Under consideration for publication in Formal Aspects of Computing

Angelic Nondeterminism in the Unifying Theories of Programming

Ana Cavalcanti[1], Jim Woodcock[1] and Steve Dunne[2]

[1]Department of Computer Science, University of York, York, YO10 5DD, England
[2]School of Computing, University of Teesside, Middlesbrough, TS1 3BA, England

Keywords: semantics, refinement, relations, predicate transformers.

Abstract. Hoare and He's unifying theories of programming (UTP) is a model of alphabetised relations expressed as predicates; it supports development in several programming paradigms. The aim of Hoare and He's work 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 is isomorphic to the monotonic predicate transformers model and can express angelic and demonic nondeterminism.

1. Introduction

Angelic nondeterminism is a specification and programming concept that is typically available in unified languages of refinement calculi [Mor94, BW98], and in concurrent constraint programming languages [JSS91]. 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 [MGW96] explores angelic nondeterminism in 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 [GM91], Gardiner and Morgan identify angelic choice with the least upper bound in the lattice of monotonic predicate transformers. In [MG90], they use this construct to define logical constants, which are pervasive in refinement techniques, and sometimes named logical, auxiliary, or angelic variables. They 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 [Mor94] 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

Correspondence and offprint requests to: Ana Cavalcanti.

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 [BW98] 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 angelic or demonic, and a complete base language, which can describe any monotonic predicate transformer [BW89, BW90]. More recently, they have suggested the use of angelic choice 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 [CW99]. It is incorporated in *Circus* [WC02], 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 [CSW03]. Extensions of *Circus* include constructs to handle, for example, time and mobility.

The semantics of *Circus* is based on Hoare and He's unifying theories of programming (UTP) [HJ98, WC04]. This is a predicate-based relational model that links constructs in several paradigms: imperative, concurrent, logical, and others. By providing a framework for the study of state and reactive aspects of a program, the UTP is a solid basis for the model of *Circus* and 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 [Hes92, CW98]. These results establish that a straightforward relational model that associates initial with final states 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 establish that the general model of UTP relations does not cover angelic nondeterminism.

In [CW05] we have proposed a UTP theory that can cover both angelic and demonic nondeterminism based on the model of binary multirelations introduced in [Rew03]. We based our proposal on an isomorphism between binary multirelations and predicate transformers suggested in [Rew03]. We have studied refinement and some programming operators, including sequence and angelic nondeterminism in that theory. It was unfortunate that the refinement relation had a definition different from that adopted in all other UTP theories: implication, instead of reverse implication. Also, we had a quite elaborate definition for sequence.

Here, we consider a different isomorphism between binary multirelations and predicate transformers; in the new UTP theory that it suggests, refinement is reverse implication. From the point of view of unification, which is, of course, a central concern in the UTP, this is very pleasing. It means that the new theory can be combined with the existing UTP theories using the approach already illustrated in [HJ98]. In the context of the new theory, we consider the definition of all programming operators studied in the general theory of relations, and designs, which are basically specification statements. We also give a definition of sequence that is much simpler than that in [CW05]; this makes our theory much more tractable and attractive.

In the next section, we present an overview of the UTP. 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. The definition of programming operators in the new theory is the subject of Section 5. Finally, in Section 6 we present our conclusions and directions for future work.

2. Unifying theories of programming

The objective of Hoare and He's work on 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 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. 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.

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

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 $I\!I \ \widehat{=}\ (v' = v)$ 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 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.

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

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 $Q(v_0)$. In all cases, v, v', and v_0 stand for lists of variables.

The nondeterministic choice $P \sqcap Q \ \widehat{=}\ P \lor Q$ 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 $[Q \Rightarrow P]$; in that case we write $P \sqsubseteq Q$. 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 greatest lower 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, if the alphabet is $\{ x, x' \}$, then 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 $P \vdash Q$.

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

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

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

$$x := e \ \widehat{=}\ \mathbf{true} \vdash x' = e \land y' = y \qquad\qquad I\!I \ \widehat{=}\ \mathbf{true} \vdash v' = v$$

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

Four healthiness conditions on relations P 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 P only need to hold if it has started. The second healthiness condition states that P 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 can be written as a design.

The healthiness conditions H3 and H4 are expressed as equations between programming constructs. Results presented in [HJ98] 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.

Designs form a UTP theory 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

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

Table 1. UTP Healthiness conditions

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 predicate transformers, with predicates characterised by sets. These models further clarify the role of the healthiness conditions in Table 1 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 a possible observation of a later state.

The model for *abort* is the universal relation: $S_{in\alpha} \times S_{out\alpha}$; 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 [CW04]; it is based on the functions $p2sb$ and $sb2p$.

Definition 3.1.

$$p2sb.(\alpha P, P) \;\widehat{=}\; (\alpha P, \{\, s_1 : S_{in\alpha P};\; s_2 : S_{out\alpha P} \mid P[s_1, s_2/in\alpha P, out\alpha P]\,\})$$
$$sb2p.(\alpha R, R) \,\widehat{=}\, (\alpha R, \exists\, s_1 : S_{in\alpha R}, s_2 : S_{out\alpha R} \bullet (s_1, s_2) \in R \,\wedge$$
$$(\textstyle\bigwedge x : in\alpha R \bullet x = s_1.x) \wedge (\textstyle\bigwedge x : out\alpha R \bullet x = s_2.x))$$

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 model is used by Hoare and He when they discuss denotational semantics.

The set-based relation defined by $p2sb$ for a predicative relation P is formed by pairs of states s_1 and s_2 such that P holds when the observational variables take the values associated to them by s_1 and s_2. 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_1 and s_2 in R. For each pair, a conjunction of equalities requires that each observational variable takes the value in the corresponding state. Since alphabets are finite, the conjunction is finite. If we use θA as an abbreviation for the state on alphabet A that associates each component of name x in A with the value of the variable x, then $sb2p.(\alpha, R)$ can be expressed as $(\theta in\alpha R, \theta out\alpha R) \in R$. The proof is a straightforward application of the one-point rule. The existential quantification probably accounts for a clearer definition for $sb2p$; in proofs the shorter formulation is more convenient.

Standard work on relational semantics [HH85] singles out a special state to indicate non-termination; this

SBH1	$\forall s_1, s_2 \mid s_1.ok = \mathit{false} \bullet (s_1, s_2) \in R$
SBH2	$\forall s_1, s_2 \mid (s_1, s_2) \in R \land s_2.ok' = \mathit{false} \bullet (s_1, s_2 \oplus \{ok' \mapsto \mathit{true}\}) \in R$
SBH3	$\forall s_1 \mid (\exists s_2 \bullet s_2.ok' = \mathit{false} \land (s_1, s_2) \in R) \bullet \forall s_2 \bullet (s_1, s_2) \in R$

Table 2. Set-based healthiness conditions

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 characterised by $p2sb$ and $sb2p$ suggests that the general UTP model of relations is not able to capture non-termination. As already mentioned, 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 value of $(\mu X \bullet X)$ is the bottom of the lattice $\bot$, which 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 SBH1, SBH2, and SBH3 over such relations; we omit the obvious types of s_1 and s_2. The theorem below, proved in [CW04], establishes that H1, H2, and H3 correspond to SBH1, SBH2, and SBH3 in the set-based model.

Theorem 3.1. 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.

The condition SBH1 requires that, in a healthy relation R, all states s_1 for which $s_1.ok$ is false 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_1 is related to a state s_2 for which $s_2.ok'$ is false, then s_1 is also related to $s_2 \oplus \{ok' \mapsto \mathit{true}\}$. This is the same state as s_2, except that the value of ok' is true. This means that if it is possible not to terminate from s_1, it is also possible to terminate. Its behaviour, however, may not be completely arbitrary: it is not required that R relates s_1 to all possible final states; this is what is required by SBH3.

We believe that it is not difficult to observe that SBH3 relations are necessarily SBH2. If the initial state s_1 is related to all possible final states, then it is also related to $s_2 \oplus \{ok' \mapsto \mathit{true}\}$. 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 [Dij76].

Isomorphisms between predicate transformers and set-based relational models have been studied [Hes92]; the one below is similar to that in [CW98]. 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 are maintained by both functions.

Definition 3.2. $sb2pt.R.\psi \mathrel{\widehat{=}} \neg\ \mathrm{dom}(R \rhd \psi)$

$$pt2sb.PT = \{ s_1 : S_{in\alpha PT};\ s_2 : S_{out\alpha PT} \mid s_1 \in \neg\ PT.(\neg \{ s_2 \}) \}$$

In the definition of $sb2pt$, ψ is a postcondition, or rather, a set of states on $out\alpha R$, which is given as argument

PTH1	$PT.\psi \subseteq \{ s_1 : S_{in\alpha PT} \mid s_1.ok = true \}$ provided $\psi \neq S_{out\alpha PT}$
PTH3	$PT.\psi = PT.\{ s_2 : \psi \mid s_2.ok' = true \}$ provided $\psi \neq S_{out\alpha PT}$

Table 3. Predicate transformers healthiness conditions

to the transformer $sb2pt.R$. The relation $R \rhd \psi$ models all executions of R that do not lead to a 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 $\neg\ \mathrm{dom}(R \rhd \psi)$ contains all initial states in which we are guaranteed to reach a state that satisfies ψ: the required weakest precondition.

The relation $pt2sb.PT$ associates an initial state s_1 to a final state s_2 if s_1 is not in the weakest precondition that guarantees that PT does not establish s_2. Since it is not guaranteed that PT will not establish s_2, 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 as well as demonic 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 [BW92]. 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_1 for which $s_1.ok$ is true. In other words, in order to 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_2 in ψ for which $s_2.ok'$ is false. In other words, even if we have s_2 and $s_2 \oplus \{ok' \mapsto true\}$ in ψ, so that termination is not required, if PT can guarantee s_2 or $s_2 \oplus \{ok' \mapsto true\}$, then it can guarantee $s_2 \oplus \{ok' \mapsto true\}$. Moreover, if s_2 is in ψ, but $s_2 \oplus \{ok' \mapsto true\}$ is not, so that non-termination is actually required, then PT cannot do it. Consequently, predicate transformers do not capture information related to the possibility of non-termination. Again, the postcondition $S_{out\alpha PT}$ is an exception.

As stated in the theorem below, which is proved in [CW04], PTH1 and PTH3 correspond to H1 and H3.

Theorem 3.2. For every set-based relation R that satisfies SBH1, $sb2pt.R$ satisfies PTH1. Conversely, for every predicate transformer PT that satisfies PTH1, $pt2sb.PT$ satisfies SBH1. The same holds for SBH3 and PTH3.

The healthiness conditions PTH1 and PTH3 restrict the behaviour of the predicate transformers for postconditions different from $S_{out\alpha PT}$. This postcondition, however, is of special interest.

Standard 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 is 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 situation is different. The postcondition $S_{out\alpha}$ does not require termination: it accepts any final state s_2, even those for which $s_2.ok' = false$. 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.

As an aside, we observe that when we consider H3-healthy designs, we get a model isomorphic to standard conjunctive weakest preconditions; in [CW04] we present an isomorphism between the predicate transformers above and those on postconditions and preconditions that do not refer to ok and ok'. In [Dun01], 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 [Rew03]; in that work, the relations are called binary multirelations. We studied that model in the context of the UTP in [CW05]. We defined 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 captured the behaviour of a program by associating each initial state with all the postconditions that the program can angelically choose to satisfy. The encoding of this model in the UTP leads to a theory in which refinement is captured by implication, instead of reverse implication.

In this section, we explore a similar model of binary multirelations in which behaviour is captured by relating an initial state to all the sets of states from which an angelic choice can be made to determine a final state for the program. The choice between the sets of states themselves is demonic.

The model for $abort$, for example, is the universal relation; this means that we can demonically choose any set of states as options open for the angelic choice. In other words, the demonic choice prevails, since any set of options whatsoever, including the empty set, can be left for the angelic choice. Miracle, on the other hand, is the empty relation; this means that there are no demonic choices to be made. In general, a computation characterised by a binary multirelation BM is at risk of not terminating when executed from any starting state s such that $(s, \emptyset) \in BM$. On the other hand, execution is miraculous from any state outside the domain of BM. All this is, of course, in sympathy with the set-based model of Section 3.

The binary multirelation for an assignment $x := e$ relates every initial state s_1 with every set that includes $s_1' \oplus \{x' \mapsto e\}$. The state s_1' is obtained from s_1 by dashing the names of each of the variables in its domain. Therefore, $s_1' \oplus \{x' \mapsto e\}$ is a final state in which the value for each variable v' of $out\alpha$ is $s_1.v$, except for x', whose value is e. If executed in s_1, the assignment $x := e$ reaches the final state $s_1' \oplus \{x' \mapsto e\}$. The fact that the binary multirelation associates s_1 to all sets that include this state, instead of just to the singleton set $\{\, s_1' \oplus \{x' \mapsto e\} \,\}$, needs further explanation.

In fact, given any two states s_2 and s_3, providing the set $\{\, s_2, s_3 \,\}$ of angelic choices, in addition to the set $\{\, s_2 \,\}$, as an extra option available for demonic choice is immaterial. Since s_3 cannot be guaranteed to be available for the angelic choice, there can be no guarantee that the program will achieve s_3. More generally, in algebraic terms, we have that $P \sqcap (P \sqcup Q) = P$, where $\sqcap$ and $\sqcup$ represent demonic and angelic choice; this property can be easily proved in the predicate transformer model, for example.

In general, in the binary multirelation model, if an initial state s_1 is associated with a set of states ss, then associating s_1 to a superset of ss does not add to the options that are actually available for angelic choice. We could provide a definition of refinement that takes this fact into account, and regards the relation that associates s_1 only to ss_1 and a relation that associates s_1 to ss_1 and to one or more of its supersets as equal. We are striving, however, for a simple definition of refinement. Therefore, we choose to identify one of those binary multirelations as the unique model of the program that actually only provides the states in ss for angelic choice. Inspired by the model in [Rew03], we choose the set of binary multirelations that are upward closed. This is captured in the following healthiness condition.

$$\mathsf{BMH} \quad \forall\, s_1, ac_1, ac_2 \mid (s_1, ac_1) \in BM \wedge ac_1 \subseteq ac_2 \bullet (s_1, ac_2) \in BM$$

This states that, if from an initial state s_1, the set of angelic choices ac_1 is available for demonic choice, so are all the supersets ac_2 of ac_1.

The binary multirelation that models the angelic choice $x := 0 \sqcup x := 1$, with alphabet $\{\, x, x' \,\}$, is $\{\, s_1, ac \mid \{\, (x' \mapsto 0), (x' \mapsto 1) \,\} \subseteq ac \,\}$. It associates to each initial state s_1 the sets of angelic choices that include $(x' \mapsto 0)$ and $(x' \mapsto 1)$. This is because the angel can ensure the final value of x to be either 0 or 1, as required. We use $(x' \mapsto v)$ to denote a record with a single component named x' whose value is v. For the demonic choice, $x := 0 \sqcap x := 1$, the range of the binary multirelation includes the supersets of $\{\, (x' \mapsto 0) \,\}$ and $\{(x' \mapsto 1)\,\}$. In this case, the demon is in control: the final value of x is arbitrarily chosen to be 0 or 1.

For $x := 0 \sqcup (x := 1 \sqcap x := 2)$, which is a program that involves an angelic and a demonic choice, the model is $\{ s_1, ac \mid \{(x' \mapsto 0), (x' \mapsto 1)\} \subseteq ac \lor \{(x' \mapsto 0), (x' \mapsto 2)\} \subseteq ac \}$. The demonic choices available cannot prevent the angelic choice of 0 for the final value of x: all sets ac that can be demonically chosen include the state $(x' \mapsto 0)$. The options $(x' \mapsto 1)$ and $(x' \mapsto 2)$, however, are left open for demonic choice. The functions below define an isomorphism between binary multirelations and predicate transformers.

Definition 4.1. $bm2pt.BM.\psi = \{ s_1 \mid (s_1, \neg\, \psi) \notin BM \}$

$pt2bm.PT = \{ (s_1, \psi) \mid s_1 \in \neg\, PT.(\neg\, \psi) \}$

The function $bm2pt$ converts a binary multirelation to a weakest precondition transformer. We have that $bm2pt.BM$ is guaranteed to establish a postcondition ψ in all initial states s_1 for which there is not a set of states disjoint from ψ that can be demonically chosen. If, in all sets of states available for demonic choice, there is at least one state that is acceptable from the point of view of ψ, or in other words, belongs to ψ, the angelic choice is guaranteed to select such a state to satisfy ψ. In the definition of $bm2pt$, we consider specifically whether the complement of ψ is associated to s_1. If any set disjoint from ψ is associated to s_1, then upward closedness guarantees that the complement of ψ is also associated with s_1.

Conversely, the multirelation $pt2bm.PT$ associates an initial state s_1 with all the postconditions that PT is not guaranteed not to establish from s_1. These are the sets of states that may be reached from s_1. They are taken as available for demonic choice.

This isomorphism is simpler than that presented in [Rew03], 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, although slightly more complex than that in [CW05].

Theorem 4.1. $pt2bm.(bm2pt.BM) = BM$

Proof.

$pt2bm.(bm2pt.BM)$ [definition of $pt2bm$]

$= \{ (s_1, \psi) \mid s_1 \in \neg\, bm2pt.BM.(\neg\, \psi) \}$ [definition of $bm2pt$]

$= \{ (s_1, \psi) \mid s_1 \in \neg \{ s_1 \mid (s_1, \neg\,\neg\, \psi) \notin BM \} \}$ [property of sets]

$= \{ (s_1, \psi) \mid s_1 \in \{ s_1 \mid (s_1, \psi) \in BM \} \}$ [property of set comprehension]

$= \{ (s_1, \psi) \mid (s_1, \psi) \in BM \}$ [property of sets]

$= BM$

□

Theorem 4.2. $bm2pt.(pt2bm.PT) = PT$

Proof.

$bm2pt.(pt2bm.PT).\psi$ [definition of $bm2pt$]

$= \{ s_1 \mid (s_1, \neg\, \psi) \notin pt2bm.PT \}$ [definition of $pt2bm$]

$= \{ s_1 \mid (s_1, \neg\, \psi) \notin \{ (s_1, \psi) \mid s_1 \in \neg\, PT.(\neg\, \psi) \} \}$ [property of set comprehension]

$= \{ s_1 \mid s_1 \notin \neg\, PT.(\neg\,\neg\, \psi) \}$ [property of sets]

$= \{ s_1 \mid s_1 \in PT.\psi \}$ [property of sets]

$= PT.\psi$

□

The following two theorems establish that monotonic predicate transformers correspond to BMH-healthy

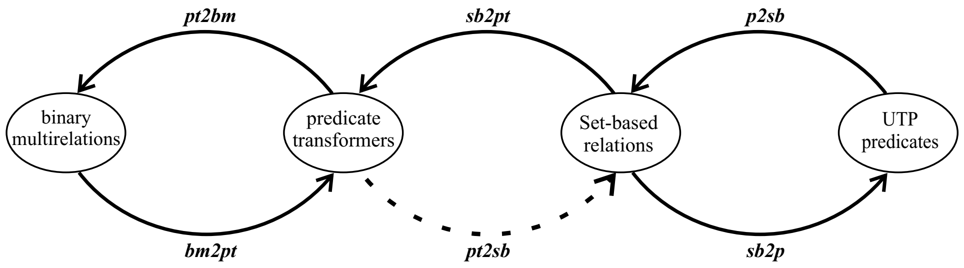

Fig. 1. Models and isomorphisms

multirelations. First of all, healthy binary multirelations define monotonic predicate transformers.

Theorem 4.3. For a BMH-healthy binary multirelation BM, $bm2pt.BM$ is monotonic.

Proof. We consider two postconditions ψ_1 and ψ_2.

$\psi_1 \subseteq \psi_2$ [property of sets]

$\Rightarrow \neg\, \psi_2 \subseteq \neg\, \psi_1$ [BM is healthy]

$\Rightarrow \forall\, s_1 \bullet (s_1, \neg\, \psi_2) \in BM \Rightarrow (s_1, \neg\, \psi_1) \in BM$ [predicate calculus]

$\Rightarrow \forall\, s_1 \bullet (s_1, \neg\, \psi_1) \notin BM \Rightarrow (s_1, \neg\, \psi_2) \notin BM$ [definition of $bm2pt$]

$\Rightarrow bm2pt.BM.\psi_1 \subseteq bm2pt.BM.\psi_2$

□

Now, a monotonic predicate transformer corresponds to a healthy binary multirelation.

Theorem 4.4. For a monotonic PT, the binary multirelation $pt2bm.PT$ is BMH-healthy.

Proof. We consider two postconditions ψ_1 and ψ_2, and an initial state s_1.

$\psi_1 \subseteq \psi_2$ [property of sets]

$\Rightarrow \neg\, \psi_2 \subseteq \neg\, \psi_1$ [PT is monotonic]

$\Rightarrow PT.(\neg\, \psi_2) \subseteq PT.(\neg\, \psi_1)$ [property of sets]

$\Rightarrow \neg\, PT.(\neg\, \psi_1) \subseteq \neg\, PT.(\neg\, \psi_2)$ [property of sets]

$\Rightarrow s_1 \in \neg\, PT.(\neg\, \psi_1) \Rightarrow s_1 \in \neg\, PT.(\neg\, \psi_2)$ [definition of $pt2bm$]

$= (s_1, \psi_1) \in pt2bm.PT \Rightarrow (s_1, \psi_2) \in pt2bm.PT$

□

In conclusion, we have a model isomorphic to monotonic predicate transformers. 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 multirelation as a relation between a state on an alphabet $in\alpha$ and a state on $\{\, ac' \,\}$. The value of ac' is the set of angelic choices available to the program: a set of states on an alphabet $out\alpha$.

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

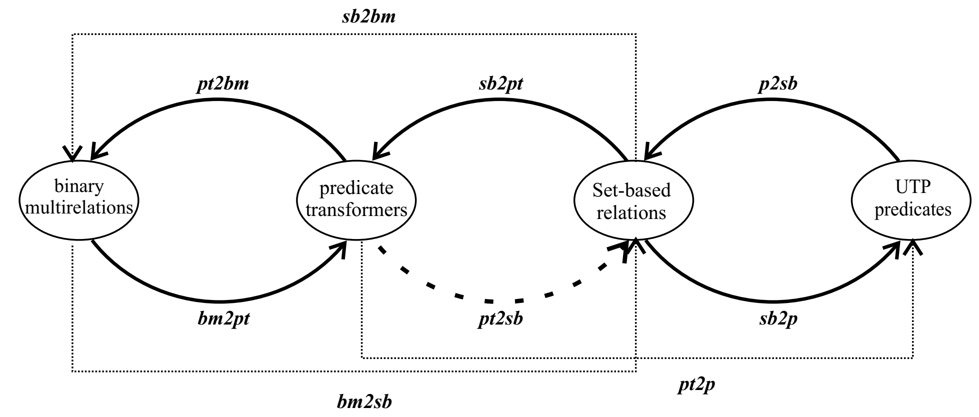

Fig. 2. Extra isomorphism

and set-based relations with alphabet $in\alpha \cup \{ ac' \}$. It is based on the functions below.

Definition 4.2.

$$bm2sb.BM = \{ s_1 : S_{in\alpha};\ s_2 : S_{\{ac'\}} \mid (s_1, s_2.ac') \in BM \}$$
$$sb2bm.ACR = \{ s_1 : S_{in\alpha};\ ss : \mathbb{P}\, S_{out\alpha} \mid (s_1, (ac' \mapsto ss)) \in ACR \}$$

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 ac'; 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 aim at expressing predicate transformers as predicates using $pt2bm$, $bm2sb$, and $sb2p$. In our calculations, we name the composition of $pt2bm$, $bm2sb$, and $sb2p$ as $pt2p \mathrel{\widehat{=}} sb2p \circ bm2sb \circ pt2bm$. The next theorem is useful.

Theorem 4.5. $pt2p.PT = \theta in\alpha \in \neg\ PT.(\neg\ ac')$

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 **true**.

Theorem 4.6. $pt2p.abort = \mathbf{true}$.

Proof.

$pt2p.abort$ [Theorem 4.5]

$= \theta in\alpha \in \neg\ abort.(\neg\ ac')$ [definition of *abort*]

$= \theta in\alpha \in \neg\ \emptyset$ [property of sets]

$= \mathbf{true}$

□

The everywhere miraculous program is represented by **false**. Other relations are considered in Section 5.

4.2. Healthiness condition

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

PBMH $P;\ (ac \subseteq ac') = P$

This requires that, if, after executing P, we execute a program that enlarges ac', then the result could have been obtained by P itself. A healthy P characterises ac' 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 4.7. If BM is BMH-healthy, then $sb2p.(bm2sb.BM)$ is PBMH-healthy.

Proof.

$sb2p.(bm2sb.BM);\ (ac \subseteq ac')$ [definition of $bm2sb$]

$= sb2p.\{\, s_1 : S_{in\alpha};\ s_2 : S_{\{ac'\}} \mid (s_1, s_2.ac') \in BM \,\};\ (ac \subseteq ac')$ [definition of $sb2p$]

$= (\theta in\alpha, \theta\{\, ac' \,\}) \in \{\, s_1 : S_{in\alpha};\ s_2 : S_{\{ac'\}} \mid (s_1, s_2.ac') \in BM \,\};\ (ac \subseteq ac')$ [property of sets]

$= ((\theta in\alpha,\ ac') \in BM);\ (ac \subseteq ac')$ [definition of sequence]

$= \exists\, ac_0 \bullet (\theta in\alpha, ac_0) \in BM \wedge ac_0 \subseteq ac'$ [BM is BMH-healthy and predicate calculus]

$= (\theta in\alpha, ac') \in BM$ [definitions of $sb2p$ and $bm2sb$]

$= sb2p.(bm2sb.BM)$

□

This proof is simpler than that of the corresponding theorem in [CW05].

Theorem 4.8. If P is a PBMH-healthy predicate, then $sb2bm.(p2sb.P)$ is BMH-healthy.

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

$(s_1, \psi_1) \in sb2bm.(p2sb.P)$ [definition of $p2sb$]

$= (s_1, \psi_1) \in sb2bm.\{\, s_1, s_2 \mid P[s_1, s_2/in\alpha, ac'] \,\}$ [definition of $sb2bm$]

$= (s_1, \psi_1) \in \{\, s_1 : S_{in\alpha};\ ss : \mathbb{P}\, S_{\{ac'\}} \mid (s_1, (ac' \mapsto ss)) \in \{\, s_1, s_2 \mid P[s_1, s_2/in\alpha, ac'] \,\}\}$ [property of sets]

$= (s_1, (ac' \mapsto \psi_1)) \in \{\, s_1, s_2 \mid P[s_1, s_2/in\alpha, ac'] \,\}$ [property of sets]

$= P[s_1, \psi_1/in\alpha, ac']$ [P is PBMH-healthy]

$= (P;\ ac \subseteq ac')[s_1, \psi_1/in\alpha, ac']$ [substitution]

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

$= \exists\, ac_0 \bullet P[s_1, ac_0/in\alpha, ac'] \wedge ac_0 \subseteq \psi_1$ [$\psi_1 \subseteq \psi_2$]

$\Rightarrow \exists\, ac_0 \bullet P[s_1, ac_0/in\alpha, ac'] \wedge ac_0 \subseteq \psi_2$ [definition of sequential composition, and substitution]

$= (P;\ (ac \subseteq ac'))[s_1, \psi_2/in\alpha, ac']$ [P is PBMH-healthy]

$= P[s_1, \psi_2/in\alpha, ac']$ [definitions of $p2sb$ and $sb2bm$]

$= (s_1, \psi_2) \in sb2bm.(p2sb.P)$

□

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

4.3. Refinement

The refinement relation is reverse implication, as in all theories of the UTP. We prove that this corresponds to the refinement relation of the model of binary multirelations.

Definition 4.3. $BM_1 \sqsubseteq_{BM} BM_2 \mathrel{\widehat{=}} BM_2 \subseteq BM_1$

The pre-order proposed in [Rew03] for binary multirelations becomes a partial order in the restricted setting of healthy binary multirelations; also, it collapses to set inclusion. We have adopted the inverse order here, which is also the standard definition of refinement for set-based relations.

It is reassuring that this order corresponds to the usual refinement relation in the model of predicate transformers, which we present below.

Definition 4.4. $PT_1 \sqsubseteq_{PT} PT_2 \mathrel{\widehat{=}} \forall \psi \bullet PT_1.\psi \subseteq PT_2.\psi$

The next theorem establishes that the above notions of refinement are indeed compatible.

Theorem 4.9. $BM_1 \sqsubseteq_{BM} BM_2$ if, and only if, $bm2pt.BM_1 \sqsubseteq_{PT} bm2pt.BM_2$.

Proof.

$$\begin{array}{lr}
bm2pt.BM_1 \sqsubseteq_{PT} bm2pt.BM_2 & [\text{definition of } \sqsubseteq_{PT}]\\
= \forall \psi \bullet bm2pt.BM_1.\psi \subseteq bm2pt.BM_2.\psi & [\text{definition of } bm2pt]\\
= \forall \psi \bullet \{ s_1 \mid (s_1, \neg \psi) \notin BM_1\} \subseteq \{ s_1 \mid (s_1, \neg \psi) \notin BM_2\} & [\text{property of sets}]\\
= \forall \psi, s_1 \bullet (s_1, \neg \psi) \notin BM_1 \Rightarrow (s_1, \neg \psi) \notin BM_2 & [\text{property of sets}]\\
= \forall \psi, s_1 \bullet (s_1, \neg \psi) \in BM_2 \Rightarrow (s_1, \neg \psi) \in BM_1 & [\text{predicate calculus}]\\
= \forall \psi, s_1, \phi \mid \phi = \neg \psi \bullet (s_1, \phi) \in BM_2 \Rightarrow (s_1, \phi) \in BM_1 & [\text{property of sets}]\\
= \forall \psi, s_1, \phi \mid \psi = \neg \phi \bullet (s_1, \phi) \in BM_2 \Rightarrow (s_1, \phi) \in BM_1 & [\text{predicate calculus}]\\
= \forall s_1, \phi \bullet (s_1, \phi) \in BM_2 \Rightarrow (s_1, \phi) \in BM_1 & [\text{property of sets}]\\
= BM_2 \subseteq BM_1 & [\text{definition of } \sqsubseteq_{BM}]\\
= BM_1 \sqsubseteq_{BM} BM_2 &
\end{array}$$

□

The correspondence between UTP and binary multirelation refinement is established below.

Theorem 4.10. $P \sqsubseteq Q$ if, and only if, $sb2bm.(p2sb.P) \sqsubseteq_{BM} sb2bm.(p2sb.Q)$.

Proof.

$$\begin{array}{lr}
sb2bm.(p2sb.P) \sqsubseteq_{BM} sb2bm.(p2sb.Q) & [\text{definition of } \sqsubseteq_{BM}]\\
= sb2bm.(p2sb.Q) \subseteq sb2bm.(p2sb.P) & [\text{property of sets}]\\
= \forall s_1, \psi \bullet (s_1, \psi) \in sb2bm.(p2sb.Q) \Rightarrow (s_1, \psi) \in sb2bm.(p2sb.P) & [\text{definitions of } sb2bm \text{ and } p2sb]\\
= \forall s_1, \psi \bullet Q[s_1, \psi / in\alpha, ac'] \Rightarrow P[s_1, \psi / in\alpha, ac'] & [\text{predicate calculus}]\\
= \forall x : in\alpha, ac' \bullet Q \Rightarrow P & [\text{the alphabet is } in\alpha \cup \{ac'\}]\\
= [Q \Rightarrow P] & [\text{definition of refinement in the UTP}]\\
= P \sqsubseteq Q &
\end{array}$$

□

Refinement $\sqsubseteq_R$ in the set-based model of the UTP is also reverse set inclusion, like in the binary multirelation model. That this relation corresponds to the others is not a surprising result; the proof that it corresponds to refinement in the predicate model and in the binary multirelation model, for example, is a direct consequence of subset inclusion properties. In the next section, we use the following result.

Theorem 4.11. $PT_1 \sqsubseteq_{PT} PT_2$ if, and only if, $pt2p.PT_1 \sqsubseteq pt2p.PT_2$.

Proof.

$pt2p.PT_1 \sqsubseteq pt2r.PT_2$ [definition of $pt2p$]

$= sb2p.(bm2sb.(pt2bm.PT_1)) \sqsubseteq sb2p.(bm2sb.(pt2bm.PT_2))$ [definition of $sb2p$ and property of sets]

$= bm2sb.(pt2bm.PT_1) \sqsubseteq_R bm2sb.(pt2bm.PT_2)$ [definition of $bm2sb$ and property of sets]

$= pt2bm.PT_1 \sqsubseteq_{BM} pt2bm.PT_2$ [Theorem 4.9]

$= bm2pt.(pt2bm.PT_1) \sqsubseteq_{PT} bm2pt.(pt2bm.PT_2)$ [Theorem 4.2]

$= PT_1 \sqsubseteq_{PT} PT_2$

□

Now, we have a UTP theory that corresponds to monotonic predicate transformers. In the next section, we explore the definition of the operators in our new theory; besides angelic choice, we consider operators defined the general theory of UTP relations and designs.

5. Operators

We have already calculated the definition of *abort* in our new theory; the calculation for *miracle* is equally simple. In this section, we use the function $pt2p$ to justify the definitions of other relations and relational operators in our theory of angelic nondeterminism.

5.1. Choice: angelic and demonic

Of course, angelic choice $P \sqcup Q$ is the first operator of interest. In the predicate transformer model, it is characterised by disjunction (or union), which is the least upper bound operator. In our new UTP theory, it is characterised by conjunction. 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 5.1. $pt2p.(P \sqcup Q) = pt2p.P \wedge pt2p.Q$

Proof.

$pt2p.(P \sqcup Q)$ [Theorem 4.5]

$= \theta in\alpha \in \neg\,(P \sqcup Q).(\neg\, ac')$ [predicate transformer semantics of $\sqcup$]

$= \theta in\alpha \in \neg\,(P.(\neg\, ac') \vee Q.(\neg\, ac'))$ [property of sets]

$= \theta in\alpha \in \neg\, P.(\neg\, ac') \cap \neg\, Q.(\neg\, ac')$ [property of sets]

$= \theta in\alpha \in \neg\, P.(\neg\, ac') \wedge \theta in\alpha \in \neg\, Q.(\neg\, ac')$ [Theorem 4.5]

$= pt2p.P \wedge pt2p.Q$

□

Like in the original UTP model, demonic choice is captured by disjunction. In the predicate transformer model, it 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 5.2. $pt2p.(P \sqcap Q) = pt2p.P \lor pt2p.Q$

Proof. Similar to that of Theorem 5.1. $\square$

Logical constants are defined as the least upper bound operator in the complete lattice of monotonic predicate transformers [GM91], which is equal to that in the complete boolean lattice of predicate transformers [BW90]. It also corresponds to the least upper bound operator in our theory: universal quantification.

Theorem 5.3. $pt2p.(\mathbf{con} X \bullet P(X)) = \forall X \bullet pt2p.P(X)$

Proof.

$$
\begin{aligned}
& pt2p.(\mathbf{con} X \bullet P(X)) && \text{[Theorem 4.5]}\\
&= \theta in\alpha \in \neg\, (\mathbf{con} X \bullet P(X)).(\neg\, ac') && \text{[predicate transformer semantics of } \mathbf{con}\text{]}\\
&= \theta in\alpha \in \neg\, (\bigsqcup\{\, X \bullet P(X)\,\}).(\neg\, ac') && \text{[property of lattice of predicate transformers]}\\
&= \theta in\alpha \in \bigcap\{\, X \bullet \neg\, P(X).(\neg\, ac')\,\} && \text{[property of sets]}\\
&= \forall X \bullet \theta in\alpha \in \neg\, P(X).(\neg\, ac') && \text{[Theorem 4.5]}\\
&= \forall X \bullet pt2p.P(X)
\end{aligned}
$$

$\square$

A similar proof establishes that the greatest lower bound operator in the predicate transformer model corresponds to the greatest lower bound operator in our theory.

5.2. Assignment

Assignment can be defined as follows as a predicate transformer.

$$(x := e).\psi = \{\, s \mid s' \oplus \{x' \mapsto e\} \in \psi \,\} \tag{2}$$

This corresponds to substitution: the standard weakest precondition semantics of assignment, but it is expressed using sets. Moreover, there is a slight complication due to the fact that postconditions and preconditions are predicates on different variables, or rather, states on $in\alpha$ and on $out\alpha$. A similar weakest precondition semantics is considered in [CW99] for Z. In the above notation, $x := e$ is guaranteed to establish ψ when executed in an initial state s, if the final state $s' \oplus \{x' \mapsto e\}$ obtained by dashing the variables of s and associating e to x' belongs to ψ.

The theorem below gives a definition for assignment in our UTP theory of angelic nondeterminism.

Theorem 5.4. $pt2p.(x := e) = (\theta in\alpha)' \oplus \{x' \mapsto e\} \in ac'$

Proof.

$$
\begin{aligned}
& pt2p.(x := e) && \text{[Theorem 4.5]}\\
&= \theta in\alpha \in \neg\, ((x := e).ac') && \text{[predicate transformer semantics of } x := e \text{ (2)]}\\
&= \theta in\alpha \in \neg\, \{\, s \mid s' \oplus \{x' \mapsto e\} \in \neg\, ac' \,\} && \text{[property of sets]}\\
&= (\theta in\alpha)' \oplus \{x' \mapsto e\} \in ac'
\end{aligned}
$$

$\square$

The assignment is a deterministic command, which does not really involve either demonic or angelic choices. Therefore, the uniquely determined final state of the assignment is in all sets of angelic choices available for demonic choice. Moreover, since any set that includes that final state is available for demonic choice, the angelic choice can provide no interesting guarantees.

5.3. Conditional

We consider the conditional command $P \lhd b \rhd Q$, which behaves like P, if the condition b holds, and like Q otherwise. This is the form of conditional studied in the UTP, where b is a condition: a predicate over the input alphabet, only. To convert b to a set, we use the function $c2sb$, which is similar to $p2sb$, but it results in sets of states, instead of sets of pairs of states; its inverse is $sb2c$, which is similar to $sb2p$.

Definition 5.1. $c2sb.P = \{ s \mid P[s/\alpha] \}$
$sb2c.b = \theta in\alpha \in b$

It is not difficult to establish an isomorphism between sets of states and conditions based on $c2sb$ and $sb2c$.

The semantics of conditionals in our new theory is the subject of the next theorem.

Theorem 5.5. $pt2p.(P \lhd b \rhd Q) = (b \Rightarrow pt2p.P) \wedge (\neg\ b \Rightarrow pt2p.Q)$

Proof.

$pt2p.(P \lhd b \rhd Q)$ [Theorem 4.5]

$= \theta in\alpha \in \neg\ (P \lhd b \rhd Q).(\neg\ ac')$ [predicate transformer semantics of $P \lhd b \rhd Q$]

$= \theta in\alpha \in \neg\ (c2sb.b \cap P.(\neg\ ac')\ \cup\ \neg\ c2sb.b \cap Q.(\neg\ ac'))$ [property of sets]

$= \theta in\alpha \in \neg\ (c2sb.b \cap P.(\neg\ ac'))\ \wedge\ \theta in\alpha \in \neg\ (\neg\ c2sb.b \cap Q.(\neg\ ac'))$ [property of sets]

$= (\theta in\alpha \in \neg\ c2sb.b \vee \theta in\alpha \in \neg\ P.(\neg\ ac')) \wedge (\theta in\alpha \in c2sb.b \vee \theta in\alpha \in \neg\ Q.(\neg\ ac'))$
[property of sets and definition of $sb2c$]

$= (\neg\ sb2c.(c2sb.b) \vee \theta in\alpha \in \neg\ P.(\neg\ ac')) \wedge (sb2c.(c2sb.b) \vee \theta in\alpha \in \neg\ Q.(\neg\ ac'))$
[definitions of $sb2c$ and $c2sb$]

$= (\neg\ b \vee \theta in\alpha \in \neg\ P.(\neg\ ac')) \wedge (b \vee \theta in\alpha \in \neg\ Q.(\neg\ ac'))$ [definition of $pt2p$]

$= (\neg\ b \vee pt2p.P) \wedge (b \vee pt2p.Q)$ [predicate calculus]

$= (b \Rightarrow pt2p.P) \wedge (\neg\ b \Rightarrow pt2p.Q)$

□

Basically, the semantics of conditional is the same as that in the general theory of relations.

5.4. Sequence

Sequential composition cannot correspond to relational composition, since the relations are not homogeneous. We provide here a much simpler definition than that suggested in [CW05], though.

The weakest precondition semantics of sequence is function composition. In our setting, since preconditions are over states on $in\alpha$ and postconditions are over states on outα, the composition is not direct. The definition is as follows.

$$(P;\ Q).\psi = P.(Q.\psi)' \tag{3}$$

As usual, the weakest precondition for $P;\ Q$ to establish ψ is the weakest precondition for P to establish the weakest precondition for Q to establish ψ. However, the weakest precondition for Q to establish ψ is not a postcondition, since it is a set of initial states. The corresponding postcondition is $(Q.\psi)'$. For a set of initial states ss, the set ss' contains states s', for each initial state s in ss; more formally, in terms of the relational image operator: $ss' = \_' (\!|\ ss\ |\!)$.

In the context of our UTP theory, the definition can be surprisingly simple. The definition of sequence for binary multirelations is very intuitive.

$$BM_1;\ BM_2 = \{ s_1, ss \mid \exists\, ss_0 \bullet (s_1, ss_0') \in BM_1 \wedge ss_0 \subseteq \{ s_1 \mid (s_1, ss) \in BM_2 \} \} \tag{4}$$

An initial state s_1 is associated to a set of angelic choices ss in $(BM_1;\ BM_2)$ if BM_1 associates s_1 to a set

ss_0' of angelic choices such that, whatever state from ss_0 is chosen, the execution of BM_2 from that state may lead to the availability of ss for angelic choice. This is the definition in [Rew03]. For healthy binary multirelations, it can be simplified as shown below.

$$BM_1;\ BM_2 = \{\, s_1, ss \mid (s_1, \{\, s_1 \mid (s_1, ss) \in BM_2 \,\}') \in BM_1 \} \tag{5}$$

In words, the set of angelic choices ss is available for $(BM_1;\ BM_2)$ from an initial state s_1 if all the initial states of BM_2 from which ss is available is a set of angelic choices available for BM_1 from s_1. This can be expressed in the predicative theory using substitution.

Theorem 5.6. $pt2p.(P;\ Q) = (pt2p.P)[\{\, s' \mid (pt2p.Q)[s/in\alpha] \,\}/ac']$

Proof.

$$\begin{array}{ll}
pt2p.(P;\ Q) & [\text{Theorem 4.5}]\\
= \theta in\alpha \in \neg\ (P;\ Q).(\neg\ ac') & [\text{predicate transformer semantics of sequence (3)}]\\
= \theta in\alpha \in \neg\ P.(Q.(\neg\ ac'))' & [\text{property of sets}]\\
= \theta in\alpha \in \neg\ P.(\neg\ (\neg\ Q.(\neg\ ac'))') & [\text{property of sets}]\\
= \theta in\alpha \in \neg\ P.(\neg\ \{\, s \mid s \in \neg\ Q.(\neg\ ac') \}') & [\text{property of substitution}]\\
= \theta in\alpha \in \neg\ P.(\neg\ \{\, s' \mid (\theta in\alpha \in \neg\ Q.(\neg\ ac'))[s/in\alpha] \}) & [\text{Theorem 4.5}]\\
= \theta in\alpha \in \neg\ P.(\neg\ \{\, s' \mid (pt2p.Q)[s/in\alpha] \}) & [\text{property of substitution}]\\
= (\theta in\alpha \in \neg\ P.(\neg\ ac'))[\{\, s' \mid (pt2p.Q)[s/in\alpha] \}/ac'] & [\text{Theorem 4.5}]\\
= (pt2p.P)[\{\, s' \mid (pt2p.Q)[s/in\alpha] \,\}/ac'] &
\end{array}$$

□

In conclusion, this theorem supports the following definition for sequence.

$$P;\ Q \mathrel{\widehat=} P[\,\{\, s' \mid Q[s/in\alpha] \,\}/ac'\,]$$

It states that a set of angelic choices ac' for $P;\ Q$ is a set that is available for Q when it is executed in any of the states s of a set of angelic choices for P. This is a definition that is possibly not obvious, but could be calculated using the isomorphism between predicate transformers and the UTP predicative theory.

An example of a simple sequence of two assignments can be illuminating; we consider $x := 2;\ x := x + 1$. We assume that x is the only variable in the input alphabet.

$$\begin{array}{ll}
x := 2;\ x := x + 1 & [\text{semantics of assignment and sequence}]\\
= ((x' \mapsto 2) \in ac')[\{\, s' \mid ((x' \mapsto x + 1) \in ac')[s/x] \,\}/ac'] & [\text{property of substitution}]\\
= ((x' \mapsto 2) \in ac')[\{\, s' \mid (x' \mapsto s.x + 1) \in ac' \,\}/ac'] & [\text{property of substitution}]\\
= (x' \mapsto 2) \in \{\, s' \mid (x' \mapsto s.x + 1) \in ac' \,\} & [\text{property of sets}]\\
= (x' \mapsto (x \mapsto 2).x + 1) \in ac' & [\text{property of states}]\\
= (x' \mapsto 2 + 1) \in ac' & [\text{semantics of assignment}]\\
= x := 3 &
\end{array}$$

As should be expected, the sequence of assignments is equivalent to $x := 3$. In our second example, we consider a sequence involving an angelic choice.

$$\begin{array}{ll}
(x := 0 \sqcup x := 1);\ x := x + 1 & [\text{semantics of assignment, angelic choice, and sequence}]\\
= ((x' \mapsto 0) \in ac' \wedge (x' \mapsto 1) \in ac')[\{\, s' \mid (x' \mapsto s.x + 1) \in ac' \,\}/ac'] & [\text{property of substitution}]\\
= (x' \mapsto 0) \in \{\, s' \mid (x' \mapsto s.x + 1) \in ac' \,\} \wedge (x' \mapsto 1) \in \{\, s' \mid (x' \mapsto s.x + 1) \in ac' \,\} & [\text{property of sets}]\\
= (x' \mapsto (x \mapsto 0).x + 1) \in ac' \wedge (x' \mapsto (x \mapsto 1).x + 1) \in ac' & [\text{property of states}]\\
= (x' \mapsto 1) \in ac' \wedge (x' \mapsto 2) \in ac' & [\text{semantics of assignment and angelic choice}]\\
= x := 1 \sqcup x := 2 &
\end{array}$$

Since the angelic choice $x := 0 \sqcup x := 1$ is followed by an assignment that increments x, the program actually guarantees x to take the value 1 or 2, as required.

5.5. Recursion

Finally, we consider recursion.

Theorem 5.7. $pt2p.(\mu\, X \bullet F(X))) = \mu\, X \bullet (pt2p.F))(X)$

Proof.

$$
\begin{aligned}
& pt2p.(\mu\, X \bullet F(X)) && \text{[property of } \mu\text{]}\\
&= pt2p.(\sqcap\{\, X \mid F(X)\ \sqsubseteq_{PT} X \,\}) && \text{[Theorem 5.3]}\\
&= \sqcap\{\, pt2p.X \mid F(X)\ \sqsubseteq_{PT} X \,\} && \text{[Theorem 4.11]}\\
&= \sqcap\{\, pt2p.X \mid pt2p.F(X)\ \sqsubseteq pt2p.X \,\} && \text{[property of sets]}\\
&= \sqcap\{\, X \mid (pt2p.F)(X)\ \sqsubseteq X \,\} && \text{[property of } \mu\text{]}\\
&= \mu\, X \bullet (pt2p.F))(X)
\end{aligned}
$$

□

As usual, recursion is given by the least fixed point operator.

5.6. Designs

The theory of angelic nondeterminism captures termination; this should not come as a surprise since we have a model isomorphic to monotonic predicate transformers. A program that aborts includes the empty set as an option for demonic choice. For example, we have already established that *abort* is **true**; on the other hand, the program that can lead to an arbitrary final state, but always terminates is $ac' \neq \emptyset$.

In particular, *abort* is the left zero for sequence.

Theorem 5.8. $abort;\ Q = abort$

Proof.

$$
\begin{aligned}
& abort;\ Q && \text{[definitions of } abort \text{ and sequence]}\\
&= \mathbf{true}[\{\, s' \mid Q[s/in\alpha] \,\}/ac'] && \text{[property of substitution]}\\
&= \mathbf{true} && \text{[definition of } abort\text{]}\\
&= abort
\end{aligned}
$$

□

As as consequence of this result, the paradox that motivated the definition of the theory of designs is not a concern in our theory. Therefore, there is no need to include the extra observational variables; at least, not just to model termination.

In order to give the weakest precondition semantics of a design, we need to define universal quantification and alphabet extension for predicates defined as sets of states. The usual semantics of designs, or rather, of specifications given by a precondition and a postcondition, is as follows.

$$(P \vdash Q).\psi = P \cap \overline{\forall} out\alpha \bullet \neg\, Q \cup (\psi \dagger in\alpha) \tag{6}$$

This is basically in direct correspondence with the perhaps more familiar predicative definition; a similar set-based formulation is used in [CN02]. The definition of universal quantification is as follows.

$$s \in (\overline{\forall} x \bullet P) = \forall\, v \bullet s \oplus \{x \mapsto v\} \in P \tag{7}$$

In (6) we use a universal quantification over the whole output alphabet; the extension of the above definition for sets of variables is straightforward. Also, in (6), Q is a set of states over the joint alphabet $in\alpha \cup out\alpha$; the postcondition ψ, however, is a set of states on $out\alpha$. We use the † operator to extend the alphabet of ψ to $in\alpha \cup out\alpha$. Its definition is as follows; basically, the values of the extra variables are left unconstrained.

$$s \in (P \dagger x) = \{ x \} \lhd s \in P \tag{8}$$

Again, in (6) we apply † to a set of names $in\alpha$, instead of to a single variable x. The definition above can be extended in the obvious way.

The next theorem gives a semantics for designs in our new theory. We take P and Q to be predicates, and use $c2sb$ to convert then to sets of states.

Theorem 5.9. $pt2p.(P \vdash Q) = P \Rightarrow \exists\, out\alpha \bullet Q \land \theta out\alpha \in ac'$

Proof.

$$\begin{aligned}
& pt2p.(P \vdash Q) && [\text{definition of } pt2p]\\
&= \theta in\alpha \in \neg\,(P \vdash Q).(\neg\, ac') && [\text{predicate transformer semantics of designs}]\\
&= \theta in\alpha \in \neg\,(c2sb.P \cap \overline{\forall} out\alpha \bullet \neg\, c2sb.Q \cup (\neg\, ac' \dagger in\alpha)) && [\text{property of sets}]\\
&= \theta in\alpha \in \neg\, c2sb.P \lor \theta in\alpha \notin \overline{\forall} out\alpha \bullet \neg\, c2sb.Q \cup (\neg\, ac' \dagger in\alpha) && [\text{definition of } sb2c]\\
&= \neg\, sb2c.(c2sb.P) \lor \theta in\alpha \notin \overline{\forall} out\alpha \bullet \neg\, c2sb.Q \cup (\neg\, ac' \dagger in\alpha) && [sbc2.(c2sb.P) = P]\\
&= \neg\, P \lor \theta in\alpha \notin \overline{\forall} out\alpha \bullet \neg\, c2sb.Q \cup (\neg\, ac' \dagger in\alpha) && [\text{definition of } \overline{\forall}\ (7) \text{ and predicate calculus}]\\
&= \neg\, P \lor \exists\, v \bullet \theta in\alpha \oplus \{out\alpha \mapsto v\} \notin \neg\, c2sb.Q \cup (\neg\, ac' \dagger in\alpha) && [\text{property of sets}]\\
&= \neg\, P \lor \exists\, v \bullet \theta in\alpha \oplus \{out\alpha \mapsto v\} \in c2sb.Q \land \theta in\alpha \oplus \{out\alpha \mapsto v\} \in \neg\,(\neg\, ac' \dagger in\alpha)\\
& && [\text{definition of } \dagger\ (8) \text{ and property of sets}]\\
&= \neg\, P \lor \exists\, v \bullet \theta in\alpha \oplus \{out\alpha \mapsto v\} \in c2sb.Q \land \theta in\alpha \oplus \{out\alpha \mapsto v\} \in (ac' \dagger in\alpha) && [\text{definition of } \dagger\ (8)]\\
&= \neg\, P \lor \exists\, v \bullet \theta in\alpha \oplus \{out\alpha \mapsto v\} \in c2sb.Q \land in\alpha \lhd \theta in\alpha \oplus \{out\alpha \mapsto v\} \in ac' && [\text{property of } \lhd]\\
&= \neg\, P \lor \exists\, v \bullet \theta in\alpha \oplus \{out\alpha \mapsto v\} \in c2sb.Q \land \{out\alpha \mapsto v\} \in ac' && [\text{predicate calculus}]\\
&= \neg\, P \lor \exists\, out\alpha \bullet \theta in\alpha \oplus \theta out\alpha \in c2sb.Q \land \theta out\alpha \in ac' && [\text{definition of } sb2c]\\
&= \neg\, P \lor \exists\, out\alpha \bullet sb2c.(c2sb.Q) \land \theta out\alpha \in ac' && [sbc2.(c2sb.P) = P]\\
&= \neg\, P \lor \exists\, out\alpha \bullet Q \land \theta out\alpha \in ac' && [\text{predicate calculus}]\\
&= P \Rightarrow \exists\, out\alpha \bullet Q \land \theta out\alpha \in ac'
\end{aligned}$$

□

In words, if P holds, then ac' is any set that contains a state that satisfies Q; the nondeterminism in a design is demonic. We observe that Q is not a predicate over $in\alpha \cup ac'$, but over $in\alpha \cup out\alpha$, where $out\alpha$ is the alphabet of the states in ac'.

6. 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 theories of alphabetised relations and 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 [Rew03]. 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 *Circus* [WC02], our combined formalism, and prove refinement laws.

It is unavoidable that the definition of sequence is more complicated than that 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. Moreover, our calculations revealed a tractable definition based on substitution.

In [BW98], 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_1 to the set of postconditions that can be satisfied when P is executed in s_1. The choice semantics is, of course, isomorphic to binary multirelations. Since in 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 [MGW96] 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 [MCR04], 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 [Rew03] and [MCR04] 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. Moreover, our comparatively simple definition of sequence takes advantage of the healthiness condition of the model of binary multirelations. We also go further in that we consider logical constants, recursion, assignments, conditionals, and designs.

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

[BW89] 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 *Lecture Notes in Computer Science*, pages 139 – 156, Groningen, The Netherlands, 1989. Springer-Verlag.

[BW90] R. J. R. Back and J. Wright. Duality in Specification Languages: A Lattice-theoretical Approach. *Acta Informatica*, 27(7):583 – 625, 1990.

[BW92] R. J. R. Back and J. Wright. Combining angels, demons and miracles in program specifications. *Theoretical Computer Science*, 100:365 – 383, 1992.

[BW98] R. J. R. Back and J. Wright. *Refinement Calculus: A Systematic Introduction.* Graduate Texts in Computer Science. Springer-Verlag, 1998.

[CN02] A. L. C. Cavalcanti and D. A. Naumann. Forward simulation for data refinement of classes. In L. Eriksson and P. A. Lindsay, editors, *FME 2002: Formal Methods — Getting IT Right*, volume 2391 of *Lecture Notes in Computer Science*, pages 471 – 490. Springer-Verlag, 2002.

[CSW03] 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.

[CW98] A. L. C. Cavalcanti and J. C. P. Woodcock. A Weakest Precondition Semantics for Z. *The Computer Journal*, 41(1):1 – 15, 1998.

[CW99] A. L. C. Cavalcanti and J. C. P. Woodcock. ZRC—A Refinement Calculus for Z. *Formal Aspects of Computing*, 10(3):267—289, 1999.

[CW04] A. L. C. Cavalcanti and J. C. P. Woodcock. Angelic Nondeterminism and Unifying Theories of Programming (Extended Version). Technical report, University of Kent - Computing Laboratory, 2004.

[CW05] A. L. C. Cavalcanti and J. C. P. Woodcock. Angelic Nondeterminism and Unifying Theories of Programming . In J. Derrick and E. Boiten, editors, *REFINE 2005*, volume 137 of *Eletronic Notes in Theoretical Computer Science*. Elsevier, 2005.

[Dij76] E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.

[Dun01] 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.

[GM91] P. H. B. Gardiner and C. C. Morgan. Data Refinement of Predicate Transformers. *Theoretical Computer Science*, 87:143 – 162, 1991.

[Hes92] 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.

[HH85] C. A. R. Hoare and Jifeng He. The Weakest Prespecification. Technical Monograph TM-PRG-44, Oxford University Computing Laboratory, Oxford – UK, 1985.

[HJ98] C. A. R. Hoare and He Jifeng. *Unifying Theories of Programming*. Prentice-Hall, 1998.

[JSS91] R. Jagadeesan, V. Shanbhogue, and V. Saraswat. Angelic non-determinism in concurrent constraint programming. Technical report, Xerox Park, January 1991.

[MCR04] C. E. Martin, S. A. Curtis, and I. Rewitzky. Modelling Nondeterminism. In *Mathematics of Program Construction*, Lecture Notes in Computer Science, pages 228 – 251, 2004.

[MG90] C. C. Morgan and P. H. B. Gardiner. Data Refinement by Calculation. *Acta Informatica*, 27(6):481—503, 1990.

[MGW96] A. P. Martin, P. H. B. Gardiner, and J. C. P. Woodcock. A Tactical Calculus. *Formal Aspects of Computing*, 8(4):479–489, 1996.

[Mor94] C. C. Morgan. *Programming from Specifications*. Prentice-Hall, 2nd edition, 1994.

[Rew03] 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 *Lecture Notes in Computer Science*, pages 256 – 271, 2003.

[WC02] 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 *Lecture Notes in Computer Science*, pages 184 – 203. Springer-Verlag, 2002.

[WC04] J. C. P. Woodcock and A. L. C. Cavalcanti. A Tutorial Introduction to Designs in Unifying Theories of Programming. In E. A. Boiten, J. Derrick, and G. Smith, editors, *IFM 2004: Integrated Formal Methods*, volume 2999 of *Lecture Notes in Computer Science*, pages 40 – 66. Springer-Verlag, 2004. Invited tutorial.