r} \text{Election}) \\
\Leftrightarrow & \{ \text{definition of Sufficient} \} \\
& \text{MoneyEnough}_{w,rs} \in \text{ran } (Q \circ_{\mathfrak{g}_r} \text{Election}) \\
\Leftarrow & \{ \text{definition of } \mathfrak{g}_- \} \\
& \exists Y : Y \in \text{ran } Q : (\forall st : st \in Y : (r, st) \text{Election MoneyEnough}_{w,rs}) \\
\Leftarrow & \{ \text{claim} \} \\
& \exists Y : Y \in \text{sp}(1, Q) : (\forall st : st \in Y : \text{MoneyEnough}_{w,r:rs}(st)) \\
\Leftarrow & \{ \text{strongest postconditions, } Q \text{ representable} \} \\
& \text{MoneyEnough}_{w,r:rs} \in \text{ran } Q \\
\Leftarrow & \{ \text{definition of Sufficient} \} \\
& \text{Sufficient}(w, r : rs, Q)
\end{aligned}$$

For the claim, it suffices to prove that

$$\text{MoneyEnough}_{w,r:rs}(st) \Rightarrow (r, st) \text{Election MoneyEnough}_{w,rs}$$

This can be shown by choosing a particular strongest postcondition P from $\text{sp}((r, st), \text{Election})$, and showing that when $\text{MoneyEnough}_{w,r:rs}(st)$, it is the case that $P \Rightarrow \text{MoneyEnough}_{w,rs}$.

The strongest postconditions for *Elections* are characterised by the angel’s choices of how much to spend at that election, and the choice of P corresponding to the choice of $\text{angelmoney}(st)/(\#rs + 1)$ (which might be guessed from the above discussion) allows (4) to be proved with careful detailed predicate calculus calculations (omitted). \square

4.4 Further Notes

We can use the duality theorem for multirelations (Theorem 3.1), to express *ElectionSeries* as a *foldr* too

$$ElectionSeries = foldr Election initially$$

since it is the case that

Proposition 4.1. *For all $Q : \mathbf{1} \Rightarrow Results$ and $x, y : Region$, it is the case that*

$$Q \circ_x Election \circ_y Election = Q \circ_y Election \circ_x Election$$

Proof

The details of this proof are omitted, to spare the reader much tedious predicate calculus, but here is a proof sketch:

It is sufficient to prove that for all $st : Results$ and $X : \{Results\}$,

$$(x, st)(Election \circ_y Election)X \Leftrightarrow (y, st)(Election \circ_x Election)X$$

One side of the above is expanded using the strongest postconditions for the representable multirelation *Election* (see Lemma 3.1), and it is demonstrated that this is symmetrical in x and y . Note that this necessitates election results being represented as sets of regions won so far, rather than a list, otherwise the guarantees of the two above multirelations are different. \square

The choice of postconditions in the proof provides advice for these sorts of elections: if you have a lot more money than your opponent does, then your best chance of guaranteeing a win comes from spreading it out evenly over the regions. However, if you have less money than your opponent, the best chance of defeating your opponent comes from concentrating your money in fewer regions.

5 Initial Algebras and Folds

This section describes the theoretical concepts that underly the definitions of the operations given in Section 3. In particular, it will be shown that functors (like *map*) and initial algebras (like *foldr*) have a unique extension from relations to multirelations. These results are well-known properties of predicate transformers [GMdM94, deM92] which are isomorphic to multirelations [Rew03], but it is instructive to rework some of the proofs directly for multirelations because they turn out quite differently. For instance, the extension of initial algebras is much simpler to derive for multirelations.

There are at least two well established ways to extend both functors and initial algebras from total functions to relations: via Kleisli categories [Fo94]

and using span categories [CK84,FrS93]. The same is true of multirelations. The latter method has the advantage that it produces a unique extension of a functor which is itself a weak kind of functor. This section describes a version of this method, based on the factorisation of multirelations into an angelic part followed by a demonic part.

It is beyond the scope of this paper to explain the details of categorical concepts, see [BaW90] for example, for an introduction to category theory. The notation $p ; q$ will be used for the composite of each pair of arrows with compatible types $p : A \rightarrow B$ and $q : B \rightarrow C$, and id_A is the identity arrow on object A . Function application will be denoted by juxtaposition.

5.1 Order-enriched Categories

Both relations and multirelations will be modelled using *order enriched* categories, which are those with a partial order defined on homsets with respect to which the categorical composition is monotonic.

The order enriched category of relations, **Rel**, is defined as follows. The objects of **Rel** are sets, arrows are relations, composition is defined by $;$ and the identity arrow for each set A is the identity function id_A . The order on arrows in **Rel** is subset inclusion.

The order enriched category of up-closed multirelations, **Mul**, is defined as follows. The objects of **Mul** are sets, arrows are up-closed multirelations, composition is defined by \circledast , and the identity arrow for each set A is the set membership relation \in_A . The order on arrows in **Mul** is subset inclusion.

The collection of all angelic and demonic multirelations each form subcategories of **Mul** which we will call **Ang** and **Dem** respectively. Both of the lifting functions of Definitions 3.3 can be thought of as functors that define an isomorphism between **Rel** and each of these subcategories.

5.2 Undoing the Demon's Handiwork

Order enriched categories accommodate a greater range of constructs than ordinary categories because all of the standard definitions can be weakened by substituting inequality for equality. An example of such a definition is that of a *map* [CK84,FrS93] which is a weak analogue of the concept of an isomorphism. (It is unfortunate that the word *map* is used elsewhere in this paper for a very different purpose, but since both are standard it is hoped that the meaning will be clear from the context.)

Definition 5.1 (categorical map). *Let (\mathcal{C}, \subseteq) be an order enriched category. An arrow $m : A \rightarrow B$ is a map if and only if it has a comap $m^* : B \rightarrow A$ such that both*

$$id_A \subseteq m ; m^* \text{ and } m^* ; m \subseteq id_B. \quad (4)$$

It is immediate from the following shunting rule that every map uniquely determines its comap and vice versa. Let $M : A \rightarrow B$, $N : D \rightarrow C$, $r : B \rightarrow C$ and $s : A \rightarrow D$ in an order enriched category (\mathcal{C}, \subseteq) , where r and s are maps, then

$$M ; r \subseteq s ; N \quad \Leftrightarrow \quad s^* ; M \subseteq N ; r^* \quad (5)$$

In **Mul** the maps are precisely the demonic multirelations, Every demonic multirelation $[R]$ has comap $\langle R^\circ \rangle$, and conversely, it can be shown that every map is demonic. So the concept of a map gives a succinct algebraic characterisation of demonic multirelations, and dually, comaps characterise angelic multirelations.

Intuitively, a comap can be thought of as the angelic multirelation that allows the angel to choose to undo the choices made by its corresponding demonic map.

Furthermore, the $*$ operator can be considered as a (contravariant) functor that defines an isomorphism between the categories **Dem** and **Ang**. In future, demonic multirelations will be written in lowercase, by analogy with the common convention for total functions, which are the maps in the category of relations.

5.3 Factorisation

Just as every ordinary relation can be factorised into an inverse function followed by a function, every multirelation can be factorised into an angelic part followed by a demonic part: let $M : A \rightrightarrows B$, then since $[\exists_B] = \subseteq_B$

$$\begin{aligned} M & \\ &= \{ \text{up-closure} \} \\ & \quad M ; [\exists_B] \\ &= \{ \text{Lemma 3.2} \} \\ & \quad \langle M \rangle \circ [\exists_B] \end{aligned}$$

This is not the only way to construct a factorisation of this kind, for example, let C be the subset of $A \times \mathbb{P}B$ corresponding to M , and $outl : C \rightarrow A$, $outr : C \rightarrow \mathbb{P}B$ be the projection functions, and let $^\circ$ denote relational converse, then

$$\begin{aligned} M & \\ &= \{ \text{definition of } ; \} \\ & \quad outl^\circ ; outr \\ &= \{ \text{calculation above} \} \\ & \quad \langle outl^\circ ; outr \rangle \circ [\exists_B] \\ &= \{ \text{angelic lifting is a functor} \} \\ & \quad \langle outl^\circ \rangle \circ \langle outr \rangle \circ [\exists_B] \\ &= \{ \text{lifting functors agree on total functions} \} \\ & \quad \langle outl^\circ \rangle \circ [outr] \circ [\exists_B] \\ &= \{ \text{demonic lifting is a functor} \} \\ & \quad \langle outl^\circ \rangle \circ [outr ; \exists_B] \end{aligned}$$

This factorisation appears to be quite different from the previous one, but the two are related in a sense that is captured by the following definition:

Definition 5.2 (unique map factorisation). *Let $(\mathcal{C}, \sqsubseteq)$ be an order enriched category. Then \mathcal{C} has unique map factorisation if, for every arrow $M : A \rightarrow B$, there exists a pair of maps $t : C \rightarrow A$ and $u : C \rightarrow B$ such that*

$$M = t^* ; u$$

and for any other pair of maps r and s , $r^ ; s \subseteq M$ if and only if there exists a map h such that $h ; t \subseteq r$ and $s \subseteq h ; u$.*

The proof of this uniqueness property of multirelations is included in the appendix. Factorisation is useful because it provides a mechanism for extending the definitions of functors like *map* to multirelations, but unfortunately the result is not guaranteed to be a functor itself. Instead, it is necessary to introduce a weak analogue of the notion of a relator.

5.4 Relators

Recall the following definition from, for example [BdM97].

Definition 5.3 (relator). *A relator is defined to be a monotonic functor.*

It is easy to check that every relator F preserves comaps, which is to say that for all maps r ,

$$F(r^*) = (Fr)^*$$

This property motivates the following weaker notion.

Definition 5.4 (uprelator). *Let $(\mathcal{C}, \sqsubseteq)$ be an order enriched category. An uprelator $F : \mathcal{C} \rightarrow \mathcal{C}$ is a monotonic graph morphism such that for all arrows M , N and maps r ,*

$$\begin{aligned} Fid_A &= id_{FA} \\ F(M ; N) &\subseteq FM ; FN \\ F(r^*) &= (Fr)^* \end{aligned}$$

So whereas relators distribute through composition, uprelators do not necessarily do so. The product is an example of such an operator, since for instance if \emptyset denotes the empty set, **magic** = $[\emptyset]$ and **abort** = $\langle \emptyset \rangle$

$$(id \ast \text{magic}) \circ (\text{abort} \ast \text{abort}) \neq (id \circ \text{abort}) \ast (\text{magic} \circ \text{abort})$$

It will be assumed that relators and uprelators bind more tightly than any other operations, and since there is no difference between $F(r^*)$ and $(Fr)^*$, the notation Fr^* will be used from now on to mean either. Note that the last clause in the definition of an uprelator is equivalent to saying that for all arrows M , N and maps r ,

$$F(M ; r) = FM ; Fr \quad \text{and} \quad F(r^* ; N) = Fr^* ; FN \quad (6)$$

If \mathcal{C} is a category with unique map factorisation and $\text{Map } \mathcal{C}$ is the subcategory of all maps in \mathcal{C} , then every relator $F : \text{Map } \mathcal{C} \rightarrow \text{Map } \mathcal{C}$ has a well-defined extension to an uprelator $\widehat{F} : \mathcal{C} \rightarrow \mathcal{C}$ defined for all $r^* ; s$ by

$$\widehat{F}(r^* ; s) = (Fr)^* ; Fs \quad (7)$$

So every relator $F : \mathbf{Rel} \rightarrow \mathbf{Rel}$ can be extended to an uprelator $\mathbb{F} : \mathbf{Mul} \rightarrow \mathbf{Mul}$ by the above method via the lifting functors: for all $M : A \rightrightarrows B$,

$$\mathbb{F} M = \langle F M \rangle_{\mathfrak{g}} [F \ni_B] \quad (8)$$

The unique map factorisation property guarantees that this is an uprelator and the following lemma ensures that it is the only one that coincides with F on relations.

Lemma 5.1. *If \mathcal{C} is a category with unique map factorisation and two uprelators $F, G : \mathcal{C} \rightarrow \mathcal{C}$ agree on maps, that is $Fm = Gm$ for all maps m , then $F = G$.*

The proof of this lemma is in the appendix. It shows that the behaviour of every uprelator on multirelations is uniquely determined by its behaviour on relations. For example, the *product*, *sum* and *map* uprelators of Section 3 were derived via equation (8) from their counterparts on relations.

It is natural to consider whether it is really worth using these categorical concepts to formulate definitions like *map* when they could probably have been guessed quite easily without them. One reason for doing so is that each operator defined by equation (8) is guaranteed to be an uprelator, and so, for example, it preserves the refinement ordering and obeys the laws of equation (6). Moreover, each such operator is known to be the only one that agrees with its analogue for relations. It can also be shown that if a relator F satisfies any law that is expressed as a natural transformation, then its extension \widehat{F} will inherit a weaker form of the law, but it is beyond the scope of this paper to include such laws. Similar arguments can be used to justify the use of initial algebras to formulate the definition of *fold*. Perhaps most importantly, they give a generic definition of *fold*, so it is valid across all regular datatypes.

5.5 Initial Algebras

The definition given below generalises the standard one slightly by using uprelators instead of functors.

Definition 5.5 (initial algebra). *Let F be an uprelator from some order enriched category \mathcal{C} to itself. By definition, an F -algebra is an arrow of type $FA \rightarrow A$ for some A . An F -algebra $\tau : FT \rightarrow T$ is initial if, for each F -algebra $p : FA \rightarrow A$ there exists an arrow $(\downarrow p) : T \rightarrow A$ that satisfies the equivalence*

$$(\tau ; q = F q ; p) \equiv (q = (\downarrow p))$$

Arrows of the form $\langle p \rangle$ are called *catamorphisms* and correspond to the familiar fold and reduce operators.

It has been known for some time that initial algebras are preserved under the extension of functors from total functions to relations [EW67]. More recently, it was shown that the extension of functors from relations to predicate transformers transforms initial algebras into final coalgebras [deM92]. The restatement of this result for multirelations uses only initial algebras and is given below:

Lemma 5.2. *Let $F : \mathbf{Rel} \rightarrow \mathbf{Rel}$ be a relator with initial algebra $\tau : FT \rightarrow T$. Then $\mathbb{F} : \mathbf{Mul} \rightarrow \mathbf{Mul}$ is an uprelator with initial algebra $\langle \tau \rangle$.*

Proof

Let $K : T \rightrightarrows A$ and $H : FA \rightrightarrows A$, then

$$\begin{aligned} \langle \tau \rangle \circledast K &= \mathbb{F}K \circledast H \\ &\equiv \{ \text{Definition of } \mathbb{F} \text{ (8)} \} \\ \langle \tau \rangle \circledast K &= \langle FK \rangle \circledast [F \ni] \circledast H \\ &\equiv \{ \text{Lemma 3.2} \} \\ \tau ; K &= FK ; ([F \ni] \circledast H) \\ &\equiv \{ \text{Definition 5.5} \} \\ K &= \langle [F \ni] \circledast H \rangle \end{aligned}$$

□

This proof is much shorter than its counterpart for predicate transformers and the corresponding fold operator is also more useful because it has a more familiar form, as we saw in Definition 3.8, which stated that a *foldr* is a catamorphism of the initial algebra associated with the uprelator $F_A : \mathbf{Mul} \rightarrow \mathbf{Mul}$, defined by

$$\begin{aligned} F_A(B) &= 1 \uplus (A \ast B) \\ F_A(M) &= id_1 \uplus (id_B \ast M) \end{aligned}$$

The fusion law below is a fairly immediate consequence of Definition 5.5. Unfortunately, it is less succinct than usual because of the relaxation of the condition that F should be a relator.

Lemma 5.3 (fusion). *Let $F : \mathbf{Mul} \rightarrow \mathbf{Mul}$ be an uprelator, and let $M : FA \rightrightarrows A$, $N : A \rightrightarrows B$ and $P : FB \rightrightarrows B$, then*

$$(M \circledast N = FN \circledast P \wedge F(\langle M \rangle \circledast N) = F\langle M \rangle \circledast FN) \Rightarrow \langle M \rangle \circledast N = \langle P \rangle$$

Clearly, if F is a functor then the second part of the antecedent above can be dropped. By equation (6) this is also true if M is angelic or N is demonic. A more concrete example of this law, in the context of lists, is given in the following section,

5.6 Fusion for Lists

The fusion laws associated with the definition of *foldr* on relations have been widely used in the derivation of functional programs [BdM97], and ultimately the same might be true of the corresponding laws for multirelations. Unfortunately, the law given below is less succinct than its analogue on relations because the product operator is an uprelator, rather than a relator, and so an extra condition is necessary. Nevertheless, it is still much easier to apply than its counterpart for predicate transformers.

Lemma 5.4 (fusion). *Let $M : A \times B \rightrightarrows B$, $M' : A \times B' \rightrightarrows B'$, $N : 1 \rightrightarrows B$, $N' : 1 \rightrightarrows B'$, and $K : B \rightrightarrows B'$, then*

$$\begin{aligned} N \circledast K &= N' \wedge M \circledast K = (id_A \ast K) \circledast M' \\ \wedge \\ &(\forall R, S : R \ast (S \circledast K) \supseteq (R \ast S) \circledast (id_A \ast K)) \\ \Rightarrow \\ &(fold\ M\ N) \circledast K = fold\ M'\ N' \end{aligned}$$

The statement of this law for non-empty lists is identical apart from the types of N and N' . Although the quantified expression above looks rather cumbersome, it can be dropped completely if M and N are angelic or K is demonic. This fact is used in the statement of the following special case of the fusion law.

Lemma 5.5 (map fusion). *Let $M : A \times B \rightrightarrows B$, $N : 1 \rightrightarrows B$ and $K : A' \rightrightarrows A$, then if K is angelic or M and N are demonic,*

$$map\ K \circledast fold\ M\ N = fold\ (K \ast id) \circledast M\ N$$

6 Conclusions

This paper has introduced *map* and *fold* operators for multirelations, illustrated with examples of their use in modelling and proofs. These operators have been shown to be canonical, in the sense that they are the only ones that agree with their counterparts for relations and functions. Both operators had been defined previously in the equivalent framework of predicate transformers, but they did not take on such a familiar form in that context, and the uniqueness proofs did not work out so simply. The datatype of lists has been used for illustration in this paper, but the generic nature of the definitions means that they are equally applicable to any other regular datatype, such as trees.

We have also presented some theorems showing some of the properties of maps and folds. One topic for future work is the meticulous cataloguing of the various laws of multirelations, including further exploration of those concerning the *map* and *fold* operators. In particular, it would be interesting to see whether any other well-known theorems of *foldr* translate to multirelations in the same way as the duality theorem of [Bi98] and the fusion laws of [BdM97]. The definition of *unfold* also remains to be derived, possibly through the use of Kleisli

categories [Fo94]. But perhaps the most pressing concern is to discover more applications of multirelations, like the voting example considered here, in order to demonstrate their potential value. There are areas which could benefit from the application of multirelations, including security, voting systems, transmission protocols, resource-sharing protocols and games.

References

- [BvW98] Back, R. J. R. and von Wright, J. (1998) *Refinement Calculus: A Systematic Introduction*. Graduate Texts in Computer Science. New York: Springer-Verlag.
- [Bi98] Bird, R. S. (1998) *Introduction to Functional Programming*. Prentice Hall.
- [BdM97] Bird, R. S. and de Moor, O. (1997) *Algebra of Programming*. Prentice Hall.
- [BB94] Back, R. J. R. and Butler, M. J. (1995) Exploring Summation and Product Operators in the Refinement Calculus. In *Proceedings of Mathematics of Program Construction 94*. Mller, B., Eds.
- [BaW90] Barr, M. and Wells, C. (1990). *Category theory for computing science*. Prentice-Hall.
- [BF02] Brams, S. and Fishburn, P.C. (2002) Voting procedures. In Arrow, K., Sen, A. and Suzumura, K., editors, *Handbook of Social Choice and Welfare*, Vol 1, North Holland.
- [CK84] Carboni, A and Kasangian, S. (1984) Bicategories of Spans and Relations. *Journal of Pure and Applied Algebra* 33, 259-267.
- [DaP02] Davey, B. A. and Priestley, H. A. (2002) *Introduction to Lattices and Order (Second Edition)*. Cambridge University Press.
- [DiS90] Dijkstra, E. W. and Scholten, C. S. (1990) *Predicate Calculus and Program Semantics*. Springer Verlag.
- [EW67] Eilenberg, S. and Wright, J. B. (1967) Automata in general algebras *Information and Control*, 11(4), 452-470.
- [Fo94] M. M. Fokkinga. (1994) Monadic maps and folds for arbitrary datatypes. *Memoranda Informatica*, 94-28, University of Twente. *Categories, Allegories*. Springer Verlag.
- [FrS93] Freyd, P. and Šcedrov, A. (1993) *Categories, Allegories*. Springer Verlag.
- [GMdM94] Gardiner, P. H. B., Martin, C. E. and de Moor, O. (1994) An algebraic construction of predicate transformers. *Science of Computer Programming*, 22(1-2):21-44.
- [GPC90] Gordon, G., Pressman, I. and Cohen, S. (1990) *Quantitative Decision Making For Business*, Prentice-Hall.
- [Mar91] Martin, C. E. (1991) Preordered Categories and Predicate Transformers. D. Phil Thesis. Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD
- [Mar95] Martin, C. E. (1995) Towards a Calculus of Predicate Transformers *Lecture Notes in Computer Science* 969 : 489-498.
- [MCR04] Martin, C. E., Curtis, S. A. and Rewitzky, I. (2004) Modelling Nondeterminism *Lecture Notes in Computer Science* 3125 : 228-251.
- [deM92] de Moor, O. (1992) Inductive Data Types for Predicate Transformers. *Information Processing Letters* 43(3): 113-117.
- [P03] Pauly, M. (2002) Programming and verifying Subgame Perfect Mechanisms. <http://www.csc.liv.ac.uk/pauly/>

- [Rew03] Rewitzky, I. (2003) Binary Multirelations. In: Theory and Application of Relational Structures as Knowledge Instruments. (H de Swart, E Orłowska, G Schmidt, M Roubens (eds)). *Lecture Notes in Computer Science* 2929 : 259-274
- [W94] Ward, N. T. E. (1994) *A Refinement Calculus for Nondeterministic Expressions*.
www.dstc.monash.edu.au/staff/nigel-ward/nwthesis.pdf

Appendix

Just after Definition 5.2 it was claimed that **Mul** has unique map factorisation.

Proof First suppose that $r : D \rightrightarrows A$, $s : D \rightrightarrows B$, $t : C \rightrightarrows A$, $u : C \rightrightarrows B$ and $h : D \rightrightarrows C$ are demonic multirelations such that

$$h \circledast t \subseteq r \text{ and } s \subseteq h \circledast u \quad (9)$$

then we can calculate that

$$\begin{aligned} & h \circledast t \subseteq r \text{ and } s \subseteq h \circledast u \\ \equiv & \{ * \text{ is a contravariant functor} \} \\ & r^* \subseteq t^* \circledast h^* \text{ and } s \subseteq h \circledast u \\ \Rightarrow & \{ \text{Monotonicity of } \circledast \} \\ & r^* \circledast s \subseteq t^* \circledast h^* \circledast h \circledast u \\ \Rightarrow & \{ \text{Definition of map (4)} \} \\ & r^* \circledast s \subseteq t^* \circledast u \end{aligned}$$

Notice that nothing in the above calculation depends on multirelations, and so it is true in any order enriched category. For the converse, suppose that

$$r^* \circledast s \subseteq t^* \circledast u$$

then by the shunting rule (5), we have that

$$r^* \subseteq t^* \circledast u \circledast s^*$$

Since comaps are angelic, by Lemma 6.1 below, there exists h^* such that

$$\begin{aligned} & r^* \subseteq t^* \circledast h^* \text{ and } h^* \subseteq u \circledast s^* \\ \equiv & \{ * \text{ is a contravariant functor and shunting (5)} \} \\ & h \circledast t \subseteq r \text{ and } s \subseteq h \circledast u \end{aligned}$$

as required. \square

Lemma 6.1. *Let $R : A \leftrightarrow C$, $S : A \leftrightarrow B$ and $M B \rightrightarrows C$ then*

$$\begin{aligned} & \langle R \rangle \subseteq \langle S \rangle \circledast M \\ \Rightarrow \exists H : B \leftrightarrow C : & \\ & \langle R \rangle \subseteq \langle S \rangle \circledast \langle H \rangle \text{ and } \langle H \rangle \subseteq M \end{aligned}$$

Proof

$$\begin{aligned} & \langle R \rangle \\ &= \{ \in ; \{ \}^\circ = id \} \\ & \langle R ; \in ; \{ \}^\circ \rangle \\ &= \{ \text{Definition of } \langle \rangle \} \\ & \langle \langle R \rangle ; \{ \}^\circ \rangle \\ &\subseteq \{ \text{Assumption and monotonicity of } \langle \rangle \} \\ & \langle \langle S \rangle \circ M ; \{ \}^\circ \rangle \\ &= \{ \text{Lemma 3.2 and associativity of } ; \} \\ & \langle S ; M ; \{ \}^\circ \rangle \\ &= \{ \langle \rangle \text{ distributes through composition} \} \\ & \langle S \rangle \circ \langle M ; \{ \}^\circ \rangle \end{aligned}$$

which establishes the result since $\langle \{ \}^\circ \rangle \subseteq [\exists]$ and so

$$\langle M ; \{ \}^\circ \rangle \subseteq \langle M \rangle \circ [\exists] = M$$

□

Lemma 5.1

Proof $r^* ; s$ is a map factorisation of M , then

$$\begin{aligned} & F M \\ &= \{ \text{map factorisation} \} \\ & F (r^* ; s) \\ &= \{ (6) \} \\ & F r^* ; F s \\ &= \{ F \text{ and } G \text{ agree on maps (and hence comaps)} \} \\ & G r^* ; G s \\ &= \{ (6) \} \\ & G (r^* ; s) \\ &= \{ \text{map factorisation} \} \\ & G M \end{aligned}$$

□