Available online at www.sciencedirect.com
ScienceDirect
Science of Computer Programming 65 (2007) 140–158
Science of Computer Programming
www.elsevier.com/locate/scico

Modelling angelic and demonic nondeterminism with multirelations

C.E. Martin[a,*], S.A. Curtis[a], I. Rewitzky[b]

[a] *Department of Computing, Oxford Brookes University, United Kingdom*
[b] *Department of Mathematics, University of Stellenbosch, South Africa*

Received 31 January 2005; received in revised form 15 January 2006; accepted 30 January 2006
Available online 28 November 2006

Abstract

This paper presents an introduction to a calculus of binary multirelations, which can model both angelic and demonic kinds of non-determinism. The isomorphism between up-closed multirelations and monotonic predicate transformers allows a different view of program transformation, and program transformation calculations using multirelations are easier to perform in some circumstances. Multirelations are illustrated by modelling both kinds of nondeterministic behaviour in games and resource-sharing protocols.

Keywords: Multirelation; Predicate transformer; Agent; Strongest postcondition; Angelic nondeterminism; Demonic nondeterminism; Resource sharing

1. Introduction

For many years the realm of total functions was considered to be the natural framework in which to reason about functional programs [6]. The limitation of this framework is that it can only be used to describe deterministic programs: those that deliver only one output for each input. More recently, the development of the relational calculus for program derivation [7] has allowed programmers to reason about programs together with their specifications, which may be nondeterministic: they may offer a choice of outputs for each input. Specifications expressed in this calculus can be manipulated and refined into functions that can be translated into an appropriate functional programming language. We now propose to go one step further, since relations can be used to describe only angelic or demonic nondeterminism, but not both.

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 one must be prepared for the worst possible outcome. It is well known that both these kinds of behaviour can be described in the domain of monotonic predicate transformers [2,14], but this is usually associated with the derivation of imperative, rather than functional programs.

* Corresponding author.
E-mail address: cemartin@brookes.ac.uk (C.E. Martin).

0167-6423/$ - see front matter
doi:10.1016/j.scico.2006.01.007

An equivalent relational model that has recently been proposed consists of up-closed multirelations [16]. Relational equivalents of predicate transformers have been introduced in the past (see [12] for example), but Rewitzky has extended this equivalence to show that any monotonic function over a boolean algebra has an alternative representation as a multirelation.

This paper is devoted to showing how specifications involving both angelic and demonic nondeterminism can be expressed and manipulated as multirelations. Such specifications can be refined until they contain only one kind of nondeterministic choice. Then they can be translated into programs in which those remaining choices are made interactively by an external agent at run time.

This is not the first attempt to introduce angelic and demonic nondeterminism into a calculus for the derivation of functional programs. Ward [18] developed a refinement calculus for this purpose, in a similar style to that of [2,14], and with a corresponding predicate transformer semantics. However, we argue that multirelations will provide a better model for several reasons. First, one of the beauties of the relational calculus (for examples of its use see [3,7,17]) is that specifications are simply relations: there is no separate command language. As such, they can be manipulated by all the familiar operations on relations, such as composition and inverse, as well as those developed specifically for the derivation of functional programs. In contrast, in the refinement calculus of both [2,14] and [18] the command language is distinct from its predicate transformer semantics. So specifications can only be manipulated using laws that were previously derived from the semantics. This places a burden on the developer to memorise all the laws, and can complicate the reasoning about operations as fundamental as composition, which can be expressed simply for predicate transformers, but much less so for specifications in the command language. One solution to this problem could be to represent specifications directly as predicate transformers, but this might be counter-intuitive because they model programs in reverse, mapping postconditions to preconditions. In contrast, multirelations can be used to relate inputs, or initial states, to outputs, or final states, so we propose to blur the distinction between multirelations and the specifications they represent.

As an illustration of our theory, we will give examples demonstrating how multirelations can be used to specify and manipulate some games and resource-sharing protocols. For example, in a two-player game the choices of our player could be modelled by angelic nondeterminism, and those of our opponent by demonic nondeterminism. The resulting specification would allow us to reason about the game as a whole, and then derive an implementation that achieves some goal, such as an optimal strategy for the player. The implementation would then consist of a computerised player, who would play according to the strategy against a human opponent who could make demonic choices in an attempt to defeat the machine.

The rest of this paper is organised as follows. Section 2 introduces multirelations and shows how they can be used to model contracts between two agents. Section 3 discusses refinement, together with the associated issues of correctness and feasibility. Two applications are considered in Section 4, which is followed by a description of the isomorphism between the categories of multirelations and predicate transformers in Section 5. Some concluding remarks are given in Section 6.

2. Nondeterministic specifications

2.1. Agents

We are primarily interested in specifying systems that contain both angelic and demonic choice. One way to think of such specifications is as a contract [2] between two agents, both of whom are free to make various choices. One of the agents represents our interests, but the other may not. In this context, the angelic choices are interpreted as those made by our agent, since we assume that he will always make the choice that results in the best outcome for us. Conversely, the demonic choices are those made by the other agent, and since we have no control over them we must be prepared for all possibilities, including the worst.

For example, the specification of a simple vending machine from a customer's viewpoint might contain an angelic choice of coin insertion and chocolate bar selection, followed by a demonic choice of coins returned by the machine as change. This can be viewed as a contract where our agent is the customer and the other agent is the machine designer who can choose how the machine gives out change.

Specifications like this, featuring both angelic and demonic nondeterminism, have been modelled previously using monotonic predicate transformers (see [2,14] for example), but until now there has been no equivalent relational

model. The problem with the ordinary relational calculus is that only one kind of nondeterminism can be described at a time, but fortunately this is not the case for the multirelational calculus as we shall now see.

2.2. Multirelations

Multirelations were introduced in [16] as an alternative to predicate transformers for reasoning about programs. As the name suggests, they are to relations what multifunctions (relations) are to functions: that is to say, they are relations whose target type is a powerset type.

Definition 1 (*Multirelation*). A *multirelation* with source A and target B is a subset of the cartesian product $A \times \mathbb{P}\, B$. As such, it is a set of ordered pairs (x, X) where $x \in A$ and $X \subseteq B$.

We will interpret values of the source type as input values to a program, and subsets of the target type as predicates on that type (a given value satisfies a predicate if and only if it lies within the set).

Multirelations model programs in a very 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. In contrast, a multirelation M relates two values x and *post*, written $x\ M\ post$, if and only if the corresponding program can terminate with an output value that satisfies the predicate *post* given input x. Expressed in terms of a contract between two agents, this means that, given x, our agent can choose that the output value satisfies *post* if $x\ M\ post$; subsequently the other agent chooses a value within *post*. The other agent's goals may not be directly opposed to ours, but since we have no influence over the choices he makes we must be prepared for the worst possible outcome. Therefore we can only say that a specification can achieve a given postcondition from some initial state if our agent can ensure this under all circumstances.

For example, suppose that, given an input value of x, two predicates that our agent can choose between are $\{x-2, x+2\}$ and $\{x-1\}$. If he chooses the former, then the choice of output value is $x-2$ or $x+2$ and determined by the other agent. In the latter case the other agent has no choice and $x-1$ is the output value.

Not every multirelation is suitable for the specification of a contract. Suppose that $post, post' \subseteq B$ are two postconditions such that $post \subseteq post'$. Clearly, if our agent can make choices to ensure that an output value satisfies *post*, it must also satisfy $post'$. In the model proposed here, a multirelation only represents a contract between two agents if it is up-closed in the following sense:

Definition 2 (*Up-closure*). A multirelation M with source A and target B is *up-closed* if for all $x \in A$ and $X, Y \subseteq B$,

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

We will denote the types of all relations and all up-closed multirelations with source A and target B by $A \leftrightarrow B$ and $A \rightrightarrows B$ respectively.

We will use the standard lattice theoretic notation [8] to denote the *upward closure* ($\uparrow$) of a set $X \in \mathbb{P}\, B$:

$$\uparrow X = \{Y \mid X \subseteq Y\}.$$

Before moving on to some examples, we describe how the nondeterminism in an up-closed multirelation corresponds to angelic and demonic nondeterminism. The key to understanding some of the choices offered comes from considering strongest postconditions [10], since every strongest postcondition of a specification can be thought of as a choice for our agent.

Definition 3 (*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) $(\forall post' : x\ M\ post' : post' \not\subset post)$.

So, for example, let *Int* denote the type of all integers, $M : Int \rightrightarrows Int$, and suppose that for all $x : Int$ and $X : \mathbb{P}\, Int$

$$x\ M\ X \Leftrightarrow (\{x-1\} \subseteq X \vee \{x-2, x+2\} \subseteq X)$$

then our agent can choose between the two strongest postconditions $\{x-1\}$ and $\{x-2, x+2\}$ or a weaker predicate. In general, we shall refer to the set of all strongest postconditions of a multirelation M with respect to an initial state x by $sp(x, M)$.

To show how up-closed multirelations can be used to model specifications, we will now look at a few simple examples, some deterministic and some nondeterministic. In the former case, our agent is not offered any choice, and neither is the other agent, and thus the deterministic specifications are guaranteed to satisfy any postcondition which holds for an output value corresponding to the given input value.

Examples 4. (1) Each type A has identity specification $\in_A : A \rightrightarrows A$, where $\in_A$ (also denoted by id_A) represents the set membership relation on subsets of A. So, given an input value $x \in A$, the (sole) strongest postcondition it can achieve is $\{x\}$, which means that our agent can guarantee that the value x is output.

(2) 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 $\{b\}$, which ensures that our agent can guarantee that the value b is output.

(3) Let $Bool$ denote the type of booleans, and let $guard : A \to Bool$. We define the specification $(\!|guard|\!) : A \rightrightarrows A$, which asserts that a given input value satisfies $guard$ for all $x : A$, $X : \mathbb{P}\, A$ by

$$x\ (\!|guard|\!)\ X \Leftrightarrow guard\ x\ \wedge\ x \in X.$$

So, if x satisfies the guard, this specification offers a guarantee to output x, but it will not output anything if x does not satisfy the guard.

(4) Consider the multirelation $A \times \uparrow \{1, 2\}$, for some source type A. Here the strongest postcondition for any input value is $\{1, 2\}$, so our agent must choose this postcondition (or a weaker one), and this ensures that the other agent is given the choice between output values 1 and 2 (and possibly other values). This is an example of demonic nondeterminism.

(5) In contrast, now consider the multirelation $A \times (\uparrow \{1\} \cup \uparrow \{2\})$, for some source type A. Here the strongest postconditions for any input value are $\{1\}$ and $\{2\}$, so our agent can always choose between output values 1 and 2, with no choice available for the other agent. This is an example of angelic nondeterminism.

It is worth pausing at this stage to offer a note of caution about strongest postconditions. Intuitively, we might expect every non-empty multirelation to have at least one strongest postcondition, but this is not necessarily the case. For example, 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 $up\ y = \{n : Int \mid y \leq n\}$, and let $post \in sp(x, M)$. Then there exists some $y > x$ such that $up\ y \subseteq post$, but $x\ M\ up\,(y + 1)$ as well, and so $post$ does not satisfy the second condition of Definition 3. So this multirelation has no strongest postconditions because it is always possible to find a postcondition that is stronger than any other given postcondition. The choices offered to our agent in this case include those in the infinite sequence $up\,(x + 1)$, $up\,(x + 2), \ldots$ of postconditions, each stronger than its predecessor.

The definition below characterises those multirelations that do have a complete set of strongest postconditions.

Definition 5. Let $M : A \rightrightarrows B$. Then M is *representable* if, for all $x : A$, $X : \mathbb{P}\, B$

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

We will now introduce some operations on multirelations so that we can show how they can be used to model more interesting specifications.

2.3. Operations on multirelations

All of the definitions that follow are taken directly from [16].

Definition 6 (*Angelic Choice*). The *angelic choice* of two multirelations $M, N : A \rightrightarrows B$ is the union $M \cup N : A \rightrightarrows B$.

Given any input value, our agent can ensure that $M \cup N$ establishes any postcondition that could be achieved by either M or N alone, given the same input value.

Example 7. For each pair of types A, B and values $a, b : B$, we have

$$const\ a \cup const\ b = A \times \uparrow \{a\} \cup A \times \uparrow \{b\}.$$

Here, our agent can choose between the two strongest postconditions $\{a\}$ and $\{b\}$. So he can ensure that this specification establishes any postcondition that is satisfied by either of the values a or b, depending on what outcome we aim to achieve.

The dual of angelic choice is demonic choice:

Definition 8 (*Demonic Choice*). The *demonic choice* of two multirelations $M, N : A \rightrightarrows B$ is the intersection $M \cap N : A \rightrightarrows B$.

Given any input value, our agent can ensure that $M \cap N$ establishes any postcondition that could be satisfied by both M and N, given the same input value.

Example 9. For each pair of types A, B and values $a, b : B$, we have

$$\begin{aligned} const\ a \cap const\ b &= A \times \uparrow \{a\} \cap A \times \uparrow \{b\} \\ &= A \times \uparrow \{a, b\}. \end{aligned}$$

This specification will establish any postcondition that is satisfied by both a and b. The strongest postcondition it satisfies is $\{a, b\}$, showing that our agent can guarantee one of the values a or b being output, but cannot say which.
Demonic choice can also be defined using the *power union* (for more details see [16]), which is the same as intersection, and defined for all $M, N : A \rightrightarrows B$, $x : A$ and $X : \mathbb{P}\, B$ by

$$x\ (M \cup^{+} N)\ X \Leftrightarrow (\exists Y, Z : x\ M\ Y \wedge x\ N\ Z : X = Y \cup Z).$$

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

Definition 10 (*Composition*). The *composition* of two multirelations $M : A \rightrightarrows B$, $N : B \rightrightarrows C$ is denoted by $M \fatsemi N : A \rightrightarrows C$ where for all $x : A$, $X : \mathbb{P}\, C$

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

So, given input value x, our agent can only guarantee that $M \fatsemi N$ 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.

It is routine to check that the composition operator is associative, with identity id_A for each A, monotonic with respect to set inclusion, and preserves up-closure. An alternative formulation of its definition that can be useful in calculations involving representable multirelations is given below.

Lemma 11. *Let $M : A \rightrightarrows B$ be representable, and let $N : B \rightrightarrows C$, then for all $x : A$, $X : \mathbb{P}\, C$*

$$x\ (M \fatsemi N)\ X \Leftrightarrow (\exists Z : Z \in sp(x, M) : (\forall z : z \in Z : z\ N\ X)).$$

The proof of this lemma is omitted, as is that of the following lemma, which gives some distributivity properties of composition.

Lemma 12. *Let $M, N : A \rightrightarrows B$ and $P : B \rightrightarrows C$, then*

$$\begin{aligned} (M \cap N) \fatsemi P &= M \fatsemi P \cap N \fatsemi P \\ (M \cup N) \fatsemi P &= M \fatsemi P \cup N \fatsemi P. \end{aligned}$$

We are now in a position to give some more interesting examples of specifications.

Examples 13. In these examples, we will show how multirelations can be used to model part of a simplified version of the game of Nim [2]. For details of the proper game, see [5] for example. The simplified game involves two players and a pile of matches. The players take it in turns to remove either one or two matches from the pile. The loser is the player who removes the last match. So suppose that for all $n : Int$, $sub\ n : Int \rightrightarrows Int$ represents the program that subtracts the value n from its argument: for all $x : Int$, $X : \mathbb{P}\, Int$

$$x\ (sub\ n)\ X \Leftrightarrow (x - n) \in X.$$

We will model the choices of our player by angelic nondeterminism and our opponent by demonic nondeterminism, so we have

$$player : Int \rightrightarrows Int$$
$$player = sub\ 1 \cup sub\ 2$$
$$opponent : Int \rightrightarrows Int$$
$$opponent = sub\ 1 \cap sub\ 2.$$

More explicitly, for all $x : Int$, $X : \mathbb{P}\, Int$

$$x\ player\ X \Leftrightarrow \{x - 1\} \subseteq X \vee \{x - 2\} \subseteq X$$
$$x\ opponent\ X \Leftrightarrow \{x - 1, x - 2\} \subseteq X.$$

In each case the input value x is an integer which represents the number of matches in the pile at the start of that player's move.

Although Nim appears to be a symmetric game, the following examples illustrate that the guarantees for our player are very different depending on whether he moves first or second.

(1) First suppose that our player has the first move, then a round of the game is defined by

$$round : Int \rightrightarrows Int$$
$$round = player \fatsemi opponent$$

and we can calculate that for all $x : Int$, $X : \mathbb{P}\, Int$

$$\begin{array}{ll} & x\ round\ X \\ \Leftrightarrow & \{\text{Definition of } round\} \\ & x\ (player \fatsemi opponent)\ X \\ \Leftrightarrow & \{\text{Lemma 11}\} \\ & (\exists Z : Z \in sp(x, player) : (\forall z : z \in Z : z\ opponent\ X)) \\ \Leftrightarrow & \{\text{Definition of } player\} \\ & (\exists Z : Z \in \{\{x - 1\}, \{x - 2\}\} : (\forall z : z \in Z : z\ opponent\ X)) \\ \Leftrightarrow & \{\exists, \text{range}\} \\ & ((x - 1)\ opponent\ X) \vee ((x - 2)\ opponent\ X) \\ \Leftrightarrow & \{\text{Definition of } opponent\} \\ & \{x - 2, x - 3\} \subseteq X \vee \{x - 3, x - 4\} \subseteq X. \end{array}$$

We will use this multirelation to reason about the game of Nim as a whole in Section 4.

(2) Now suppose that the opponent has the first move, and so a round of the game is now defined by

$$newround : Int \rightrightarrows Int$$
$$newround = opponent \fatsemi player.$$

By a similar calculation to that for *round* we can deduce that for all $x : Int$, $X : \mathbb{P}\, Int$

$$x\ newround\ X \Leftrightarrow \{x - 2, x - 4\} \subseteq X \vee \{x - 3\} \subseteq X.$$

This is exactly what we would expect from our intuition about angelic and demonic nondeterminism, since the value input to *player* is either $x-1$ or $x-2$, and so our player can always choose to output $x-3$, or alternatively one of $x-2$ or $x-4$.

The notation M^n will be used to denote the composition of a multirelation $M : A \rightrightarrows A$ with itself n times. In addition, M^0 is defined to be id_A. Since composition is associative, this operator satisfies the following fusion law:

Lemma 14. *Suppose* $M, N, K : A \rightrightarrows A$, *and let* $n : \mathbb{N}^+$, *then*

$$M \fatsemi K = K \fatsemi N \wedge K \fatsemi N = N \Rightarrow M^n \fatsemi K = N^n.$$

The proof is omitted since it is a simple inductive argument.

Two more operators that are frequently useful for defining repetition in program specifications are transitive closure and reflexive transitive closure:

Definition 15. Let $M : A \rightrightarrows A$.

(1) The *transitive closure* of M is denoted by $M^+ : A \rightrightarrows A$, where

$$M^+ = \bigcup \{M^n | n > 0\}.$$

(2) The *reflexive transitive closure* of M is denoted by $M^* : A \rightrightarrows A$, where

$$M^* = \bigcup \{M^n | n \geq 0\}.$$

Thus the number of iterations of M performed in M^+ or M^* is chosen by the angel. Dual operators can be defined using demonic choice, but they will not be used in this paper. The closure operators satisfy many laws, including the following, which will be used in Section 4:

Lemma 16. *Suppose* $M : A \rightrightarrows A$, *then*

$$M \subseteq M^+$$
$$M^+ \subseteq M \fatsemi M^*.$$

Proof. Let $M : A \rightrightarrows A$, then $M \subseteq M^+$ by definition of M^+. Now let $n > 0$, then

$$\begin{array}{ll} & M^n \\ = & \quad \{\text{Definition of } M^n\} \\ & M \fatsemi M^{n-1} \\ \subseteq & \quad \{\text{Monotonicity of } \fatsemi\} \\ & M \fatsemi M^* \end{array}$$

and hence $M^+ \subseteq M \fatsemi M^*$, by monotonicity of $\cup$.

It can be shown by counter example that $M^+ \neq M \fatsemi M^*$: for example, taking $M =$ *opponent* from Examples 13 and using an input value of x, the multirelation $M \fatsemi M^*$ allows our agent to guarantee the predicate $\{x-2, x-3\}$ on the output, unlike M^+. Another pair of operators that are useful in program specifications are the following well-known functions for lifting relations to multirelations [2]:

Definitions 17. Let $R : A \leftrightarrow B$, then

(1) the *angelic lifting* $\langle R \rangle : A \rightrightarrows B$ is defined for all $x : A$, $X : \mathbb{P}\, B$ by

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

(2) the *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.

The difference between the two lifting operators can be illustrated by the game of Nim of Examples 13:

Example 18. Let $x, y : Int$ and define

$$move : Int \leftrightarrow Int$$
$$x\ move\ y \Leftrightarrow y = x - 1 \vee y = x - 2$$

then an alternative definition of the moves is given by

$$player = \langle move \rangle \quad \text{and} \quad opponent = [move].$$

We have now seen that the set of up-closed multirelations is closed under angelic choice, demonic choice and composition, the latter of which can be used to define the familiar transitive closure operations. They are not closed under negation or set difference, and the converse operator cannot be defined, but union and intersection can be lifted to multirelations from the target set in an obvious way. Although negation cannot be lifted directly from relations to multirelations, it does have an analogue in the *dual* operator, which can be thought of as performing an allegiance swap, so that our (angelic) agent becomes demonic, and the other (demonic) agent becomes the angel:

Definition 19 (*Dual*). The *dual* of a specification $M : A \rightrightarrows B$ is denoted by $M^\circ : A \rightrightarrows B$ where for all $x \in A$, $X \in \mathbb{P} B$,

$$x\ M^\circ\ X \Leftrightarrow \neg(x\ M\ \bar{X})$$

where $\bar{X}$ denotes the complement of X in B.

The dual of a multirelation corresponds to the conjugate operator on predicate transformers [10]. The significance of this operator will become more apparent after the discussion of correctness in the following section.

3. Correctness, refinement and feasibility

There are many different notions of refinement for ordinary relations. For example, two opposing definitions are given in [2], depending on whether the relations are being used to model angelic or demonic nondeterminism. In the former case, it is defined by subset inclusion, and in the latter by its inverse. Since the predicate transformer model of specifications is capable of expressing both kinds of nondeterminism within a single framework, it has not previously been thought necessary to endow it with more than one refinement ordering. Likewise, it has only one notion of correctness and feasibility. However, we argue that any specification containing both angelic and demonic nondeterminism must offer certain guarantees of behaviour to both the angel and the demon, just as in a game, both players have guarantees. These guarantees can be formalised as two contrasting correctness conditions, each of which has an associated refinement ordering and feasibility condition. The choice of which one to use depends on whether we see the specification from the viewpoint of the angel or the demon. Either way, this choice must be made at the start of program development and stay fixed throughout the design process.

3.1. Correctness

To demonstrate the need for a second correctness criterion, we consider again the following simple example of a multirelation. For $x : Int$ and $X : \mathbb{P} Int$, define $M : Int \rightrightarrows Int$ by

$$x\ M\ X \Leftrightarrow (\{x - 1\} \subseteq X \vee \{x - 2, x + 2\} \subseteq X).$$

The angel has many guarantees to choose from, but the two strongest ones are the postconditions $\{x - 1\}$ and $\{x - 2, x + 2\}$, which offer different guarantees on the output value. Dually, the demon can choose between the two strongest postconditions $\{x - 1, x - 2\}$ and $\{x - 1, x + 2\}$, given the same input value x.

Both these kinds of guarantee will be formalised by the notions of angelic and demonic correctness respectively, which are defined below. One requirement of such conditions is that they should be monotonic in the following sense: suppose that a specification $M : A \rightrightarrows B$ is correct with respect to an initial state $x \in A$ and postcondition $post \subseteq B$, and suppose that $post' \subseteq B$ is another postcondition with the property that $post \subseteq post'$. Then M must also be correct with respect to x and $post'$.

3.1.1. Angelic correctness

Correctness for the angel is easily defined in the multirelational model. By definition, $x\ M\ post$ if and only if the angel can make choices to ensure that the output value satisfies *post*. So we can simply say that a given specification $M : A \rightrightarrows B$ is *angelically correct* with respect to an initial state $x \in A$ and postcondition $post \subseteq B$ if

$$x\ M\ post. \tag{1}$$

The upward closure of M ensures that this condition is monotonic. It is equivalent to the traditional notion of correctness for predicate transformers in the refinement calculus [2,14].

3.1.2. Demonic correctness

Correctness for the demon is more difficult to express because of the asymmetry of the multirelational model. Conditions that can be guaranteed by the demon are given by the following definition of correctness: a specification $M : A \rightrightarrows B$ is *demonically correct* with respect to an initial state $x \in A$ and postcondition $post \subseteq B$ if

$$x\ M^{\circ}\ post. \tag{2}$$

This means that the angel can never establish the complement of *post*, thereby ensuring that the demon can establish *post*. It is easy to check that this definition of correctness satisfies the monotonicity condition imposed above.

The following example gives an illustration of both the foregoing correctness criteria.

Example 20 (*Cake Cutting Protocol*). We will look at a well-known two-participant protocol which can be described as "I cut; you choose".

A cake can be modelled as the unit interval $I = \{p : \mathbb{R} \mid 0 \leq p \leq 1\}$, and slices of cake as sub-intervals, which will be represented by pairs of bounding values $(x, y) : I \times I$, where $x \leq y$. A single cut dividing the cake into two pieces can be modelled by the ordinary relation $cut : 1 \leftrightarrow I$, defined to be

$$cut = 1 \times I.$$

Thus, given the (null) input, *cut* may output any cutting position from 0 to 1. The selection of one of the two resulting pieces of cake can be modelled by the ordinary relation $choose : I \leftrightarrow I \times I$, where if $p : I$ and $i : I \times I$

$$p\ choose\ i \Leftrightarrow i = (0, p) \vee i = (p, 1).$$

The following multirelation describes the cake cutting protocol: our agent cuts the cake first, followed by the other agent selecting one of the resulting pieces:

$$\langle cut \rangle\ {}^{\circ}_{9}\ [choose]. \tag{3}$$

The two agents may have different preferences for cake slices (for example, one may prefer chocolate sprinkles and another may like hazelnuts but not sprinkles). So two different functions are used for measuring the value of cake slices. The value function for our agent will be denoted by $value_a$, and that for the opposing agent by $value_d$. Both functions have type $I \times I \to I$, and they must be continuous and additive. This means that if the cake is divided into slices, the sum of the values of the slices is the value of the whole cake (defined to be 1). In particular, we have for each b:

$$(\forall p : p \in I : value_b\ (0, p) + value_b\ (p, 1) = 1). \tag{4}$$

The set of strongest postconditions offered to our (angelic) agent by the protocol can be calculated as

$$\{\{(0, p), (p, 1)\} \mid p \in I\}, \tag{5}$$

which describes predicates on the demon's chosen cake slice.

A common aim of a cake cutting protocol is to achieve fair division: for n participants, this means that each participant receives a slice that they consider to be worth at least $1/n$ of the whole. So for two participants this means that each agent b should be able to achieve the postcondition $fair_b$ (on the slice chosen by the demon agent), where

$$fair_d = \{i | \exists p : p \in I : (i = (0, p) \vee i = (p, 1)) \wedge value_d i \geq 0.5\}$$
$$fair_a = \{i | \exists p : p \in I : (i = (0, p) \vee i = (p, 1)) \wedge value_a i \leq 0.5\}.$$

Thus $fair_d$ expresses that the demon agent wants his own slice of cake to be at least half the whole cake with respect to $value_d$, and $fair_a$ expresses that the angelic agent wants the demon's slice of cake to be no more than half the whole with respect to $value_a$. More precisely, to achieve fair division of the cake, the multirelation $\langle cut \rangle \fatsemi [choose]$ must be angelically correct with respect to $fair_a$, and demonically correct with respect to $fair_d$. We will now show that this is indeed the case.

First, consider our agent. As $value_a$ is continuous, our agent can choose a value x such that

$$value_a\ (0, x) = value_a\ (x, 1) = 0.5 \tag{6}$$

and thus our agent can achieve the postcondition $\{(0, x), (x, 1)\}$. By upward closure and (1) the protocol is therefore angelically correct with respect to $fair_a$.

Now consider the other agent. By (2), the protocol is demonically correct with respect to $fair_d$ if and only if our agent cannot establish the postcondition $\overline{fair_d}$. By (4), one of the slices of cake must be worth at least 0.5 to the other agent, that is to say,

$$(\forall p : p \in I : value_d\ (0, p) \geq 0.5 \vee value_d\ (p, 1) \geq 0.5).$$

Hence by (5), every postcondition that our agent can establish contains a value in $fair_d$, and thus we cannot establish $\overline{fair_d}$. So the other agent is also able to ensure receiving what he considers to be a fair slice of cake.

3.2. Refinement

One requirement of a refinement relation is that it should preserve correctness, but there are now two possible interpretations of this requirement, corresponding to the two notions of correctness defined above. Both are valid in different circumstances.

The ideas that follow will be motivated using the concept of an internal agent, and an opposing external agent. Consider an arbitrary specification involving two agents. Usually, one of the agents represents an external decision maker of some kind, such as a human user of the system; we will refer to this as the *external* agent. The choices offered to this agent could include menu options, passwords, or moves in a game. The second agent in a typical specification usually represents the system designer; we will refer to this as the *internal* agent. The choices offered to this agent represent deferred design decisions, which will be made at a later stage of the system development. As such, the designer is free to implement them however he chooses. If the system is to be implemented in a deterministic language, then ultimately all such choices must be removed completely, or at least hidden so that they depend on information that is not visible to the external agent. A refinement of a specification represents a move towards such an implementation, so it must consist of a decrease in internal choice, but not of external choice, which may even be increased. Either way, a refinement must preserve any guarantee of system behaviour that has been offered to the external agent by the original specification. So if the external and internal agents are angelic and demonic respectively, refinement must preserve angelic correctness, and dually if the allegiances are reversed. We will refer to these two separate concepts as *angelic* and *demonic* refinement respectively. We do not consider any other combinations of agents and allegiances, such as when there are both internal and external demonic choices.

3.2.1. Angelic refinement

The situation where the external agent is angelic is shared by many specifications. For example, consider a customer who is contracting a programmer to deliver a piece of software to accomplish some task. If such a specification includes choices, then it is assumed that the software user will make the best (angelic) choice to achieve the desired goal, regardless of any (demonic) design choices made by the programmer. This is the interpretation of nondeterminism used in the refinement calculus: our agent is external, and the other one is internal. So a refinement

consists of a possible decrease in demonic choice and a possible increase in angelic choice, and can be defined for all $M, M' : A \rightrightarrows B$ by

$$M \sqsubseteq_A M' \equiv M \subseteq M' \qquad (\textit{angelic refinement}).$$

Clearly, this preserves the corresponding notion of angelic correctness (1). Intuitively, this definition means that a refinement can only increase the set of postconditions that our agent can choose to satisfy and hence makes it easier to satisfy the specification.

As an example, suppose that a programmer wishes to refine the cake-cutting protocol of Example 20 to a system that optimises the demon's interests, no matter what cutting position is input by the user. This could be achieved by the obvious refinement

$$\begin{array}{rcl} [\mathit{choose}] & \sqsubseteq_A & [\mathit{choosebest}] \\ \mathit{where\ choosebest}\ p & = & \mathit{if}\ \mathit{value}_d(0, p) \geq \mathit{value}_d(p, 1)\ \mathit{then}\ (0, p)\ \mathit{else}\ (p, 1). \end{array}$$

The refined system $\langle \mathit{cut} \rangle \fatsemi [\mathit{choosebest}]$ maintains (and possibly increases) all of the guarantees offered to the angel by the original specification, and also guarantees the demon half of the cake.

3.2.2. Demonic refinement

A less common situation is that where we are concerned with developing a program that is as uncooperative as possible with the user, who is potentially hostile and might be capable of preventing our program from achieving its goal. For example, consider the game of Nim (Examples 13), where the external agent is the human opponent and the internal agent is the programmer of the computerised player. Here, the goal of the programmer is to implement a winning strategy to defeat the human adversary who is free to experiment with different strategies each time he plays. Any such implementation of the player must still offer some guarantee of behaviour to the user. For instance, the player must not remove more than two matches.

Another kind of specification that must guard against harmful users might be a security application such as a password protection mechanism. Here, the programmer will make the best choices to achieve a secure system that minimises the probability of a successful breach by an attacker.

In both these examples we assume that if the programmer wishes to achieve some outcome then he will implement the angelic choice that does so regardless of the choice made by the user. So a refinement must consist of a possible decrease in angelic choice and a possible increase in demonic choice: for all $M, M' : A \rightrightarrows B$

$$M \sqsubseteq_D M' \equiv M \supseteq M' \qquad (\textit{demonic refinement}).$$

Once again, this preserves the corresponding notion of correctness (2). Here, a refinement narrows down the selection of achievable postconditions to exclude those that our agent definitely does not want to satisfy.

For example, suppose we wish to refine the cake-cutting protocol to a system that makes the best cut on behalf of our agent, and then leaves the other agent (considered here as the external user) free to choose a piece. This could be achieved by the obvious refinement

$$\langle \mathit{cut} \rangle \sqsubseteq_D \langle \mathit{cuthalf} \rangle$$

where $\mathit{cuthalf}\ 1 = x$ and x is chosen to be a position that divides the value of the cake in half for our agent, as in (6). The refined system $\langle \mathit{cuthalf} \rangle \fatsemi [\mathit{choose}]$ guarantees the angel half of the cake, whilst maintaining (and possibly increasing) the guarantees offered to the demon in the original system.

It is interesting to note that the definitions of angelic and demonic refinement given above are identical to those given for ordinary relations in [2], even though they are used here for an entirely different purpose.

3.3. Feasibility

A specification is *feasible* if it can be translated into an equivalent implementation in the target programming language. So if the language is deterministic, a feasible program must be free of all internal choice. It may still contain some external choice however, since this can be resolved through user interaction at run-time. Therefore, if the external agent is our agent, a specification is feasible if and only if it contains purely angelic choice. Dually, if the external agent is not our agent, a specification is feasible if and only if it contains purely demonic choice. Hence each of the refinement rules of Section 3.2 can be used (on its own) to transform any specification into one that is feasible. For example, both the specifications $\langle \mathit{cut} \rangle \fatsemi [\mathit{choosebest}]$ and $\langle \mathit{cuthalf} \rangle \fatsemi [\mathit{choose}]$ are feasible.

4. Applications

4.1. Nim

The version of the Nim considered here was introduced in Examples 13. It involves two players and a pile of matches. The players take it in turns to remove either one or two matches from the pile and the loser is the player who removes the last match. The model that follows differs from that in [2] in that the number of matches in the pile is allowed to become negative. So a game can be modelled as an indefinite number of rounds:

$$nim : Int \rightrightarrows Int$$
$$nim = round^*$$

where *round* was defined previously in Examples 13(1). At each round, our player can guarantee to establish one of two postconditions:

Lemma 21. *For all* $x : Int$, $n : \mathbb{N}^+$ *and* $X : \mathbb{P}\, Int$,

$$x\ round^n\ X \Leftrightarrow \{y, y+1\} \subseteq X \vee \{y-1, y\} \subseteq X$$

where $y = x - 3n$.

The proof of this lemma is in the Appendix.

If the game starts in initial state $x = 1$, then our player must remove the last match and cannot win. If $x > 1$, then our player can choose to win the game if and only if $x\ round^*\ \{2, 3\}$, since then he can remove either one or two matches, forcing the opponent to remove the last match. By Lemma 21 and the definition of *, this postcondition can be guaranteed if and only if there exists a natural number n for which y is equal to 2 or 3, which is equivalent to saying that x is positive and satisfies $x \bmod 3 \neq 1$.

Suppose we wish to refine the specification of *nim* to an implementation that achieves the postcondition $\{2, 3\}$ whenever possible. The refined program should always make the best choice for our player, assuming that the opponent's moves are made by an external user. As such, it will be a demonic refinement, so it will preserve demonic correctness, but will not maintain the angel's guarantees unless designed specifically to do so. One refinement that will achieve the postcondition $\{2, 3\}$ whenever possible is the following:

$$nim \sqsubseteq_D nim \fatsemi choosewin$$

where *choosewin* is defined using the notation for converting guards to multirelations of Examples 4:

$$winner : Int \to Bool$$
$$winner\ x = x \in \{2, 3\}$$
$$choosewin = (\!|winner|\!).$$

This refinement will guarantee a win for our player if possible, but it does not suggest a strategy for him. This can be achieved by introducing an invariant:

$$nearwin : Int \to Bool$$
$$nearwin\ x = x \bmod 3 \neq 1$$
$$inv = (\!|nearwin|\!)$$

which can be used to rewrite the refinement:

$$\begin{aligned} & nim \fatsemi choosewin \\ = \quad & \{choosewin = inv \fatsemi choosewin\} \\ & nim \fatsemi inv \fatsemi choosewin \\ = \quad & \{round \fatsemi inv = inv \fatsemi round \fatsemi inv \text{ and Lemmas 12 and 14}\} \\ & (round \fatsemi inv)^* \fatsemi choosewin \end{aligned}$$

where the two claims used to justify the above derivation are omitted for reasons of brevity.

If we also observe that *nearwin* satisfies the following property

$$\neg(nearwin\ x) \Leftrightarrow (nearwin(x-1) \wedge nearwin(x-2)) \tag{7}$$

for all $x \in \mathbb{N}$, we may calculate further that

$$\begin{array}{ll} & round \fatsemi inv \\ = & \quad \{\text{definition of } round\} \\ & player \fatsemi opponent \fatsemi inv \\ = & \quad \{\text{definition of } inv\} \\ & player \fatsemi opponent \fatsemi \langle\!|nearwin|\!\rangle \\ = & \quad \{\text{composition and (7)}\} \\ & player \fatsemi \langle\!|\neg nearwin|\!\rangle \fatsemi opponent. \end{array}$$

The justification of the final step may not be immediately obvious but can be verified easily by expanding the definition of composition.

So this refinement replaces the player's move by the new, winning move $playerwin = player \fatsemi \langle\!|\neg nearwin|\!\rangle$. Expanding this definition, we obtain that for all $x : Int$, $X : \mathbb{P}\, Int$,

$$x\ playerwin\ X \Leftrightarrow (x \bmod 3 = 0 \wedge x - 2 \in X) \vee (x \bmod 3 = 2 \wedge x - 1 \in X)$$

which is the well known strategy of making the number of matches always satisfy the invariant $x \bmod 3 = 1$.

This strategy still has a problem, which is that *playerwin* is not demonic, and hence not feasible in the sense of Section 3.3. The reason it is not demonic is that our player has not chosen a move to make in the case that $x \bmod 3 = 1$, since he cannot guarantee to win in this case. For the refinement to be feasible, it must be modified so that our player chooses a particular move for this case. Suppose for simplicity that the move that the player chooses is to remove one match. The two cases can be considered separately by observing that the invariant can be used to factor the original specification as follows:

$$\begin{array}{ll} & nim \\ = & \quad \{\langle\!|nearwin|\!\rangle \cup \langle\!|\neg nearwin|\!\rangle = id\} \\ & (\langle\!|nearwin|\!\rangle \cup \langle\!|\neg nearwin|\!\rangle) \fatsemi nim \\ = & \quad \{\text{Lemma 12}\} \\ & \langle\!|nearwin|\!\rangle \fatsemi nim \cup \langle\!|\neg nearwin|\!\rangle \fatsemi nim \\ \sqsubseteq_D & \quad \{\text{previous calculation and } player \sqsubseteq_D sub\ 1\} \\ & \langle\!|nearwin|\!\rangle \fatsemi (playerwin \fatsemi opponent)^* \fatsemi choosewin \cup \\ & \langle\!|\neg nearwin|\!\rangle \fatsemi (sub\ 1 \fatsemi opponent)^*. \end{array}$$

This is a feasible specification, with a deterministic strategy for our player which is guaranteed to win whenever possible.

4.2. Team selection

The following resource sharing protocol concerns the selection of two sports teams from a list of available players. Each team is to have n players, and there are more than $2n$ available players for the team managers to choose from.

For this team selection problem, fair division is not always possible. For example, if one player was so talented that both team managers would value any team containing that player more highly than any team without, then one of the managers will end up disappointed. We will consider Dawson's team selection protocol from [9], which does however satisfy correctness conditions that are close to fair, as we shall see. The following is a slightly simplified version of Dawson's protocol:

Step 1. The manager for the first team (Manager 1) selects two (disjoint) teams of n players.

Step 2. Manager 2 selects one of the two teams, giving the other team to Manager 1.

Step 3. Beginning with Manager 2, the managers take turns to swap some (possibly none) of the players on their own team with some from the pool of leftover players. The protocol ends if and when neither manager wishes to make any further swaps.

Given a type *Player*, a relation describing the first step of the protocol is $partition : \mathbb{P}\, Player \leftrightarrow \mathbb{P}\,\mathbb{P}\, Player \times \mathbb{P}\, Player$, where

$$ps\ partition\ (u, pool) \Leftrightarrow u = \{t, t'\} \wedge ps = t \uplus t' \uplus p \wedge \#t = \#t' = n$$

where ps represents the input list of players, and $\uplus$ is disjoint set union.

The second step of the team selection protocol can be described by the relation $select : \mathbb{P}\,\mathbb{P}\, Player \times \mathbb{P}\, Player \leftrightarrow \mathbb{P}\, Player \times \mathbb{P}\, Player \times \mathbb{P}\, Player$, where

$$(u, p)\ select\ (t_1, t_2, q) \Leftrightarrow u = \{t_1, t_2\} \wedge p = q.$$

Thus t_1 is the team for Manager 1, t_2 is the team for Manager 2, and p is the pool of unselected players.

The swapping described in the third step of the protocol can be defined with the two relations

$$(t_1, t_2, p)\ swap_1\ (u_1, u_2, q) \Leftrightarrow t_1 \uplus p = u_1 \uplus q \wedge t_2 = u_2 \wedge \#p = \#q$$
$$(t_1, t_2, p)\ swap_2\ (u_1, u_2, q) \Leftrightarrow t_2 \uplus p = u_2 \uplus q \wedge t_1 = u_1 \wedge \#p = \#q.$$

If we now deem Manager 1 (say) to be the angel, and Manager 2 to be the demon, then the description of the protocol can be translated into the multirelation P, where

$$P = \langle partition \rangle \fatsemi [select] \fatsemi ([swap_2] \fatsemi \langle swap_1 \rangle)^+.$$

Instead of starting from a specification of some kind and refining it to produce a protocol, here we have simply modelled an existing protocol by expressing it in the calculus of multirelations. As discussed above, this protocol cannot possibly guarantee a fair division for all input sets of players, but we would still like to be able to prove that certain other correctness conditions do hold.

We will use a simplistic model, assuming that each Manager i has a value function $value_i : Player \to \mathbb{R}$ and values a whole team by the sum of its players' values. Each manager may or may not know the value function of the other manager. It could then be said that the protocol achieves fair division if each manager can select a team containing half of his $2n$ most highly valued players. It turns out that this is achievable for Manager 2, and nearly achievable for Manager 1. Defining N_2 to be half the sum of the values of the best $2n$ players (with respect to $value_2$), this is the (demonic) correctness condition C_2 we shall prove that Manager 2 can satisfy:

$$C_2\ (t_1, t_2, p) \equiv \left(\sum_{q \in t_2} value_2\ q \right) \geq N_2.$$

Turning to the interests of Manager 1, it would be ill-advised to let high value players lie unchosen in the pool at the first step of the protocol, because they can be swapped into the team of Manager 2 at the first swap. The best Manager 1 can hope for is to partition the teams from the $2n$ players he considers best, making the teams as even as possible. Thus, we define N_1 to be the largest possible sum of a team composed of n of the $2n$ best players (with respect to $value_1$), such that the other n players have a total value at least as high. Thus this is the (angelic) correctness condition that we will use for Manager 1:

$$C_1\ (t_1, t_2, p) \equiv \left(\sum_{q \in t_1} value_1\ q \right) \geq N_1.$$

In order to show that the protocol P is correct with respect to C_1 it is sufficient to show, for some other multirelation P_1 such that $P_1 \sqsubseteq_A P$, that P_1 is correct with respect to C_1, because correctness is preserved by refinement. We can

calculate

$$
\begin{array}{ll}
 & P \\
= & \quad \{\text{Definition}\} \\
 & \langle partition \rangle \fatsemi [select] \fatsemi ([swap_2] \fatsemi \langle swap_1 \rangle)^+ \\
\supseteq & \quad \{\text{Transitive closure, Lemma 16}\} \\
 & \langle partition \rangle \fatsemi [select] \fatsemi [swap_2] \fatsemi \langle swap_1 \rangle.
\end{array}
$$

Denoting the resulting multirelation by P_1, it is straightforward to see that P_1 satisfies C_1: if Manager 1 partitions the input set of players ps to produce two teams containing his $2n$ best players that are as evenly matched as possible, then by definition each team will have total value (with respect to $value_1$) at least N_1, so that $ps(\langle partition \rangle \fatsemi [select])C_1$. Subsequently, $[swap_2]$ cannot change t_1, and $\langle swap_1 \rangle$ has the option of leaving t_1 unchanged, so Manager 1 can guarantee to be able to satisfy condition C_1.

For Manager 2, we must look at demonic correctness, and we now perform a similar calculation, this time with respect to demonic refinement:

$$
\begin{array}{ll}
 & P \\
= & \quad \{\text{Definition}\} \\
 & \langle partition \rangle \fatsemi [select] \fatsemi ([swap_2] \fatsemi \langle swap_1 \rangle)^+ \\
\subseteq & \quad \{\text{Closure, Lemma 16}\} \\
 & \langle partition \rangle \fatsemi [select] \fatsemi [swap_2] \fatsemi \langle swap_1 \rangle \fatsemi ([swap_2] \fatsemi \langle swap_1 \rangle)^* \\
= & \quad \{\text{Lifting distributes through composition}\} \\
 & \langle partition \rangle \fatsemi [select \,; swap_2] \fatsemi \langle swap_1 \rangle \fatsemi ([swap_2] \fatsemi \langle swap_1 \rangle)^*.
\end{array}
$$

Denoting the result by P_2, we thus have that $P_2 \sqsubseteq_D P$, and so if we can prove that Manager 2 can guarantee P_2 to satisfy C_2, then that must also be the case for P. The correctness of P_2 is now easier to demonstrate, by considering demonic guarantees for $[select \,; swap_2]$.

At the first step, Manager 1 (the angel) has complete control over the partitioning. To complete the subsequent step, $select \,; swap_2$, Manager 2 should first note where, within the suggested teams and pool, his $2n$ most highly valued players are. Suppose that this set of players is represented by the disjoint union

$$b_1 \uplus bp_1 \uplus b_2 \uplus bp_2$$

for player sets satisfying

$$
\begin{array}{l}
b_1 \subseteq t \wedge b_2 \subseteq t' \wedge bp_1 \uplus bp_2 \subseteq p \\
\wedge \#b_1 + \#bp_1 = \#b_2 + \#bp_2 = n
\end{array}
$$

for teams and pool $(\{t, t'\}, p)$.

Manager 2 should then choose the most valuable of $b_1 \uplus bp_1$ and $b_2 \uplus bp_2$ for his team (t_2) during his execution of $[select; swap_2]$. As the value of the players in $b_1 \uplus bp_1 \uplus b_2 \uplus bp_2$ is $2N_2$, this offers Manager 2 a guarantee of being able to satisfy C_2, and therefore $\langle partition \rangle \fatsemi [select; swap_2]$ satisfies the demonic correctness condition C_2. Subsequently, for $\langle swap_1 \rangle \fatsemi ([swap_2] \fatsemi \langle swap_1 \rangle)^*$, a similar argument to that for Manager 1 can be used, as Manager 1 cannot guarantee to change t_2 subsequently and Manager 2 always has the option to keep t_2 unchanged. Thus Manager 2 can guarantee to satisfy C_2, and the protocol is demonically correct for C_2.

5. Categories, multirelations and predicate transformers

This section describes the bijective correspondence between multirelations and predicate transformers. The relationship between multirelations and monotonic functions has already been demonstrated in [16], in the context of boolean algebras. We present this result in the setting of category theory, because it provides a foundation for further work (not presented in this paper) that translates some well-known properties of coinductive datatypes in the category of predicate transformers to that of multirelations.

It is beyond the scope of this paper to explain the details of categorical concepts. For example, see [4] for an introduction to category theory.

5.1. Order-enriched categories

We will model multirelations and predicate transformers 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 up-closed multirelations **Mul** is defined as follows. The objects of **Mul** are sets, arrows are up-closed multirelations, composition is defined by $\fatsemi$, and the identity arrow for each set A is the set membership relation $\in_A$. The order on arrows in **Mul** is a refinement relation which can be instantiated either to subset inclusion or its inverse.

The order enriched category of monotonic predicate transformers **Tran** also has sets as its objects, but its arrows are monotonic functions between powersets.

So the set of all arrows with source A and target B is defined by

$$\{p : \mathbb{P}A \to \mathbb{P}B \mid \forall X, Y : X \subseteq Y : p\ X \subseteq p\ Y\}.$$

Composition and identity are the ordinary functional composition and identity. The order on arrows in **Tran** is a refinement relation, which can be instantiated either to the pointwise extension of the inclusion ordering on subsets or its reverse.

5.2. An isomorphism between Mul and Tran

The following result is a simple extension to categories of the isomorphism given in [16], restricted to the special case of powerset boolean algebras.

Theorem 22. *The categories* **Mul** *and* $\mathbf{Tran}^{op}$ *are order isomorphic.*

Proof sketch. It is sufficient to show that the functions used to define the isomorphism in [16] are both contravariant monotonic functors. The functions are defined as follows: for each $M : A \rightrightarrows B$ in **Mul**, $\phi_M : B \to A$ is the arrow in **Tran** defined for all $X : \mathbb{P}B$ by

$$\phi_M X = \{x : A \mid x\ M\ X\}$$

and for each $p : B \to A$ in **Tran**, $\theta_p : A \rightrightarrows B$ is defined for all $x : A,\ X : \mathbb{P}B$ by

$$x\ \theta_p\ X \Leftrightarrow x \in p\ X.$$

It is routine to check that these are indeed contravariant monotonic functors. □

When presented in this simple context of powerset boolean algebras, this isomorphism seems rather obvious. This point is also made in [11], together with the interesting observation that it can be considered as a special case of the well-known isomorphism between domains $(X \times Y) \to Z$ and $X \to (Y \to Z)$.

It is also arguable that these two alternative views of a specification are not necessarily very different from each other. For example, consider the specifications of *player* and *opponent* in Examples 13. Converting these multirelations to predicate transformers ϕ_p and ϕ_o respectively gives, for all $X : \mathbb{P}\,Int$,

$$\phi_p X = \{y : Int \mid \{y-1\} \subseteq X \vee \{y-2\} \subseteq X\}$$
$$\phi_o X = \{y : Int \mid \{y-1, y-2\} \subseteq X\}.$$

Although these specifications look rather similar to their multirelational counterparts, there is a difference when forming their composition. It was a trivial matter to calculate the value of $round = player \fatsemi opponent$ in Examples 13, but the corresponding calculation for predicate transformers leads to the expression

$$\begin{array}{ll} & \phi_o(\phi_p X) \\ = & \\ & \{z : Int \mid \{z-1, z-2\} \\ & \qquad \subseteq \{y : Int \mid \{y-1\} \subseteq X \vee \{y-2\} \subseteq X\}\} \end{array}$$

which does not simplify quite so immediately.

This illustrates that there are times when calculations are easier to do on multirelations than predicate transformers, but of course there are other times when the predicate transformer model is the easier one to use, for example composition of functions in the predicate transformer model is simpler than multirelational composition. We argue that multirelations can be a useful alternative, depending on the calculation.

6. Conclusion

This paper consists of some preliminary steps towards the construction of a new calculus for modelling nondeterministic specifications as multirelations. The model is appealing because it combines some of the merits of the existing models of relations and predicate transformers. Like relations, multirelations model specifications from initial to final state, and like predicate transformers, they can model two kinds of nondeterminism within a single framework. This is achieved by mapping input values to postconditions rather than output values. So the multirelational model is more expressive than that of relations, but not quite as transparent. Therefore it may be useful for descriptions of systems that cannot be expressed more simply using ordinary relations.

The two forms of nondeterminism in a multirelational specification represent two potentially conflicting interests that can be viewed in a number of different ways. Typically, we think of one of these interests as representing an angel and the other as a demon, where the angel is on our side but the demon is not. Not all systems can be uniquely categorised in this way however. For example, in the case of a game or resource sharing protocol involving two participants with opposing interests, either participant could be modelled as the angel, depending on the circumstances. Consequently, it is sometimes useful to associate the different interests in a specification with different agents, without necessarily labelling them as angelic or demonic. The number of agents involved in such a specification is not limited to two, but it can only be modelled as a multirelation if the allegiance of each agent is known, with the choices made by our allies treated as angelic, and those of our enemies as demonic. This observation is made in [2], where the choices of multiple agents are first represented by indexed operators, but later categorised as simply angelic or demonic.

The concept of agents is also useful for distinguishing between the internal and external choices in a specification. We use this idea to motivate the need for a new notion of refinement: we claim that a refinement is equivalent to a reduction in internal choice or an increase in external choice. This leads to two opposing definitions of angelic and demonic refinement that are identical to those that have been suggested for relations [2]. The choice of which one to use must be made at the start of program development and stay fixed throughout the design process. We have provided corresponding new notions of correctness and feasibility, and demonstrated that refinement is a correctness preserving operation. Feasibility is interpreted to mean the absence of internal choice, since external choices can remain in the program to be executed interactively at run-time.

Although none of the calculus presented so far is tied exclusively to the derivation of functional programs, this is one of the areas we plan to develop in the future. The central result that we wish to exploit concerns the extension of initial algebras from relations to predicate transformers [13]. This result is equally true of multirelations, because of the isomorphism of categories presented in Section 5. The reason that we feel that it is important is that its counterpart for total functions and relations is one of the primary building blocks of the relational calculus of [7]: it provides the key to the extension of the fold operator on regular datatypes of functional programming to relations. The corresponding extension of fold to multirelations has a very familiar form, unlike its predicate transformer equivalent. Associated with this definition is a universal property and fusion law that can be used in the calculation of programs. More details of this will appear in a forthcoming paper.

Some of the areas that we are currently exploring as applications of this theory are security protocols, resource-sharing protocols and game theoretic mechanisms such as voting procedures. We hope to use multirelations to reason about such systems and derive implementations that meet various requirements. In the case of a game, this could be interpreted as the implementation of a winning strategy, and in the case of resource-sharing protocols, it could be a protocol that achieves fair division. In practice, few games actually have a winning strategy, but there are weaker game theoretic concepts that are desirable in an implementation. For example, Pauly [15] proposes the existence of a subgame equilibrium as an alternative notion of correctness. Similarly, there are a number of different criteria that are desirable in a resource-sharing protocol. Further connections to game theory may emerge by examining the link between multirelations and game semantics [1].

Acknowledgements

We would like to thank the anonymous referees, whose thorough reports greatly improved the presentation of this paper.

Appendix

Proof of Lemma 21. The proof is by induction on n.
Base case. By the definition of *round*, the result holds if $n = 1$.
Inductive step. Suppose that

$$x\ round^n\ X \Leftrightarrow \{y, y+1\} \subseteq X \vee \{y-1, y\} \subseteq X$$

where $y = x - 3n$.

$$\begin{array}{ll}
& x\ round^{n+1}\ X \\
\Leftrightarrow & \quad \{\text{Definition of } round^{n+1}\} \\
& x\ (round^n \fatsemi round)\ X \\
\Leftrightarrow & \quad \{\text{Lemma 11}\} \\
& (\exists Z : Z \in sp(x, round^n) : (\forall z : z \in Z : z\ round\ X)) \\
\Leftrightarrow & \quad \{\text{Inductive hypothesis}\} \\
& (\exists Z : Z \in \{\{y, y+1\}, \{y-1, y\}\} : (\forall z : z \in Z : z\ round\ X)) \\
\Leftrightarrow & \quad \{\text{logic}\} \\
& ((y-1)\ round\ X \wedge y\ round\ X) \vee (y\ round\ X \wedge (y+1)\ round\ X) \\
\Leftrightarrow & \quad \{\text{Distributive law, commutativity}\} \\
& y\ round\ X \wedge ((y-1)\ round\ X \vee (y+1)\ round\ X) \\
\Leftrightarrow & \quad \{\text{Definition of } round\} \\
& \begin{array}{ll}
(\{y-2, y-3\} \subseteq X \vee \{y-3, y-4\} \subseteq X) & \wedge \\
(\{y-1, y-2\} \subseteq X \vee \{y-2, y-3\} \subseteq X & \vee \\
\{y-3, y-4\} \subseteq X \vee \{y-4, y-5\} \subseteq X) &
\end{array} \\
\Leftrightarrow & \quad \{\text{Absorption}\} \\
& (\{y-2, y-3\} \subseteq X \vee \{y-3, y-4\} \subseteq X) \\
\Leftrightarrow & \quad \{\text{Let } w = x - 3(n+1)\} \\
& (\{w, w+1\} \subseteq X \vee \{w-1, w\} \subseteq X). \quad \square
\end{array}$$

References

[1] S. Abramsky, G. McCusker, Game Semantics, in: H. Schwichtenberg, U. Berger (Eds.), Computational Logic: Proceedings of the 1997 Marktoberdorf Summer School, Springer-Verlag, 1999, pp. 1–56.
[2] R.J.R. Back, J. von Wright, Refinement Calculus: A Systematic Introduction, in: Graduate Texts in Computer Science, Springer-Verlag, New York, 1998.
[3] R. Backhouse, P. Hoogendijk, Elements of a relational theory of datatypes, in: H. Partsch, B. Mller, S. Schumann (Eds.), FIP TC2/WG2.1 State-of-the-Art Report on Formal Program Development, in: LNCS, vol. 755, Springer-Verlag, 1993, pp. 7–42.
[4] M. Barr, C. Wells, Category Theory for Computing Science, Prentice-Hall, 1990.
[5] E.R. Berlekamp, J.H. Conway, R.K. Guy, Winning Ways for Your Mathematical Plays, second ed., A.K. Peters Ltd, 2001.
[6] R.S. Bird, A calculus of functions for program derivation, in: D.A. Turner (Ed.), Research Topics in Functional Programming, in: University of Texas at Austin Year of Programming Series, Addison-Wesley, 1990, pp. 287–308.
[7] R.S. Bird, O. de Moor, Algebra of Programming, Prentice Hall, 1997.
[8] B.A. Davey, H.A. Priestley, Introduction to Lattices and Order, second ed., Cambridge University Press, 2002.
[9] C. Bryan Dawson, A better draft: Fair division of the talent pool, College Mathematics Journal 28 (2) (1997) 82–88.
[10] E.W. Dijkstra, C.S. Scholten, Predicate Calculus and Program Semantics, Springer-Verlag, 1990.

[11] W.H. Hesselink, Multirelations are predicate transformers, 2004, Available from http://www.cs.rug.nl/~wim/pub/whh318.pdf.gz (Unpublished manuscript).
[12] N. Lynch, F. Vaandrager, Forward and backward simulations for timing-based systems, in: J.W. de Bakker, W.P. de Roever, C. Huizing, G. Rozenberg (Eds.), Proceedings of Real-Time: Theory in Practice (REX Workshop, Mook, The Netherlands, June 1991), in: Lecture Notes in Computer Science, vol. 600, Springer-Verlag, 1992, pp. 397–446.
[13] O. de Moor, Inductive data types for predicate transformers, Information Processing Letters 43 (3) (1992) 113–117.
[14] C.C. Morgan, Programming from Specifications, Prentice Hall, 1998.
[15] M. Pauly, Programming and verifying subgame perfect mechanisms, 2002. Available from http://www.csc.liv.ac.uk/~pauly/.
[16] I. Rewitzky, Binary multirelations, in: H. de Swart, E. Orlowska, G. Schmidt, M. Roubens (Eds.), Theory and Application of Relational Structures as Knowledge Instruments, in: Lecture Notes in Computer Science, vol. 2929, 2003, pp. 259–274.
[17] G. Schmidt, T. Ströhlein, Relationen und Grafen, Springer-Verlag, 1988.
[18] N.T.E. Ward, A refinement calculus for nondeterministic expressions, 1994. Available from www.dstc.monash.edu.au/staff/nigel-ward/nwthesis.pdf.