Angelic Nondeterminism and Unifying Theories of Programming

Ana Cavalcanti and Jim Woodcock

University of York
Department of Computer Science
York, UK
{Ana.Cavalcanti,jim}@cs.york.ac.uk

Abstract. Hoare and He's unifying theories of programming (UTP) is a model of alphabetised relations expressed as predicates, which supports development in several programming paradigms. The aim is the unification of languages and techniques, so that we can benefit from results in different contexts. In this paper, we investigate the integration of angelic nondeterminism in the UTP; we propose the unification of a model of binary multirelations, which sis isomorphic to the monotonic predicate transformers model and can express angelic and demonic nondeterminism.

Keywords. semantics, refinement, relations, predicate transformers.

1 Introduction

Angelic nondeterminism is a specification and programming concept that is typically available in unified languages of refinement calculi [18, 4], and in concurrent constraint programming languages [15]. In program development techniques, it is reflected in choice constructs in which the choice is not arbitrary, but made to guarantee success, if possible. In programming languages, it is reflected in the use of backtracking in exhaustive searches. The work in [16] explores angelic nondeterminism in a language for definition of tactics of proofs.

In contrast, demonic nondeterminism is related to an arbitrary choice construct that provides no guarantees; success is still a possibility, but it does not influence the choice. Demonic choice is commonly used to model abstraction and information hiding; in this case, choice is used in a specification to explicitly indicate options that are left open to the programmer.

In [11], Gardiner and Morgan identify angelic choice with the least upper bound in the lattice of monotonic predicate transformers. In [19], they use this construct to define logical constants, which are pervasive in refinement techniques, and are sometimes named logical, auxiliary, or angelic variables. The logical constants play a fundamental rôle in the formalisation of data refinement of recursive programs, and, more importantly, they are used in calculational simulation rules for specification statements and guarded commands.

In [18] Morgan proposes an algebraic approach to refinement. In that work, logical constants are at the heart of the formalisation of initial variables, which are used in specification statements: they appear in postconditions to refer to values of variables before the execution of the program. Logical constants are also central to the stepwise calculational development of sequences and loops.

Back and von Wright's work on refinement [4] has also explored the use of angelic nondeterminism. They have extensively studied the set of monotonic predicate transformers as a lattice with the refinement ordering. They have identified interesting sublattices, in which choice can be either angelic or demonic, and a complete

base language, which can describe any monotonic predicate transformer [1, 2]. More recently, they have suggested the use of angelic nondeterminism to model user interactions with a system, and game-like situations.

Morgan's refinement calculus has been adapted to handle Z specifications; the resulting calculus is called ZRC [7]. It is incorporated in *Circus* [21], a combination of Z and CSP that supports refinement of state-rich, reactive programs. The design of *Circus* follows the trend to combine notations; it has been successfully applied in case studies, and has a refinement technique that supports decomposition of the state and behaviour of centralised systems [5].

Departing from standard work in refinement calculi, the semantics of *Circus* is based on Hoare and He's unifying theories of programming (UTP) [14, 22]. This is a predicate-based relational model for programming that links constructs in several programming paradigms: imperative, concurrent, logical, and others. By providing a framework for the study of state and reactive aspects of a program, the UTP has proved to be very adequate as a basis for the *Circus* model, and for several of its extensions. Nevertheless, logical constants and, more generally, angelic nondeterminism are not considered. Since we adopt Morgan's calculational refinement style, we have pursued the possibility of modelling angelic nondeterminism in the UTP.

Angelic nondeterminism has been extensively studied using weakest precondition semantics. There are results on the relationship between relational and predicate transformer models in which relations are sets of pairs of states and predicates are sets of states [12, 6]. These results establish that the UTP relational model cannot capture angelic and demonic nondeterminism.

In this paper, firstly, we consider a set-based relational model for the UTP. Secondly, we propose a predicate transformer model; conjunctive predicate transformers correspond to the set-based relations, and therefore to UTP relations. These models clarify some aspects of the UTP, and provide guidance on the use of the model of binary multirelations introduced in [20] for the UTP. Based on this model, we propose a UTP theory for angelic nondeterminism.

In the next section, we present an overview of the unifying theories of programming. In Section 3, we consider a set-based relational model and a predicate transformer model for the UTP. In Section 4 we enrich the UTP with a theory to cope with angelic and demonic nondeterminism. Finally, in Section 5 we present our conclusions and directions for future work.

2 Unifying theories of programming

The objective of Hoare and He's unifying theories of programming is to study and compare programming paradigms. The main concern is with program development; using the framework of the UTP, it should be possible to take advantage of different techniques and approaches whenever convenient.

In the general theory of relations of the UTP, a relation is a pair $(\alpha P, P)$, where αP is a set of names of observational variables, and P is a predicate. The set of variables is the alphabet of the relation; it contains both the set $in\alpha P$ of undashed names of the observational variables, and the set $out\alpha P$ of dashed names. The free variables of P must be contained in αP.

Each observational variable records information relevant to characterise the behaviour of a program. For example, program variables are observational variables; the model of an assignment $x := e$, if the program variables are x, y, and z, is as follows. The undecorated name of a variable refers to its value before the execution of the program, and the dashed name refers to its value in a subsequent observation.

$$x := e \ \widehat{=} \ (x' = e \land y' = y \land z' = z)$$

The alphabet is $\{ x, y, z, x', y', z' \}$. The assignment sets the final value of x, which

is represented by x', to e; all the other variables are unchanged.

The program $\mathit{II} \;\widehat{=}\; (v' = v)$ skips: it does not change the observational variables v. We write $v' = v$ as an abbreviation for a conjunction of equalities that state that the final value of each variable is equal to its initial value.

A sequence $P \,;\, Q$ is defined simply as relational composition, if, for each dashed variable in the alphabet of P, the undashed variable is in the alphabet of Q. The set $in\alpha' Q$ is obtained by dashing all variables in αQ.

$$P(v') \,;\, Q(v) \;\widehat{=}\; \exists\, v_0 \bullet P(v_0) \wedge Q(v_0) \quad \text{provided } out\alpha P = in\alpha' Q = \{\, v' \,\}$$

The notation $P(v')$ emphasises that P may have free occurrences of observational variables v'; the later reference to $P(v_0)$ refers to the predicate obtained by substituting v_0 for the free occurrences of v' in P. Similarly, for $Q(v)$ and $Q(v_0)$.

The nondeterministic choice $P \sqcap Q \;\widehat{=}\; P \vee Q$ of relations P and Q with the same alphabet is demonic. It behaves like either P or Q.

The set of relations with a particular alphabet is a complete lattice, with order $\Leftarrow$; this is the refinement ordering in this setting. More formally, the program denoted by P is refined by that denoted by Q when $[Q \Rightarrow P]$. As a matter of fact, P and Q can be either programs (assignments, sequence, choices, and others) or any relation used to specify a program; they are all relations. The square brackets denote universal quantification over all the alphabet.

In contrast with the other operators, the least upper bound $\sqcap S$ of a set S of relations is defined algebraically: $[P \Leftarrow \sqcap S] \;\widehat{=}\; ([P \Leftarrow X] \text{ for all } X \text{ in } S)$. The bottom of this lattice is the program $\bot \;\widehat{=}\; \mathbf{true}$, which is called abort. Incidentally, the top element is **false**; it is written $\top$ and called miracle.

Recursion is modelled using least fixed points. If $F(X)$ is a relation, in which X is used as a recursion variable, the recursive program is written $\mu\; X \bullet F(X)$. This is the least fixed point of the function F.

Hoare and He point out what they regard an infelicity. The recursive program $\mu\, X \bullet X$ is supposed to model an infinite loop; it is equivalent to $\bot$ or **true**. Nonetheless, the sequence $(\mu\, X \bullet X) \,;\, x' = 3$ is equivalent to $x' = 3$, even though it should not be possible to recover from a program that does not terminate.

The solution proposed by Hoare and He is the introduction of an extra boolean observational variable ok to record termination. If ok has value *true*, it means that the program has started; if ok' has value *true*, then the program has terminated. In this new theory, relations take the form of designs $P \vdash Q$.

$$(P \vdash Q) \;\widehat{=}\; (ok \wedge P) \Rightarrow (ok' \wedge Q)$$

The predicates P and Q are the program's pre and postcondition. If the design has started and P holds, then it terminates and establishes Q.

In this new theory, assignment and skip are redefined. Below, y and y' stand for the observational variables other than x and x'.

$$x := e \;\widehat{=}\; \mathbf{true} \vdash x' = e \wedge y' = y \qquad\qquad \mathit{II} \;\widehat{=}\; \mathbf{true} \vdash v' = v$$

The new definitions use designs to take ok and ok' into account.

Four healthiness conditions on relations R are regarded of interest in the theory of designs; they are summarised in Table 1. Healthiness condition H1 states that any restrictions on the behaviour of R only need to hold if it has started. The second healthiness condition states that R cannot require non-termination: if it holds when ok' is *false*, then it also holds when ok' is *true*. Together, H1 and H2 characterise the designs: a predicate is H1 and H2 if and only if it is a design.

The healthiness conditions H3 and H4 are expressed as equations between programming constructs. Results presented in [14] clarify that H3 designs can be expressed using preconditions that do not refer to dashed observational variables, and that H4 designs model feasible or implementable programs.

H1 $R = (ok \Rightarrow R)$	No predictions before startup
H2 $[R[false/ok'] \Rightarrow R[true/ok']]$	Non-termination is not required
H3 $R = R \,;\, \mathit{II}$	Preconditions do not use dashes
H4 $R \,;\, \mathbf{true} = \mathbf{true}$	Feasibility

Table 1. UTP Healthiness conditions

Designs form a UTP theory that is characterised by an alphabet that includes ok and ok', and by the healthiness conditions H1 and H2. For reactive programs, for instance, we have a theory of relations whose alphabets include six other observational variables, and that satisfy two other healthiness conditions. Alphabets and healthiness conditions are the basis to compare and combine different theories. Later on, we present a theory for angelic (and demonic) nondeterminism; beforehand, we study set-based models for the UTP.

3 Set-based models

In this section, we consider two set-based models for the UTP: relations, characterised by sets of pairs, and predicates transformers, with predicates characterised by sets. These models further clarify the role of healthiness conditions and the internalized model of nontermination based on ok and ok'. Most importantly, however, they provide guidance in the definition of a UTP theory based on binary multirelations. It is this theory that can capture both angelic and demonic nondeterminism.

3.1 Relations

The set-based relational model is that of sets of pairs of states. A state associates names (of observational variables) to their values. The set S_A of all states on an alphabet A contains the records with a component for each variable in A. Each such state is an observation of the behaviour of a program. A relation, like a UTP predicate, is a pair $(\alpha R, R)$, where αR is the alphabet, and R is a relation between the elements of $S_{in\alpha R}$ and $S_{out\alpha R}$. Such a relation models a program by associating an observation of an initial state with an observation of a possible final state.

The model for abort is the universal relation: $\mathbb{P}\, S_{in\alpha} \times S_{out\alpha}$; when the predicate P (or relation R) is not relevant, instead of writing $in\alpha P$ (or $in\alpha R$) and $out\alpha P$ (or $out\alpha R$), we simply write $in\alpha$ and $out\alpha$. Partiality models miracles. If a state is not in the domain of the relation, then it is miraculous at that state: it can achieve any required result. In particular, the model of miracle is the empty relation.

It is not difficult to see that the first general predicate-based theory of the UTP is isomorphic to this set-based model. A simple proof is presented in [8]; it is based on the functions $p2sb$ and $sb2p$.

Definition 1.

$$p2sb.(\alpha P, P) \;\widehat{=}\; (\alpha P, \{\, s : S_{in\alpha P};\; s' : S_{out\alpha P} \mid P[s, s'/in\alpha P, out\alpha P] \,\})$$

$$sb2p.(\alpha R, R) \mathrel{\widehat{=}} (\alpha R, \exists\, s : S_{in\alpha R}, s' : S_{out\alpha R} \bullet (s, s') \in R \wedge (\bigwedge x : in\alpha R \bullet x = s.x) \wedge (\bigwedge x : out\alpha R \bullet x = s'.x))$$

The first, $p2sb$, transforms a UTP relation into a set-based relation; the second, $sb2p$ is its inverse: it transforms a set-based relation into a UTP relation. Both $p2sb$ and $sb2p$ do not change the alphabet of the relations. A similar set-based relational model is used by Hoare and He when they discuss denotational semantics.

$$\mathsf{SBH1}\ \forall\, s, s' \mid s.ok = \mathit{false} \bullet (s, s') \in R$$

$$\mathsf{SBH2}\ \forall\, s, s' \mid (s, s') \in R \land s'.ok' = \mathit{false} \bullet (s, s' \oplus \{ok' \mapsto \mathit{true}\}) \in R$$

$$\mathsf{SBH3}\ \forall\, s \mid (\exists\, s' \bullet s'.ok' = \mathit{false} \land (s, s') \in R) \bullet \forall\, s' \bullet (s, s') \in R$$

Table 2. Set-based healthiness conditions

The set-based relation defined by $p2sb$ for a predicative relation P is formed by pairs of states s and s' such that P holds when the observational variables take the values associated to them by s and s'. The predicate $P[s/A]$ is obtained by replacing x with $s.x$, for all x in A.

The predicate defined by $sb2p$ for a relation R is an existential quantification over pairs of states s and s' in R. For each pair, a conjunction of equalities requires that each observational variable takes the value in the corresponding initial or final state. Since alphabets are finite, the conjunction is finite.

Standard work on relational semantics [13] singles out a special state to indicate non-termination; this is not the case in our model. If an initial state is associated with all possible final states, then we cannot say whether the final state is simply arbitrary or we have a possibility of non-termination. In standard relational semantics, the model for abort that we presented above is actually the model for a program that always terminates, but whose final state is arbitrary.

The isomorphism confirms that the general UTP model is not able to capture non-termination. Hoare and He pointed out a paradox in the fact that, if the alphabet is $\{ x, x' \}$, then $(\mu\, X \bullet X);\ x := 3$ is equivalent to $x := 3$. This is not really a paradox: the bottom of the lattice $\bot$ is not an aborting program, but the program that terminates and gives an arbitrary value to x. If, in sequence, we assign 3 to x, then the arbitrariness is irrelevant. Their model is sensible, for terminating programs. (Their attempt to solve the supposed paradox by giving a strongest fixed point semantics to recursion was always doomed to fail.)

For designs, the alphabet includes ok and ok'; therefore, these variables are also part of the alphabet of the corresponding set-based relations. In Table 2, we present healthiness conditions; we omit the obvious types of s and s'.

The healthiness condition SBH1 requires that all states s for which $s.ok$ is false are in the domain of R, and are related to all possible final states. This means that a state in which the program has not started is not miraculous and leads to no controlled behaviour. In relations that are SBH2-healthy, if a state s is related to a state s' for which $s'.ok'$ is false, then s is also related to $s' \oplus \{ok' \mapsto \mathit{true}\}$. This is the same state as s', except that the value of ok' is *true*. This means that if it is possible not to terminate from s, it is also possible to terminate. Its behaviour, however, may not be completely arbitrary: it is not required that R relates s to all possible final states; this is what is required by SBH3.

The theorem below, proved in [8], establishes that H1, H2, and H3 correspond to SBH1, SBH2, and SBH3.

Theorem 1. *For every UTP relation $(\alpha P, P)$ that satisfies H1, $p2sb.(\alpha P, P)$ satisfies SBH1. Conversely, for every set-based relation $(\alpha R, R)$ that satisfies SBH1, $sb2p.(\alpha R, R)$ satisfies H1. The same holds for H2 and SBH2, and for H3 and SBH3.*

We believe that it is not difficult to observe that SBH3 relations are necessarily SBH2. If the initial state s is related to all possible final states, then it is also related to $s' \oplus \{ok' \mapsto \mathit{true}\}$. This rather obvious result seems to be not so clear in the predicate setting. It means that, at least for the purpose of the study of total correctness of sequential programs, Hoare and He did not need to consider four healthiness conditions, but only three of them: H1, H3, and H4. It turns out,

however, that non-H3 designs are important for the modelling of more sophisticated programming paradigms like CSP, for instance.

The healthiness condition H4 requires feasibility. It is not relevant for us, as miracles are an important part of Morgan's refinement calculus and ZRC.

3.2 Predicate transformers

In the model of predicate transformers, we regard predicates as sets of states. The model is composed of pairs $(\alpha PT, PT)$, where αPT is the alphabet of the transformer, and PT is a total monotonic function from $\mathbb{P}\, S_{out\alpha PT}$ to $\mathbb{P}\, S_{in\alpha PT}$. A program is modelled by its weakest precondition transformer [9].

Isomorphisms between predicate transformers and set-based relational models have already been studied [12]. The isomorphism that we propose here is similar to that in [6]. We define functions $sb2pt$ and $pt2sb$; the first transforms a set-based relation into a weakest precondition, and the second transforms a weakest precondition back into a set-based relation. For simplicity, we ignore alphabets, which, strictly speaking, should be maintained by both functions.

Definition 2. $sb2pt.R.\psi \mathrel{\widehat{=}} \overline{\mathrm{dom}(R \rhd \psi)}$

$$pt2sb.PT = \{\, s : S_{in\alpha PT};\ s' : S_{out\alpha PT} \mid s \in \overline{PT.\overline{\{\, s' \,\}}} \,\}$$

In the definition of $sb2pt$, ψ is a postcondition, or rather, a set of states, which is given as argument to the transformer $sb2pt.R$. The relation $R \rhd \psi$ models all executions of R that do not lead to a final state that satisfies ψ; the operator $\_ \rhd \_$ is range subtraction. In $\mathrm{dom}(R \rhd \psi)$, we have all initial states in which it is possible not to achieve ψ. The complement contains all initial states in which we are guaranteed to reach a final state that satisfies ψ: the required weakest precondition.

The relation $pt2sb.PT$ associates an initial state s to a final state s' if s is not in the weakest precondition that guarantees that PT does not establish s'. Since it is not guaranteed that PT will not establish s', then it is possible that it will. The possibility is captured in the relation.

Since the general set-based relations can only model terminating programs, we cannot expect an isomorphism between them and the whole set of predicate transformers. In fact, we prove that they are isomorphic to the set of universally conjunctive predicate transformers PT: those that satisfy the property below.

$$PT.(\bigcap\{\, i \bullet \psi_i \,\}) = \bigcap\{\, i \bullet PT.\psi_i \,\} \tag{1}$$

An important and well-known consequence of this isomorphism is that UTP relations cannot model angelic nondeterminism. Since we have an isomorphism between UTP relations and set-based relations, and another between set-based relations and universally conjunctive predicate transformers, then UTP relations are isomorphic to universally conjunctive predicate transformers.

As already said, the angelic choice in which we are interested is the least upper bound of the lattice of monotonic predicate transformers. Joins in the lattice of universally conjunctive predicate transformers are not preserved in the lattice of monotonic predicate transformers [3]. We need a relational model isomorphic to the monotonic predicate transformers.

We investigate, next, the set of predicate transformers that correspond to UTP designs. In this case, ok is in the alphabet of the states in a precondition, and ok' is in the alphabet of the states in a postcondition. Table 3 gives healthiness conditions over such predicate transformers PT. The first healthiness condition, PTH1 requires that the weakest precondition for PT to establish any ψ is included in the set of initial states s for which $s.ok$ is true. In other words, in order to

$$\mathsf{PTH1}\ PT.\psi \subseteq \{\, s : S_{in\alpha PT} \mid s.ok = true \,\} \text{ provided } \psi \neq S_{out\alpha PT}$$

$$\mathsf{PTH3}\ PT.\psi = PT.\{\, s' : \psi \mid s'.ok' = true \,\} \text{ provided } \psi \neq S_{out\alpha PT}$$

Table 3. Predicate transformers healthiness conditions

guarantee a postcondition, PT must start. The only exception is the postcondition $S_{out\alpha PT}$, which imposes no restrictions whatsoever.

The healthiness condition PTH3 states that, in calculating $PT.\psi$, we can ignore all the states s' in ψ for which $s'.ok'$ is false. In other words, even if we have s' and $s' \oplus \{ok' \mapsto true\}$ in ψ, so that termination is not required, if PT can guarantee s' or $s' \oplus \{ok' \mapsto true\}$, then it can guarantee $s' \oplus \{ok' \mapsto true\}$. Consequently, predicate transformers do not capture information related to the possibility of nontermination. Again, the postcondition $S_{out\alpha PT}$ is an exception.

As expected, PTH1 and PTH3 correspond to H1 and H3 [8]. They restrict the behaviour of the predicate transformers for postconditions different from $S_{out\alpha PT}$. This particular postcondition, however, is of special interest.

Universally conjunctive predicate transformers can only model terminating programs; this is because, if (1) holds for the empty set, then $PT.S_{out\alpha} = S_{in\alpha}$. In words, for the postcondition that does not impose any restrictions, any initial state should be satisfactory. Nevertheless, the postcondition that does not impose any restriction still requires termination. Therefore, it is required that the program always terminates. In the context of predicate transformers that involve states on ok and ok', however, the postcondition $S_{out\alpha}$ does not require termination: it accepts any final state s', even those for which $s'.ok' = false$. Similarly, the precondition $S_{in\alpha}$ does not even require the program to start.

Therefore, the universal conjunctivity of the predicate transformers corresponding to designs does not imply that only terminating programs can be modelled. Unfortunately, conjunctivity is still an issue: the predicate transformers that are PTH1 and PTH3 healthy are conjunctive. As a consequence, they cannot model angelic nondeterminism. We need a model isomorphic to monotonic, not necessarily conjunctive, predicate transformers. This is pursued in the next section.

When we consider H3-healthy designs, we get a model isomorphic to standard weakest preconditions; in [8] we present an isomorphism between the predicate transformers above and those on postconditions and preconditions that do not refer to ok and ok'. In [10], different healthiness conditions that lead to a theory of general correctness are proposed.

4 Binary Multirelations

A relational model isomorphic to monotonic predicate transformers is presented in [20]; in that work, the relations are called binary multirelations. In our setting, we define a binary multirelation as a pair $(\alpha BM, BM)$, where αBM is an alphabet, and BM is a relation between $S_{in\alpha BM}$ and postconditions: elements of $\mathbb{P}\, S_{out\alpha BM}$. Intuitively, BM captures the behaviour of a program by associating each initial state with all the postconditions that the program can angelically choose to satisfy.

If a postcondition ψ can be satisfied, so can all postconditions weaker than ψ. Therefore, we have the following healthiness condition.

$$\mathsf{BMH} \quad \forall\, s, \psi_1, \psi_2 \mid (s, \psi_1) \in BM \land \psi_1 \subseteq \psi_2 \bullet (s, \psi_2) \in BM$$

The model for abort, for example, is the empty relation; miracle relates each initial state with every subset of $S_{out\alpha}$; it is the universal relation. The binary multirelation for an assignment $x := e$ relates every initial state s with every set that includes $[s]' \oplus \{x' \mapsto e\}$. This is a final state in which the value for each variable v' of $out\alpha$ is

$s.v$, except for x', whose value is e. If executed in s, the assignment $x := e$ reaches a final state that satisfies the postcondition $\{\, [s]' \oplus \{x' \mapsto e\}\,\}$, and any other weaker postcondition represented by one of its supersets.

The binary multirelation that models the angelic choice $x := 0 \sqcup x := 1$ is $\{\, s, \psi \mid \{(x' \mapsto 0)\} \subseteq \psi \vee \{(x' \mapsto 1)\} \subseteq \psi \,\}$. It associates to each initial state s the postconditions that include $(x' \mapsto 0)$ or $(x' \mapsto 1)$. We use $(x' \mapsto v)$ to denote a record with a single component named x' whose value is v. This is because the angel can ensure the final value of x to be either 0 or 1, as required. For the demonic choice, $x := 0 \sqcap x := 1$, the range of the binary multirelation includes the supersets of $\{\, (x' \mapsto 0), (x' \mapsto 1)\,\}$. In this case, the demon is in control: the final value of x is arbitrarily chosen to be 0 or 1.

For $x := 0 \sqcap (x := 1 \sqcup x := 2)$, which involves a demonic and an angelic choice, the model is $\{\, s, \psi \mid \{(x' \mapsto 0), (x' \mapsto 1)\} \subseteq \psi \vee \{(x' \mapsto 0), (x' \mapsto 2)\} \subseteq \psi \,\}$. If either 0 or 1 is an acceptable final value for x, then the angel can help. Similarly, if 0 and 2 are acceptable, we are guaranteed success. Nevertheless, 1 or 2 only cannot be guaranteed; of course, a requirement for 0, 1, or 2 is successful, and indeed the postcondition $\{\, (x' \mapsto 0), (x' \mapsto 1), (x' \mapsto 2)\,\}$ is a superset of both $\{\, (x' \mapsto 0), (x' \mapsto 1)\,\}$ and $\{\, (x' \mapsto 0), (x' \mapsto 2)\,\}$ and therefore is included in the range of the binary multirelation.

Here, we consider the isomorphism between binary multirelations and predicate transformers characterised by the functions below.

Definition 3. $bm2pt.BM.\psi = \{\, s \mid (s, \psi) \in BM \,\}$

$$pt2bm.PT = \{\, (s, \psi) \mid s \in PT.\psi \,\}$$

The function $bm2pt$ converts a binary multirelation to a weakest precondition: we have that $bm2pt.BM$ is guaranteed to establish a postcondition ψ in all initial states s associated to ψ in BM; in these states BM will angelically choose to establish ψ if required. Conversely, the multirelation $pt2bm.PT$ associates an initial state s with all the postconditions that PT is guaranteed to establish from s.

This isomorphism is simpler than that presented in [20], which constructs the binary multirelation corresponding to a predicate transformer using prime filter representations of states. Our proof that $bm2pt$ and $pt2bm$ characterise an isomorphism between predicate transformers and binary multirelations is very simple.

Theorem 2. $pt2bm.(bm2pt.BM) = BM$

Proof.

$$\begin{aligned}
&pt2bm.(bm2pt.BM) && \text{[definition of } pt2bm\text{]}\\
&= \{\, (s, \psi) \mid s \in bm2pt.BM.\psi \,\} && \text{[definition of } bm2pt\text{]}\\
&= \{\, (s, \psi) \mid s \in \{\, s \mid (s, \psi) \in BM \,\}\,\} && \text{[property of set comprehension]}\\
&= \{\, (s, \psi) \mid\ (s, \psi) \in BM \,\} && \text{[property of sets]}\\
&= BM && \square
\end{aligned}$$

Theorem 3. $bm2pt.(pt2bm.PT) = PT$

Proof.

$$\begin{aligned}
&bm2pt.(pt2bm.BM).\psi && \text{[definition of } bm2pt\text{]}\\
&= \{\, s \mid (s, \psi) \in pt2bm.PT \,\} && \text{[definition of } pt2bm\text{]}\\
&= \{\, s \mid (s, \psi) \in \{\, (s, \psi) \mid s \in PT.\psi \,\}\,\} && \text{[property of set comprehension]}\\
&= \{\, s \mid s \in PT.\psi \,\} && \text{[property of sets]}\\
&= PT.\psi && \square
\end{aligned}$$

The following two theorems establish that monotonic predicate transformers cor-

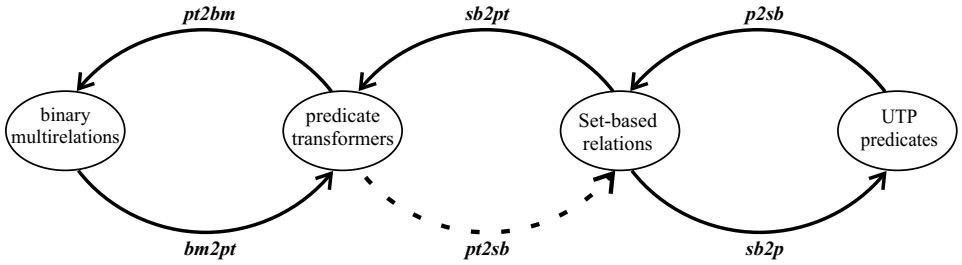

Fig. 1. Models and isomorphisms

respond to BMH-healthy multirelations. They conclude our argument; we have a model isomorphic to monotonic predicate transformers.

Theorem 4. *For a BMH-healthy binary relation BM, $bm2pt.BM$ is monotonic.*

Proof. Let ψ_1 and ψ_2 be such that $\psi_1 \subseteq \psi_2$.

$$
\begin{array}{ll}
bm2pt.BM.\psi_1 & [\text{definition of } bm2pt] \\
= \{\, s \mid (s, \psi_1) \in BM \,\} & [BM \text{ is healthy and } \psi_1 \subseteq \psi_2] \\
\subseteq \{\, s \mid (s, \psi_2) \in BM \,\} & [\text{definition of } bm2pt] \\
= bm2pt.BM.\psi_2 & \Box
\end{array}
$$

Theorem 5. *For a monotonic PT, the binary multirelation $pt2bm.PT$ is BMH-healthy.*

Proof. Let ψ_1 and ψ_2 be such that $\psi_1 \subseteq \psi_2$.

$$
\begin{array}{ll}
(s, \psi_1) \in pt2bm.PT & [\text{definition of } pt2bm] \\
= (s, \psi_1) \in \{\, (s, \psi) \mid s \in PT.\psi \,\} & [\text{property of set comprehension}] \\
= s \in PT.\psi_1 & [PT \text{ is monotonic and } \psi_1 \subseteq \psi_2] \\
\Rightarrow s \in PT.\psi_2 & [\text{property of set comprehension}] \\
= (s, \psi_2) \in \{\, (s, \psi) \mid s \in PT.\psi \,\} & [\text{definition of } pt2bm] \\
= (s, \psi_2) \in pt2bm.PT & \Box
\end{array}
$$

What we need now is a way of expressing multirelations as alphabetised predicates.

4.1 Predicative theory

The key point to define a UTP theory based on binary multirelations is the choice of alphabet. We propose a view of a binary multirelations as a relation between a state on an alphabet $in\alpha$ and a state on $\{\, dc' \,\}$. The value of dc' is the set of demonic choices available to the program: a set of states on an alphabet $out\alpha$. For example, in a theory of designs in which we can handle angelic nondeterminism, the alphabet is $\{\, v, ok, dc' \,\}$, where v stands for the list of program variables. In dc', the states are records that give values to the variables v' and ok'.

Figure 1 summarises the isomorphisms we have defined so far. We are looking

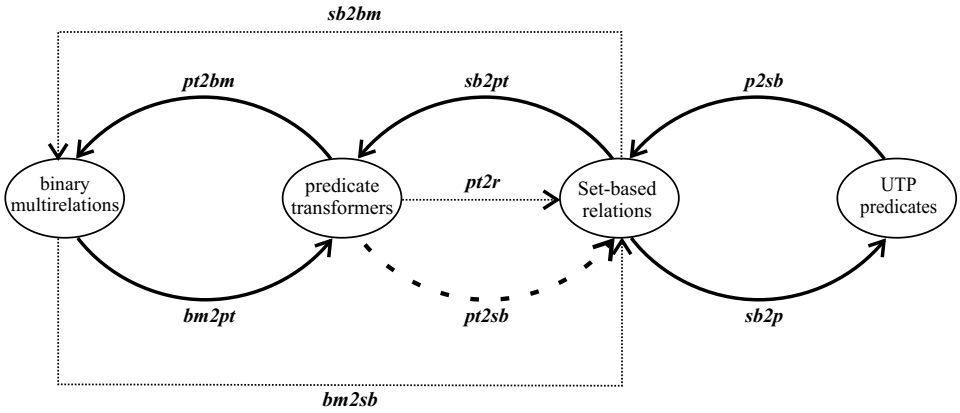

Fig. 2. Extra isomorphism

for a way of representing binary multirelations as UTP predicates. We cannot use $pt2sb$ in the transformation because it cannot handle non-conjunctive predicate transformers. Instead, we define an isomorphism between binary multirelations and set-based relations with alphabet $in\alpha \cup \{\, dc' \,\}$. It is based on the functions below.

Definition 4.

$$bm2sb.BM = \{\, s : S_{in\alpha};\ s' : S_{\{dc'\}} \mid (s, s'.dc') \in BM \,\}$$
$$sb2bm.DCR = \{\, s : S_{in\alpha};\ ss : \mathbb{P}\, S_{out\alpha} \mid (s, (dc' \mapsto ss)) \in DCR \,\}$$

Using $bm2sb$, we get a standard set-based relation in which the sets in the range of the original binary multirelation are wrapped in records with a single component dc'; the function $sb2bm$ unwraps these records. The proof that $bm2sb$ and $sb2sm$ establish an isomorphism is trivial.

Since predicate transformers are the standard setting for the study of angelic nondeterminism, we actually aim at expressing predicate transformers as UTP predicates using $pt2bm$, $bm2sb$, and $sb2p$. In our calculations, we name the composition of $pt2bm$ and $bm2sb$ as $pt2r \mathrel{\widehat{=}} bm2sb \circ pt2bm$. The next theorem is useful.

Theorem 6. $pt2r.PT = \{\, s : S_{in\alpha};\ s' : S_{\{dc'\}} \mid s \in PT.(s'.dc') \,\}$

For conciseness, we omit its simple proof. Figure 2 shows the additional isomorphism and function that we use in the sequel.

For example, the predicate transformer *abort* maps all postconditions to the empty set: it can never guarantee anything. In the UTP, it corresponds to **false**.

Theorem 7. $sb2p.(pt2r.abort) = \mathbf{false}$.

Proof.

$$\begin{aligned}
& sb2p.(pt2r.abort) && \text{[definition of } pt2r\text{]} \\
&= sb2p.\{\, s, s' \mid s \in abort.(s'.dc') \,\} && \text{[definition of } abort\text{]} \\
&= sb2p.\emptyset && \text{[definition of } sb2p\text{]} \\
&= \exists\, s, s' \bullet (s, s') \in \emptyset \wedge (\bigwedge x : in\alpha \bullet x = s.x) \wedge dc' = s'.dc' && \text{[property of sets]} \\
&= false && \square
\end{aligned}$$

Therefore, partiality models abortion. The miraculous program is **true**.

4.2 Healthiness condition

In the UTP, the healthiness condition for binary multirelations is as follows.

PBMH $\quad P;\ dc \subseteq dc' = P$

This requires that, if, after executing P, we execute a program that enlarges dc', then the result could have been obtained by P itself. A healthy P characterises dc' not by defining a particular value for it, but the smallest set of elements it should include. All the supersets should be allowed.

Healthy binary multirelations correspond to PBMH-healthy predicates.

Theorem 8. *If BM is BMH-healthy, then $sb2p.(bm2sb.BM)$ is PBMH-healthy.*

Proof.

$$sb2p.(bm2sb.BM);\ dc \subseteq dc' \qquad \text{[definition of } bm2sb\text{]}$$

$$= sb2p.\{\ s : S_{in\alpha};\ s' : S_{\{dc'\}} \mid (s, s'.dc') \in BM\ \};\ dc \subseteq dc' \qquad \text{[definition of } sb2p\text{]}$$

$$\begin{array}{l} = (\exists\, s, s' \bullet \qquad \text{[property of set comprehension]} \\ \quad (s, s') \in \{\ s : S_{in\alpha};\ s' : S_{\{dc'\}} \mid (s, s'.dc') \in BM\ \} \land \\ \quad (\bigwedge x : in\alpha R \bullet x = s.x) \land s'.dc' = dc'); \\ \quad dc \subseteq dc' \end{array}$$

$$\begin{array}{l} = (\exists\, s, s' \bullet (s, s'.dc') \in BM \land (\bigwedge x : in\alpha \bullet x = s.x) \land s'.dc' = dc'); \\ \quad dc \subseteq dc' \end{array}$$

[definition of sequential composition]

$$\begin{array}{l} = \exists\, s, s', dc_0 \bullet \qquad \text{[predicate calculus]} \\ \quad (s, s'.dc') \in BM \land (\bigwedge x : in\alpha \bullet x = s.x) \land s'.dc' = dc_0 \land dc_0 \subseteq dc' \end{array}$$

$$= \exists\, s, s' \bullet (s, s'.dc') \in BM \land (\bigwedge x : in\alpha \bullet x = s.x) \land s'.dc' \subseteq dc'$$

[BM is BMH-healthy and predicate calculus]

$$= \exists\, s, s' \bullet (s, dc') \in BM \land (\bigwedge x : in\alpha \bullet x = s.x) \land s'.dc' = dc'$$

[predicate calculus]

$$= \exists\, s, s' \bullet (s, s'.dc') \in BM \land (\bigwedge x : in\alpha \bullet x = s.x) \land s'.dc' = dc'$$

[property of sets]

$$\begin{array}{l} = \exists\, s, s' \bullet \qquad \text{[definitions of } sb2p \text{ and } bm2sb\text{]} \\ \quad (s, s') \in \{\ s : S_{in\alpha};\ s' : S_{\{dc'\}} \mid (s, s'.dc') \in BM\ \} \land \\ \quad (\bigwedge x : in\alpha \bullet x = s.x) \land s'.dc' = dc' \end{array}$$

$$= sb2p.(bm2sb.BM) \qquad \square$$

Theorem 9. *If P is a PBMH-healthy predicate, then $sb2bm.(p2sb.P)$ is BMH-healthy.*

Proof. Let ψ_1 and ψ_2 be such that $\psi_1 \subseteq \psi_2$.

$$(s, \psi_1) \in sb2bm.(p2sb.P) \qquad \text{[definition of } p2sb\text{]}$$

$$= (s, \psi_1) \in sb2bm.\{\ s, s' \mid P[s, s'/in\alpha, dc']\ \} \qquad \text{[definition of } sb2bm\text{]}$$

$$= (s, \psi_1) \in \{\, s : S_{in\alpha};\ ss : \mathbb{P}\, S_{\{dc'\}} \mid (s, (dc' \mapsto ss)) \in \{\, s, s' \mid P[s, s'/in\alpha, dc'] \,\}\}$$
[property of sets]

$$= (s, (dc' \mapsto \psi_1)) \in \{\, s, s' \mid P[s, s'/in\alpha, dc'] \,\}$$ [property of sets]

$$= P[s, \psi_1/in\alpha, dc']$$ [P is PBMH-healthy]

$$= (P;\ dc \subseteq dc')[s, \psi_1/in\alpha, dc']$$ [substitution]

$$= P[s/in\alpha];\ dc \subseteq \psi_1$$ [definition of sequential composition]

$$= \exists\, dc_0 \bullet P[s, dc_0/in\alpha, dc'] \wedge dc_0 \subseteq \psi_1$$ [$\psi_1 \subseteq \psi_2$]

$$\Rightarrow \exists\, dc_0 \bullet P[s, dc_0/in\alpha, dc'] \wedge dc_0 \subseteq \psi_2$$
[definition of sequential composition, and substitution]

$$= (P;\ dc \subseteq dc')[s, \psi_2/in\alpha, dc']$$ [P is PBMH-healthy]

$$= P[s, \psi_2/in\alpha, dc']$$ [definitions of $p2sb$ and $sb2bm$]

$$= (s, \psi_2) \in sb2bm.(p2sb.P)$$ □

It is pleasing that the healthiness condition can be cast in a quite simple way, and also in terms of the fixpoint of the idempotent $\mathsf{PHBM}(X) = X;\ dc \subseteq dc'$. This is important for the approach to linking theories encouraged by the UTP.

4.3 Refinement

The refinement relation is implication in the reverse direction from that adopted in the UTP. Still, it is just implication, and, more importantly, it corresponds to refinement in the predicate transformer model.

As usual, we define predicate transformer refinement as follows.

Definition 5. $PT_1 \sqsubseteq_{PT} PT_2 \mathrel{\widehat{=}} \forall\, \psi \bullet PT_1.\psi \subset PT_2.\psi$

For healthy binary multirelations, we have the following definition.

Definition 6. $BM_1 \sqsubseteq_{BM} BM_2 \mathrel{\widehat{=}} BM_1 \subseteq BM_2$

The next theorem establishes that these orders are compatible.

Theorem 10. *$BM_1 \sqsubseteq_{BM} BM_2$ if, and only if, $bm2pt.BM_1 \sqsubseteq_{PT} bm2pt.BM_2$.*

Proof.

$$bm2pt.BM_1 \sqsubseteq_{PT} bm2pt.BM_2$$ [definition of $\sqsubseteq_{PT}$]

$$= \forall\, \psi \bullet bm2pt.BM_1.\psi \subseteq bm2pt.BM_2.\psi$$ [definition of $bm2pt$]

$$= \forall\, \psi \bullet \{\, s \mid (s, \psi) \in BM_1 \} \subseteq \{\, s \mid (s, \psi) \in BM_2 \}$$ [property of sets]

$$= \forall\, \psi, s \bullet (s, \psi) \in BM_1 \Rightarrow (s, \psi) \in BM_2$$ [property of sets]

$$= BM_1 \sqsubseteq_{BM} BM_2$$ □

Finally, we define angelic refinement in the UTP theory.

Definition 7. $P \sqsubseteq_A Q \mathrel{\widehat{=}} [\, P \Rightarrow Q \,]$

The correspondence between this refinement relation and that for binary multire-

lations is established below.

Theorem 11. *$P \sqsubseteq_A Q$ if, and only if, $sb2bm.(p2sb.P) \sqsubseteq_{BM} sb2bm(p2sb.Q)$.*

Proof.

$$sb2bm.(p2sb.P) \sqsubseteq_{BM} sb2bm(p2sb.Q) \qquad \text{[definition of } \sqsubseteq_{BM}\text{]}$$

$$= sb2bm.(p2sb.P) \subset sb2bm(p2sb.Q) \qquad \text{[property of sets]}$$

$$= \forall\, s, \psi \bullet (s, \psi) \in sb2bm.(p2sb.P) \Rightarrow (s, \psi) \in sb2bm.(p2sb.Q) \qquad \text{[definition of } sb2bm\text{]}$$

$$= \forall\, s, \psi \bullet (s, dc' \mapsto \psi) \in p2sb.P \Rightarrow (s, dc' \mapsto \psi) \in p2sb.Q \qquad \text{[definition of } p2sb\text{]}$$

$$= \forall\, s, \psi \bullet P[s, \psi / in\alpha, dc'] \Rightarrow Q[s, \psi / in\alpha, dc'] \qquad \text{[predicate calculus]}$$

$$= \forall\, x : in\alpha, dc' \bullet P \Rightarrow Q \qquad \text{[the alphabet is } in\alpha \cup \{dc'\}\text{]}$$

$$= [P \Rightarrow Q] \qquad \square$$

The pre-order proposed in [20] for binary multirelations becomes a partial order in the restricted setting of healthy binary multirelations. Also, it collapses to set inclusion, which is the order we adopt here.

4.4 Operators

Angelic choice $P \sqcup Q$ is characterised by disjunction. The program $P \sqcup Q$ gives all the guarantees that can be provided by choosing P, together with those that arise from the possibility of choosing Q.

Theorem 12. *$sb2p.(pt2r.(P \sqcup Q)) = sb2p.(pt2r.P) \vee sb2p.(prt2.Q)$*

Proof.

$$sb2p.(pt2r.(P \sqcup Q)) \qquad \text{[Theorem 6]}$$

$$= sb2p.\{\, s, s' \mid s \in (P \sqcup Q).(s'.dc') \,\} \qquad \text{[predicate transformer semantics of } \sqcup\text{]}$$

$$= sb2p.\{\, s, s' \mid s \in P.(s'.dc') \cup Q.(s'.dc') \,\} \qquad \text{[definition of } sb2p\text{]}$$

$$= \exists\, s, s' \bullet s \in P.(s'.dc') \cup Q.(s'.dc') \wedge (\bigwedge x : in\alpha R \bullet x = s.x) \wedge s'.dc' = dc' \qquad \text{[property of sets and predicate calculus]}$$

$$= (\exists\, s, s' \bullet s \in P.(s'.dc') \wedge (\bigwedge x : in\alpha R \bullet x = s.x) \wedge s'.dc' = dc') \vee (\exists\, s, s' \bullet s \in Q.(s'.dc') \wedge (\bigwedge x : in\alpha R \bullet x = s.x) \wedge s'.dc' = dc') \qquad \text{[definitions of } pt2r \text{ and } sb2p\text{]}$$

$$= sb2p.(pt2r.P) \vee sb2p.(pt2r.Q) \qquad \square$$

Demonic choice is captured by conjunction; a postcondition is guaranteed by $P \sqcap Q$

only if both P and Q can guarantee it, so that the arbitrary choice is not a problem.

Theorem 13. $sb2p.(pt2r.(P \sqcap Q)) = sb2p.(pt2r.P) \wedge sb2p.(prt2.Q)$

Proof.

$$sb2p.(pt2r.(P \sqcap Q)) \qquad \text{[Theorem 6]}$$

$$= sb2p.\{\, s, s' \mid s \in (P \sqcap Q).(s'.dc') \,\} \qquad \text{[predicate transformer semantics of } \sqcap\text{]}$$

$$= sb2p.\{\, s, s' \mid s \in P.(s'.dc') \cap Q.(s'.dc') \,\} \qquad \text{[definition of } sb2p\text{]}$$

$$= \exists\, s, s' \bullet s \in P.(s'.dc') \cap Q.(s'.dc') \wedge (\bigwedge x : in\alpha R \bullet x = s.x) \wedge s'.dc' = dc'$$
[property of sets and predicate calculus]

$$= (\theta x : in\alpha \bullet x \mapsto x) \in P.dc' \cap Q.dc' \qquad \text{[property of sets]}$$

$$= (\theta x : in\alpha \bullet x \mapsto x) \in P.dc' \wedge (\theta x : in\alpha \bullet x \mapsto x) \in Q.dc'$$
[property of sets and predicate calculus]

$$(\exists\, s, s' \bullet s \in P.(s'.dc') \wedge (\bigwedge x : in\alpha R \bullet x = s.x) \wedge s'.dc' = dc') \wedge$$
$$(\exists\, s, s' \bullet s \in Q.(s'.dc') \wedge (\bigwedge x : in\alpha R \bullet x = s.x) \wedge s'.dc' = dc')$$
[definitions of $pt2r$ and $sb2p$]

$$= sb2p.(pt2r.P) \wedge sb2p.(pt2r.Q) \qquad \square$$

In this proof, we use the notation $(\theta x : A \bullet x \mapsto v)$ to describe the record that associates each name x in the alphabet A to the corresponding value v. Above, the value is that of the variable x itself; we have a predicate on the variables x and dc'.

Sequential composition cannot correspond to relational composition. It uses the operator * to lift Q to a predicate on dc and dc'. It is inspired on the UTP treatment of logic programming, and is defined as follows.

$$Q^* \;\widehat{=}\; \mu\, X \bullet \mathbf{true} \lhd dc = \emptyset \rhd \mathbf{var}\ s \bullet s' \in dc;\ (v := s.v;\ Q) \sqcup (dc := dc \setminus \{\, s \,\};\ X)\ \mathbf{end}$$

Sequential composition is $P;\ Q^*$: after the execution of P, Q^* recursively selects a state in dc' and executes Q. The program $P \lhd c \rhd Q$ is a conditional: it executes P if c holds, else it executes Q. A variable s is declared to hold a state in dc. The observational variables are initialised as in s before Q is executed. The demonic choice of all the outcomes of the executions of Q is the result of the sequence.

It is unavoidable that the definitions of some operators are more complicated than those in the original UTP model. It is part of the philosophy of the UTP to study constructs and concepts in isolation: we have provided a theory for angelic nondeterminism which can be incorporated to the other theories as needed. We have also established that we do need a more elaborate relational model to capture angelic nondeterminism.

5 Conclusions

The central objective of Hoare and He's UTP is to formalise different programming paradigms within a common semantic framework, so that they may be directly compared and new compound programming languages and refinement calculi may be developed. This ambitious research programme has only just been started. An important question to ask is: what are the theoretical limits to this investigation?

Angelic nondeterminism is a valuable concept: it plays an important rôle in refinement calculi, and it is used as an abstraction in search-based and constraint-oriented programming, hiding details of how particular strategies are implemented. The main contribution of this paper is a predicative account of binary multirelations that allows the unification of angelic nondeterminism into the UTP.

We describe the UTP predicative theory of alphabetised relations and the theory of designs, where it is possible to observe the start and termination of a program. Designs enable reasoning about total correctness, and a set-based model of relations brings this fact sharply into focus. We show that there is an isomorphism between our set-based relations and universally conjunctive predicate transformers. This establishes a connection with an existing result: conjunctive predicate transformers cannot capture angelic nondeterminism.

A relational model that can capture both angelic and demonic nondeterminism is presented in [20]. We cast that model in the UTP predicative style, including a healthiness condition and the refinement relation. This allows its use in an integrated framework that covers, for instance, concurrency and higher-order programming. We are going to use this model to extend the existing semantics of our combined formalism [21], and prove refinement laws.

In [4], Back and von Wright present another relational model isomorphic to predicate transformers; it is actually a functional model called choice semantics. In that work, a program P is a function from initial states s to the set of post-conditions that can be satisfied when P is executed in s. The choice semantics is, of course, isomorphic to binary multirelations. Since in the predicative style of the UTP relations are defined punctually, it was more convenient to base our work on binary multirelations rather than on choice semantics.

The work in [16] presents a functional semantics for a tactic language which includes angelic nondeterminism. The semantics of angelic choice is a list that contains all the options available to the angel; demonic nondeterminism is not included. In [17], the set-based model of binary relations is used to support angelic and demonic nondeterminism in a calculus for functional programs. They adopt two refinement relations, one of which is the same as ours.

Both [20] and [17] present operations that model, for example, angelic nondeterminism and sequence. Our contribution is to cast these operations at the level of UTP predicates, where they can be integrated into more powerful theories of programming.

Acknowledgements

The authors are grateful to Will Harwood for extensive discussions, and to Carroll Morgan for pointing out the work on binary multirelations. This work is partially funded by QinetiQ and the Royal Society.

References

1. R. J. R. Back and J. Wright. A Lattice-theoretical Basis for a Specification Language. In J. L. A. van de Snepscheut, editor, *Mathematics of Program Construction: 375th Anniversary of the Groningen University*, volume 375 of *LNCS*, pages 139 – 156, Groningen, The Netherlands, 1989. Springer-Verlag.
2. R. J. R. Back and J. Wright. Duality in Specification Languages: A Lattice-theoretical Approach. *Acta Informatica*, 27(7):583 – 625, 1990.
3. R. J. R. Back and J. Wright. Combining angels, demons and miracles in program specifications. *Theoretical Computer Science*, 100:365 – 383, 1992.

4. R. J. R. Back and J. Wright. *Refinement Calculus: A Systematic Introduction*. Graduate Texts in Computer Science. Springer-Verlag, 1998.
5. A. L. C. Cavalcanti, A. C. A. Sampaio, and J. C. P. Woodcock. A Refinement Strategy for *Circus*. *Formal Aspects of Computing*, 15(2 – 3):146 – 181, 2003.
6. A. L. C. Cavalcanti and J. C. P. Woodcock. A Weakest Precondition Semantics for Z. *The Computer Journal*, 41(1):1 – 15, 1998.
7. A. L. C. Cavalcanti and J. C. P. Woodcock. ZRC—A Refinement Calculus for Z. *Formal Aspects of Computing*, 10(3):267—289, 1999.
8. A. L. C. Cavalcanti and J. C. P. Woodcock. Angelic Nondeterminism and Unifying Theories of Programming (Extended Version). Technical report 13-04, University of Kent - Computing Laboratory, 2004.
9. E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.
10. S. Dunne. Recasting Hoare and He's Unifying Theories of Programs in the Context of General Correctness. In A. Butterfield and C. Pahl, editors, *IWFM'01: 5th Irish Workshop in Formal Methods*, BCS Electronic Workshops in Computing, Dublin, Ireland, July 2001.
11. P. H. B. Gardiner and C. C. Morgan. Data Refinement of Predicate Transformers. *Theoretical Computer Science*, 87:143 – 162, 1991.
12. W. H. Hesselink. *Programs, Recursion and Unbounded Choice – Predicate Transformation Semantics and Transformation Rules*. Cambridge Tracts in Theoretical Computer Science 27. Cambridge University Press, 1992.
13. C. A. R. Hoare and Jifeng He. The Weakest Prespecification. Technical Monograph TM-PRG-44, Oxford University Computing Laboratory, Oxford – UK, 1985.
14. C. A. R. Hoare and He Jifeng. *Unifying Theories of Programming*. Prentice-Hall, 1998.
15. R. Jagadeesan, V. Shanbhogue, and V. Saraswat. Angelic non-determinism in concurrent constraint programming. Technical report, Xerox Park, January 1991.
16. A. P. Martin, P. H. B. Gardiner, and J. C. P. Woodcock. A Tactical Calculus. *Formal Aspects of Computing*, 8(4):479–489, 1996.
17. C. E. Martin, S. A. Curtis, and I. Rewitzky. Modelling Nondeterminism. In *Mathematics of Program Construction*, LNCS, 2004.
18. C. C. Morgan. *Programming from Specifications*. Prentice-Hall, 2nd edition, 1994.
19. C. C. Morgan and P. H. B. Gardiner. Data Refinement by Calculation. *Acta Informatica*, 27(6):481—503, 1990.
20. I. Rewitzky. Binary Multirelations. In H. Swart, E. Orlowska, G. Schmidt, and M. Roubens, editors, *Theory and Application of Relational Structures as Knowledge Instruments*, volume 2929 of *LNCS*, pages 256 – 271, 2003.
21. J. C. P. Woodcock and A. L. C. Cavalcanti. The Semantics of *Circus*. In D. Bert, J. P. Bowen, M. C. Henson, and K. Robinson, editors, *ZB 2002: Formal Specification and Development in Z and B*, volume 2272 of *LNCS*, pages 184—203. Springer-Verlag, 2002.
22. J. C. P. Woodcock and A. L. C. Cavalcanti. A Tutorial Introduction to Designs in Unifying Theories of Programming. In *IFM 2004: Integrated Formal Methods*, volume 2999 of *LNCS*, pages 40 – 66. Springer-Verlag, 2004. Invited tutorial.