Angelic Processes

Pedro Fernando de Oliveira Salazar Ribeiro

Doctor of Philosophy

University of York

Computer Science

December 2014

Abstract

In the formal modelling of systems, demonic and angelic nondeterminism play fundamental roles as abstraction mechanisms. The angelic nature of a choice pertains to the property of avoiding failure whenever possible. As a concept, angelic choice first appeared in automata theory and Turing machines, where it can be implemented via backtracking. It has traditionally been studied in the refinement calculus, and has proved to be useful in a variety of applications and refinement techniques. Recently it has been studied within relational, multirelational and higher-order models. It has been employed for modelling user interactions, game-like scenarios, theorem proving tactics, constraint satisfaction problems and control systems.

When the formal modelling of state-rich reactive systems is considered, it only seems natural that both types of nondeterministic choice should be considered. However, despite several treatments of angelic nondeterminism in the context of process algebras, namely Communicating Sequential Processes, the counterpart to the angelic choice of the refinement calculus has been elusive.

In this thesis, we develop a semantics in the relational setting of Hoare and He's Unifying Theories of Programming that enables the characterisation of angelic nondeterminism in CSP. Since CSP processes are given semantics in the UTP via designs, that is, pre and postcondition pairs, we first introduce a theory of angelic designs, and an isomorphic multirelational model, that is suitable for characterising processes. We then develop a theory of reactive angelic designs by enforcing the healthiness conditions of CSP. Finally, by introducing a notion of divergence that can undo the history of events, we obtain a model where angelic choice avoids divergence. This lays the foundation for a process algebra with both nondeterministic constructs, where existing and novel abstract modelling approaches can be considered. The UTP basis of our work makes it applicable in the wider context of reactive systems.

Contents

List of Tables

List of Figures

Acknowledgements

Although the work presented in this thesis is mainly the result of three years of hard work, passion and dedication, there is a whole lot more to tell, remember and appreciate. I truly feel that the fact that I was surrounded by the best inspiring minds is the main reason why my seven year-long journey into Computer Science continues to flourish to this day.

I would like to thank my supervisor, Ana Cavalcanti, for her continued support and prompt attention to detail, which in the context of this work, has been of utmost importance. Our insightful discussions have always been extremely productive and have often led to new ideas for future work. I would also like to thank Jim Woodcock, Frank Zeyda and Simon Foster for their insightful discussions and suggestions regarding my work. Frank has not only been a friend and a source of inspiration, but also extremely knowledgeable and helpful in discussing technical aspects, like those of the UTP. I would also like to thank my examiners Professor Steve Schneider and Dr. Andrew Butterfield for their professional, rigorous and helpful feedback. In addition, I would like to thank my assessor Dr. Detlef Plump for his prompt feedback and positive contributions over the past three years. I am also grateful for the financial support from EPSRC, UK, which gave me an invaluable sense of comfort in my daily life.

I would not have been able to reach this milestone if it were not for my parents, whose unconditional support for my dreams has been pivotal since an early age. My childhood curiosity has grown and with it so have my science and technology dreams. Despite the geographical distance, their encouragement has always played a key role in being able to study abroad, and for that I will be eternally grateful.

My partner Zhishuang Chen has been an essential source of inspiration and support. It is thanks to her unconditional love and support that I have made through some of the most anxious and tough times during the course of this degree. Being a PhD student herself, I hope to be able to equally and positively contribute towards her achievements.

Finally, I would also like to thank the following friends, and in no particular order, who have in one way or another, contributed positively to my well-being while living and studying in York: Artur Goulão Ferreira, Frank Soboczenski, Theodora Lee, Ruofan Jin, Miguel Prôa and Luis Carlos Rodrigues.

Author's Declaration

I hereby declare that the work presented in this thesis is based on my original contributions, unless otherwise stated. The following material has been previously published.

[1] P. Ribeiro and A. Cavalcanti, "Designs with Angelic Nondeterminism," in *Theoretical Aspects of Software Engineering (TASE), 2013 International Symposium on.* IEEE, 2013, pp. 71–78.

[2] ——, "Angelicism in the Theory of Reactive Processes," in *Unifying Theories of Programming*, ser. Lecture Notes in Computer Science, D. Naumann, Ed. Springer International Publishing, 2015, vol. 8963, pp. 42–61.

[3] ——, "UTP Designs for Binary Multirelations," in *Theoretical Aspects of Computing ICTAC 2014*, ser. Lecture Notes in Computer Science, G. Ciobanu and D. Méry, Eds. Springer International Publishing, 2014, vol. 8687, pp. 388–405.

Chapter 1

Introduction

In this chapter we discuss the motivation and objectives underlying our work on a semantic model for CSP processes with angelic nondeterminism. Furthermore, we provide an overview of all semantic models of interest in the context of this thesis and their relationships. Finally, an outline of this document's structure is presented.

1.1 Motivation

In an increasingly connected world, where software-driven systems are ubiquitous, it is imperative that their behaviour is rigorously studied. Since the software crisis of the seventies [4], significant attention has been devoted to this problem with the development of several theories, techniques and tools. The earliest contributions can be found in the works of Floyd, Hoare and Dijkstra. In 1967, Floyd [5] proposed techniques for rigorously characterizing and analysing programs specified as flowcharts, by considering propositions associated with the entrance and exit of commands in the flowchart, akin to pre and postconditions. Hoare [6] would later propose a formal system, known as Hoare logic, capable of proving partial correctness of program statements for a sequential programming language. Inspired by Hoare's work, Dijkstra [7] introduced weakest precondition semantics with his language of guarded commands, an imperative language that allows for the existence of repetitive and nondeterministic constructs.

As systems present several aspects of interest, ranging from the intended functional behaviour to the actual operating environment, modelling approaches focus on specific properties of interest, at suitable levels of abstraction. For instance, there are several formal notations catering for the specification of functional behaviour, such as Z [8, 9], Object-Z [10], Vienna Development Method (VDM) [11], Abstract State Machine (ASM) [12, 13] and B [14, 15]. Concurrent and reactive systems have also been extensively studied with formalisms such as Communicating Sequential Processes (CSP) [16–18], Calculus of Concurrent Systems (CCS) [19] and Algebra of Communicating Processes (ACP) [20]. Several works have also focused on combining both state-based and concurrent formalisms as found in the literature [10, 21–27].

The successful characterisation of a particular system relies on appropriate abstraction mechanisms being available, such that a system can be decomposed into manageable parts with the appropriate level of detail. Formal specifications are, in this sense, at the very top of the hierarchy, and provide the highest-level and most abstract model of a system. Since the foundational works of Back [28], Morris [29] and Morgan [30], however, it has been possible to consider both specifications and programs within the same formal model.

An essential abstraction mechanism that is pervasive across modelling approaches is that of nondeterministic choice. It can be used to specify purely nondeterministic behaviour, such that no particular choice is guaranteed, but also to describe concisely a set of choices, such that, if there are options that lead to success, they are guaranteed to be chosen. The former is traditionally referred to as being demonic, while the latter is referred to as angelic. Operationally, both nondeterministic choices embody some notion of failure, and success.

Demonic choice has traditionally been used for the underspecification of behaviour, and plays an essential role in the contractual approach between users and developers. In the context of refinement, the behaviour of a specification can be made more deterministic while adhering to the externally observable behaviour. In other words, given a particular set of choices, the user is unable to force any particular choice and must accept any subset, including failure, if this is a possibility. This corresponds to the semantics of nondeterminism in Dijkstra's [7] guarded commands, and internal choice in CSP [17], for example.

On the other hand, angelic choice is driven by success. Given a set of choices, as long as there is at least one choice that leads to success, then the angel can achieve a satisfying outcome. Thus, operationally, angelic nondeterminism can be interpreted as a backtracking mechanism. Indeed this is similar to the underlying concept involved in searching for solutions in a given space. Another typical application of this concept can be found in the context of nondeterministic finite state automatons, where acceptance is successful if, and only if, the system reaches an accepting state.

The concept of angelic nondeterminism has traditionally been studied in the refinement calculus [29, 31, 32], where angelic choice is defined as the least upper bound of the lattice of monotonic predicate transformers. Its dual is demonic choice, which is defined as the greatest lower bound of the lattice. In [33, 34] the least upper bound is used to define logical variables, which enable the postcondition of a specification statement to refer to the initial value of a program variable. This is central to the refinement technique of Gardiner and Morgan [33], and, in particular, to their calculational data-refinement approach.

In [35] Rewitzky introduces binary multirelations for modelling both forms of nondeterminism. Unlike relational models, which relate initial states to final states, multirelations relate initial states to sets of final states. A number of models are explored in [36], of which the model of upward-closed binary multirelations is the most important as it has a lattice-theoretic structure. A generalised algebraic structure has also been proposed by Guttmann [37], where the monotonic predicate transformers and multirelations are characterised as instances.

Cavalcanti et al. [38] have proposed a predicative encoding of binary multirelations in the context of Hoare and He's [39] Unifying Theories of Programming (UTP), a relational framework suitable for characterising several programming paradigms. This is achieved by encoding program variables as record components. First an isomorphism is established between the new UTP model and a set-based relational model. Afterwards an isomorphism is established between the set-based model and the monotonic predicate transformers. Finally an isomorphism is established between the predicate transformers model and upward-closed binary multirelations. This is then used to establish the correspondence between the semantics of statements in the predicate transformers model and in the proposed UTP model.

Angelic choice has also been considered at the expression, or term, level by Morris [40, 41]. In [41], an axiomatic basis is presented for defining operators for both angelic and demonic nondeterminism within a term language. Each type is represented as a partially ordered set, and an ordering is given. This is then lifted into a Free Completely Distributive (FCD) lattice where the refinement relation corresponds to the ordering relation imposed on the type, demonic choice is the meet, and angelic choice is the join. In [40] this model is shown to be isomorphic to higher-order models of predicate transformers, binary multirelations and state transformers. While it is possible to cast typical sequential programming constructs into this theory, its focus is on functional languages. Hesselink [42] further studies this model and provides a different construction of the FCD.

In [43], Tyrrell et al., inspired on the previous work on the FCD by Morris [41], provide an axiomatization for an algebra, similar to CSP, where external choice is referred to as "angelic choice". The definitions are then lifted from a partially ordered set into the FCD lattice. Just as the authors point out, this model is quite different from the traditional CSP model whose complete semantics is based on failures-divergences [17, 18]. In the model proposed, *Stop* is the bottom of the refinement ordering, rather than divergence. Thus, it is impossible to distinguish divergence from deadlock.

Roscoe [18] has proposed an angelic choice operator $P \boxdot Q$ through an operational combinator semantics for CSP. It is an alternative to the external choice operator of CSP that behaves as follows: as long as the environment chooses events offered by both P and Q, then the choice between P and Q is unresolved. The possibility of divergence or otherwise has no effect on the choice.

Despite the various models where angelic nondeterminism is employed in the context of process algebras, and the different semantics considered in the literature [18, 43], the counterpart to the angelic choice of the refinement calculus has been elusive. The notion of failure of interest here is that of divergence as required for a characterisation of angelic nondeterminism in the context of state-rich reactive systems for both data and behavioural refinement.

The UTP of Hoare and He [39] provides an ideal framework to study the concept of angelic nondeterminism in a theory of CSP [39, 44]. The UTP is a predicative framework of alphabetized relations suitable for characterising different programming aspects, such as functionality, concurrency, logic programming, higher-order programming, object-orientation [45, 46], pointers [47], time [48–50] and others. It supports the engineering of theories by enabling results to be related through linking functions, while allowing different concerns to be studied in isolation. The theory of designs [39, 51], which characterises total correctness, is one of the most important. In general, a UTP theory is a complete lattice where we can use joins and meets to model dual choices.

While sequential computations can be characterised by a relation between their initial and final states, the formal characterisation of reactive systems requires a richer model that accounts for the continuous interactions with their environment. In the UTP this is achieved through the theory of reactive processes [39, 44]. Together with the theory of designs, these two theories enable the specification of CSP processes in an assertional style, that is, in terms of designs that characterise the pre and postcondition of processes.

The theory of angelic nondeterminism presented in [38] is a starting point for the development of a model of CSP with both nondeterministic constructs. However, this model is focused on correctness of sequential programs and is not directly applicable to reactive processes. It is an encoding that caters for termination, so that designs are not considered as a separate theory.

In summary, a suitable treatment of angelic nondeterminism is yet to be considered in the context of process algebras for state-rich reactive systems. The UTP presents itself as a natural domain for the development of such a model, as existing theories, and their results, can be easily exploited. Our hypothesis is as follows.

Research Hypothesis

> ***A model can be defined to give a semantics to CSP that caters for both angelic and demonic nondeterminism, that is applicable in the wider context of any algebra of state-rich reactive systems for refinement, and that preserves the existing semantics of CSP processes, particularly within the subset of nondivergent processes.***

This concludes the discussion of the motivation underlying our work. In what follows we discuss the objectives in more detail.

1.2 Objectives

As already mentioned, the overall objective of our work is to define a semantic model suitable for state-rich process algebras, and CSP in particular, where both nondeterministic choices can be expressed. In contrast with some of the existing approaches [43], we do not intend to propose an entirely new semantic model for CSP, rather we aim to extend the current model while conserving the existing semantics. Therefore, our construction must be appropriately justified in the context of the existing model [38, 39].

With this in mind, the UTP framework and its CSP model provide a solid basis for studying the concept of angelic nondeterminism in the context of process algebras. We also observe that a UTP theory is a complete lattice where both angelic and demonic choice can be modelled as the meet and join, respectively.

The UTP supports work in the wider context of semantic models that consider behaviour and other aspects, such as data, security, mobility, and so on. Examples of such heterogeneous semantic models built using the UTP include *Circus* [22], which combines CSP with the Z specification language. Our aim to is to enable such semantic models to benefit from our treatment of angelic nondeterminism.

We also aim to enable existing modelling approaches and refinement techniques to be reused. This is central to the relevance and applicability of our semantic model. An important factor in UTP theories, for example, is that the refinement order is

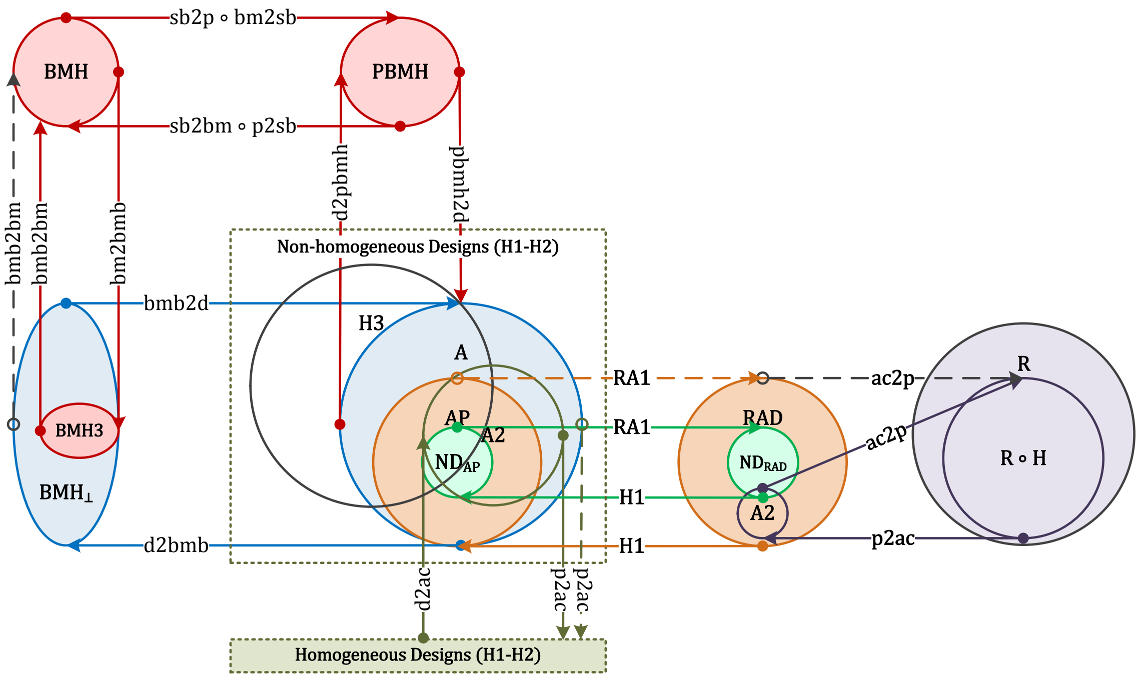


Figure 1.1: Theories and their relationship through linking functions

common across all theories. Our emphasis on maintaining a compatible semantics is essential in order to enable the scenario of reusing existing refinement techniques.

Our goal ultimately consists in developing a conservative extension of the CSP theory [39, 44] through a predicative encoding of multirelations that is suitable for characterising CSP processes. Of particular importance is the treatment of divergence where angelic choice can avoid potentially divergent processes. We seek a theory of CSP with both angelic and demonic nondeterminism, which is applicable to any algebra of state-rich reactive processes. In the following section we discuss our theories, by showing their relationship with other semantic models of interest, namely CSP.

1.3 Overview of Semantic Models

In this section we provide an overview of all the semantic models of interest in the context of our work. This includes both existing models as well as those we propose.

In the UTP [39] theories are characterised by three components: an alphabet, which is a set of variables available for recording the observations of programs in a particular paradigm, including program variables; a set of healthiness conditions, which are idempotent and monotonic functions, usually with a name written in boldface, whose fixed points are the the valid predicates of a theory; and a set of operators. For a relation P, the alphabet is split into two disjoint subsets, $in\alpha(P)$ which contains undashed variables corresponding to the initial observations, and $out\alpha(P)$ containing dashed counterparts for after or final observations.

Each theory of interest is depicted in Figure 1.1, and also individually in the subsequent Figures 1.2 to 1.6, by an ellipse, and labelled according to the name of its characterising healthiness condition. Subset theories correspond to enclosed ellipses. While the formal definition of each healthiness condition is deferred to later chapters, in Tables 1.1 to 1.6 we informally describe the healthiness conditions of each theory. In Figure 1.1 arrows denote linking functions established between theories. Pairs of solid arrows denote isomorphic models, while pairs with a dashed arrow indicate an adjoint (that is part of a Galois connection).

In the next Section 1.3.1 we describe the theory of designs. Section 1.3.2 focuses on the theory of CSP as reactive designs. In Section 1.3.3 we discuss the relationship between the theory of binary multirelations, the predicative encoding of [38], and the relationship with our theory of extended binary multirelations. In Section 1.3.4 we discuss our theory of angelic designs, which is the basis for extending the concept of angelic nondeterminism to CSP through the theory of reactive angelic designs, summarized in Section 1.3.5. Finally, Section 1.3.6 discusses our theory of angelic processes.

1.3.1 Designs

Since CSP processes are expressed in the UTP through reactive designs, the first theory of interest is that of designs, which models total correctness. Designs are relations whose alphabet contains not only program variables, but also auxiliary Boolean variables to capture termination. Its characterising healthiness conditions are **H1** and **H2**, whose composition is called **H**, as summarized in Table 1.1. In

	Description
H1	Meaningful observations can only be made once a design has been started.
H2	A design may not require non-termination.
H3	A design must have arbitrary behaviour when it does not terminate.

Table 1.1: Healthiness Conditions of Designs

general, this is a theory that encompasses programs whose preconditions can refer to the after or final observations of a computation. As a consequence these observations can be ascertained irrespective of termination. Such designs do not satisfy the

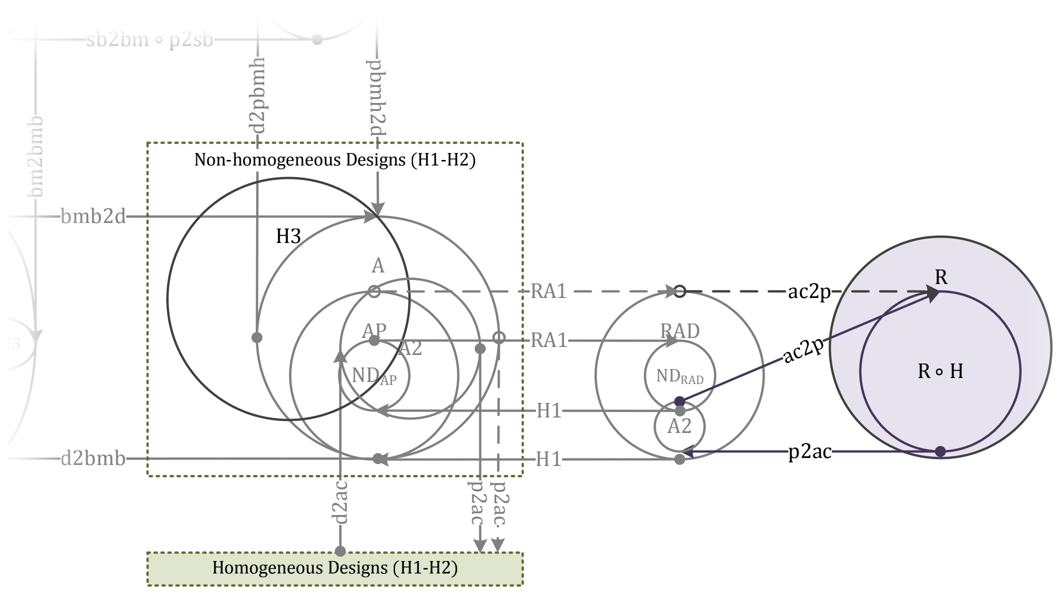


Figure 1.2: Theories of designs and reactive designs

healthiness condition **H3**. This is precisely the case when characterising a CSP process through reactive designs, such as $a \rightarrow Chaos$, whose precondition requires that no after observation of the trace of events is prefixed by the event a otherwise, it diverges.

The subset of designs whose preconditions may not refer to the after or final observations of a computation is characterised by **H3**. These designs correspond to standard pre and postcondition pairs as found in notations like Z [8] and VDM [11].

In the context of our work, we consider a theory of designs whose relations are not homogeneous, that is, their input and output alphabet differ. This is because of the multirelational nature of our encoding of angelic nondeterminism. In Figure 1.2 we highlight the theories of homogeneous and non-homogeneous designs in the context of other theories previously depicted in Figure 1.1.

1.3.2 CSP Processes as Reactive Designs

The second theory of interest is that of reactive processes, whose combination with the theory of designs provides the characterisation of CSP processes in the UTP. In the theory of reactive processes the alphabet is extended with observational variables to record the interactions with the environment: a trace of events, a set of events refused, and a Boolean variable that records whether the process is waiting for an interaction. Its healthiness conditions, which we informally describe in Table 1.2, are **R1**, **R2** and **R3**, whose functional composition is **R**.

	Description
R1	A process can only extend the trace of events.
R2	A process must be insensitive to the initial trace of events.
R3	A process must only start executing once any previous interactions with the environment have finished.
R	Functional composition of **R1**, **R2** and **R3** that characterises reactive processes.

Table 1.2: Healthiness Conditions of Reactive Processes

In order to characterise CSP processes, another two healthiness conditions are necessary. They are **CSP1** and **CSP2**, whose informal description is included in Table 1.3. Together, these healthiness conditions allow the characterisation of CSP processes as the image of designs through the function **R** [39, 44], that is, in terms of pre and postcondition pairs.

	Description
CSP1	A process that is in a divergent state can only extend the trace of events.
CSP2	A recast of **H2** within the model of reactive processes.

Table 1.3: Healthiness Conditions of CSP Processes

Since it is our goal to keep the semantics unchanged for the subset of nondivergent processes, in each theory of processes that we study, we identify such a subset. This is characterised by the healthiness condition **ND**, which is tailored to the theory of interest by adding a subscript corresponding to the characterising healthiness condition of the theory it applies to.

1.3.3 Binary Multirelations and their UTP Encoding

To achieve our goal we have developed a predicative encoding of multirelations suitable for characterising processes. Our starting point was the predicative encoding of Cavalcanti et al. [38], whose theory is characterised by the healthiness condition **PBMH**. This is essentially a predicative version of **BMH**, that characterises a set-based model of upward-closed binary multirelations [35].

In [38] the authors establish that both models are isomorphic through a stepwise construction of models, as previously discussed in Section 1.1. This is achieved through the composition of the linking functions, $sb2p \circ bm2sb$ and $sb2bm \circ p2sb$, which we include in Figures 1.1 and 1.3 for completeness. The first contribution of

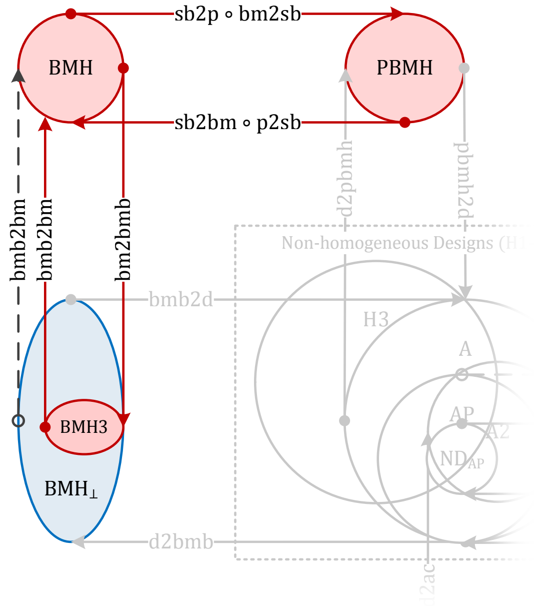


Figure 1.3: Theories related to binary multirelations

this thesis is a theory of extended binary multirelations that caters for potentially non-terminating computations. This theory is isomorphic to the theory of angelic designs, which we describe in the next section. It is characterised by the healthiness condition $\mathbf{BMH}_\perp$, which corresponds to the conjunction of **BMH0**, **BMH1** and **BMH2** as described in Table 1.4. Finally, we establish that the subset of **BMH3**

	Description
BMH0	The set of final states must be upward-closed.
BMH1	Similarly to **H2** forbids the specification of non-termination.
BMH2	Appropriately characterises two complementary notions of abortion.
BMH3	Characterises the subset of $\mathbf{BMH}_\perp$ that is isomorphic to the original theory of binary multirelations.
$\mathbf{BMH}_\perp$	Conjunction of **BMH0**, **BMH1** and **BMH2**.

Table 1.4: Healthiness Conditions of Extended Binary Multirelations

multirelations is isomorphic to the original theory of binary multirelations, via the pair of linking functions *bmb2bm* and *bm2bmb*. In general, a Galois connection can also be established between $\mathbf{BMH}_\perp$ and **BMH**. Figure 1.3, which highlights the theories in the context of Figure 1.1, illustrates these connections.

1.3.4 Angelic Designs

Our approach for developing a model of CSP with angelic nondeterminism closely follows that of the UTP model of CSP. Based on the the encoding proposed in [38], we have developed a theory of angelic designs where we reintroduce the auxiliary Boolean variables of the original theory of designs. Furthermore, we also generalise that model to cope with non-**H3** designs, as required for specifying CSP processes. This theory is characterised by the healthiness conditions **A0** and **A1**, whose functional composition is **A** (as described in Table 1.5), and **H1** and **H2** of the original theory of designs.

	Description
A0	Whenever the precondition of a design is satisfied, then the set of angelic choices must not be empty.
A1	The set of angelic choices must be upward-closed.
A2	Characterises the subset of relations that effectively do not have any angelic choices.
A	Functional composition of **A0** and **A1**

Table 1.5: Healthiness Conditions of Angelic Designs

The additional healthiness condition **A2** characterises the subset of **A**-designs that do not exhibit angelic nondeterminism. This is useful to establish that the subset of **A2** angelic designs is isomorphic to the original theory of homogeneous designs, via the linking functions $d2ac$ and $p2ac$. In general, these adjoints also enable a Galois connection to be established with the set of **A**-designs. As part of validating our approach, we also establish that the subset of angelic designs that is **H3**-healthy is isomorphic to the theory of **PBMH** [38]. This is achieved by introducing two linking functions, $d2pbmh$ and $pbmh2d$, that map predicates in that theory to angelic designs, and vice versa. In Figure 1.4 we highlight the theory of angelic designs in the context of Figure 1.1 and show its relationship with the **PBMH** theory, the extended theory of binary multirelations, and the original theory of homogeneous designs.

In addition, and as already discussed, we have developed an extended set-based model of binary multirelations that is isomorphic to **A**-healthy designs. This complementary model is useful to understand the implications of non-homogeneous relations and also to validate certain aspects of the model of angelic designs, such as the notion of sequential composition, which is not entirely trivial in the context of a predicative encoding of multirelations. We establish that these two models are

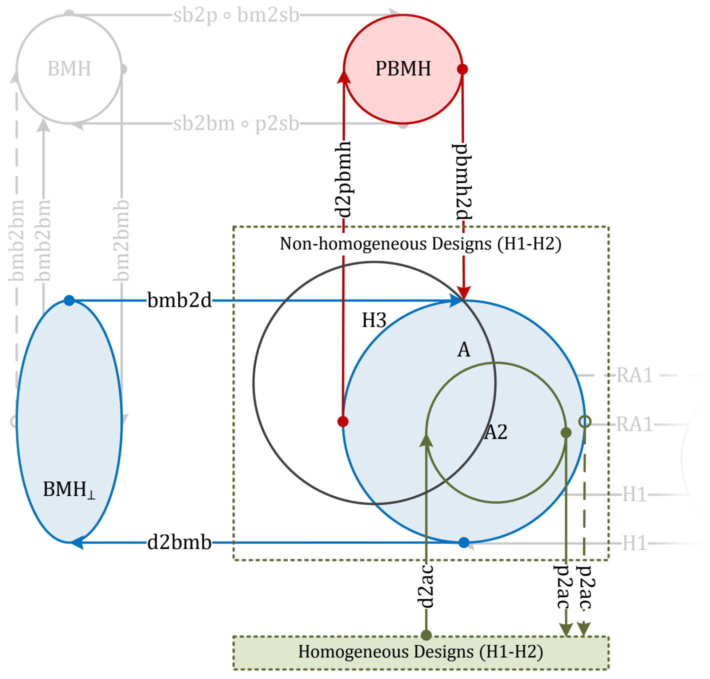


Figure 1.4: Theory of angelic designs and links

isomorphic through the pair of linking functions *bmb*2*d* and *d*2*bmb*.

1.3.5 Reactive Angelic Designs

Having established a theory of angelic designs, we introduce a conservative extension of CSP with angelic nondeterminism. This is achieved by considering an encoding of the observational variables of reactive processes, based on that used for angelic designs, and expressing every healthiness condition of CSP with this encoding. For each healthiness condition **R1**, **R2**, **R3**, **CSP1** and **CSP2**, we introduce a counterpart in this model, as summarized in Table 1.6. The theory is characterised by **RAD**, which is defined by the composition of all healthiness conditions of interest, including **PBMH** that guarantees upward-closure for the sets of final states. As part of our validation approach, we establish that the subset of **RAD** with no angelic nondeterminism, characterised by **A2**, is isomorphic to the theory of CSP. This is achieved by introducing the linking functions *ac*2*p* and *p*2*ac*. In general, if we consider the superset **RAD**, a Galois connection exists between the theories. This relationship is illustrated in Figure 1.5.

The theory of reactive angelic designs corresponds to a natural extension of

	Description
RA1	There must be some set of angelic choices available to the angel, and in any such set, the trace of events can only be extended.
RA2	A process must be insensitive to the initial value of the trace of events.
RA3	A process must not start executing before its predecessor has stopped interacting with its environment.
RA	Functional composition of **RA1**, **RA2** and **RA3**.
CSPA1	When in an unstable state, **RA1** must be enforced.
CSPA2	A recast of **H2** within this model.
RAD	Functional composition of all of the above healthiness conditions and **PBMH**.
$\mathbf{ND_{RAD}}$	Characterises the subset of non-divergent reactive angelic designs.

Table 1.6: Healthiness Conditions of Reactive Angelic Designs

the CSP theory with both angelic and demonic nondeterminism. In this theory it is possible to establish that angelic choice avoids divergence. For example, the angelic choice $a \rightarrow \mathit{Chaos} \sqcup b \rightarrow \mathit{Skip}$ becomes $a \rightarrow \mathit{Skip}$, provided that a and b are equal. However, since **RA1** requires under all circumstances that no trace of events may be undone, if a and b are different events, then the possibility to observe the event a cannot be entirely excluded, and so divergence is still a possibility. In order to lift this restriction we have relaxed **RA1** in case of divergence, which is the motivation for the theory of angelic processes that we discuss in the next section.

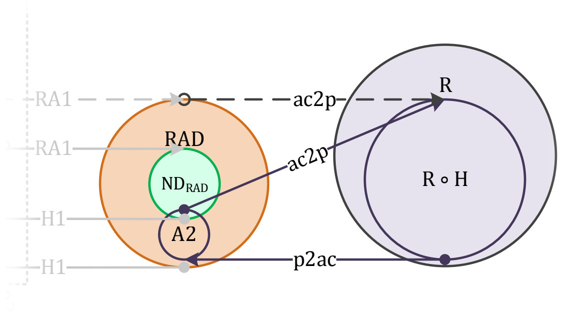


Figure 1.5: Theory of reactive angelic designs and links with CSP

1.3.6 Angelic Processes

In order to allow angelic choice to exclude potentially divergent processes, we relax the theory of reactive angelic designs by allowing the history of events to be undone whenever there is the potential to diverge. This is achieved by not enforcing **RA1** in all cases. Therefore, we redefine **RA3** to cope with this fact as $\mathbf{RA3_{AP}}$, and define the healthiness condition of this theory as **AP**, as summarized in Table 1.7.

	Description
$\mathbf{RA3_{AP}}$	A recast of **RA3** in the theory of angelic processes.
AP	Functional composition of $\mathbf{RA3_{AP}}$, **RA2**, **A** and, **H1** and **H2** of the theory of designs (with the corresponding alphabet of this theory).
$\mathbf{ND_{AP}}$	Characterises the subset of non-divergent angelic processes.

Table 1.7: Healthiness Conditions of Angelic Processes

The consequence of the functional composition underlying **AP** is that this model is effectively a theory of angelic designs, where **RA1** is only required in the postcondition. This is a direct consequence of the definition of **A**, as it requires that the set of angelic choices in the postcondition of an **A**-design is not empty.

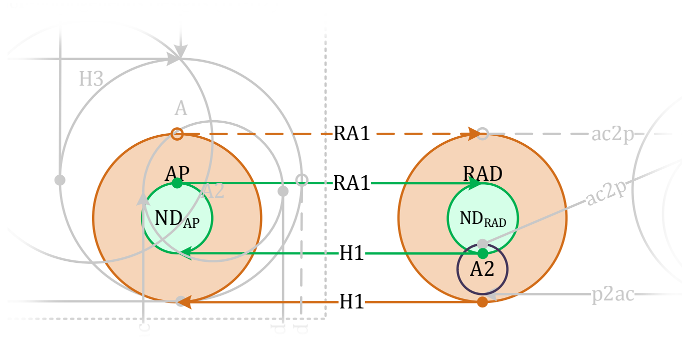


Figure 1.6: Theory of angelic processes and link with reactive angelic designs

The resulting theory is more generic than that of reactive angelic designs, since it does require **RA1**. As part of our validation approach, we establish a Galois connection with the theory of reactive angelic designs, and also prove that an isomorphism exists with respect to the subsets of non-divergent processes, characterised by $\mathbf{ND_{RAD}}$ and $\mathbf{ND_{AP}}$, respectively. This is achieved by turning reactive angelic designs into designs, through **H1**, while in the opposite direction we just enforce **RA1**. These links are depicted in Figure 1.6 where we highlight both theories in the context of Figure 1.1.

A detailed account of all the new theories is presented in the sequel as described below.

1.4 Outline

In Chapter 2, we provide an overview of the concept of angelic nondeterminism as found in the literature. In addition, we discuss the most important semantic models in the context of our work by introducing: weakest precondition semantics, binary multirelations, the UTP, and the existing models of CSP.

Chapter 3 presents the extended model of binary multirelations that handles non-terminating computations. We introduce the healthiness condition $\mathbf{BMH}_\perp$ as well as the most important operators of this theory. Finally, we establish its relationship with the existing model of binary multirelations via linking functions (see Figure 1.3).

Chapter 4 introduces the theory of angelic designs, the first new UTP theory developed in this thesis. We introduce the alphabet of the theory, followed by the healthiness conditions **A0** to **A2**. The relationship with the extended model of binary multirelations is studied before introducing the most important operators. We conclude this chapter by studying the relationship of the subset of angelic designs that are **H3**-healthy and the **PBMH** theory of [38].

In Chapter 5 the theory of reactive angelic designs is presented. This is a natural extension of the UTP model of CSP in the context of a theory with angelic nondeterminism, where the healthiness conditions of CSP are expressed using this new encoding. The resulting healthiness condition is **RAD**. Finally, we discuss the operators and study the link with the existing theory of reactive designs.

Our final contribution is found in Chapter 6, where we present the theory of angelic processes, whose healthiness condition is **AP**. This chapter concludes by exploring the relationship with the theory of reactive angelic designs and the main algebraic properties.

Finally, in Chapter 7 we summarize the main contributions of this thesis and further contextualize our work. We conclude with pointers for future work.

Chapter 2

Angelic Nondeterminism

In this chapter we provide an account of angelic nondeterminism as found in the literature, and introduce the foundations upon which our theories are built. Section 2.1 discusses the concept of angelic nondeterminism and its applications. In Section 2.2 we introduce Dijkstra's weakest preconditions and the predicate transformers of the refinement calculus. Section 2.3 introduces Rewitzky's theory of binary multirelations. In Section 2.4 we provide an introduction to the UTP of Hoare and He. Finally, Section 2.5 contains a short introduction to CSP and a discussion on the different semantic approaches to characterising angelic nondeterminism in CSP.

2.1 Definition and Applications

The earliest use of angelic nondeterminism can be found in the theories of computation, more specifically in automata theory [52] and Turing machines [53]. For example, in pushdown stack automata, the addition of nondeterminism enables the automaton to accept arbitrary context-free languages [54], while for Turing machines it helps characterise the class of NP-problems [53] whose solutions can be found efficiently given an angelically nondeterministic machine.

Angelic nondeterminism has been used as a specification and programming construct in several applications, including parsing [55], modelling of game-like scenarios [32] and user interactions, theorem proving tactics [56, 57], constraint programming [58], logic programming [59] and others. These are problems where finding solutions often involves a combination of search and backtracking. For instance, in Angel [56, 57], theorem proof tactics can be combined through angelic choice, such that failure leads to backtracking.

While this is a perfectly reasonable interpretation of angelic choice, backtracking is not the only possibility, nor is it always desired. Irrespective of the actual operation of an angelic choice, its distinguishing feature across the different applications is its capability to provide a high degree of abstraction while still guaranteeing success.

Already in 1967, Floyd [60] envisioned angelic choice as a mechanism for the abstract specification of algorithms, with actual executable programs being produced mechanically, perhaps by a compiler. In the context of his formal characterisation of programs as flowcharts, Floyd introduced explicit nondeterministic choice points, and appropriate notions of success and failure, in order to avoid implementation details of particular execution strategies. Although angelic nondeterminism is usually interpreted operationally as a backtracking mechanism, it can also be implemented through some form of parallelism [61].

Almost at the same time, important contributions were being made to the theoretical understanding of programs. In 1969, Hoare proposed his formal system for proving partial correctness in the context of sequential programming languages [6]. While in 1975 Dijkstra [7, 62] introduced his language of guarded commands, an imperative language with repetitive and nondeterministic constructs. Unlike Floyd's choice points, Dijkstra's nondeterministic choice was no longer angelic.

Dijkstra [7, 62] fundamentally changed the approach to establishing total correctness by calculation through his weakest precondition semantics. His model restricted itself to feasible programs by excluding the existence of miracles (with the so called "*Law of the Excluded Miracle*"). Miracle is the theoretical counterpart to abort and corresponds to the infeasible program that can never be executed, while abort represents the worst possible program whose behaviour, in the context of a theory of total correctness, is completely arbitrary.

When Back [28, 63], Morris [29] and Morgan [31] introduced the refinement calculus, miracles were introduced back into their models. This enabled their models to become more generic, and paved the way for the development of models that are complete lattices under the refinement order. The most important was, perhaps, the lattice of monotonic predicate transformers where angelic and demonic choice are modelled as the least upper bound and greatest lower bound of the lattice. Back and von Wright [32] extensively studied sublattices, where choice can be either angelic or demonic. They have also considered angelic nondeterminism in the context of game-like scenarios and modelling of user interactions.

Angelic choice also plays a significant role amongst data refinement techniques, such as that of Gardiner and Morgan [33], where the least upper bound is used to define logical variables. These enable the postcondition of a specification statement to refer to the initial value of a program variable.

Ward and Hayes, in their work [61] on applications of angelic nondeterminism, clearly emphasize that unlike Floyd's choice points, the angelic choice of the refinement calculus can "look ahead" and guide choices to avoid divergence, if at all possible. This is not restricted to explicit choice points, but rather applies to any angelic construct, such as the angelic assignment of values to program variables, which they explore in the refinement of programs from high-level specifications.

In the context of theories of total correctness, computations can also be specified through relations between initial states and final states. This is the notion adopted in formal notations like Z [8] and VDM [11], where there is an explicit relation between the initial and final value of a computation. However, as Back [32] and Cavalcanti et al. [64] have noted, relations can only capture one type of nondeterminism, either angelic or demonic, but not both.

When Cavalcanti et al. [64] proposed the introduction of angelic nondeterminism into the relational setting of Hoare and He's UTP [39], a multirelational encoding had to be considered. They first established that, in general, UTP relations are isomorphic to conjunctive predicate transformers. Their solution to the problem consisted in defining a predicative encoding of Rewitzky's [35] upward-closed binary multirelations, which is the basis for the work that we describe in this thesis.

As already mentioned, Rewitzky's [35] multirelations are relations between initial states and sets of final states. In [36] several models of binary multirelations are considered, of which the model of upward-closed multirelations is the most important due to its lattice-theoretic structure. In this model, the refinement order is reverse subset inclusion, and angelic and demonic choice correspond to set union and intersection, respectively. We discuss this model in more detail in Section 2.3.

More recently, Guttmann [37] has proposed a generalised algebraic structure that has both the monotonic predicate transformers and multirelations as instances. Guttmann has also extensively studied the relational properties of multirelations, and proposed an extension catering for non-terminating computations [65] in the setting of general correctness. This involves extending the set of final states to record whether a computation does not terminate: a similar idea is used in our extended model of binary multirelations [3] where we record whether a computation may not terminate and still establish some final value. This model is part of the first contribution of this thesis and is discussed in detail in Chapter 3.

In Section 2.5 we come back to the topic of angelic nondeterminism by reviewing the existing approaches to characterising angelic nondeterminism in CSP. Next we introduce Dijkstra's weakest precondition semantics.

2.2 Weakest Preconditions

As already discussed, one of the earliest treatments of total correctness is due to Dijkstra [7, 62], through his language of guarded commands and weakest precondition semantics. The underlying idea is that for each program statement S and postcondition q, it is possible to establish the weakest precondition $wp(S, q)$, such that, starting S in a state satisfying $wp(S, q)$ achieves postcondition q. A weakest precondition characterises all possible initial states that lead to successful termination with the postcondition holding. In Dijkstra's model [7, 62], predicates are characterised by functions on all points of a state space, which in his original presentation [62] are defined through Cartesian products.

If we consider the program *Skip*, which does not change the state and always terminates successfully, its weakest precondition semantics is defined as follows.

Definition 1 $wp(\mathit{Skip}, q) = q$

That is, the weakest precondition corresponds exactly to the intended outcome q. A simple assignment statement, where a program variable x is assigned the value of an expression e, is given semantics for a postcondition q as follows.

Definition 2 $wp(x := e, q) \mathrel{\widehat{=}} q[e/x]$

In other words, the weakest precondition of the assignment is given as the substitution of expression e for variable x in the corresponding postcondition q.

In general, not all possible weakest preconditions are valid, in the sense that the semantic model must obey certain fundamental properties of interest, such as monotonicity. In what follows, we review the original properties of Dijkstra's model [62].

2.2.1 Healthiness Conditions

Dijkstra's semantics [62] insist on four healthiness conditions, which we discuss in this section. The first property, reproduced below, corresponds to the "*Law of the Excluded Miracle*", which forbids miraculous behaviour from being specified.

Definition 3 (Non-miraculous) $wp(S, F) = F$

If program statement S could achieve F, the predicate which is false everywhere, then there must be no such initial state where $wp(S, F)$ that can be satisfied. This is precisely one of the properties that Back [32], Morris [29] and Morgan [31] relaxed in order to introduce the lattice of monotonic predicate transformers.

The fundamental property of interest in models for refinement is monotonicity. The definition [62] is reproduced below.

Definition 4 (Monotonicity) $(q \Rightarrow r) \Rightarrow (wp(S, q) \Rightarrow wp(S, r))$

For every state and program statement S, whenever q is a stronger predicate than r, then the weakest precondition $wp(S, q)$ is also stronger than $wp(S, r)$. In other words, if q is a postcondition stronger than r, then, the set of initial states guaranteed to establish q is a subset of those that establish r.

The next healthiness condition that Dijkstra presents is conjunctivity, whose formal definition is reproduced below [62].

Definition 5 (Conjunctivity) $wp(S, q) \wedge wp(S, r) \Leftrightarrow wp(S, q \wedge r)$

The right-hand side implication follows directly from monotonicity and properties of the predicate calculus. However, the left-hand side implication is not necessarily satisfied in general. In fact, this property is precisely what prevents angelic nondeterminism from being specified in Dijkstra's model, as noted by Back [63]. This result follows from the definition of the angelic statement whose semantics, as given, for example, in [61, 66], is defined using an existential quantification.

The counterpart to conjunctivity is disjunctivity, whose definition is as follows.

Definition 6 (Disjunctivity) $wp(S, q) \vee wp(S, r) \Leftrightarrow wp(S, q \vee r)$

Since weakest preconditions observing this property cannot model demonic nondeterminism, Dijkstra [62] uses a weaker version where only the left-hand side implication is enforced. Similarly to the angelic statement, the demonic specification statement is defined, for example, in [61, 66] using a universal quantification.

In [63] Back and von Wright extensively study different models of weakest preconditions with different properties, including models with and without miracles, conjunctivity and disjunctivity. They conclude that by considering a model that is neither conjunctive nor disjunctive, both forms of nondeterminism can be modelled together. Furthermore, by considering a model with miracles, a complete lattice exists where angelic and demonic choice correspond to the meet and join, respectively. This is a result explored in all versions of the refinement calculus [29, 31, 32]. Our remaining discussion on weakest preconditions is mostly based on Back and von Wright's work [32].

2.2.2 Predicate Transformers

The *wp* function of Dijkstra is a predicate transformer as it maps predicates to predicates. Back and von Wright [62], in their presentation of the refinement calculus introduce the notion of contracts which can be either specifications or programs. The satisfaction of a contract S by establishing postcondition q when started from an initial state σ is denoted by $\sigma\ \{\!|\ S\ |\!\}\ q$. They characterise $wp : \mathbb{P}\,\Sigma \to \mathbb{P}\,\Sigma$, where the state space is Σ, for a contract S as follows.

Definition 7 (Weakest Precondition) $wp(S, q) \mathrel{\widehat{=}} \{\sigma \mid \sigma\ \{\!|\ S\ |\!\}\ q\}$

That is, the set of all initial states σ, from which S is guaranteed to establish q. Weakest precondition semantics can then be given to their language of contracts [32], which we reproduce in the following definition.

Definition 8 (Basic Weakest Preconditions)

$$\begin{aligned}
wp(\langle f\rangle, q) &= f^{-1}(q)\\
wp(\{g\}, q) &= g \cap q\\
wp([g], q) &= \neg\ g \cup q\\
wp(S_1\ ;\ S_2, q) &= wp(S_1, wp(S_2, q))\\
wp(S_1 \sqcup S_2, q) &= wp(S_1, q) \cup wp(S_2, q)\\
wp(S_1 \sqcap S_2, q) &= wp(S_1, q) \cap wp(S_2, q)
\end{aligned}$$

The first construct $\langle f\rangle$ is a functional update that changes the state according to function f. An example is the identity *id*, which does not change the state.

The following construct $\{g\}$ is an assertion, which has no effect on the state if g holds. Otherwise the program aborts. The assertion $\sigma\ \{\!|\ \{g\}\ |\!\}\ q$ holds if, and only if, the state σ is in the intersection of g and the postcondition q.

Its dual is the assumption $[g]$; it has no effect if g holds and otherwise the contract is satisfied trivially. Hence, the weakest precondition is given by $\sigma \in q$ and otherwise, if g fails to hold then $\sigma \in \neg\ g$.

The sequential composition of S_1 and S_2 is given as the weakest precondition of S_1, with respect to the postcondition characterised by the weakest precondition of S_2. That is, $wp(S_2, q)$ is an intermediate condition that needs to be satisfied in order to achieve q.

Finally, angelic and demonic choice are defined as $\sqcup$ and $\sqcap$, respectively. In an angelic choice, it is sufficient that either the precondition of S_1 or S_2 is satisfied in order to achieve q, whereas in a demonic choice both need to be satisfied.

2.2.3 Predicate Transformers Lattice

In Back and von Wright's model [32], the notion of refinement is given for two contracts S_1 and S_2 as follows.

Definition 9 $S_1 \sqsubseteq S_2 \Leftrightarrow \forall \sigma, q \bullet \sigma \ \{\!|\ S_1\ |\!\}\ q \Rightarrow \sigma\ \{\!|\ S_2\ |\!\}\ q$

A contract S_1 is refined by S_2 if, and only if, for all initial states σ and postconditions q, if σ is an initial state of contract S_1 leading to postcondition q, then it is also an initial state of S_2 leading to q. As this order is reflexive, transitive and antisymmetric [32, 67], it is a partial order. The bottom element is the assertion $\{\mathit{false}\}$, which can never be satisfied in any initial state, while the top element is the assumption $[\mathit{false}]$, so that it is trivially satisfied in any initial state for any final condition q.

When Back and von Wright [32] introduce their model of predicate transformers, they actually consider the target state space as being potentially different from the initial state space, as required, for instance, to model states with scoped variables. Thus, the set of predicate transformers from an initial state space Σ, to a final state space Γ is defined by $\mathbb{P}\,\Gamma \to \mathbb{P}\,\Sigma$.

The refinement order for predicate transformers is defined by considering the pointwise extension of the subset ordering; for predicate transformers T_1 and T_2, we have the following definition.

Definition 10 $T_1 \sqsubseteq T_2 \mathrel{\widehat{=}} \forall\, q \in \mathbb{P}\,\Gamma \bullet\ T_1(q) \subseteq T_2(q)$

That is, T_1 is refined by T_2, if, and only if, the set of initial states that characterise the weakest precondition for q to be established according to T_1 is a subset of that characterised by T_2. This order forms a complete Boolean lattice [32]. Thus the lattice operators on predicate transformers are pointwise extensions of the corresponding operators on predicates [32].

Finally, in [32] Back and von Wright consider the complete sublattice of monotonic predicate transformers. What is particularly important about their result is that every basic statement is monotonic and so are the sequential composition, meet, and join of predicate transformers [32].

This concludes our discussion of the lattice of monotonic predicate transformers as the standard model where angelic and demonic nondeterminism have traditionally been studied. In the following Section 2.3 we discuss the theory of upward-closed binary multirelations, which is effectively a relational characterisation of the predicate transformers model [35].

2.3 Binary Multirelations

As already discussed, it is not possible to model both angelic and demonic nondeterminism in a purely relational model. However, multirelational models can be used to characterise both forms of nondeterminism in a relational setting.

In [35] Rewitzky introduces the theory of binary multirelations, which are relations between initial states and sets of final states. In our presentation we define these relations through the following type *BM*, where *State* is a type of records with a component for each program variable.

Definition 11 $BM \mathrel{\widehat{=}} State \leftrightarrow \mathbb{P}\, State$

An example of a program in this model is the assignment of the value 1 to the only program variable x when started from any initial state.

Example 1 $x :=_{BM} 1 = \{s : State, ss : \mathbb{P}\, State \mid (x \mapsto 1) \in ss\}$

This assignment, which we subscript with *BM* to distinguish it from assignment statements in other models that we discuss later, is defined by relating every initial state s to a set of final states ss where the component x is set to the value 1. For conciseness, in the examples and definitions that follow, the types of s and ss may be omitted where it is clear that the composite type is *BM*.

The target set of a binary multirelation can be interpreted as either encoding angelic or demonic choices [35, 64]. Here we present a model where the set of final states encodes angelic choices. This decision is justified in [38] as maintaining the refinement order of the isomorphic UTP model of Cavalcanti et al. [38], which we discuss in Section 2.4.4.

Demonic choices are encoded by the different ways in which the set of final states can be chosen. For example, consider the following program which angelically assigns the value 1 or 2 to the only program variable x; it uses $\sqcup_{BM}$ the angelic choice operator for binary multirelations.

Example 2 $x :=_{BM} 1 \sqcup_{BM} x :=_{BM} 2 = \{s, ss \mid (x \mapsto 1) \in ss \land (x \mapsto 2) \in ss\}$

In this multirelation, every initial state s is associated with all sets ss in which we can find the choice of a final state where x is assigned the value 1 or 2. Irrespective of the set of final states chosen by the demon, the angel is always able to enforce this choice. As illustrated, for a particular initial state, the choices available to the angel correspond to those in the distributed intersection over all possible sets of final states.

2.3.1 Healthiness Conditions

Example 2 above illustrates a fundamental property of binary multirelations: upward-closure [35]. This property is captured by the following healthiness condition for a multirelation B.

Definition 12 $\mathbf{BMH} \mathrel{\widehat{=}} \forall s, ss_0, ss_1 \bullet ((s, ss_0) \in B \land ss_0 \subseteq ss_1) \Rightarrow (s, ss_1) \in B$

If an initial state s is related to a set of final states ss_0, then it is also related to any superset ss_1. This reflects the fact that if it is possible to terminate in some final state in ss_0, then the addition of any other final states to that set does not change the actual states available for angelic choice.

Upward-closure ensures that there is a complete lattice under the subset order, with angelic and demonic choice corresponding to the least upper bound and greatest lower bound, respectively. Moreover, in [35] Rewitzky establishes that there is a bijection between upward-closed binary multirelations and monotonic unary operators. Since, as explained in Section 2.2 predicate transformer semantics can be given in terms of monotonic unary operators, this establishes that the multirelational model is in fact a relational characterisation for commands with both forms of nondeterminism.

2.3.2 Refinement

In the model of upward-closed binary multirelations, refinement is defined for healthy multirelations B_0 and B_1 by reverse subset inclusion as follows [35].

Definition 13 $B_0 \sqsubseteq_{BM} B_1 \mathrel{\widehat{=}} B_0 \supseteq B_1$

A multirelation B_0 is refined by B_1 if, and only if, B_1 is a subset of B_0.

This partial order forms a complete lattice. The bottom element $\bot_{BM}$, corresponding to the notion of **abort**, is defined by the universal relation, which associates every initial state to every possible set of final states.

Definition 14 $\bot_{BM} \mathrel{\widehat{=}} State \times \mathbb{P}\, State$

The top element $\top_{BM}$ is defined by the empty relation and corresponds to the notion of **miracle**, the infeasible program.

Definition 15 $\top_{BM} \mathrel{\widehat{=}} \emptyset$

Via refinement, the degree of angelic nondeterminism of a program can be increased, while the degree of demonic nondeterminism can be decreased, that is, a program can be refined into a demonically more deterministic one. In particular, the infeasible program $\top_{BM}$ refines every other program, while every program refines $\bot_{BM}$.

2.3.3 Operators

In this section we present the main operators of the theory of binary multirelations and discuss their most important properties.

Assignment

The first operator of interest, which we have briefly discussed in Example 2, is assignment. Its complete definition is as follows.

Definition 16 $x :=_{BM} e \mathrel{\widehat{=}} \{s, ss \mid s \oplus (x \mapsto e) \in ss\}$

Every initial state s is related to every set of final states ss that includes a state where s is overridden to define that x has the value of expression e.

Angelic Choice

The angelic choice operator is defined as set intersection.

Definition 17 $B_0 \sqcup_{BM} B_1 = B_0 \cap B_1$

This operator corresponds to the least upper bound of the lattice. Intuitively, the final states available for angelic choice are those in the intersection of all choices available for demonic choice. The operator satisfies the following property.

Lemma L.2.3.1 $B_0 \sqsubseteq_{BM} B_0 \sqcup_{BM} B_1$

That is, the degree of angelic nondeterminism can be increased.

Demonic Choice

Its dual, demonic choice, is the greatest lower bound and is defined as set union.

Definition 18 $B_0 \sqcap_{BM} B_1 = B_0 \cup B_1$

For a given initial state, the sets of final states available for demonic choice correspond to those in either B_0 or B_1. Demonic choice observes the following property.

Lemma L.2.3.2 $B_0 \sqcap_{BM} B_1 \sqsubseteq_{BM} B_0$

That is, the degree of demonic nondeterminism can be decreased. Finally, angelic and demonic choice distribute over one another.

Lemma L.2.3.3 $B_0 \sqcap_{BM} (B_1 \sqcup B_2) = (B_0 \sqcap_{BM} B_1) \sqcup_{BM} (B_0 \sqcap_{BM} B_2)$

This property follows from the distributive properties of set union and set intersection. It is equally valid in the theory of predicate transformers and the isomorphic UTP model of [38].

Sequential Composition

Although this is a relational model, since states are related to sets of states, the definition of sequential composition is not relational composition. Instead it is defined as follows.

Definition 19

$$B_0 \mathbin{;_{BM}} B_1 \mathrel{\widehat=} \{s_0, ss_1 \mid \exists\, ss_0 \bullet (s_0, ss_0) \in B_0 \land ss_0 \subseteq \{s_1 \mid (s_1, ss_1) \in B_1\}\}$$

It considers every initial state s_0 in B_0 and set of final states ss_1, such that there is some intermediate set of states ss_0 that is related from s_0 in B_0, and ss_0 is a subset of those initial states of B_1 that achieve ss_1. As noted in [38] for healthy multirelations this definition can be simplified further as shown in the following lemma.

Lemma L.2.3.4 *Provided B_0 satisfies* **BMH***,*

$$B_0 \mathbin{;_{BM}} B_1 \mathrel{\widehat=} \{s_0, ss_1 \mid (s_0, \{s_1 \mid (s_1, ss_1) \in B_1\}) \in B_0\}$$

Proof. Equation 5 in [38]. □

This definition is the basis for the definition of sequential composition in the isomorphic UTP model of [38], and for the definition of sequential composition in the extended model of binary multirelations that we discuss in Chapter 3.

2.4 The Unifying Theories of Programming

As previously discussed, the UTP of Hoare and He [39] is a framework of alphabetized relations suitable for characterising different programming paradigms. The UTP promotes unification of results while enabling different aspects of programs to be considered in isolation. In [39] a collection of theories is presented that targets multiple aspects of different programming paradigms, such as functionality, concurrency, logic programming and higher-order programming. Several other theories have since been developed which cater for other aspects, such as angelic nondeterminism [38], object-orientation [45, 46], pointers [47] and time [48–50].

The UTP is based on the principle of observation, and so the discourse for recording observations is defined by an alphabet whose variables determine the observable parameters of a system. These can be either program variables, or alternatively, auxiliary variables that capture information like termination and execution time. A UTP theory is characterised by three components: an alphabet, a set of healthiness conditions and a set of operators.

For a given relation P, its alphabet is given by $\alpha(P)$. Similar to the conventions of Z, in the UTP an alphabet is split into two disjoint subsets: $in\alpha(P)$, which contains undashed variables for characterising the initial observations, and $out\alpha(P)$, which contains the dashed counterparts of each variable that characterise the final or subsequent observations of a system. For example, a program whose purpose is to increment the initial value of the only program variable x can be specified by the relation: $x' = x + 1$. This relation concisely describes all pairs of values (x, x') that satisfy this predicate. Thus relations characterise the possible observations of a program.

When the input and output alphabets of a relation are exactly the same, except for the fact that variables are undashed and dashed in either set, respectively, a relation is said to be homogeneous.

Definition 20 (Homogeneous Relation) *A relation P is homogeneous if, and only if, $(in\alpha(P))' = out\alpha(P)$.*

This is captured by Definition 20, where $(in\alpha(P))'$ is the set of variables obtained by dashing every variable in the set $in\alpha(P)$.

The remainder of this section is organised as follows. In Section 2.4.1 we discuss the other two components of UTP theories, namely healthiness conditions and operators. In Section 2.4.2 we introduce the theory of designs which captures total correctness. In Section 2.4.3 we discuss the approach to linking theories in the UTP.

Finally, Section 2.4.4 discusses the theory of angelic nondeterminism of [38].

2.4.1 Theories

The second component of a UTP theory is a set of healthiness conditions that characterise the predicates of a theory. These are normally specified by idempotent and monotonic functions whose fixed points are the valid predicates of the theory.

Healthiness Conditions

For instance, in the context of theories concerning time, it is often possible to make observations of a system in discrete-time units recorded using a variable t. It is expected that any plausible theory describing such a system must guarantee that time is increasingly monotonic. This property can be described by the following healthiness condition HC.

Example 3 $\mathbf{HC}(P) \mathrel{\widehat=} P \land t \leq t'$

It requires that under all circumstances, it must be the case that the initial value of t is less than or equal to the final or after value t'. This healthiness condition is defined in terms of conjunction, so it is called a conjunctive healthiness condition [47]. A general result on conjunctive healthiness conditions [47] enables us to establish that HC is idempotent and monotonic with respect to refinement. An observation in this theory is valid if, and only if, it is a fixed point of HC.

Refinement

The theory of relations forms a complete lattice [39], with the order given by (reverse) universal implication. The top of the lattice is *false* and the bottom is *true*. This order corresponds to the notion of refinement. Its definition is presented below, where the square brackets stand for universal quantification over all the variables in the alphabet [39].

Definition 21 (Refinement) $P \sqsubseteq Q \mathrel{\widehat=} [Q \Rightarrow P]$

Refinement can be understood as capturing the notion of correctness in the sense that, if a predicate Q refines P, then all possible behaviours exhibited by Q are permitted by P. This notion is paramount for the UTP framework and it is the same across all theories. The relation *true* imposes no restriction and permits the observation of any value for all variables in the alphabet, while *false* permits none.

Operators

A UTP theory comprises a number of operators that characterise how the theory may be used algebraically to specify more complex behaviours. In the theory of relations there are a number of core operators that correspond to typical constructs found in programming languages, such as assignment (:=), conditional ($A \lhd c \rhd B$), and sequential composition (;). In what follows we present some of the most important operators of the theory of relations.

Sequential Composition

In UTP theories whose relations are homogeneous, sequential composition is defined as relational composition. The definition is shown below through substitution.

Definition 22 (Sequential Composition) $P \mathbin{;} Q \mathrel{\widehat=} \exists v_0 \bullet P[v_0/v'] \land Q[v_0/v]$

The intuition here is that the sequential composition of two relations P and Q involves some intermediate, unobservable state, whose vector of variables is represented by v_0. This vector is substituted in place for the final values of P, as represented by v', as well as substituted for the initial values of Q, as represented by v. It is finally hidden by the existential quantifier.

Skip

An important construct in the relational theory is the program $I\!I_{\mathcal{R}}$, otherwise also known as **Skip**, whose definition is presented below.

Definition 23 (Skip) $I\!I_{\mathcal{R}} \mathrel{\widehat=} (v' = v)$

This is a program that keeps the value of all variables unchanged. The most interesting property of $I\!I_{\mathcal{R}}$ is that it is the left-unit for sequential composition [39].

Demonic Choice

Due to the lattice-theoretic approach of the UTP, demonic choice ($\sqcap$) corresponds to the greatest lower bound. This means that its definition is simply disjunction.

Definition 24 (Demonic choice) $P \sqcap Q \mathrel{\widehat=} P \lor Q$

Unfortunately the least upper bound, which is conjunction, does not correspond to the notion of angelic choice. As mentioned previously, it is not possible to represent both choices directly within the relational model [38].

Recursion

Recursion is defined in the UTP as the weakest fixed point. Since we have a complete lattice, it is possible to find a complete lattice of fixed points as established by a result due to Tarski [39, 67]. In the following definition, F is a monotonic function and $\bigsqcap$ is the greatest lower bound.

Definition 25 (Recursion) $\mu X \bullet F(X) \mathrel{\widehat{=}} \bigsqcap\{X \mid [F(X) \sqsubseteq X]\}$

A non-terminating recursion, such as $(\mu\, Y \bullet Y)$, is equated with the bottom of the lattice, *true* [39]. Intuitively this means that it does not terminate, but if we sequentially compose this recursion with another program, then it becomes possible to recover from the non-terminating recursion as shown in the following example [51].

Example 4

$$\begin{aligned}
&(\mu\, Y \bullet Y) \mathbin{;} x' = 0 && \{\text{Definition of recursion}\}\\
&= \bigsqcap\{X \mid [(\mu\, Y \bullet Y)(X) \sqsubseteq X]\} \mathbin{;} x' = 0 && \{\text{Function application}\}\\
&= \bigsqcap\{X \mid [X \sqsubseteq X]\} \mathbin{;} x' = 0 && \{\text{Reflexivity of } \sqsubseteq\}\\
&= \bigsqcap\{X \mid \mathit{true}\} \mathbin{;} x' = 0 && \{\text{Property of } \sqcap\}\\
&= \mathit{true} \mathbin{;} x' = 0 && \{\text{Definition of sequential composition}\}\\
&= \exists\, v_0 \bullet \mathit{true} \land x' = 0 && \{\text{Propositional calculus}\}\\
&= x' = 0
\end{aligned}$$

This issue motivated Hoare and He [39] to propose the theory of designs that we present in the following Section 2.4.2.

2.4.2 Designs

As already mentioned, when considering theories of total correctness for reasoning about programs, the theory of relations is not appropriate due to the fact that it allows unrealistic observations of recovery from non-terminating programs [39, 51]. In other words, the bottom of the lattice, *true*, is not necessarily a left-zero of sequential composition as would be needed. As a result, Hoare and He [39] have introduced the theory of designs, which addresses this issue.

Alphabet

The theory of designs is defined by considering the addition of two auxiliary Boolean variables to the alphabet: ok and ok'. Their purpose is to track whether a program has been started, in which case ok is $true$, and whether a program has successfully terminated, in which case ok' is $true$.

In what follows we present the healthiness conditions that define the theory of designs. Finally we discuss the notion of refinement in the context of designs.

Healthiness Conditions

Any valid predicate of this theory has to obey two basic principles: that no guarantees can be made by a program before it has started, and, that no program may require non-termination. These two principles are formally characterised by the healthiness conditions **H1**, and **H2**, respectively [39]. We reproduce their definitions below.

Definition 26 $\mathbf{H1}(P) \mathrel{\widehat=} ok \Rightarrow P$

The definition of **H1** states that any guarantees made by P can only be established once it has started. Otherwise, any observation is permitted and it behaves like the bottom of the lattice, which is the same as the one for relations: $true$.

Definition 27 $\mathbf{H2}(P) \mathrel{\widehat=} [P[false/ok'] \Rightarrow P[true/ok']]$

The definition of **H2** states that if it is possible for a program P not to terminate, that is for ok' to be $false$, then it must also be possible for it to terminate, that is for ok' to be true $true$. This healthiness condition can alternatively be expressed using the J-split of [44] as $\mathbf{H2}(P) = P \ ; \ J$, where $J \mathrel{\widehat=} (ok \Rightarrow ok') \wedge v' = v$. That is, the value of ok can increase monotonically, while every other variable v is unchanged.

A predicate that is both **H1** and **H2** satisfies the following property.

Lemma L.2.4.1 (Design)

$$\mathbf{H1} \circ \mathbf{H2}(P) = (ok \wedge \neg\, P[false/ok']) \Rightarrow (P[true/ok'] \wedge ok')$$

Proof. Theorem 3.2.3 in [39]. □

Here the design is split into two parts: a precondition and a postcondition. It is defined using the notation of Hoare and He [39] as shown in the following definition.

Definition 28 (Design) $(P \vdash Q) \mathrel{\widehat=} (ok \land P) \Rightarrow (ok' \land Q)$

A design can also be written using the following notation, where we use the shorthand notation $P^a = P[a/ok']$, with $t = true$ and $f = false$, as introduced by Woodcock and Cavalcanti [51], which emphasises that we can assume without loss of generality, that ok' is not free in pre and postconditions. Furthermore, it is usually assumed that ok is also not free in either P or Q.

Lemma L.2.4.2 (Design) *A predicate P is a design if, and only if, it can be written in the following form:* $(\neg P^f \vdash P^t)$.

Proof. Theorem 3.2.3 in [39] and definition of design. □

We observe that the functions **H1** and **H2** (and indeed all of the healthiness conditions of designs) are idempotent and monotonic with respect to refinement [39]. Furthermore, none of the proofs establishing these results rely on the property of homogeneity. Therefore it is possible to define a non-homogeneous theory of designs.

Hoare and He [39] identified another two healthiness conditions of interest which we discuss further below. The third healthiness condition **H3** requires $I\!I_{\mathcal{D}}$, the **Skip** of designs, to be a right-unit for sequential composition [39].

Definition 29 (Skip) $I\!I_{\mathcal{D}} \mathrel{\widehat=} (true \vdash v' = v)$

Skip is the program that always terminates successfully and does not change the program variables. It is essentially the counterpart to $I\!I_{\mathcal{R}}$ in the theory of designs.

Definition 30 $\mathbf{H3}(P) \mathrel{\widehat=} P \ ; I\!I_{\mathcal{D}}$

From this definition it may not be immediately obvious how designs are further restricted by **H3**. In fact, it requires the precondition not to have any dashed variables (as confirmed by Theorem T.2.4.1). In order to understand the intuition behind it we consider an example of a design that is not **H3**-healthy.

Example 5

$$\begin{aligned}
&(x' \neq 2 \vdash true) && \{\text{Definition of designs}\}\\
&= (ok \land x' \neq 2) \Rightarrow ok' && \{\text{Propositional calculus}\}\\
&= ok \Rightarrow (x' = 2 \lor ok')
\end{aligned}$$

In this case we have a program that upon having started can either terminate and any final values are permitted, or can assign the value 2 to the variable x and termination is then not required. In the context of a theory of total correctness for sequential programs this is a behaviour that would not normally be expected. However it is worth noting that in the context of CSP non **H3**-designs are important, since they enable the specification of CSP processes such as $a \rightarrow Chaos$.

The healthiness condition **H3** can also be interpreted as guaranteeing that if a program may not terminate, then it has arbitrary behaviour. Thus a predicate that is **H3**-healthy is also necessarily **H2**-healthy [38].

If we expand the definition of **H3** by applying the definition of sequential definition for designs we obtain the following result [39, 51].

Theorem T.2.4.1 $((\neg P^f \vdash P^t) = (\neg P^f \vdash P^t) \; ; \; I\!I_{\mathcal{D}}) \Leftrightarrow (\neg P^f = \exists v' \bullet \neg P^f)$

Proof. Theorem 3.2.4 in [39] and proof in Section 6.3 of [51]. □

This theorem shows that the value of any dashed variables in $\neg P^f$ must be irrelevant. Therefore any design that is **H3**-healthy can only have a condition as its precondition, that is, a predicate that only mentions undashed variables, and thus can only impose restrictions on previous programs.

Finally, the last healthiness condition of interest is **H4**, which restricts designs to feasible programs. It is defined by the following algebraic equation [39] that requires that *true* is a right-zero for sequential composition.

Definition 31 (H4) $P \; ; \; true = true$

The intuition here is that this prevents the top of the lattice, $\top_D$, itself a trivial refinement of any program, from being healthy. In order to explain the intuition for this, we consider the definition of $\top_D$.

Definition 32 (Miracle)

$$\begin{aligned} \top_D &\mathrel{\widehat=} (true \vdash false) && \{\text{Property of designs}\} \\ &= ok \Rightarrow false && \{\text{Propositional calculus}\} \\ &= \neg ok \end{aligned}$$

The top $\top_D$ denotes a program that could never be started ($\neg ok$). Furthermore, if it could, and indeed its precondition makes no restriction, it would establish the impossible: *false*. Any conceivable implementable program must not behave in this way. However, miracle is an important construct in refinement calculi [38, 51].

For completeness we also provide the definition of the bottom of the lattice of designs, which is usually named **Abort**.

Definition 33 (Abort) $\bot_D \mathrel{\widehat=} (\mathit{false} \vdash \mathit{true})$

The bottom $\bot_D$ provides no guarantees at all: it may fail to terminate, and if it does terminate there are no guarantees on the final values. Indeed it is not required to guarantee anything at all since its precondition is *false*.

Operators

In the following theorems we introduce the meet and join of the lattice of designs as presented in [51]. Like in the lattice of relations, the greatest lower bound corresponds to demonic choice.

Theorem T.2.4.2 (Greatest lower bound) $\bigsqcap_i (P_i \vdash Q_i) = (\bigwedge_i P_i) \vdash (\bigvee_i Q_i)$

Proof. Theorem 1 in [51]. □

Theorem T.2.4.3 (Least upper bound) $\bigsqcup_i (P_i \vdash Q_i) = (\bigvee_i P_i) \vdash (\bigvee_i P_i \Rightarrow Q_i)$

Proof. Theorem 1 in [51]. □

Sequential Composition The definition of sequential composition for designs can be deduced from Definition 22. Here we present the result as proved in [39, 51].

Theorem T.2.4.4 (Sequential composition of designs) *Provided ok and ok′ are not free in P_0, P_1, Q_0 and Q_1,*

$$(P_0 \vdash P_1) \mathbin{;} (Q_0 \vdash Q_1) = (\neg\,(\neg\,P_0 \mathbin{;} \mathit{true}) \wedge \neg\,(P_1 \mathbin{;} \neg\,Q_0) \vdash P_1 \mathbin{;} Q_1)$$

Proof. Law $T3$ in [51]. □

This definition can be interpreted as establishing P_1 followed by Q_1 provided that P_0 holds and P_1 satisfies Q_0. As pointed out in [51], if P_0 is a condition then the definition can be further simplified.

Theorem T.2.4.5 (Sequential composition of designs) *Provided ok and ok′ are not free in P_0, P_1, Q_0 and Q_1, and P_0 is a condition,*

$$(P_0 \vdash P_1) \mathbin{;} (Q_0 \vdash Q_1) = (P_0 \wedge \neg\,(P_1 \mathbin{;} \neg\,Q_0) \vdash P_1 \mathbin{;} Q_1)$$

Proof. Law $T3'$ in [51]. □

Refinement

As in all UTP theories, the refinement order in the theory of designs is: universal (reverse) implication. Thus the following result can be established [51].

Theorem T.2.4.6 (Refinement)

$$(P_0 \vdash P_1) \sqsubseteq (Q_0 \vdash Q_1) = [P_0 \land Q_1 \Rightarrow P_1] \land [P_0 \Rightarrow Q_0]$$

Proof. Law 5 in [51]. □

Theorem T.2.4.6 confirms the intuition about refinement as found in other calculi: preconditions can be weakened while postconditions can be strengthened.

This section concludes our overview of the theory of designs. In the following section we focus on how theories can be related and combined.

2.4.3 Linking Theories

The UTP provides a very powerful framework that allows relationships to be established between different theories. This means that results in different theories can be reused. We elaborate on some of principles behind the linking of theories in the following paragraphs. A full account is available in [39].

Following the convention of Hoare and He [39], we assume the existence of a pair of functions L and R that map one theory into another: L maps the (potentially) more expressive theory into the (potentially) weaker theory, and R, vice-versa.

Subset Theories

The simplest form of relationship that can be established is that between subset theories [39]. We consider the case where a theory T is a subset of S, it is then possible to find a function $R : T \to S$: it is simply the identity [39]. Defining $L : S \to T$ for the reverse direction may be slightly more complicated as the subset theory is normally less expressive.

Hoare and He [39] pinpoint the most important properties of such a function $L : S \to T$: weakening or strengthening, idempotence and, ideally, monotonicity. As highlighted in [39], monotonicity is not always necessarily observed. We reproduce the respective definitions below.

Definition 34 (Weakening) $\forall X \in S \bullet L(X) \sqsubseteq X$

Definition 35 (Strengthening) $\forall X \in S \bullet X \sqsubseteq L(X)$

We follow Hoare and He's convention and refer to a function that is both weakening and idempotent as a link and, if it is also monotonic we refer to it as a retract.

Bijective Links

When two theories have equal expressive power, the pair of linking functions between them can be proved to form a bijection. In other words, each function undoes exactly the effect of the application of the other and, thus, as expected, the following identities hold.

Definition 36 (Bijection) *A function L is a bijection if, and only if, the inverse function $R = L^{-1}$ exists, and the following hold for all P,*

$$L \circ R(P) = P \wedge R \circ L(P) = P$$

A bijection constitutes the strongest form of relationship between theories. It can apply even when the alphabets are different or when the theories are presented in different styles [39]. Indeed this is often what is sought: proving that two theories have exactly the same expressive power, yet their shape may suit different applications better.

Galois Connections

Often, though, and as explained previously in the discussion of subset theories, we want to relate theories with different expressivity. Therefore the linking function is not a bijection, as there has to be some weakening or strengthening in either direction. A pair of functions describing this relationship constitutes what is known as a Galois connection. Here we reproduce the definition of [39] and provide a pictorial illustration in Figure 2.1.

Definition 37 (Galois Connection) *For lattices S and T, a pair (L, R) of functions $L : S \rightarrow T$ and $R : T \rightarrow S$ is defined to be a Galois connection if, and only if, for all $X \in S$ and $Y \in T$:*

$$R(Y) \sqsubseteq X \Leftrightarrow Y \sqsubseteq L(X)$$

As pointed out earlier, a bijection presents a stronger relationship than a Galois connection. However, it is not the case that every bijection is a Galois connection [39]. Hoare and He [39] give the example of negation whose inverse is precisely itself, however negation is not monotonic. It is a known property of Galois connections that

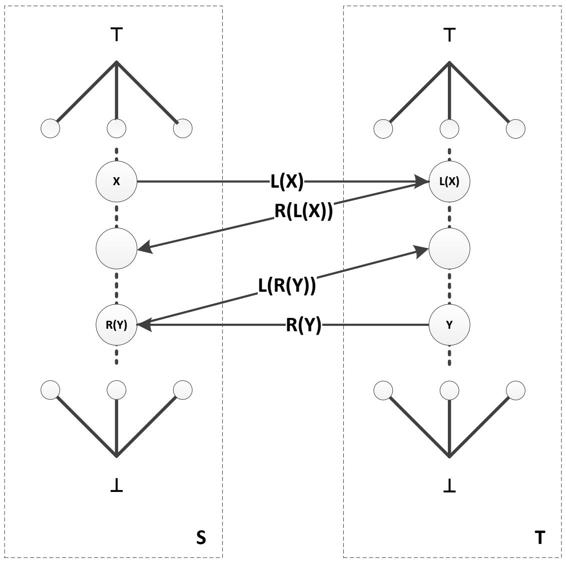


Figure 2.1: Galois connection between two lattices, S and T

the functions are monotonic. In addition, the composition of Galois connections is also a Galois connection (Theorem 4.2.5 in [39]).

2.4.4 Angelic Nondeterminism

In order to model both angelic and demonic nondeterminism in the relational setting of the UTP, Cavalcanti et al. [38] have proposed an encoding of upward-closed binary multirelations through non-homogeneous relations. The alphabet of that theory consists of the undashed program variables, whose set is given by $in\alpha$, and of the sole dashed variable ac', which is a set of final states whose components range over $out\alpha$, the output variables of a program. The final states in ac' are those available for angelic choice, while the demonic choices are those over the value of ac'. Similarly to our presentation of binary multirelations in Section 2.3, a state is a record whose components are program variables.

Despite being a theory which does not include the variables ok and ok', it directly captures termination. The intuition here is that a program may fail to terminate if there are no choices available to the angel. In other words, if ac' may be empty, then non-termination is a possibility. Conversely, if the program terminates, then there must be at least one final state available for angelic choice.

Healthiness Conditions

Since the theory is essentially a relational encoding of binary multirelations, in order for it to observe the essential properties of binary multirelations, the set of final choices ac' needs to be upward-closed. So the only healthiness condition of the theory is defined as follows [38].

Definition 38 $\mathbf{PBMH}(P) \mathrel{\widehat=} P \mathbin{;} ac \subseteq ac'$

This is a predicative version of **BMH**, which is defined using the sequential composition operator. If it were possible for P to establish some set of final states ac', then any superset could have also been obtained.

One immediate consequence of **PBMH** illustrated is that no well-behaved program can require the set of final states ac' to be empty as illustrated in the following Lemma L.2.4.3, which establishes that $ac' \neq \emptyset$ is not a fixed point of **PBMH**.

Lemma L.2.4.3 $\mathbf{PBMH}(ac' = \emptyset) = true$

Proof.

$$\begin{array}{ll}
\mathbf{PBMH}(ac' = \emptyset) & \{\text{Definition of } \mathbf{PBMH}\} \\
= ac' = \emptyset \mathbin{;} ac \subseteq ac' & \{\text{Definition of sequential composition}\} \\
= \exists\, ac_0 \bullet (ac' = \emptyset)[ac_0/ac'] \wedge (ac \subseteq ac')[ac_0/ac] & \{\text{Substitution}\} \\
= \exists\, ac_0 \bullet ac_0 = \emptyset \wedge ac_0 \subseteq ac' & \{\text{Property of sets}\} \\
= true &
\end{array}$$

□

In other words, this corresponds to the same condition enforced by **H2** of the theory of designs. Moreover, because non-termination involves ac' being empty, and since there is a requirement on ac' being upward-closed, this theory also satisfies the condition enforced by **H3** of the theory of designs: arbitrary behaviour when there is non-termination. In the following, where we discuss the operators of the theory, we establish this result by proving that the *Skip* of this theory is a right-unit for sequential composition, essentially a recast of **H3**.

Operators

The operators of the UTP theory presented in [38] are calculated from their corresponding predicate transformer's definition through a composition of linking functions that establish isomorphisms between predicate transformers, binary multirelations and the proposed UTP model. In the following paragraphs we reproduce the most important operators, whose definitions are subscripted with **A**.

Since this theory is a complete lattice, the angelic choice operator is the least upper bound, conjunction, while demonic choice corresponds to the greatest lower bound, disjunction. Furthermore, the bottom of the lattice is *true* and corresponds to abort, while *false* is the top and corresponds to miracle.

Skip The program that terminates successfully without changing the state is defined as follows.

Definition 39 $\mathit{II}_{\mathbf{A}} \mathrel{\widehat{=}} (\theta in\alpha)' \in ac'$

The definition requires that the dashed version of the initial state $\theta in\alpha$ is available for angelic choice in ac'. The notation $\theta in\alpha$ is used to denote a state where each name x in $in\alpha$ is a component associated with the corresponding program variable x, while the notation $(\theta in\alpha)'$ denotes the state obtained from $\theta in\alpha$ by dashing the name of each state component.

This operator was originally not considered in [38], but is useful, for example, to show that this theory observes the same property as **H3** of the theory of designs. This is presented following the introduction of the sequential composition operator.

Assignment The next operator of interest is assignment. An assignment of the value of an expression e to a program variable x is defined as follows.

Definition 40 (Assignment) $x :=_{\mathbf{A}} e \mathrel{\widehat{=}} (\theta in\alpha)' \oplus (x' \mapsto e) \in ac'$

The definition requires that there is a final state available for angelic choice in ac', where the dashed version of the initial state $(\theta in\alpha)$ is overridden with a component of name x' with value e.

Sequential Composition The operator that is perhaps most challenging is sequential composition. Since the theory is non-homogeneous, sequential composition is no longer relational composition as in other UTP theories. Instead, the authors in [38] have calculated the following definition, which uses substitution.

Definition 41 $P \mathbin{;_{\mathbf{A}}} Q \mathrel{\widehat{=}} P[\{s' \mid Q[s/in\alpha]\}/ac']$

The set of angelic choices resulting from composing P and Q corresponds to the angelic choices of Q, such that they can be reached from an initial state s of Q that is available for P as a set ac' of angelic choices. The states in Q are obtained by considering the substitution in Q over all variables x in $in\alpha$ with their corresponding state component $s.x$. Since states in ac' have dashed components, the set construction considers the dashed s' version of s. This definition can be interpreted as back propagating the necessary information regarding the final states.

We consider the following example, where there is a choice between angelically assigning the value 1 or 2 to the only program variable x, followed by a sequential composition with an assumption, where the program terminates successfully only when the initial value of x is 1 and otherwise aborts. For simplicity, we consider x to be the only program variable.

Example 6

$$
\begin{aligned}
&(x :=_{\mathbf{A}} 1 \sqcup x :=_{\mathbf{A}} 2) \mathbin{;_{\mathbf{A}}} (x = 1 \Rightarrow I\!I_{\mathbf{A}}) && \{\text{Definition of } \sqcup \text{ and assignment}\}\\
&= ((x' \mapsto 1) \in ac' \land (x' \mapsto 2) \in ac') \mathbin{;_{\mathbf{A}}} (x = 1 \Rightarrow I\!I_{\mathbf{A}})\\
& && \{\text{Definition of } ;_{\mathbf{A}} \text{ and } I\!I_{\mathbf{A}}\}\\
&= \begin{pmatrix} (x' \mapsto 1) \in ac' \\ \land \\ (x' \mapsto 2) \in ac' \end{pmatrix} [\{s' \mid (x = 1 \Rightarrow (x' \mapsto x) \in ac')[s/in\alpha]\}/ac']\\
& && \{\text{Substitution}\}\\
&= ((x' \mapsto 1) \in ac' \land (x' \mapsto 2) \in ac')[\{s' \mid s.x = 1 \Rightarrow (x' \mapsto s.x) \in ac'\}/ac']\\
& && \{\text{Property of substitution}\}\\
&= \begin{pmatrix} ((x' \mapsto 1) \in ac')[\{s' \mid s.x = 1 \Rightarrow (x' \mapsto s.x) \in ac'\}/ac'] \\ \land \\ ((x' \mapsto 2) \in ac')[\{s' \mid s.x = 1 \Rightarrow (x' \mapsto s.x) \in ac'\}/ac'] \end{pmatrix} && \{\text{Substitution}\}\\
&= \begin{pmatrix} ((x' \mapsto 1) \in \{s' \mid s.x = 1 \Rightarrow (x' \mapsto s.x) \in ac'\}) \\ \land \\ ((x' \mapsto 2) \in \{s' \mid s.x = 1 \Rightarrow (x' \mapsto s.x) \in ac'\}) \end{pmatrix} && \{\text{Property of sets}\}\\
&= \begin{pmatrix} (x \mapsto 1).x = 1 \Rightarrow (x' \mapsto (x \mapsto 1).x) \in ac' \\ \land \\ (x \mapsto 2).x = 1 \Rightarrow (x' \mapsto (x \mapsto 2).x) \in ac' \end{pmatrix} && \{\text{Record component } x\}\\
&= (1 = 1 \Rightarrow (x' \mapsto 1) \in ac') \land (2 = 1 \Rightarrow (x' \mapsto 2) \in ac') && \{\text{Predicate calculus}\}\\
&= (x' \mapsto 1) \in ac' && \{\text{Definition of assignment}\}\\
&= x :=_{\mathbf{A}} 1
\end{aligned}
$$

The result is that the angel avoids assigning 2 to x, since that would lead to abortion. So effectively, the information regarding the sets available for angelic choice is back propagated from the assumption through the sequential composition.

Finally, we show that this theory observes the property of **H3** of the theory of designs by expressing **H3** in this model.

Definition 42 $\mathbf{H3}_{\mathbf{A}}(P) \mathrel{\widehat{=}} P \mathbin{;_{\mathbf{A}}} I\!I_{\mathbf{A}}$

This requires the identity of the theory $I\!I_{\mathbf{A}}$ to be a right-unit, which we prove in the following lemma for healthy predicates.

Lemma L.2.4.4 $P = P \mathbin{;_{\mathbf{A}}} I\!I_{\mathbf{A}}$

Proof.

$$
\begin{array}{ll}
P \mathbin{;_{\mathbf{A}}} I\!I_{\mathbf{A}} & \{\text{Definition of } I\!I_{\mathbf{A}} \text{ and } ;_{\mathbf{A}}\} \\
= P[\{s' \mid ((\theta in\alpha)' \in ac')[s/in\alpha]\}/ac'] & \{\text{Expand } \theta in\alpha \text{ for each } x_i \text{ in } in\alpha\} \\
= P[\{s' \mid ((x_0 \mapsto x_0, \ldots, x_i \mapsto x_i)' \in ac')[s/in\alpha]\}/ac'] & \{\text{Dash state components}\} \\
= P[\{s' \mid ((x_0' \mapsto x_0, \ldots, x_i' \mapsto x_i) \in ac')[s/in\alpha]\}/ac'] & \{\text{Substitution}\} \\
= P[\{s' \mid (x_0' \mapsto s.x_0, \ldots, x_i' \mapsto s.x_i) \in ac'\}/ac'] & \{\text{Dash state components}\} \\
= P[\{s \mid (x_0' \mapsto s.x_0', \ldots, x_i' \mapsto s.x_i') \in ac'\}/ac'] & \{\text{State components}\} \\
= P[\{s \mid s \in ac'\}/ac'] & \{\text{Property of sets}\} \\
= P[ac'/ac'] & \{\text{Property of substitution}\} \\
= P &
\end{array}
$$

□

This concludes the discussion of the healthiness conditions of the theory. In what follows we discuss the relationship between this theory, binary multirelations and the predicate transformers.

Relationship with Binary Multirelations

As previously discussed, the theory of [38] is isomorphic to the theory of upward-closed binary multirelations. We depict this relationship in Figures 1.1 and 1.3 where both theories, characterised by their respective healthiness conditions **PBMH** and **BMH** are related through a pair of composed linking functions [38]. For completeness, we reproduce the result of these linking results in what follows, while the definition of each individual linking function is available in [38].

The first composition maps from this theory into the model of binary multirelations; this result is reproduced below [38].

Theorem T.2.4.7 $sb2bm \circ p2sb(P) \mathrel{\widehat=} \{s : State, ss : \mathbb{P}\, State \mid P[s, ss/in\alpha, ac']\}$

Proof. Part of Theorem 4.8 in [38], following the definitions of $p2sb$ and $sb2bm$. □

It considers every initial state s and set of final states ss, such that P holds when every initial variable x in $in\alpha$ is substituted with its corresponding state component $s.x$, and the set of final states ss is substituted for ac'.

The inverse link is established by the composition of the respective inverse linking functions $sb2p$ and $bm2sb$, whose functional composition is shown below [38].

Theorem T.2.4.8 $sb2p \circ bm2sb(B) \mathrel{\widehat=} (\theta in\alpha, ac') \in B$

Proof. Part of Theorem 4.7 in [38], following the definitions of $bm2sb$ and $sb2p$. □

For a binary multirelation B, the corresponding UTP predicate requires that every pair of initial states $\theta in\alpha$ and set of final states ac' is in B.

Relationship with Predicate Transformers

The last relationship that we discuss in this section pertains to the links between the UTP model of [38] and the monotonic predicate transformers. This is achieved in [38] through a pair of linking functions, $pt2p$, which maps from the predicate transformers model into this one, and a functional composition in the opposite direction, whose combined result we call $p2pt$. The definition of $pt2p$ is the result of Theorem 4.5 in [38], which we reproduce below.

Theorem T.2.4.9 $pt2p(PT) = \theta in\alpha \in \neg\, PT.(\neg\, ac')$

Proof. Theorem 4.5 in [38]. □

For a predicate transformer PT, $pt2p$ defines the predicate that requires that the initial state $\theta in\alpha$ is associated with all postconditions ac' that PT is not guaranteed not to establish from the initial state [38]. In this treatment of predicate transformers, predicates are modelled by their characteristic sets, such that PT is a monotonic function from sets of final states to sets of initial states [38].

The function mapping in the opposite direction is not presented in [38], however it can be calculated from the definitions of $p2sb$, $sb2bm$ and $bm2pt$, which leads to the following definition.

Definition 43 $p2pt(P)(\psi) = \{s \mid \neg\, P[s, \neg\, \psi/in\alpha, ac']\}$

This definition is justified by the following lemma.

Lemma L.2.4.5 $bm2pt(sb2bm \circ p2sb(P), \psi) = \{s \mid \neg\, P[s, \neg\, \psi / in\alpha, ac']\}$

Proof.

$$\begin{aligned}
& bm2pt(sb2bm \circ p2sb(P), \psi) && \{\text{Theorem T.2.4.7}\} \\
&= bm2pt(\{s_1, ss \mid P[s_1, ss/in\alpha, ac']\}, \psi) && \{\text{Definition of } bm2pt \text{ [38]}\} \\
&= \{s \mid (s, \neg\, \psi) \notin \{s_1, ss \mid P[s_1, ss/in\alpha, ac']\}\} && \{\text{Property of sets}\} \\
&= \{s \mid \neg\, P[s_1, ss/in\alpha, ac'][s, \neg\, \psi/s_1, ss]\} && \{\text{Substitution}\} \\
&= \{s \mid \neg\, P[s, \neg\, \psi/in\alpha, ac']\}
\end{aligned}$$

□

This result concludes our discussion regarding the theory of angelic nondeterminism in the UTP and its relationship with the standard model of predicate transformers, where angelic and demonic nondeterminism have traditionally been characterised.

2.5 Processes: CSP and Angelic Nondeterminism

Motivated by the advances of concurrency in both hardware and software, and the lack of a clear understanding of the mechanisms involved, in 1978 Hoare [68] proposed the original version of Communicating Sequential Processes (CSP). The idea was to characterise concurrent systems as the result of sequential processes that execute in parallel, and communicate and synchronize through primitive operations of input and output. However, it was not until further contributions by Hoare [16, 69], Brookes [70] and Roscoe [17, 18] that the algebra of CSP appeared, together with a complete semantics, presented in all three main flavours: algebraic, denotational and operational. This was followed by the introduction of support for model checking through Failures-Divergence Refinement (FDR) [71, 72].

In Section 2.5.1 we provide an introduction to CSP through a presentation of its most important operators and algebraic laws. In Section 2.5.2 we discuss the standard semantics of CSP as found in [18]. The material presented here is meant as background for understanding both CSP and the existing proposals for handling angelic nondeterminism, which we discuss in Section 2.5.3. A full account of CSP can be found in [17, 18]. Finally, Section 2.5.4 explores the UTP model of CSP [39, 44].

2.5.1 Notation

As the name processes in CSP suggests, the central notion of CSP is that of processes. These include basic processes, such as *Skip*, the process that terminates successfully without influence from the environment, *Stop*, which behaves as deadlock and hence refuses to do anything, and *Chaos*, which behaves unpredictably.

The other core notion of CSP is that of communication. This is achieved by defining events, which the system can perform only with the cooperation of its environment. That is, once the environment is given the possibility to perform an event, and it agrees to do so, then the event happens instantaneously and atomically. The easiest way to express this behaviour in CSP is through prefixing of events.

Definition 44 (Prefixing) $a \to P$

This process offers the environment the possibility to perform the event a, after which it behaves like P, some other CSP process. We consider the process P_0.

Example 7 $P_0 = up \to down \to Stop$

In this case a sequence of *up* and *down* events is followed by deadlock. A direct consequence of the definition of processes in this way is that recursion can occur naturally as part of the functional style of CSP as shown in the following example.

Example 8 (Mutual Recursion)

$$\begin{aligned} P_1 &= up \to P_2 \\ P_2 &= down \to P_1 \end{aligned}$$

These processes are defined by mutual recursion. The set of possible traces of events of P_1 is a superset of Example 7. It never terminates nor deadlocks.

CSP presents a rich set of operators that allow more complex interactions to be modelled. The first that we consider in the sequel is called external choice.

Definition 45 (External Choice) $P \Box Q$

In this case the environment is offered the choice between behaving as either P or Q. This operator satisfies a number of laws as reproduced below [17].

Lemma L.2.5.1 (Laws of External Choice)

$$Idempotent : P \Box P = P$$

$$\begin{aligned} \textit{Associative} &: P \Box (Q \Box R) = (P \Box Q) \Box R \\ \textit{Symmetry} &: P \Box Q = Q \Box P \\ \textit{Unit} &: P \Box \textit{Stop} = P \end{aligned}$$

Perhaps the most interesting result here is that *Stop* is the unit of external choice. When the environment is given the choice between deadlocking or behaving as P, it can only choose to behave as P.

External choice can be used to generalize the prefixing operator of Definition 44. Instead of permitting a single event, prefixing can be of a set of events $E \subseteq \Sigma$ over some alphabet Σ as follows.

Definition 46 $x : E \rightarrow P = \Box\, x : E \bullet x \rightarrow P$

This is basically a distributed external choice over all possible events in E. Moreover, CSP permits the definition of channels, which can carry values of a certain type E. For a channel name c of type E, the set of possible events that represent communications over c is defined by considering events with composed names prefixed by c as follows: $\{c.x \mid x \in E\}$. Usually in the CSP syntax, channel communications are prefixed with ? to denote input communications while ! denotes output communications, as shown in Example 9.

Example 9 (Buffer) $P_3 = \mathit{in}?x \rightarrow \mathit{out}!x \rightarrow P_3$

These annotations are syntactic sugar for the corresponding events $\mathit{in}.x$ and $\mathit{out}.x$. In this example we have an input communication over channel *in*, which is then relayed onto the output channel *out*, effectively behaving as a one place buffer.

In addition to external choice, there is an operator in CSP known as internal choice.

Definition 47 (Internal Choice) $P \sqcap Q$

This choice is also known as demonic choice, since the environment cannot possibly force the system into behaving as either P or Q. Indeed the system can choose either at its discretion. For instance, if *Stop* is offered as a choice, then the system may deadlock. This operator satisfies a number of important laws, of which a summary is included below [17].

Lemma L.2.5.2 (Laws of Internal Choice)

$$\textit{Idempotent} : P \sqcap P = P$$

$$\begin{aligned}
\textit{Associative}: & P \sqcap (Q \sqcap R) = (P \sqcap Q) \sqcap R \\
\textit{Symmetry}: & P \sqcap Q = Q \sqcap P \\
\textit{Distributive}: & P \sqcap (Q \Box R) = (P \sqcap Q) \Box (P \sqcap R)
\end{aligned}$$

Of these, distributivity is perhaps the most important. In fact, most CSP operators distribute through internal choice, except, for example, recursion [17].

The next operator of interest is that of sequential composition; it allows the composition of processes sequentially, other than by using prefixing.

Definition 48 (Sequential Composition) $P \mathbin{;} Q$

A consequence of CSP's functional language is that it is not possible to pass local process information through sequential composition. So for instance, the following process P_4 does not behave as would intuitively be expected in CSP.

Example 10 $P_4 = in?x \rightarrow Skip \mathbin{;} out!x \rightarrow Stop$

This is because the scope of x is local to both of these processes, and not global. However, this problem can be obviated by the introduction of parallelism in CSP.

CSP provides a number of different parallel composition operators [17]. Here we consider the most generic operator, which is the alphabetised parallel composition.

Definition 49 (Alphabetised Parallel Composition) $P \llbracket \alpha P \mid \alpha Q \rrbracket Q$

Alphabetised here means that processes P and Q only need to agree on events in the intersection of the alphabet of events of each process as defined in the operator: αP and αQ, respectively. Events not in the intersection do not need the agreement of both processes. For instance, to specify the behaviour that may be expected of the process P_4 from Example 10, we can consider a third process in parallel that communicates the desired value between the two processes.

Example 11 (Parallel Composition)

$$P_5 = \left(\begin{array}{l}
((in?x \rightarrow t!x \rightarrow Skip) \mathbin{;} (t?y \rightarrow out!y \rightarrow Stop)) \\
\llbracket \{\mid in, out, t \mid\} \mid \{\mid t \mid\} \rrbracket \\
(t?z \rightarrow t!z \rightarrow Skip)
\end{array}\right)$$

In this example, we add the extra channel t that serves as an internal communication channel. However, in pursuing this style of specification we have added an externally observable set of events t, which may not always be desired. CSP provides a solution for this kind of modelling problem as well.

Events can effectively be hidden from other processes when they are not needed. This abstraction is achieved in CSP by using the hiding operator.

Definition 50 (Hiding) $P \setminus E$

Here the process P has the events in the set E hidden from other processes, such that events in E become internal events that can happen irrespective of the cooperation from the environment [17]. In the following example, we give the effect of hiding the communications over t of P_5.

Example 12 (Hiding) $P_6 = P_5 \setminus \{\!| \ t \ |\!\} = \mathit{in}?x \rightarrow \mathit{out}!x \rightarrow \mathit{Stop}$

This new process P_6 is equivalent to the process that takes a communication over channel *in*, relays over channel *out* and then deadlocks.

This concludes our discussion on the notation of CSP and the most important concepts underlying its operators and algebraic properties. In the following section we focus our attention on the denotational semantics of CSP.

2.5.2 Semantics

Many interesting properties in CSP are proved using its algebraic laws. For instance, step-laws [17] provide a mechanism for a stepwise calculation of the behaviour of operators. In addition, CSP also has a denotational semantics, which we discuss in this section.

Traces

The simplest semantic model proposed for CSP considers the observable sequences of events that a process may produce. For a *CSP* process, where Σ is the set of all possible events, the set of *traces* is given by the function $\mathit{traces} : \mathit{CSP} \rightarrow \mathbb{P}(\mathrm{seq}\, \Sigma)$. For instance, the set of traces for process P_0 from Example 7 is obtained as follows.

$$\mathit{traces}(P_0) = \{\langle\rangle, \langle \mathit{up} \rangle, \langle \mathit{up}, \mathit{down} \rangle\}$$

This includes the empty sequence followed by all possible sequences of events.

Refinement in this model allows reasoning about safety, since a process P is refined by Q if, and only if, the set of trances of Q is a subset of those of P

Definition 51 (Traces Refinement) $P \sqsubseteq_T Q \Leftrightarrow \mathit{traces}(Q) \subseteq \mathit{traces}(P)$

In other words, every behaviour of Q is a possible behaviour of P. In particular, *Stop*, refines every process in the traces model, since it is a possible behaviour of every process. This motivates the definition of the following semantic model.

Failures

The following semantic model of CSP considers the set of events that may be refused by a process after a certain trace of events. This allows reasoning about liveness, in that a process like *Stop* no longer refines every other process. For a *CSP* process, the set of failures, is given by the function $\mathit{failures} : \mathit{CSP} \rightarrow \mathbb{P}(\operatorname{seq}\Sigma \times \mathbb{P}\,\Sigma)$. For example, in the case of process P_0, and assuming that the alphabet Σ is $\{\mathit{up}, \mathit{down}\}$ the failures are obtained as follows.

$$\mathit{failures}(P_0) = \left\{ \begin{array}{l} (\langle\rangle, \{\mathit{down}\}), (\langle\rangle, \emptyset), (\langle \mathit{up}\rangle, \{\mathit{up}\}), (\langle \mathit{up}\rangle, \emptyset), \\ (\langle \mathit{up}, \mathit{down}\rangle, \{\mathit{up}, \mathit{down}\}), (\langle \mathit{up}, \mathit{down}\rangle, \{\mathit{up}\}), \\ (\langle \mathit{up}, \mathit{down}\rangle, \{\mathit{down}\}), (\langle \mathit{up}, \mathit{down}\rangle, \emptyset) \end{array} \right\}$$

In other words, once the process deadlocks it refuses every possible event. Failures allow the semantics of external and internal choice to be distinguished [17].

Refinement is defined by considering the refusal pairs in addition to the traces.

Definition 52 (Failures Refinement)

$$P \sqsubseteq_F Q \Leftrightarrow \mathit{traces}(Q) \subseteq \mathit{traces}(P) \land \mathit{failures}(Q) \subseteq \mathit{failures}(P)$$

A process P is refined by Q, if, and only if, in addition to the traces of Q being a subset of those for P, the failures of Q are also a subset of P.

This is almost the complete semantics for CSP except, for the treatment of divergence, which requires one final addition to the model [17].

Failures-Divergences

Divergence can arise in CSP in different ways. For example, the most obvious is through the process *Chaos*, whose arbitrary behaviour includes divergence, while a process such as $P = P$, with an infinite recursion and no visible events, is also a divergence. The *Chaos* process in [17] is the most non-deterministic process that does not include divergence. Here we consider the behaviour of *Chaos* to be completely arbitrary, which corresponds to **div** in the standard CSP failures-divergences semantics. The approach followed in CSP is that any two processes that can diverge immediately are equivalent and useless, and that, once a process diverges, it can perform any trace of events and refuse any event [17].

The function $divergences : CSP \to \mathbb{P}(\text{seq}\,\Sigma)$ gives the set of divergences for a *CSP* process. We consider the following example, where the process P_7 offers the event a followed by divergent behaviour.

Example 13 (Divergence) $P_7 = a \to Chaos$

Its divergences are the set of all traces that lead to divergent behaviour. In the example above this is $\{s : \text{seq}\,\Sigma \mid \langle a \rangle \leq s\}$, that is, every trace that has a as the first event. In addition, because $divergences(P)$ includes every trace on which process P can diverge, the notion of failures needs to be redefined. This is because once a process has diverged it can refuse anything. These failures are obtained by the following function $failures_\perp$.

Definition 53 $failures_\perp(P) = failures(P) \cup \{s : \text{seq}\,\Sigma, ss : \Sigma \mid s \in divergences(P)\}$

A process P can then be characterised through a pair $(failures\perp(P), divergences(P))$.

Finally, the refinement order for processes P and Q in the failures-divergences model is given as follows.

Definition 54 (Failures-Divergences Refinement)

$$P \sqsubseteq_{FD} Q \Leftrightarrow failures_\perp(Q) \subseteq failures_\perp(P) \land divergences(Q) \subseteq divergences(P)$$

Process P is refined by Q if, and only if, the set of $failures_\perp$ and $divergences$ for Q are a subset of those of P. Consequently, *Chaos* is refined by every other process.

This concludes our discussion on the standard CSP semantic model of failures-divergences [17]. A full account of the CSP semantics, including the operational semantics, which is the basis for the FDR model checker, is available in [17]. In Section 2.5.4 we present the UTP model of CSP.

2.5.3 Angelic Nondeterminism in CSP

As we have previously discussed, the concept of angelic nondeterminism has also been considered in the context of CSP. Here we consider in more detail the different approaches proposed and discuss their properties.

Lattice-Theoretic Model

In [43] Tyrrell et al. present an axiomatized model for an algebra resembling CSP. At the core of their proposal is the notion that external choice, referred to as angelic choice, is a dual of internal choice in a lattice-theoretic model. This is achieved by a stepwise construction that begins with proper processes, that is, processes without choice, parallelism or recursion, which are modelled as finite sequences of events that terminate with either an empty sequence $\langle\rangle$ or with Ω. This is sufficient to give semantics to the following processes [43], where $[\_] : \mathit{Proc}(\Sigma) \to \operatorname{seq}\Sigma$ is the semantic denotation for a process, $\mathit{Proc}(\Sigma)$ is the set of all processes constructed from *Skip*, *Stop* and prefixing of events in Σ, and $\frown$ is sequence concatenation.

Definition 55 (Proper Processes)

$$\begin{aligned}
&[\mathit{Skip}] \mathrel{\widehat=} \langle\rangle\\
&[\mathit{Stop}] \mathrel{\widehat=} \Omega\\
&[a \to P] \mathrel{\widehat=} a \frown [P]
\end{aligned}$$

A partial order $\leq_P$ is then defined for $[\mathit{Proc}(\Sigma)]$, such that Ω is the least element, and for any two processes P and Q, their order is given recursively in terms of the suffix of the respective sequences of events.

Definition 56 (Refinement of Proper Processes)

$$\begin{aligned}
&\forall\, s \in [\mathit{Proc}(\Sigma)] \bullet \Omega \leq_P s\\
&\forall\, e \in \Sigma, s, t \in [\mathit{Proc}(\Sigma)] \bullet e \frown s \leq_P e \frown t \Leftrightarrow s \leq_P t
\end{aligned}$$

This corresponds to the refinement order for proper processes, where *Stop* is the least element of the order. The definition for other operators, such as restriction and sequential composition, is further specified in [43].

Having defined the refinement order for proper processes, an order-embedding is defined from the set of sequences into the FCD lattice. A lattice L is a free completely distributive lattice over a partially ordered set C, written $\mathit{FCD}(C)$, if, and only if, “there is a completion $\phi : C \to L$ such that for every FCD lattice M and function $f : C \to M$, there is a unique function $\phi^*_M : L \to M$ which is a complete homomorphism and satisfies $\phi^*_M \circ \phi = f$” [41, 43]. We illustrate this functional relationship in Figure 2.2. The FCD provides a number of interesting properties, namely, that each element can be described as the meet of joins of subsets of ϕC, or the join of meets of subsets of ϕC [43]. This is essential in the characterisation of

$$\begin{tikzcd}
 & M \\
C \arrow[ur, "f"] \arrow[r, "\phi"'] & L \arrow[u, "\phi^*_M"']
\end{tikzcd}$$

Figure 2.2: Free Completely Distributive Lattice completion

recursive processes, which is achieved through the weakest fixed point of the lattice that excludes the least element [43]. Liftings are then defined for unary and binary operators into the FCD lattice, such that internal and angelic choice correspond to the meet and join, respectively. Definitions are also given in [43] for the alphabetised parallel operator and recursive processes.

The construction of [43] provides for an elegant algebra, whose axiomatic description follows from the construction of the FCD lattice. However, with *Stop* as the least element of the refinement order, it is not possible to distinguish deadlock from divergence in this model. Thus, the semantics is quite different from the standard model of failures-divergences [17].

Operational CSP Combinators

In [18] Roscoe proposes an angelic choice operator through combinator style operational semantics of CSP. Traditionally [17, 18], the operational semantics of CSP has been defined through a Labelled Transition System (LTS). An LTS is a directed graph, where each edge is labelled with an action that denotes what happens when the system transitions between states. In CSP the set of possible labels includes the events in Σ and another two special events: $\checkmark$ which signals successful termination and does not require the cooperation of the environment (such as in the case of *Skip*), and τ which is an internal event invisible to the environment. Hence, $\checkmark$ is always the last event possible and leads to a special end state Ω.

Operational semantics for CSP operators can be given in the style of Plotkin's Structured Operational Semantics (SOS) [73]. For example, the process *Stop* has no actions, while *Skip* can be given the following rule [18].

$$\frac{}{\mathit{Skip} \xrightarrow{\checkmark} \Omega}$$

Since the transition relation always associates *Skip* to Ω with action $\checkmark$, the bar is empty above, while the transition below means that *Skip* can transition into the final special state Ω by doing action $\checkmark$. External choice, on the other hand, requires more rules since an internal event τ does not decide the choice [18].

$$\frac{P \xrightarrow{\tau} P'}{P \,\Box\, Q \xrightarrow{\tau} P' \,\Box\, Q}\;,\quad \frac{Q \xrightarrow{\tau} Q'}{P \,\Box\, Q \xrightarrow{\tau} P \,\Box\, Q'}$$

In these two cases, an internal action can be performed by either P or Q, in which case, the τ event is promoted, while the choice is not resolved. Any other event a, including $\checkmark$, decides the choice between processes P and Q.

$$\frac{P \xrightarrow{a} P'}{P \,\Box\, Q \xrightarrow{a} P'}(a \neq \tau),\quad \frac{Q \xrightarrow{a} Q'}{P \,\Box\, Q \xrightarrow{a} Q'}(a \neq \tau)$$

Given the number of different rules needed to specify an operator, and the fact that it is actually possible to define operators that are not conformant with the failures-divergences semantics of CSP [18], Roscoe proposes an alternative known as combinator style operational rules. The idea is that it is possible to distinguish process arguments whose actions are immediately relevant from those that are not [18]. The latter are **off**, while the former are **on**. Thus the semantics of external choice can be given as

$$((a, .), a, \mathbf{1}), ((., a), a, \mathbf{2}) \text{ for each } a \in \Sigma$$

where each triple is defined by: a tuple that denotes the actions that each **on** process performs (with . indicating none), ordered according to the indices of the arguments, the overall action performed, and the format of the resulting state given in CSP syntax. In the case of external choice, for each event a in Σ, either the first process, whose tuple is $(a, .)$, or the second process, whose tuple is $(., a)$ can decide the choice. The resulting event performed by the system is a, and the resulting state is either $\mathbf{1}$, which corresponds to the first process or $\mathbf{2}$, which corresponds to the second process.

An assumption of this style of specification is that τ events are always promoted for arguments that are **on**, so there is no need to include rules for this [18]. Finally, the specification of the external choice operator also requires rules for termination:

$$((\checkmark, .), \checkmark, \Omega), ((., \checkmark), \checkmark, \Omega)$$

In this case, the termination of either process leads to termination, in which case the system transitions to the special state Ω, with the visible action being $\checkmark$.

The interesting result about this style of operational specification, is that every such operator conforms to the failures-divergences semantics of CSP, and Roscoe [18] envisions this as a mechanism for adding new operators to FDR. Moreover, in [18] Roscoe also gives a CSP process, which is able to simulate processes specified using combinator style semantics.

Having defined his combinator-style operational rules, Roscoe [18] proposes an angelic choice operator $P \boxdot Q$ (Example 9.2 in [18]), which gives the environment a choice over both actions P and Q as long as the environment picks one that they both offer. In fact, to achieve this definition Roscoe defines a family of operators $P\ {}_{s}\boxdot\ Q$ and $P \boxdot_{s} Q$, where s is a non-empty trace that keeps track of the difference in events performed "ahead" by the other operand. The operational semantics of this angelic choice operator is reproduced below [18].

- For $\boxdot$: $\forall\, a \in \Sigma$: $((a,.), a, \mathbf{1} \boxdot_{\langle a\rangle} \mathbf{2})$, $((.,a), a, \mathbf{1}_{\langle a\rangle} \boxdot \mathbf{2})$
  $((\checkmark,.), \checkmark, \Omega)$ and $((.,\checkmark), \checkmark, \Omega)$
- For ${}_{\langle b\rangle ^\frown s}\boxdot$: $\forall\, a \in \Sigma$: $((b,.), \tau, \mathbf{1}_{s} \boxdot \mathbf{2})$, $((.,a), a, \mathbf{1}_{\langle a,b\rangle ^\frown s} \boxdot \mathbf{2})$
  $((\checkmark,.), \tau, \mathbf{2})$ and $((.,\checkmark), \checkmark, \Omega)$
- For $\boxdot_{\langle b\rangle ^\frown s}$: $\forall\, a \in \Sigma$: $((.,b), \tau, \mathbf{1} \boxdot_{s} \mathbf{2})$, $((a,.), a, \mathbf{1} \boxdot_{\langle a,b\rangle ^\frown s} \mathbf{2})$
  $((\checkmark,.), \checkmark, \Omega)$ and $((.,\checkmark), \tau, \mathbf{1})$

The first set of rules for $P \boxdot Q$ considers the case where either P or Q perform the event a, in which case the event a is visible. If P performs event a, then the resulting process $P \boxdot_{\langle a\rangle} Q$ has the sequence $\langle a\rangle$ corresponding to the events Q could catch up to. Similarly, there is a rule for the case when Q performs the event a. If either process terminates, then $\checkmark$ is observed and the system transitions to Ω.

The second set of rules for $P_{\langle b\rangle ^\frown s} \boxdot Q$ considers the case where process Q is ahead. If P performs the event b, then an internal event is observed, and the resulting process $P_{s} \boxdot Q$ considers the tail s of the sequence. Process Q could perform another a event and step further ahead, in which case a is appended to the initial sequence $\langle b\rangle ^\frown s$. If P terminates, then an internal event τ is observed and the choice is resolved in favour of Q. Otherwise if Q terminates, then $\checkmark$ is observed and the system transitions into Ω. The last set of rules describes the case where P is ahead of Q instead.

In summary, a process whose trace is behind the other is allowed to catch up, while if it terminates then the choice resolves in favour of the other process. We consider the following example, with $\Sigma = \{a, b\}$.

Example 14 $a \rightarrow \mathit{Chaos} \boxdot a \rightarrow \mathit{Skip}$

Suppose the left-hand side process $a \to \mathit{Chaos}$ performs event a first, then we arrive at the configuration $\mathit{Chaos} \boxdot_{\langle a \rangle} a \to \mathit{Skip}$. Now either $a \to \mathit{Skip}$ catches up, in which case the process can then potentially terminate, or we observe events from *Chaos* with the potential for non-termination. Similar reasoning applies to the case where the right-hand side performs event a first. In other words, an equivalent CSP process describing this behaviour would be $a \to (\mathit{Chaos} \sqcap \mathit{Skip})$, where following the event a, it may terminate or diverge. Essentially, this angelic choice operator is a variant of the external choice operator that is able to delay the choice between either branch, as long as the environment can control that choice.

It is clear from Example 14 that the angelic choice operator of Roscoe [18] is not able to avoid divergence. Ideally, a counterpart to the angelic choice of the refinement calculus should avoid divergence and favour successfully terminating processes, just like in most theories of angelic nondeterminism.

2.5.4 UTP Model

As we have previously discussed, CSP can be characterised in the UTP through the theory of reactive processes [39, 44]. In addition to the variables ok and ok' of the theory of designs, this theory includes the variables $wait$, tr, ref and their dashed counterparts, that record information about interactions with the environment.

The variable $wait$ records whether the previous process is waiting for an interaction from the environment or, alternatively, has terminated. Similarly, $wait'$ ascertains this for the current process. The variable ok indicates whether the previous process is in a stable state, while ok' records this information for the current process. If a process is not in a stable state, then it is said to have diverged. A process only starts executing in a state where ok and $\neg\, wait$ are $true$. Successful termination is characterised by ok' and $\neg\, wait'$ being $true$.

Like in standard CSP, the interactions with the environment are represented using sequences of events, recorded by tr and tr'. The variable tr records the sequence of events that took place before the current process started, while tr' records all the events that have been observed so far. Finally, ref and ref' record the set of events that may be refused by the process at the start, and currently, as required for the appropriate modelling of deadlock [17].

Healthiness Conditions

The theory of reactive processes **R** is characterised by the functional composition of the following three healthiness conditions, which we reproduce below [39, 44].

Definition 57 (Reactive Process)

$$\begin{aligned}
&\mathbf{R1}(P) \mathrel{\widehat=} P \land tr \leq tr' \\
&\mathbf{R2}(P) \mathrel{\widehat=} P[\langle\rangle, tr' - tr/tr, tr'] \\
&\mathbf{R3}(P) \mathrel{\widehat=} I\!I_{rea} \lhd wait \rhd P \\
&\mathbf{R}(P) \mathrel{\widehat=} \mathbf{R3} \circ \mathbf{R1} \circ \mathbf{R2}(P)
\end{aligned}$$

R1 requires that in all circumstances the only change that can be observed in the final trace of events tr' is an extension of the initial sequence tr, while **R2** requires that a process must not impose any restriction on the initial value of tr. Finally, **R3** requires that if the previous process is waiting for an interaction with the environment, that is *wait* is *true*, then the process behaves as the identity of the theory $I\!I_{rea}$ [39, 44], otherwise it behaves as P. The healthiness condition of the theory of reactive processes is **R**, the functional composition of **R1**, **R2** and **R3**.

CSP Processes as Reactive Designs

The theory of CSP can be described by reactive processes that in addition also satisfy two other healthiness conditions, **CSP1** and **CSP2**, whose definitions are reproduced below [39, 44].

Definition 58 (CSP)

$$\begin{aligned}
&\mathbf{CSP1}(P) \mathrel{\widehat=} P \lor \mathbf{R1}(\neg\, ok) \\
&\mathbf{CSP2}(P) \mathrel{\widehat=} P \ ; ((ok \Rightarrow ok') \land tr' = tr \land ref' = ref \land wait' = wait)
\end{aligned}$$

The first healthiness condition **CSP1** requires that if the previous process has diverged, that is, *ok* is *false*, then extension of the trace is the only guarantee. **CSP2** is **H2**, using the J-split of Cavalcanti and Woodcock [44], restated with the extended alphabet of reactive processes.

A process that is **R**, **CSP1** and **CSP2**-healthy can be described in terms of a design as proved in [39, 44]. We reproduce this result below, where we use the notation $P^o_w = P[o, w/ok', wait]$.

Theorem T.2.5.1 (Reactive Design) *For every CSP process P,*

$$\mathbf{R}(\neg\, P^f_f \vdash P^t_f) = P$$

Proof. Theorem 12 in [44], or Theorem 8.2.2 in [39]. □

This result is important as it allows CSP processes to be specified in terms of pre and postconditions, such as is the case for sequential programs, while the healthiness condition $\mathbf{R}$ enforces the required reactive behaviour.

Operators

The operators of CSP can then be defined using reactive designs. In what follows we present the most important CSP operators and discuss their specification, where use the subscript $\mathbf{R}$ to distinguish these definitions from those in other theories.

The first process of interest is $Skip_{\mathbf{R}}$, which terminates successfully.

Definition 59 (Skip) $Skip_{\mathbf{R}} \mathrel{\widehat=} \mathbf{R}(true \vdash tr' = tr \land \neg\, wait')$

Its precondition is *true* since it never diverges and its postcondition requires that the trace of events tr is unchanged while it terminates $\neg\, wait'$.

On the other hand, the process that never terminates is defined by $Stop_{\mathbf{R}}$.

Definition 60 (Stop) $Stop_{\mathbf{R}} \mathrel{\widehat=} \mathbf{R}(true \vdash tr' = tr \land wait')$

Its precondition is *true* while the postcondition requires that not only is the trace of events tr never changed, but the process is always waiting for the environment: $wait'$ is *true*.

Immediate divergence is captured by the process $Chaos_{\mathbf{R}}$.

Definition 61 (Chaos) $Chaos_{\mathbf{R}} \mathrel{\widehat=} \mathbf{R}(false \vdash true)$

In this case, the precondition is *false*, since it always diverges, then there is no way to satisfy the precondition of this process, and its postcondition is *true*. In fact, this design becomes just *true*, and the function $\mathbf{R}$ ensures that the only observation that can be made is the extension of the sequence of traces tr.

Prefixing can be described in terms of reactive designs as follows.

Definition 62 (Prefixing)

$$a \rightarrow_{\mathbf{R}} Skip_{\mathbf{R}} \mathrel{\widehat=} \mathbf{R}(true \vdash (tr' = tr \land a \notin ref') \lhd wait' \rhd (tr' = tr \frown \langle a \rangle))$$

The precondition is *true*, while in the postcondition there is a conditional, which defines two possible observations of its behaviour. When the process is still waiting for an interaction from the environment, and $wait'$ is *true*, then the trace of events remains unchanged while the event a is not in the set of refusals ref'. When the process is no longer waiting, and *wait′* is *false*, then the event a is appended to the initial trace of events *tr*.

In the case of internal choice the environment has no control over the choice.

Definition 63 (Internal Choice) $P \sqcap_{\mathbf{R}} Q \mathrel{\widehat=} \mathbf{R}(\neg P^f_f \land \neg Q^f_f \vdash P^t_f \lor Q^t_f)$

In this case the precondition requires that the precondition of both processes P and Q, $\neg P^f_f$ and $\neg Q^f_f$, holds. Moreover, the postcondition is the disjunction of the postconditions of P and Q, P^t_f and Q^t_f, respectively, as either postcondition may be established.

External choice, on the other hand, presents a more complex definition as a reactive design.

Definition 64 (External Choice)

$$P \mathbin{\Box_{\mathbf{R}}} Q \mathrel{\widehat=} \mathbf{R}(\neg P^f_f \land \neg Q^f_f \vdash (P^t_f \land Q^t_f) \lhd tr' = tr \land wait' \rhd (P^t_f \lor Q^t_f))$$

Like in the definition for internal choice, both preconditions of P and Q need to be satisfied. The postcondition defines two cases: when the process is waiting and the trace of events has not changed, and the only possible observations of the external choice are those that are admitted by the postconditions of both processes, and, once a choice is made, the observations are either those of P or Q, according to the postconditions.

The final, and perhaps most complex, yet fundamental operator that we consider in this discussion is sequential composition.

Definition 65 (Sequential Composition)

$$P \mathbin{;_{\mathbf{R}}} Q \mathrel{\widehat=} \mathbf{R}\left(\begin{array}{l} \left(\begin{array}{l} \neg\,(\mathbf{R1}(P^f_f) \ ; \mathbf{R1}(\mathit{true})) \\ \land \\ \neg\,(\mathbf{R1}(P^t_f) \ ; (\neg\, \mathit{wait} \land \mathbf{R1} \circ \mathbf{R2}(Q^f_f))) \end{array}\right) \\ \vdash \\ \mathbf{R1}(P^t_f) \ ; (\mathit{II} \lhd \mathit{wait} \rhd \mathbf{R1} \circ \mathbf{R2}(Q^t_f)) \end{array}\right)$$

The precondition is the conjunction of two terms, the first of which requires that the precondition of P is satisfied. This is similar to the sequential composition of designs (Theorem T.2.4.4), apart from the fact that **R1** is required to hold. The second term requires that the postcondition of P satisfies the precondition of Q when *wait* is no longer *true*, that is, when it actually starts executing. This is again similar to the result for designs, apart from the fact there is the variable *wait* and that **R1** must hold, and so must **R2** for the negation of the precondition of Q. Finally, the postcondition is given by the sequential composition of the postcondition of P with a conditional, where: if P is still waiting for the environment, then it behaves as the identity II, otherwise it behaves as the postcondition of Q, where both **R1** and **R2** are required to hold.

This concludes our discussion of the UTP model of CSP. We have covered the definition of the most important operators as reactive designs. In the following section we summarise the main points of this chapter.

2.6 Final Considerations

The concept of angelic nondeterminism has been employed in many different applications as we have discussed. Its original treatment made the abstract specification of algorithms in problems involving backtracking and search possible. In the context of theories of correctness, it has traditionally been studied in the refinement calculus of Back [32], Morris [29] and Morgan [31] through the universal monotonic predicate transformers, where it can be characterised as the least upper bound of the lattice.

In the context of relational theories, however, capturing both angelic and demonic nondeterminism is not entirely trivial. Rewitzky [35] provided the fundamental theory of binary multirelations in which angelic nondeterminism can be characterised in terms of relations between states and sets of states. This has been used by Cavalcanti et al. [38] to encode both angelic and demonic nondeterminism in the relational setting of Hoare and He's UTP [39], a framework suitable for studying different programming paradigms, including process algebras like CSP.

CSP has received some attention regarding the concept of angelic nondeterminism as well. In particular, Tyrrel et al. [43] have suggested a lattice-theoretic model for an algebra resembling CSP where angelic choice is the dual of internal choice. However, the semantics is quite different from the standard model of failures-divergences of CSP [17, 18]. Roscoe has also proposed an angelic choice operator, which however, does not avoid divergent behaviour. Ideally, an angelic choice counterpart to the refinement calculus should avoid divergent behaviour. This notion, however, has been elusive. We address this problem in the remainder of this thesis.

Chapter 3

Extended Binary Multirelations

In this chapter we introduce an extended model of binary multirelations that caters for sets of final states that are not necessarily terminating. This is achieved by extending Rewitzky's [35] model of upward-closed binary multirelations with a special state that denotes the possibility for non-termination.

The following Section 3.1 introduces the model. In Section 3.2 the healthiness conditions are defined; their characterisation as fixed points is discussed in Section 3.3. In Section 3.4 the refinement order is defined, while the operators are defined in Section 3.5. Section 3.6 formalizes the relationship between this model and that of [35]. Finally, we summarize our results in Section 3.7.

3.1 Introduction

Similarly to the original model of binary multirelations, a relation in this model associates to each initial program state a set of final states. The notion of a final state, however, is different, as formalised by the following type $BM_\perp$.

Definition 66 (Extended Binary Multirelation)

$$\begin{aligned}
&State_\perp == State \cup \{\perp\} \\
&BM_\perp == State \leftrightarrow \mathbb{P}\, State_\perp
\end{aligned}$$

Each initial state is related to a set of final states of type $State_\perp$, a set that may contain the special state $\perp$, which denotes non-termination. If a set of final states does not contain $\perp$, then termination in one of its states is guaranteed.

Similar to the original theory of binary multirelations, the set of final states encodes the choices available to the angel. The demonic choices are encoded by the different ways in which the set of final states can be chosen.

We consider the following example, where the value 1 is assigned to the program variable x, but termination is not guaranteed. This is specified by the following relation, where $:=_{BM_\perp}$ is the assignment operator that does not require termination.

Example 15 $x :=_{BM_\perp} 1 = \{s : State, ss : \mathbb{P}\, State_\perp \mid s \oplus (x \mapsto 1) \in ss\}$

Every initial state s is related to a set of final states ss where the state obtained from s by overriding the value of the component x with 1 is included. Since ss is of type $State_\perp$, the sets of final states ss include those with and without $\perp$. The angelic choice, therefore, cannot guarantee termination. In the following examples and definitions we may omit the type of s and ss for conciseness; they always have the same types as in Example 15.

It is also possible to specify a program that must terminate for certain sets of final states but not necessarily for others as shown in the following example, where $\sqcap_{BM_\perp}$ is the demonic choice operator of the theory.

Example 16

$$\begin{aligned}
&(x :=_{BM} 1) \sqcap_{BM_\perp} (x :=_{BM_\perp} 2)\\
&=\\
&\{s, ss \mid (s \oplus (x \mapsto 1) \in ss \land \perp \notin ss) \lor (s \oplus (x \mapsto 2) \in ss)\}
\end{aligned}$$

Since BM is in fact a subset of $BM_\perp$, it is possible to use some of the existing operators, such as the terminating assignment operator $:=_{BM}$. In this case, there is a demonic choice between the terminating assignment of 1 to x, and the assignment of 2 to x that does not require termination.

3.2 Healthiness Conditions

Having defined the type of the extended binary multirelations $BM_\perp$, in the following Sections 3.2.1 to 3.2.4 we introduce the healthiness conditions that characterise the relations in the theory.

3.2.1 BMH0

The first healthiness condition of interest is **BMH0**. It enforces the upward closure of the original theory of binary multirelations [35] for sets of final states that are necessarily terminating, and in addition enforces a similar property for sets of final states that are not required to terminate.

Definition 67 (BMH0)

$$\forall\, s, ss_0, ss_1 \bullet ((s, ss_0) \in B \land ss_0 \subseteq ss_1 \land (\bot \in ss_0 \Leftrightarrow \bot \in ss_1)) \Rightarrow (s, ss_1) \in B$$

It states that for every initial state s, and for every set of final states ss_0 in a relation B, any superset ss_1 of that final set of states is also associated with s such that $\bot$ is in ss_0 if, and only if, it is in ss_1. That is, **BMH0** requires the upward closure for sets of final states that terminate, and for those that may or may not terminate, but separately.

The definition of **BMH0** can be split into two conjunctions as established by the following Lemma L.3.2.1. **BMH** is the healthiness condition of the original theory whose definition was reproduced in Section 2.3. Proof of these and other results to follow can be found in Appendix B of the extended version of this thesis [74].

Lemma L.3.2.1

$$\begin{array}{l}
\mathbf{BMH0} \\
\Leftrightarrow \\
\left(\begin{array}{l}
\left(\begin{array}{l}
\forall\, s, ss_0, ss_1 \bullet \\
((s, ss_0) \in B \land ss_0 \subseteq ss_1 \land \bot \in ss_0 \land \bot \in ss_1) \Rightarrow (s, ss_1) \in B
\end{array}\right) \\
\land \\
\mathbf{BMH}
\end{array}\right)
\end{array}$$

This result confirms that for sets of final states that terminate this healthiness condition enforces **BMH** exactly as in the original theory of binary multirelations [35].

3.2.2 BMH1

The second healthiness condition **BMH1** requires that if it is possible to choose a set of final states where termination is not guaranteed, then it must also be possible to choose an equivalent set of states where termination is guaranteed. This healthiness condition is similar in nature to **H2** of the theory of designs.

Definition 68 (BMH1) $\forall\, s : \mathit{State}, ss : \mathbb{P}\, \mathit{State}_\bot \bullet (s, ss \cup \{\bot\}) \in B \Rightarrow (s, ss) \in B$

If it is possible to reach a set of final states $(ss \cup \{\bot\})$ from some initial state s, then the set of final states ss, without $\bot$, so that termination is required, is also associated with s.

This healthiness condition excludes relations that only offer sets of final states that may not terminate. We consider the following Example 17.

Example 17 $\{s : State, ss : \mathbb{P}\, State_\perp \mid s \oplus (x \mapsto 1) \in ss \land \perp \in ss\}$

This relation describes the assignment of 1 to the program variable x where termination is not guaranteed. It discards the inclusive situation where termination may indeed occur, and so is not **BMH1**-healthy. The inclusion of a corresponding final set of states that requires termination does not change the choices available to the angel as it is still impossible to guarantee termination.

3.2.3 BMH2

In this model, both the empty set of final states and $\{\perp\}$ characterise abortion. This redundancy, which facilitates the linking between theories, in particular with the original theory of Rewitzky [35], is captured by the following healthiness condition.

Definition 69 (BMH2) $\forall s : State \bullet (s, \emptyset) \in B \Leftrightarrow (s, \{\perp\}) \in B$

It requires that every initial state s is related to the empty set of final states if, and only if, it is also related to the set of final states $\{\perp\}$. By allowing $(s, \emptyset)$ to be part of the model, we can easily characterise the original theory of binary multirelations as a subset of ours.

If we consider **BMH1** in isolation, it covers the reverse implication of **BMH2** because if $(s, \{\perp\})$ is in the relation, so is $(s, \emptyset)$. However, **BMH2** is stronger than **BMH1** by requiring $(s, \{\perp\})$ to be in the relation if $(s, \emptyset)$ is also in the relation.

This new model of binary multirelations is characterised by the conjunction of the healthiness conditions **BMH0**, **BMH1** and **BMH2** to which we refer as $\mathbf{BMH}_\perp$. In Section 3.3 we provide alternative definitions of the healthiness conditions in terms of fixed points. This characterisation enables us, for instance, to establish that the healthiness conditions are idempotent and monotonic.

3.2.4 BMH3

The fourth healthiness condition characterises a subset of the model that corresponds to the original theory of binary multirelations of Rewitzky [35].

Definition 70 (BMH3)

$$\forall s : State \bullet (s, \emptyset) \notin B \Rightarrow (\forall ss : \mathbb{P}\, State_{\perp} \bullet (s, ss) \in B \Rightarrow \perp \notin ss)$$

If an initial state s is not related to the empty set, then it must be the case that for all sets of final states ss related to s, $\perp$ is not included in the set ss.

The healthiness condition **BMH3** excludes relations that do not guarantee termination for particular initial states, yet establish some set of final states. An example of such a relation is Example 15. This is also the case for the original theory of binary multirelations. If it is possible for a program not to terminate when started from some initial state, then execution from that state must lead to arbitrary behaviour. This is the same intuition for **H3** of the theory of designs [39].

3.3 Healthiness Conditions as Fixed Points

Having defined the healthiness conditions of the theory, in this section we consider their definitions via idempotent functions, whose fixed points are the relations in the theory. This is similar to the approach followed in UTP theories. This dual characterisation is used in Section 3.6 to establish an isomorphism between a subset of this model and the original theory of binary multirelations.

For each healthiness condition of interest, we use the notation $\mathbf{bmh_x}$ to denote the function whose fixed points correspond exactly to the relations characterised by the healthiness condition **BMHx**, that is $\mathbf{bmh_x}(B) = B \Leftrightarrow \mathbf{BMHx}$. Furthermore, the notation $\mathbf{bmh_{x,y}}$ denotes the functional composition of the functions $\mathbf{bmh_x}$ and $\mathbf{bmh_y}$, so that $\mathbf{bmh_{x,y}}(B) = \mathbf{bmh_x} \circ \mathbf{bmh_y}(B)$.

In the next Section 3.3.1, each healthiness condition is characterised by a corresponding function. A full account of the properties of the functional composition of each function is found in Appendix B.2. Moreover, in Sections 3.3.2 and 3.3.3 the two functions that characterise the model as a whole, and its subset of interest, are presented.

3.3.1 $\mathbf{bmh_0}$, $\mathbf{bmh_1}$, $\mathbf{bmh_2}$ and $\mathbf{bmh_3}$

The first function of interest is $\mathbf{bmh_0}$, whose fixed points are the **BMH0**-healthy binary multirelations. It is defined as follows.

Definition 71

$$\mathbf{bmh_0}(B) \mathrel{\widehat=} \{s, ss \mid \exists\, ss_0 \bullet (s, ss_0) \in B \land ss_0 \subseteq ss \land (\bot \in ss_0 \Leftrightarrow \bot \in ss)\}$$

For every initial state s in B, whenever it is related to a set of final states ss_0 it is also related to its superset ss, such that $\bot$ is in ss_0 if, and only if, $\bot$ is also in ss. In other words, $\mathbf{bmh_0}$ enforces the upward closure of a relation B just like **BMH0**.

The healthiness condition **BMH1** is characterised by the fixed points of $\mathbf{bmh_1}$.

Definition 72 $\mathbf{bmh_1}(B) \mathrel{\widehat=} \{s, ss \mid (s, ss \cup \{\bot\}) \in B \lor (s, ss) \in B\}$

Its definition considers all pairs (s, ss) in B, such that if a set of final states includes $\bot$ then there is also a set of final states without $\bot$.

BMH2-healthy relations are fixed points of the function $\mathbf{bmh_2}$, whose definition is presented below.

Definition 73 $\mathbf{bmh_2}(B) \mathrel{\widehat=} \{s, ss \mid (s, ss) \in B \land ((s, \{\bot\}) \in B \Leftrightarrow (s, \emptyset) \in B)\}$

The definition considers every pair (s, ss) in B and requires that $(s, \{\bot\})$ is in B if, and only if, $(s, \emptyset)$ is also in B. If the equivalence is not satisfied then $\mathbf{bmh_2}$ yields the empty set.

Finally, the **BMH3**-healthy relations are characterised by the fixed points of $\mathbf{bmh_3}$.

Definition 74 $\mathbf{bmh_3}(B) \mathrel{\widehat=} \{s, ss \mid ((s, \emptyset) \in B \lor \bot \notin ss) \land (s, ss) \in B\}$

The definition considers every pair (s, ss) in B and requires that either ss is a set of final states with guaranteed termination, and so without $\bot$, or $(s, \emptyset)$ is in B, and thus the initial state s leads to arbitrary behaviour.

The following Lemmas L.3.3.1 to L.3.3.4 establish that the fixed points of each $\mathbf{bmh_x}$ function are exactly those relations that satisfy the corresponding healthiness condition **BMHx**.

Lemma L.3.3.1 **BMH0** $\Leftrightarrow \mathbf{bmh_0}(B) = B$

Lemma L.3.3.2 **BMH1** $\Leftrightarrow \mathbf{bmh_1}(B) = B$

Lemma L.3.3.3 **BMH2** $\Leftrightarrow \mathbf{bmh_2}(B) = B$

Lemma L.3.3.4 **BMH3** $\Leftrightarrow \mathbf{bmh_3}(B) = B$

Furthermore, the following Lemmas L.3.3.5 to L.3.3.8 establish that each $\mathbf{bmh_x}$ function is idempotent.

Lemma L.3.3.5 $\mathbf{bmh_0} \circ \mathbf{bmh_0}(B) = \mathbf{bmh_0}(B)$

Lemma L.3.3.6 $\mathbf{bmh_1} \circ \mathbf{bmh_1}(B) = \mathbf{bmh_1}(B)$

Lemma L.3.3.7 $\mathbf{bmh_2} \circ \mathbf{bmh_2}(B) = \mathbf{bmh_2}(B)$

Lemma L.3.3.8 $\mathbf{bmh_3} \circ \mathbf{bmh_3}(B) = \mathbf{bmh_3}(B)$

This section concludes our discussion regarding the definition of the $\mathbf{bmh_x}$ functions. Properties of their functional composition are studied in detail in Appendix B.2. In the following Sections 3.3.2 and 3.3.3 we focus our attention only on the functional compositions that characterise the theory of **BMH0-BMH2** multirelations and the subset, that in addition, satisfies **BMH3**.

3.3.2 $\mathbf{bmh_{0,1,2}}$

The relations in the model of extended binary multirelations are characterised by the conjunction of the healthiness conditions **BMH0**, **BMH1** and **BMH2**, otherwise also named as $\mathbf{BMH}_\perp$ as depicted in Figure 1.1. These relations can also be expressed as fixed points of the functional composition of the functions $\mathbf{bmh_0}$, $\mathbf{bmh_1}$ and $\mathbf{bmh_2}$, as shown by the following Lemma L.3.3.9.

Lemma L.3.3.9

$$\mathbf{bmh_{0,1,2}}(B) = \left\{ s, ss \middle| \begin{array}{l} \exists ss_0 \bullet ((s, ss_0) \in B \lor (s, ss_0 \cup \{\perp\}) \in B) \\ \land ((s, \{\perp\}) \in B \Leftrightarrow (s, \emptyset) \in B) \\ \land ss_0 \subseteq ss \land (\perp \in ss_0 \Leftrightarrow \perp \in ss) \end{array} \right\}$$

The notation $\mathbf{bmh_{0,1,2}}$ denotes the functional composition $\mathbf{bmh_0} \circ \mathbf{bmh_1} \circ \mathbf{bmh_2}$. The order of this functional composition is justified by Theorem T.3.3.1, and results established in Appendices B.2.5 and B.2.6.

Theorem T.3.3.1 $\mathbf{BMH0} \land \mathbf{BMH1} \land \mathbf{BMH2} \Leftrightarrow \mathbf{bmh_{0,1,2}}(B) = B$

Proof. Follows from Lemmas L.3.3.10 to L.3.3.13 below. □

That is, a multirelation B is a fixed point of $\mathbf{bmh_{0,1,2}}$, if, and only if, it satisfies the healthiness conditions **BMH0**, **BMH1** and **BMH2**. The proof of this theorem relies on the results which we discuss in the following paragraphs.

First we establish in Lemmas L.3.3.10 to L.3.3.12 that a fixed point of $\mathbf{bmh_{0,1,2}}$ satisfies each of the healthiness conditions **BMH0**, **BMH1** and **BMH2**.

Lemma L.3.3.10 $(\mathbf{bmh_{0,1,2}}(B) = B) \Rightarrow \mathbf{BMH0}$

Lemma L.3.3.11 $(\mathbf{bmh_{0,1,2}}(B) = B) \Rightarrow \mathbf{BMH1}$

Lemma L.3.3.12 $(\mathbf{bmh_{0,1,2}}(B) = B) \Rightarrow \mathbf{BMH2}$

Moreover, we establish in Lemma L.3.3.13 that a relation that is **BMH0**, **BMH1** and **BMH2**-healthy is also a fixed point of $\mathbf{bmh_{0,1,2}}$.

Lemma L.3.3.13 *Provided B is* **BMH0** – **BMH2**-*healthy,* $\mathbf{bmh_{0,1,2}}(B) = B$.

These lemmas conclude our discussion of the healthiness conditions of the new theory of binary multirelations. In summary, these relations can be characterised either by the predicates **BMH0-BMH2** or as fixed points of $\mathbf{bmh_{0,1,2}}$. In the following section we focus our attention on the subset of the theory that contains only the multirelations that are in addition **BMH3**-healthy.

3.3.3 $\mathbf{bmh_{0,1,3,2}}$

Relations that are **BMH0**, **BMH1**, **BMH2** and **BMH3**-healthy can be characterised as fixed points of the functional composition $\mathbf{bmh_{0,1,3,2}}$. The result of this composition is given by the following Lemma L.3.3.14.

Lemma L.3.3.14

$$\mathbf{bmh_0} \circ \mathbf{bmh_1} \circ \mathbf{bmh_3} \circ \mathbf{bmh_2}(B)$$

$$=$$

$$\left\{ s, ss \;\middle|\; \begin{array}{l} ((s, \emptyset) \in B \land (s, \{\bot\}) \in B) \\ \lor \\ \left(\begin{array}{l} (s, \{\bot\}) \notin B \land (s, \emptyset) \notin B \\ \land \\ (\exists\, ss_0 \bullet (s, ss_0) \in B \land ss_0 \subseteq ss \land \bot \notin ss_0 \land \bot \notin ss) \end{array} \right) \end{array} \right\}$$

The set construction considers a disjunction, where, either s is an aborting state, and hence it is related to the empty set and $\{\bot\}$, and otherwise, if it is not aborting, it satisfies the same property of upward-closure as required by $\mathbf{bmh_0}$. The particular order of this functional composition is justified by the following Theorem T.3.3.2.

Theorem T.3.3.2 $\mathbf{BMH0} \land \mathbf{BMH1} \land \mathbf{BMH2} \land \mathbf{BMH3} \Leftrightarrow \mathbf{bmh_{0,1,3,2}}(B) = B$

The proof of Theorem T.3.3.2 is split into two implications. First, we establish through Lemma L.3.3.15 that the conjunction of the predicative healthiness conditions **BMH0** to **BMH3** implies that B is a fixed point of $\mathbf{bmh_{0,1,3,2}}$.

Lemma L.3.3.15 $\mathbf{BMH0} \land \mathbf{BMH1} \land \mathbf{BMH2} \land \mathbf{BMH3} \Rightarrow \mathbf{bmh_{0,1,3,2}}(B) = B$

To prove the reserve implication, we first establish through Lemma L.3.3.16 that a fixed point of $\mathbf{bmh_{0,1,3,2}}$ is also a fixed point of $\mathbf{bmh_{0,1,2}}$, so that Lemmas L.3.3.10 to L.3.3.12 are directly applicable.

Lemma L.3.3.16 $\mathbf{bmh_{0,1,2}} \circ \mathbf{bmh_{0,1,3,2}}(B) = \mathbf{bmh_{0,1,3,2}}(B)$

Finally, Lemma L.3.3.17 establishes that every fixed point of $\mathbf{bmh_{0,1,3,2}}$ satisfies the predicative healthiness condition **BMH3**.

Lemma L.3.3.17 $(\mathbf{bmh_{0,1,3,2}}(B) = B) \Rightarrow \mathbf{BMH3}$

This concludes the proof that the subset of the theory that is in addition **BMH3**-healthy also has a counterpart characterisation via fixed points of $\mathbf{bmh_{0,1,3,2}}$. This function characterises the subset that corresponds to the original theory of binary multirelations. The relationship with the original theory of binary multirelations is explored in Section 3.6.

3.4 Refinement

The refinement order for the new binary multirelation model is defined exactly as in the original theory of binary multirelations [35].

Definition 75 (Refinement) $B_1 \sqsubseteq_{BM_\perp} B_0 \mathrel{\widehat=} B_1 \supseteq B_0$

It is reverse subset inclusion, such that a program characterised by a multirelation B_0 refines another characterised by a multirelation B_1 if, and only if, B_0 is a subset of B_1.

The extreme points of the theory as expected of a theory of designs, are the everywhere miraculous program and abort. Their definitions are presented below.

Definition 76 (Miracle) $\top_{BM_\perp} \mathrel{\widehat=} \emptyset$

As in the original theory, miracle is denoted by the absence of any relationship between any input state and any set of final states, that is, the program cannot possibly be executed.

Definition 77 (Abort) $\perp_{BM_\perp} \mathrel{\widehat=} \mathit{State} \times \mathbb{P}\, \mathit{State}_\perp$

On the other hand, abort is characterised by the universal relation, such that every initial state is related to every possible set of final states.

3.5 Operators

In this section the most important operators of the theory are introduced. Namely, we define the operators of assignment, angelic and demonic choice, and sequential composition. These enable the discussion of interesting properties observed in this model of extended binary multirelations.

As discussed in Chapter 1, the model that we propose here is isomorphic to the theory of angelic designs that we discuss in Chapter 4. In that chapter we establish that the operators discussed here are in correspondence with those in the theory of angelic designs, which we prove to be closed. Together with the respective isomorphism that we discuss in Section 4.3, these results are sufficient to establish closure of the operators with respect to the healthiness condition $\mathbf{BMH}_\perp$.

3.5.1 Assignment

The first operator of interest is assignment. As already illustrated, in this new model, there is the possibility to define two distinct assignment operators. The first one behaves exactly as in the original theory of binary multirelations $x :=_{BM} e$. This operator does not need to be redefined, since $BM \subseteq BM_\perp$. The new operator that we define below, however, behaves rather differently, in that it may or may not terminate.

Definition 78 $x :=_{BM_\perp} e \mathrel{\widehat=} \{s : \mathit{State}, ss : \mathbb{P}\, \mathit{State}_\perp \mid s \oplus (x \mapsto e) \in ss\}$

This assignment guarantees that for every initial state s, there is some set of final states available for angelic choice where x has the value of expression e. However, termination is not guaranteed. While the angel can choose the final value of x it cannot possibly guarantee termination in this case.

3.5.2 Angelic Choice

The definition of angelic choice is exactly the same as in the original theory of binary multirelations.

Definition 79 $B_0 \sqcup_{BM_\perp} B_1 \mathrel{\widehat{=}} B_0 \cap B_1$

For every set of final states available for demonic choice in B_0 and B_1, only those that can be chosen both in B_0 and B_1 are available.

An interesting property of angelic choice that is observed in this model is illustrated by the following Lemma L.3.5.1. It considers the angelic choice between two assignments of the same expression, yet only one is guaranteed to terminate.

Lemma L.3.5.1 $(x :=_{BM_\perp} e) \sqcup_{BM_\perp} (x :=_{BM} e) = (x :=_{BM} e)$

This result can be interpreted as follows: given an assignment that is guaranteed to terminate, adding a corresponding angelic choice that is potentially non-terminating does not in fact introduce any new choices.

In general, and as expected from the original model of binary multirelations, the angelic choice operator observes the following properties. As the refinement ordering in the new model is exactly the same as in the theory of binary multirelations, the angelic choice operator, being the least upper bound in both theories, has the same properties with respect to the extreme points of the lattice.

Lemma L.3.5.2 $\top_{BM_\perp} \sqcup_{BM_\perp} B = \top_{BM_\perp}$

The angelic choice between an everywhere miraculous program and any other program is still miraculous.

Lemma L.3.5.3 $\perp_{BM_\perp} \sqcup_{BM_\perp} B = B$

On the other hand, the angelic choice between abort and any other program B is the same as B. That is, the angel will avoid choosing an aborting program if possible.

3.5.3 Demonic Choice

The next operator of interest is demonic choice. It is also defined exactly like in the original theory of binary multirelations.

Definition 80 $B_0 \sqcap_{BM_\perp} B_1 \mathrel{\widehat{=}} B_0 \cup B_1$

For every initial state, a corresponding set of final states available for demonic choice in either, or both, of B_0 and B_1, is included in the result.

Similarly to the angelic choice operator, there is a general result regarding the demonic choice over the two assignment operators, terminating and not necessarily terminating. This is established by the following Lemma L.3.5.4.

Lemma L.3.5.4 $(x :=_{BM} e) \sqcap_{BM_\perp} (x :=_{BM_\perp} e) = (x :=_{BM_\perp} e)$

If there is an assignment for which termination is not guaranteed, then the demonic choice over this assignment and a corresponding one that is guaranteed to terminate is the same as the assignment that does not require termination. In other words, if it is possible for the demon to choose between two similar sets of final states, one that is possibly non-terminating and one that terminates, then the one for which termination is not guaranteed dominates the choice.

The following two laws show how the demonic choice operator behaves with respect to the extreme points of the lattice.

Lemma L.3.5.5 $\perp_{BM_\perp} \sqcap_{BM_\perp} B = \perp_{BM_\perp}$

Lemma L.3.5.6 $\top_{BM_\perp} \sqcap_{BM_\perp} B = B$

As expected, the demonic choice between abort and some other program is abort. In the case of a miracle, the demon will avoid choosing it if possible.

Since the angelic and demonic choice operators are defined as set intersection and union, respectively, they also distribute through each other. This is exactly the same property as in the original theory of binary multirelations.

3.5.4 Sequential Composition

The definition of sequential composition in this new model is not immediately obvious. We note, however, that one of the reasons for developing this theory is the fact that it allows a more intuitive account of the definition of sequential composition and, as such, an easier route to discover the definition in the theory of angelic designs. To illustrate the issue, we consider the following example from the theory of designs, where a non-**H3**-design is sequentially composed with $I\!I_{\mathcal{D}}$.

Example 18

$(x' = 1 \vdash \mathit{true}) \fatsemi I\!I_{\mathcal{D}}$ $\{\text{Definition of } I\!I_{\mathcal{D}}\}$

$= (x' = 1 \vdash \mathit{true}) \fatsemi (\mathit{true} \vdash x' = x)$ $\{\text{Sequential composition for designs}\}$

$$= (\neg\ (x' \neq 1\ ;\ \mathit{true}) \land \neg\ (\mathit{true}\ ;\ \mathit{false}) \vdash \mathit{true}\ ;\ x' = x) \qquad \{\text{Sequential composition}\}$$

$$= (\neg\ (\exists\, x_0 \bullet x_0 \neq 1 \land \mathit{true}) \land \neg\ (\exists\, x_0 \bullet \mathit{true} \land \mathit{false}) \vdash \exists\, x_0 \bullet \mathit{true} \land x' = x_0)$$
$$\{\text{Predicate calculus and one-point rule}\}$$

$$= (\neg\ \mathit{true} \land \neg\ \mathit{false} \vdash \mathit{true}) \qquad \{\text{Predicate calculus and property of designs}\}$$

$$= \mathit{true}$$

The result is *true*, the bottom of designs [39], whose behaviour is arbitrary. This arises because, since the first design can always establish a final value for x, namely 1, where termination is then not guaranteed, the *Skip* design $I\!I_{\mathcal{D}}$ that follows can never guarantee termination. This result can be generalised for a sequential composition involving any non-**H3**-design.

This provides the motivation for the definition of sequential composition in the new binary multirelational model.

Definition 81

$$B_0\ ;_{BM_\perp}\ B_1 \mathrel{\widehat=} \left\{ s_0, ss_0 \middle| \begin{array}{l} \exists\, ss \bullet (s_0, ss) \in B_0 \land \\ (\perp \in ss \lor ss \subseteq \{s_1 : \mathit{State} \mid (s_1, ss_0) \in B_1\}) \end{array} \right\}$$

For sets of final states where termination is guaranteed, that is, $\perp$ is not in the set of intermediate states ss, this definition matches that of the original theory. If $\perp$ is in ss, and hence termination is not guaranteed, then the result of the sequential composition is arbitrary as it can include any set of final states. If we assume that B_0 is **BMH0**-healthy, then the definition of sequential composition can be split into the set union of two sets as shown in Theorem T.3.5.1.

Theorem T.3.5.1 *Provided B_0 is* **BMH0***-healthy,*

$$B_0\ ;_{BM_\perp}\ B_1 = \left(\begin{array}{l} \{s_0, ss_0 \mid (s_0, \mathit{State}_\perp) \in B_0\} \\ \cup \\ \{s_0, ss_0 \mid (s_0, \{s_1 \mid (s_1, ss_0) \in B_1\}) \in B_0\} \end{array} \right)$$

The first set considers the case when B_0 leads to sets of final states where termination is not required and, therefore, to the whole of $\mathit{State}_\perp$, due to upward closure. The second set considers the case where termination is required and matches the result of Lemma L.2.3.4.

For a similar example to Example 18 expressed in the new theory, we consider the following example, where a non-terminating assignment is followed by the assignment that requires termination, but does not change the value of x.

Example 19

$$
\begin{array}{ll}
(x :=_{BM_\perp} e) \mathbin{;}_{BM_\perp} (x :=_{BM} x) & \{\text{Definition of } \mathbin{;}_{BM_\perp} \text{ (Theorem T.3.5.1)}\} \\
= \left(\begin{array}{l}
\{s_0 : State, ss_0 : \mathbb{P}\, State_\perp \mid (s_0, State_\perp) \in (x :=_{BM_\perp} e)\} \\
\cup \\
\left\{\begin{array}{l} s_0 : State, ss_0 : \mathbb{P}\, State_\perp \\ \mid (s_0, \{s_1 : State \mid (s_1, ss_0) \in (x :=_{BM} x)\}) \in (x :=_{BM_\perp} e) \end{array}\right\}
\end{array}\right) & \\
& \{\text{Definition of } :=_{BM} \text{ and } :=_{BM_\perp}\} \\
= \left(\begin{array}{l}
\left\{\begin{array}{l} s_0 : State, ss_0 : \mathbb{P}\, State_\perp \\ \mid (s_0, State_\perp) \in \{s : State, ss : \mathbb{P}\, State_\perp \mid s \oplus (x \mapsto e) \in ss\} \end{array}\right\} \\
\cup \\
\left\{\begin{array}{l} s_0 : State, ss_0 : \mathbb{P}\, State_\perp \\ \left|\begin{array}{l} (s_0, \{s_1 : State \mid (s_1, ss_0) \in (x :=_{BM} x)\}) \\ \in \\ \{s : State, ss : \mathbb{P}\, State \mid s \oplus (x \mapsto e) \in ss\} \end{array}\right. \end{array}\right\}
\end{array}\right) & \\
& \{\text{Property of sets}\} \\
= \left(\begin{array}{l}
\{s_0 : State, ss_0 : \mathbb{P}\, State_\perp \mid s_0 \oplus (x \mapsto e) \in State_\perp\} \\
\cup \\
\left\{\begin{array}{l} s_0 : State, ss_0 : \mathbb{P}\, State_\perp \\ \mid s_0 \oplus (x \mapsto e) \in \{s_1 : State \mid (s_1, ss_0) \in (x :=_{BM} x)\} \end{array}\right\}
\end{array}\right) & \\
& \{\text{Property of sets}\} \\
= \left(\begin{array}{l}
\{s_0 : State, ss_0 : \mathbb{P}\, State_\perp \mid true\} \\
\cup \\
\left\{\begin{array}{l} s_0 : State, ss_0 : \mathbb{P}\, State_\perp \\ \mid s_0 \oplus (x \mapsto e) \in \{s_1 : State \mid (s_1, ss_0) \in (x :=_{BM} x)\} \end{array}\right\}
\end{array}\right) & \\
& \{\text{Property of sets and definition of } \perp_{BM_\perp}\} \\
= \perp_{BM_\perp} &
\end{array}
$$

The result of this sequential composition is an aborting program. Like in the theory of designs, if it is possible for the first program not to terminate, then the sequential composition cannot provide any guarantees either. The properties observed by the sequential composition operator are explored in what follows.

Properties

The first property of interest considers the sequential composition of $\top_{BM_\perp}$ followed by some program B. The result is also a miraculous program as shown in the following Lemma L.3.5.7.

Lemma L.3.5.7 $\top_{BM_\perp} ;_{BM_\perp} B = \top_{BM_\perp}$

The following law expresses that the sequential composition of abort with another program is also abort.

Lemma L.3.5.8 $\perp_{BM_\perp} ;_{BM_\perp} B = \perp_{BM_\perp}$

In the following paragraphs we explore some examples with respect to the extreme points of the lattice.

The following example describes the general behaviour of some program B that is **BMH0**-healthy sequentially composed with a miraculous program.

Example 20

$$\begin{aligned}
& B ;_{BM_\perp} \top_{BM_\perp} && \{\text{Definition of } \top_{BM_\perp} \text{ and } ;_{BM_\perp} \text{ (Theorem T.3.5.1)}\}\\
&= \left(\begin{array}{l} \{s_0 : State, ss_0 : \mathbb{P}\, State_\perp \mid (s_0, State_\perp) \in B\} \\ \cup \\ \{s_0 : State, ss_0 : \mathbb{P}\, State_\perp \mid (s_0, \{s_1 : State \mid (s_1, ss_0) \in \emptyset\}) \in B\} \end{array}\right) && \{\text{Property of sets}\}\\
&= \left(\begin{array}{l} \{s_0 : State, ss_0 : \mathbb{P}\, State_\perp \mid (s_0, State_\perp) \in B\} \\ \cup \\ \{s_0 : State, ss_0 : \mathbb{P}\, State_\perp \mid (s_0, \emptyset) \in B\} \end{array}\right)
\end{aligned}$$

If B may not terminate for some set of initial states, and it is **BMH0**-healthy, then the result of the sequential composition is also abort, for those initial states. If B aborts for some particular initial state s_0, then that state is related to the empty set in B and the result of the sequential composition is also abort. Otherwise, the result is miraculous as the initial state is not in the domain of either relation in the union above.

The following example describes the behaviour of a program B sequentially composed with abort.

Example 21

$$B ;_{BM_\perp} \perp_{BM_\perp} \qquad \{\text{Definition of } \perp_{BM_\perp} \text{ and } ;_{BM_\perp} \text{ (Theorem T.3.5.1)}\}$$

$$= \left(\begin{array}{l} \{s_0 : State, ss_0 : \mathbb{P}\, State_\perp \mid (s_0, State_\perp) \in B\} \\ \cup \\ \left\{ \begin{array}{l} s_0 : State, ss_0 : \mathbb{P}\, State_\perp \\ \mid (s_0, \{s_1 : State \mid (s_1, ss_0) \in (State \times \mathbb{P}\, State_\perp)\}) \in B \end{array} \right\} \end{array} \right)$$

{Property of sets}

$$= \left(\begin{array}{l} \{s_0 : State, ss_0 : \mathbb{P}\, State_\perp \mid (s_0, State_\perp) \in B\} \\ \cup \\ \{s_0 : State, ss_0 : \mathbb{P}\, State_\perp \mid (s_0, \{s_1 : State \mid true\}) \in B\} \end{array} \right)$$ {Property of sets}

$$= \{s_0 : State, ss_0 : \mathbb{P}\, State_\perp \mid (s_0, State_\perp) \in B \lor (s_0, State) \in B\}$$

Because B is upward closed, if it definitely terminates then *State* is a superset of all sets of final states and is in B. If B may or may not terminate for some particular set of final states, then $State_\perp$ is also in B due to the upward closure guaranteed by **BMH0**. In either case, the sequential composition behaves as abort. If B is miraculous, then so is the sequential composition.

3.6 Relationship with Binary Multirelations

Having presented the most important operators of the theory, in this section we focus our attention on the relationship between the new model and the original theory of binary multirelations. The first step consists in the definition of a pair of linking functions, *bmb2bm*, which maps relations from the new model into the original theory of binary multirelations, and *bm2bm*, a mapping in the opposite direction.

As previously discussed in Chapter 1, the relationship is illustrated in Figures 1.1 and 1.3 where each theory is labelled according to its healthiness conditions. In this case, we have a bijection between the subset of $\mathbf{BMH}_\perp$ characterised by the relations that are **BMH3**-healthy and the original theory of binary multirelations characterised by **BMH**. In this section our discussion is focused on this isomorphism, while in Chapter 4 we discuss the isomorphism with the theory of angelic designs.

3.6.1 From $BM_\perp$ to BM (*bmb2bm*)

The first function of interest is *bmb2bm* that maps from binary multirelations in the new model, of type $BM_\perp$, to those in the original model of type BM.

Definition 82

$$bmb2bm : BM_\perp \to BM$$
$$bmb2bm(B) \mathrel{\widehat=} \{s : State, ss : \mathbb{P}\, State_\perp \mid (s, ss) \in B \land \perp \notin ss\}$$

Its definition considers every pair (s, ss) in B such that $\perp$ is not in ss. We consider the following example, where $bmb2bm$ is applied to the potentially non-terminating assignment of e to the program variable x.

Example 22 $bmb2bm(x :=_{BM_\perp} e) = (x :=_{BM} e)$

The result corresponds to assignment in the original theory.

In order to establish that $bmb2bm$ yields a multirelation that is **BMH**-healthy we use an alternative way to characterise the set of healthy binary multirelations as fixed points of the function $\mathbf{bmh_{up}}$.

Definition 83 $\mathbf{bmh_{up}}(B) \mathrel{\widehat=} \{s, ss \mid \exists\, ss_0 : \mathbb{P}\, State \bullet (s, ss_0) \in B \land ss_0 \subseteq ss\}$

This definition is justified by Lemma L.3.6.1.

Lemma L.3.6.1 $\mathbf{BMH} \Leftrightarrow \mathbf{bmh_{up}}(B) = B$

Finally, Theorem T.3.6.1 establishes that the application of $bmb2bm$ to a multirelation that is **BMH0**-**BMH3**-healthy yields a **BMH**-healthy relation.

Theorem T.3.6.1

$$\mathbf{bmh_{up}} \circ bmb2bm(\mathbf{bmh_{0,1,3,2}}(B)) = bmb2bm(\mathbf{bmh_{0,1,3,2}}(B))$$

In summary, $bmb2bm$ yields relations that are in the original theory.

3.6.2 From BM to $BM\perp$ ($bm2bmb$)

The mapping in the opposite direction, from BM to $BM_\perp$ is achieved by the function $bmb2bm$, whose definition is presented below.

Definition 84

$$bm2bmb : BM \to BM_\perp$$
$$bm2bmb(B) \mathrel{\widehat=} \{s : State, ss : \mathbb{P}\, State_\perp \mid ((s, ss) \in B \land \perp \notin ss) \lor (s, \emptyset) \in B\}$$

It considers every pair (s, ss) in a relation B, where $\perp$ is not in the set of final states ss, or if B is aborting for a particular state s, that is, s is related to the empty set, then it is related to every possible final state, including $\perp$, so that we have nontermination for s.

Similarly to the treatment of $bm2bmb$, Theorem T.3.6.2 establishes that the application of $bmb2bm$ to an upward-closed relation, that is **BMH**-healthy, yields a relation that is **BMH0-BMH3**-healthy.

Theorem T.3.6.2

$$\mathbf{bmh_{0,1,3,2}} \circ bm2bmb(\mathbf{bmh_{up}}(B)) = bm2bmb(\mathbf{bmh_{up}}(B))$$

This result completes the proof for healthiness of both linking functions. In the following section we discuss the isomorphism.

3.6.3 Isomorphism ($bm2bmb$ and $bmb2bm$)

Based on the results of the previous Sections 3.6.1 and 3.6.2 we can establish that $bm2bmb$ and $bmb2bm$ form a bijection for healthy relations as ascertained by the following Theorems T.3.6.3 and T.3.6.4.

Theorem T.3.6.3 *Provided B is* $\mathbf{BMH_{0,1,2,3}}$*-healthy,* $bm2bmb \circ bmb2bm(B) = B$*,*

Theorem T.3.6.4 *Provided B is* **BMH***-healthy,* $bmb2bm \circ bm2bmb(B) = B$*,*

These results show that the subset of the theory that is **BMH0-BMH3**-healthy is isomorphic to the original theory of binary multirelations [35]. This confirms that while our model is more expressive, it is still possible to express every program that could be specified using the original model.

3.7 Final Considerations

In this chapter we have introduced a new model of binary multirelations that allows the specification of sets of final states for which termination is not required. This model extends the theory of Rewiztky [35] by considering a special state $\perp$ that denotes the possibility for non-termination. The healthiness conditions have been introduced as predicates and subsequently characterised as fixed points of idempotent functions. This dual characterisation is useful for reasoning about the link between this model and the theory of [35].

The operators of the theory have been introduced and their properties studied. Notable differences with respect to the original theory include the potentially non-terminating assignment and sequential composition. The definition of the latter is perhaps the most unexpected, as the intuition comes from the UTP theory of designs. The full justification for some of the operators and the refinement order is revisited again in Chapter 4 where we introduce the isomorphic model of angelic designs.

Finally, we have studied the relationship between this new model of binary multirelations and the theory of [35]. We have found that the subset of multirelations that are, in addition, **BMH3**-healthy, is isomorphic to the original theory. While this model is more expressive, we can still reason about the existing model of binary multirelations.

Chapter 4

Angelic Designs

In this chapter we introduce a new UTP theory of designs with both angelic and demonic nondeterminism. As already indicated the starting points for this predicative model are the theory of Cavalcanti et al. [38] and the extended model of binary multirelations presented earlier in Chapter 3. For this reason, Section 4.1 begins by discussing the choice of alphabet and the relationship with the alphabet of [38]. In Section 4.2 the healthiness conditions of the theory are presented. Section 4.3 discusses the isomorphism with the model of extended binary multirelations. In Section 4.4 we explore the notion of refinement and prove that it corresponds exactly to that in the model of Chapter 3. In Section 4.5 the main operators of the theory are presented, including angelic and demonic choice. In Section 4.6 we explore the relationship with the original theory of designs. In Section 4.7 we show that the subset of **H3**-healthy designs is isomorphic to the theory of [38]. Finally, Section 4.8 concludes the chapter with a summary of the main results.

4.1 Alphabet

Our aim is to build a theory of designs. Therefore, the alphabet of our theory includes the observational variables ok and ok', like every theory of designs and two additional variables s and ac', as shown in the following definition, where the notation for a type of *State* is enriched to carry a parameterised set of variables $S\alpha$ that specifies the names of all the record components considered. The approach followed in our discussion is that a record can be represented as a set of ordered pairs where the first component is the variable name, from a set of all possible variables, and the second component corresponds to the associated value or expression.

Definition 85 (Alphabet)

$$
\begin{aligned}
&s : \mathit{State}(S\alpha) \\
&ac' : \mathbb{P}\,\mathit{State}(S\alpha) \\
&ok, ok' : \{\mathit{true}, \mathit{false}\} \\
&\mathit{State}(S\alpha) = \{x, e \mid x \in S\alpha\}
\end{aligned}
$$

The variable s encapsulates the initial values of program variables as record components of s, just like in the extended model of binary multirelations discussed in Chapter 3. The set of final states ac' is similar to that of [38] with the notable difference that we do not dash the variable names in the record components, instead we only consider these as undashed. This deliberate choice bears no consequences, other than simplifying reasoning and proofs. The set of program variables $S\alpha$ recorded in both s and final states of ac' is the same.

The set of angelic choices ac' of this new model and that of [38] can be related by dashing or undashing the variables of the components of all states in either set. This relationship is formalized by the following pair of functions.

Definition 86

$$
\begin{aligned}
\mathit{undashset}(ss) &\mathrel{\widehat=} \{z : \mathit{State}(S\alpha) \mid z \in ss \bullet \mathit{undash}(z)\} \\
\mathit{dashset}(ss) &\mathrel{\widehat=} \{z : \mathit{State}(S\alpha) \mid z \in ss \bullet \mathit{dash}(z)\}
\end{aligned}
$$

The function *undashset* maps a set ss of states whose record components are dashed variables into a set where every state has its components undashed. This is achieved by considering every state z in the set ss and applying *undash*, a function which undashes the names of every record component of a state. Similarly, *dashset* maps in the opposite direction by dashing every state in ss. A state z whose components range over the set of variables $S\alpha$ can be dashed and undashed via the functions, *dash* and *undash*.

The function $\mathit{dash}(z)$ considers every record component $z.x$ of z, and dashes the name of x into x'. Similarly, the function *undash* performs the inverse renaming, by undashing every x' to x. The functions *dash* and *undash* are bijective. They are the exact inverse of each other. Useful properties include, for instance, that $\mathit{undash}(z).x = z.x'$ and $\mathit{dash}(z).x' = z.x$. These and other properties of *dash* and *undash* are included for completeness in Appendix D.2.

These functions are important in the development of links between the theories, in particular with the theory of [38], which we explore in Section 4.7. In the following Section 4.2 we introduce the healthiness conditions.

4.2 Healthiness Conditions

Since the theory we propose is a theory of designs, at the very least predicates need to satisfy **H1** and **H2**. More important for our discussion is the fact that none of the proofs in [39] regarding **H1** and **H2** require homogeneity, so it is possible to consider a non-homogeneous theory of designs.

In addition, since we have a theory with ok and ok', the record of termination embedded in the use of ac' must be related to that in ok and ok'. This is the concern of the first healthiness condition **A0**, which we discuss in Section 4.2.1. Similarly to the theory of [38], there is a requirement for ac' to be upward-closed. This is the concern of the second healthiness condition **A1**, which we discuss in Section 4.2.2. Finally, the composition of both healthiness conditions, named as **A**, is explored in Section 4.2.3.

4.2.1 A0

The notion of termination considered in this theory is related to that of [38]. In that model, termination is always guaranteed as long as ac' is not empty. In the theory of designs termination is signalled by ok'. In order to reconcile these two notions we introduce the following healthiness condition **A0**.

Definition 87 $\mathbf{A0}(P) \mathrel{\widehat=} P \land ((ok \land \neg P^f) \Rightarrow (ok' \Rightarrow ac' \neq \emptyset))$

It states that when a design is started and its precondition $\neg P^f$ is satisfied, if it terminates, with ok' being *true*, then it must be the case that ac' is not empty. In other words, there must be at least one state in ac' available for angelic choice. If the precondition $\neg P^f$ is not satisfied, then the design aborts and there are no guarantees on the outcome, and so ac' may or may not be empty.

The function **A0** is idempotent and monotonic as established by the following Theorems T.4.2.1 and T.4.2.2. Proof of these and other results to follow can be found in Appendix C of the extended version of this thesis [74].

Theorem T.4.2.1 $\mathbf{A0} \circ \mathbf{A0}(P) = \mathbf{A0}(P)$

Theorem T.4.2.2 $(P \sqsubseteq Q) \Rightarrow (\mathbf{A0}(P) \sqsubseteq \mathbf{A0}(Q))$

More importantly, the function **A0** is closed with respect to designs.

Theorem T.4.2.3 *If P is a design so is* $\mathbf{A0}(P)$.

$$\mathbf{A0}(P) = (\neg\, P^f \vdash P^t \land ac' \neq \emptyset)$$

Therefore a design in this theory can be stated in the usual manner, with a pre and a postcondition which in this case requires ac' not to be empty. In other words, once the precondition of an angelic design is satisfied, it terminates successfully with at least one final state available for angelic choice.

Finally, **A0** is closed with respect to conjunction and disjunction as stated in the following Theorems T.4.2.4 and T.4.2.5.

Theorem T.4.2.4 *Provided P and Q are* **A0**-*healthy,*

$$\mathbf{A0}(P \land Q) = P \land Q$$

Theorem T.4.2.5 *Provided P and Q are* **A0**-*healthy designs,*

$$\mathbf{A0}(P \lor Q) = P \lor Q$$

The function **A0** distributes through conjunction, and provided that the predicate is a design, that is **H1** and **H2**-healthy, it also distributes through disjunction. This extra proviso is not a problem since this is a theory of designs. These properties conclude our discussion regarding **A0**.

4.2.2 A1

In addition to requiring a consistent treatment of termination, our theory of designs also requires that both the pre and postcondition observe the upward closure of the set of final states ac'. In order to enforce this property in the new theory we extend the original healthiness condition **PBMH** of [38] to accommodate the additional variables ok and ok' as follows.

Definition 88 $\mathbf{PBMH}(P) \mathrel{\widehat=} P \mathbin{;} ac \subseteq ac' \land ok' = ok$

In addition to requiring that the value of ac' must be upward-closed, the value of ok' is left unchanged. This is the definition of **PBMH** adopted throughout our work. Its expanded version given by Lemma L.4.2.1 is more often used directly in proofs.

Lemma L.4.2.1 $\mathbf{PBMH}(P) = \exists\, ac_0 \bullet P[ac_0/ac'] \land ac_0 \subseteq ac'$

When considering a design, with precondition P and postcondition Q, the application of **PBMH** yields a design where it is itself applied to the postcondition and the negation of the precondition, as shown in the following Lemma L.4.2.2.

Lemma L.4.2.2 $\mathbf{PBMH}(P \vdash Q) = (\neg\, \mathbf{PBMH}(\neg\, P) \vdash \mathbf{PBMH}(Q))$

The requirement on the postcondition is exactly like in the original theory of [38]. While the requirement on the negation of the precondition follows directly from the definition of designs, where for non-**H3** designs it is actually the negation of the precondition that determines what is enforced in the case of non-termination. In Section 2.4 we show in Example 5 such a scenario.

The application of **PBMH** to a design is precisely the motivation behind the definition of the following healthiness condition **A1**.

Definition 89 $\mathbf{A1}(P \vdash Q) \mathrel{\widehat=} (\neg\, \mathbf{PBMH}(\neg\, P) \vdash \mathbf{PBMH}(Q))$

Therefore **A1** and **PBMH** are synonyms and can be used interchangeably.

The function **A1** is idempotent and monotonic as established by the following Theorems T.4.2.6 and T.4.2.7.

Theorem T.4.2.6 $\mathbf{A1} \circ \mathbf{A1}(P_0 \vdash P_1) = \mathbf{A1}(P_0 \vdash P_1)$

Theorem T.4.2.7 $(P \sqsubseteq Q) \Rightarrow \mathbf{A1}(P) \sqsubseteq \mathbf{A1}(Q)$

Furthermore it is closed with respect to both conjunction and disjunction, and distributes through disjunction. In the following section we discuss the functional composition of **A0** and **A1**.

4.2.3 A

The theory of designs we propose is characterised by the functional composition of **A0**, **A1**, and **H1** and **H2** of the original theory of designs. The order in which these functions are composed is important since they to not always necessarily commute. In order to explain the reason behind this we consider the following counter-example.

Counter-example 1

$$\mathbf{A0} \circ \mathbf{A1}(\mathit{true} \vdash ac' = \emptyset) \qquad \{\text{Definition of } \mathbf{A1}\}$$

$$= \mathbf{A0}\left(\begin{array}{l} \neg\, (\mathit{false}\ ;\ ac \subseteq ac') \\ \vdash \\ ac' = \emptyset\ ;\ ac \subseteq ac' \end{array}\right) \qquad \{\text{Definition of sequential composition}\}$$

$$
\begin{aligned}
&= \mathbf{A0}\left(\begin{array}{l} \neg\,(\mathit{false} \land \exists\, ac_0 \bullet ac_0 \subseteq ac') \\ \vdash \\ \exists\, ac_0 \bullet ac_0 = \emptyset \land ac_0 \subseteq ac' \end{array}\right) && \{\text{One-point rule and predicate calculus}\} \\
&= \mathbf{A0}(\mathit{true} \vdash \mathit{true}) && \{\text{Definition of } \mathbf{A0} \text{ (Theorem T.4.2.3)}\} \\
&= \mathbf{A0}(\mathit{true} \vdash ac' \neq \emptyset)
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{A1} \circ \mathbf{A0}(\mathit{true} \vdash ac' = \emptyset) && \{\text{Definition of } \mathbf{A0} \text{ (Theorem T.4.2.3)}\} \\
&= \mathbf{A1}(\mathit{true} \vdash ac' = \emptyset \land ac' \neq \emptyset) && \{\text{Predicate calculus}\} \\
&= \mathbf{A1}(\mathit{true} \vdash \mathit{false}) && \{\text{Definition of } \mathbf{A1}\} \\
&= (\neg\,(\mathit{false}\ ;\ ac \subseteq ac') \vdash \mathit{false}\ ;\ ac \subseteq ac') && \{\text{Definition of sequential composition}\} \\
&= (\mathit{true} \vdash \mathit{false})
\end{aligned}
$$

In this example we apply the healthiness conditions in different orders to an unhealthy design ($\mathit{true} \vdash ac' = \emptyset$) whose postcondition requires non-termination: $ac' = \emptyset$. In the first case **A1** changes the postcondition into *true*, followed by the application of **A0**. While in the second case, **A0** is applied in the first place, making the postcondition *false*, a predicate that satisfies **PBMH**. The resulting predicate conforms to the definition of $\top_{\mathcal{D}}$. Thus the functions do not always commute.

If instead we consider healthy predicates, then we can ensure that **A0** and **A1** commute. The following Theorem T.4.2.8 establishes this result for predicates that are **A1**-healthy. In fact the only requirement is for the postcondition, P^t to satisfy **PBMH**.

Theorem T.4.2.8 *Provided P^t satisfies* **PBMH**, $\mathbf{A0} \circ \mathbf{A1}(P) = \mathbf{A1} \circ \mathbf{A0}(P)$

This indicates that it is appropriate to introduce the definition of **A** as the functional composition of **A1** followed by **A0**, since **A0** preserves **A1**-healthiness.

Definition 90 $\mathbf{A}(P) \mathrel{\widehat{=}} \mathbf{A0} \circ \mathbf{A1}(P)$

Theorem T.4.2.8 establishes that once the postcondition of P satisfies **PBMH** then the functions commute. Therefore by applying first **A1** first we guarantee that this is always the case.

Since the function **A** is defined by the functional composition of **A1** and **A0**, and these functions are monotonic, so is **A**. It is also idempotent as established by the following Theorem T.4.2.9.

Theorem T.4.2.9 $\mathbf{A} \circ \mathbf{A}(P) = \mathbf{A}(P)$

More importantly, it commutes with **H1** and **H2** of the theory of designs as established by the following Theorem T.4.2.10.

Theorem T.4.2.10 $\mathbf{H1} \circ \mathbf{H2} \circ \mathbf{A}(P) = \mathbf{A} \circ \mathbf{H1} \circ \mathbf{H2}(P)$

The healthiness condition of our theory is $\mathbf{H1} \circ \mathbf{H2} \circ \mathbf{A}$. Since these commute, and they are all idempotents so is their functional composition [39]. Furthermore, monotonicity also follows from the monotonicity of each function.

This concludes the main discussion on the healthiness conditions of the theory of angelic designs. Before exploring the relationship between this theory and the model of extended binary multirelations in Section 4.3, we first discuss how to define the subset of non-angelic designs of this theory in the following Section 4.2.4.

4.2.4 A2

In general, in our theory, a relation that does not exhibit angelic nondeterminism always provides at most one angelic choice. In other words, for every initial state, there must be at most one final state available in the distributed intersection over all possible values of ac'. That is, without directly considering the upward-closure of ac', there must be at most one state in ac'. This leads to the following healthiness condition **A2**.

Definition 91 $\mathbf{A2}(P) \mathrel{\widehat=} \mathbf{PBMH}(P \mathbin{;_{\mathcal{A}}} \{s\} = ac')$

This definition is given in terms of the operator $;_{\mathcal{A}}$, which we previously discussed in Section 2.4.4 and whose formal definition in the context of the theory of angelic designs is discussed in Section 4.5. The intuition behind this definition is that **A2** requires the set of final states in P to be either empty or a singleton, otherwise it becomes *false*. Since this purposedly breaks the upward-closure, **PBMH** must be applied as a result. If we consider the definition of **PBMH** and $;_{\mathcal{A}}$, the definition of **A2** can be expanded as established by the following Theorem T.4.2.11.

Theorem T.4.2.11 $\mathbf{A2}(P) = P[\emptyset/ac'] \lor (\exists\, y \bullet P[\{y\}/ac'] \land y \in ac')$

It confirms our intuition that ac' must be either empty or a singleton.

As expected of a healthiness condition, **A2** is idempotent and monotonic as confirmed by Theorems T.4.2.12 and T.4.2.13.

Theorem T.4.2.12 $\mathbf{A2} \circ \mathbf{A2}(P) = \mathbf{A2}(P)$

Theorem T.4.2.13 $P \sqsubseteq Q \Rightarrow \mathbf{A2}(P) \sqsubseteq \mathbf{A2}(Q)$

The function **A2** distributes through disjunction as established by Theorem T.4.2.14.

Theorem T.4.2.14 $\mathbf{A2}(P \lor Q) = \mathbf{A2}(P) \lor \mathbf{A2}(Q)$

Consequently it is also closed under disjunction. However, and as expected, **A2** is not necessarily closed under conjunction. As we discuss later in Section 4.5.4 angelic choice is defined through conjunction, so it is no surprise that the conjunction of two **A2**-healthy predicates can introduce angelic nondeterminism. Finally, when applied to a design, we obtain the following result of Lemma L.4.2.3.

Lemma L.4.2.3 $\mathbf{A2}(P \vdash Q) = (\neg\, \mathbf{A2}(\neg\, P) \vdash \mathbf{A2}(Q))$

That is, **A2** can be directly applied to both the negation of the precondition and the postcondition of a design.

This concludes the discussion of the healthiness conditions of the theory, and its subset of non-angelic designs. As highlighted in Figure 1.1, the function **A2** plays a fundamental role in identifying the subset of theories with no angelic nondeterminism, particularly when links are established with other theories.

4.3 Relationship with Extended Binary Multirelations

As previously discussed, the model of extended binary multirelations developed in Chapter 3 is a complementary model to that of angelic designs. In this section we show how these two models can be related and prove that they are isomorphic.

In order to do so, we define a pair of linking functions, $d2bmb$ that maps from angelic designs to binary multirelations, and $bmb2d$ mapping in the opposite direction. The latter is defined in Section 4.3.2 while the former is defined in Section 4.3.1. Finally, in Section 4.3.3 the isomorphism is established by proving that these functions form a bijection.

4.3.1 From Designs to Binary Multirelations ($d2bmb$)

The first function of interest is $d2bmb$. It maps from **A**-healthy designs into relations of type $BM_\perp$ and is defined as follows, where, as before, s is of type $State$ and ss of type $State_\perp$.

Definition 92 (d2bmb)

$$d2bmb : \mathbf{A} \to BM_{\perp}$$

$$d2bmb(P) \mathrel{\widehat=} \left\{ s, ss \;\middle|\; \begin{array}{l} (\neg\, P^f \Rightarrow P^t)[true/ok][ss/ac'] \land \perp \notin ss) \\ \lor \\ (P^f[true/ok][(ss \setminus \{\perp\})/ac'] \land \perp \in ss) \end{array} \right\}$$

For a given design P, whose precondition is $\neg\, P^f$, and postcondition is P^t, the set construction of $d2bmb(P)$ is split into two disjuncts.

The first disjunct considers the case where P is guaranteed to terminate, with ok and ok' both substituted with $true$ in the design P to obtain the implication $\neg\, P^f \Rightarrow P^t$. The resulting set of final states ss, for which termination is required ($\perp \notin ss$) is obtained by substituting ss for ac' in P.

In the second disjunct we consider the case where ok is also $true$, but ok' is false. This corresponds to the situation where P does not terminate. In this case, the set of final states is obtained by substituting $ss \setminus \{\perp\}$ for ac' and requiring $\perp$ to be in the set of final states ss.

As a consequence of P satisfying **H2**, we ensure that if there is some set of final states characterised by the second disjunct, and therefore, containing $\perp$, then there is also an equivalent set of final states without $\perp$ that is characterised by the first disjunct.

In the following Theorem T.4.3.1 we establish that the application of $d2bmb$ to **A**-healthy designs yields relations that are **BMH0**-**BMH2**-healthy.

Theorem T.4.3.1 *Provided P is a design,*

$$\mathbf{bmh_{0,1,2}} \circ d2bmb(\mathbf{A}(P)) = d2bmb(\mathbf{A}(P))$$

That is, the application of $d2bmb$ to an **A**-healthy design is a fixed point of $\mathbf{bmh_{0,1,2}}$.

We consider the following Example 23 where $d2bmb$ is applied to the program that either assigns the value 1 to the sole program variable x and terminates, or assigns the value 2 to x, in which the case termination is not required.

Example 23

$d2bmb((x \mapsto 2) \notin ac' \vdash (x \mapsto 1) \in ac')$ {Definition of $d2bmb$ (Lemma L.C.2.8)}

$$= \left\{ s, ss \;\middle|\; \begin{array}{l} ((x \mapsto 2) \notin ac' \Rightarrow (x \mapsto 1) \in ac')[ss/ac'] \land \bot \notin ss) \\ \lor \\ (((x \mapsto 2) \in ac')[ss \setminus \{\bot\}/ac'] \land \bot \in ss) \end{array} \right\}$$

{Predicate calculus and substitution}

$$= \left\{ s, ss \;\middle|\; \begin{array}{l} ((x \mapsto 2) \in ss \land \bot \notin ss) \\ \lor \\ ((x \mapsto 1) \in ss \land \bot \notin ss) \\ \lor \\ ((x \mapsto 2) \in (ss \setminus \{\bot\}) \land \bot \in ss) \end{array} \right\} \qquad \{\text{Property of sets}\}$$

$$= \left\{ s, ss \;\middle|\; \begin{array}{l} ((x \mapsto 2) \in ss \land \bot \notin ss) \\ \lor \\ ((x \mapsto 1) \in ss \land \bot \notin ss) \\ \lor \\ ((x \mapsto 2) \in ss \land (x \mapsto 2) \notin \{\bot\} \land \bot \in ss) \end{array} \right\} \qquad \{\text{Property of sets}\}$$

$$= \left\{ s, ss \;\middle|\; \begin{array}{l} ((x \mapsto 2) \in ss \land \bot \notin ss) \\ \lor \\ ((x \mapsto 1) \in ss \land \bot \notin ss) \\ \lor \\ ((x \mapsto 2) \in ss \land \bot \in ss) \end{array} \right\} \qquad \{\text{Predicate calculus}\}$$

$$= \{s, ss \mid (x \mapsto 2) \in ss \lor ((x \mapsto 1) \in ss \land \bot \notin ss)\}$$

{Definition of $\sqcap_{BM_\bot}$ and $:=_{BM_\bot}$ and $:=_{BM}$}

$$= (x :=_{BM_\bot} 2) \sqcap_{BM_\bot} (x :=_{BM} 1)$$

As expected, the function $d2bmb$ yields a program with the same behaviour specified using the binary multirelational model. It is the demonic choice over two assignments, one requires termination while the other does not.

4.3.2 From Binary Multirelations to Designs ($bmb2d$)

The second linking function of interest is $bmb2d$, which maps from relations of type $BM_\bot$ to **A**-healthy predicates. Its definition is presented below.

Definition 93

$$bmb2d : BM_\bot \to \mathbf{A}$$
$$bmb2d(B) \mathrel{\widehat=} ((s, ac' \cup \{\bot\}) \notin B \vdash (s, ac') \in B)$$

It is defined as a design, such that for a particular initial state s, the precondition requires $(s, ac' \cup \{\bot\})$ not to be in B, while the postcondition establishes that (s, ac') is in B. This definition can be expanded into a more intuitive representation according to the following Lemma L.4.3.1.

Lemma L.4.3.1 $bmb2d(B) = ok \Rightarrow \left(\begin{array}{l} ((s, ac') \in B \land \bot \notin ac' \land ok') \\ \lor \\ (s, ac' \cup \{\bot\}) \in B \end{array}\right)$

The behaviour of $bmb2d$ is split into two disjuncts. The first one considers the case where B requires termination, and hence $\bot$ is not part of the set of final states of the pair in B. While the second disjunct considers sets of final states that do not require termination, in which case ok' can be either *true* or *false*.

Theorem T.4.3.2 establishes that $bmb2d(B)$ yields **A**-healthy designs provided that B is **BMH0**-**BMH2**-healthy.

Theorem T.4.3.2 *Provided B satisfies* $\mathbf{bmh_{0,1,2}}$, $\mathbf{A} \circ bmb2d(B) = bmb2d(B)$.

This result confirms that $bmb2d$ is closed with respect to **A** when applied to relations that are **BMH0**-**BMH2**-healthy. This concludes our discussion of $bmb2d$. In the following Section 4.3.3 we focus our attention on the isomorphism.

4.3.3 Isomorphism: $d2bmb$ and $bmb2d$

In this section we show that $d2bmb$ and $bmb2d$ form a bijection. The following Theorem T.4.3.3 establishes that $d2bmb$ is the inverse function of $bmb2d$ for relations that are **BMH0**-**BMH2**-healthy.

Theorem T.4.3.3 *Provided B is* **BMH0**-**BMH2***-healthy,*

$$d2bmb \circ bmb2d(B) = B$$

Theorem T.4.3.4, on the other hand, establishes that $bmb2d$ is the inverse function of $d2bmb$ for designs that are **A**-healthy.

Theorem T.4.3.4 *Provided P is an* **A***-healthy design,*

$$bmb2d \circ d2bmb(P) = P$$

Together these results establish that the models are isomorphic. This result is of fundamental importance since it allows the same programs to be characterised using two different approaches. The binary multirelational model provides a set-theoretic approach, while the predicative theory proposed can be easily linked with other UTP theories of interest, namely the theory of reactive processes.

Furthermore, this dual approach enables us to justify the definition of certain aspects of our theory. This includes the healthiness conditions and the definition of certain operators such as sequential composition. The most intuitive and appropriate model can be used in each case. The results obtained in either model can then be related using the linking functions.

4.4 Refinement

The healthiness condition **A** can be viewed as a function from the theory of designs into our theory. The theory of designs is a complete lattice [39]. Since **A** is monotonic and idempotent, its range is also a complete lattice [39]. Therefore we can assert that the theory we propose is also a complete lattice under the universal reverse implication order.

In the following Section 4.4.1 we revisit the least and greatest elements of the of designs lattice and explore their properties within our theory. Next in Section 4.4.2 we show that the refinement order of our theory corresponds exactly to subset inclusion in the extended theory of binary multirelations of Chapter 3.

4.4.1 Extreme Points

Since we have a theory of designs, the extreme points of the lattice are exactly the same as those of any theory of designs. The bottom is defined by *true* ($\bot_{\mathcal{D}}$), whose behaviour is unpredictable and may include non-termination. While the top is the everywhere miraculous program given by $\neg\, ok$ ($\top_{\mathcal{D}}$). (In the theory of angelic nondeterminism of [38] the top is defined by *false* and the bottom by *true*.)

The bottom of the lattice *true* is an angelic design as established by the following Theorem T.4.4.1.

Theorem T.4.4.1 $\mathbf{A}(\bot_{\mathcal{D}}) = \bot_{\mathcal{D}}$

The consequence of *true* being the bottom of the lattice is that ac' may be empty. This is as expected, since a program for which there is no choice available to the angel corresponds to the possibility of non-termination.

The definition for the top of the lattice is a direct consequence of having the additional variables ok and ok'. It is also an angelic design as established by the following Theorem T.4.4.2.

Theorem T.4.4.2 $\mathbf{A}(\top_{\mathcal{D}}) = \top_{\mathcal{D}}$

Thus, such a program may never be started and its characterisation as a pre and postcondition pair is just like in the original theory of designs.

This concludes our introduction to the extreme points of the theory. In the following Section 4.4.2 we establish the relationship between the refinement order of this theory and that of the binary multirelational model.

4.4.2 Relationship with Extended Binary Multirelations

The model in Chapter 3 is meant to be as similar as possible to the original model of binary multirelations. In Section 3.4 the refinement order $\sqsubseteq_{BM_\perp}$ is defined as subset inclusion, like in the original theory. The following Theorem T.4.4.3 establishes that in fact the refinement order $\sqsubseteq_{BM_\perp}$ corresponds to the refinement order of designs $\sqsubseteq_{\mathcal{D}}$.

Theorem T.4.4.3 *Provided B_0 and B_1 are* **BMH0**-**BMH2**-*healthy,*

$$bmb2d(B_0) \sqsubseteq_{\mathcal{D}} bmb2d(B_1) \Leftrightarrow B_0 \sqsubseteq_{BM_\perp} B_1$$

It is reassuring to find that the refinement order in our theory of angelic designs corresponds to subset ordering in the binary multirelational model. This is particularly important as it confirms the intuitive definition of the theory of extended binary multirelations.

4.5 Operators

In this section we define the main operators of the theory of angelic designs. This includes the definition of assignment in the following Section 4.5.1, sequential composition in Section 4.5.2, demonic choice in Section 4.5.4, and finally angelic choice in Section 4.5.3. For these operators we show how they relate to their counterpart in the model of extended binary multirelations. In addition we also prove that they are all closed under **A**.

4.5.1 Assignment

The first operator we consider is assignment. The definition, presented below, is similar to that of [38].

Definition 94 (Assignment) $(x :=_{\mathcal{D}ac} e) \mathrel{\widehat=} (true \vdash s \oplus (x \mapsto e) \in ac')$

It is defined by a design whose precondition is *true*, and whose postcondition establishes that every set of final states ac' has a state where the component x is assigned the value of the expression e. Every such state is the result of overriding the value of x in the initial state s, while leaving every other program variable unchanged.

4.5.2 Sequential Composition

A challenging aspect of the theory of angelic designs is that it uses non-homogeneous relations. Consequently sequential composition cannot be simply defined as relational composition like in other UTP theories. The definition we propose here is layered upon the sequential composition operator $;_{\mathcal{A}}$ originally introduced in [38].

The definition of sequential composition for angelic designs is given by considering the auxiliary variables *ok* and *ok′* separately, as follows.

Definition 95 ($;_{\mathcal{D}ac}$-sequence) $P \,;_{\mathcal{D}ac}\, Q \mathrel{\widehat=} \exists\, ok_0 \bullet P[ok_0/ok'] \,;_{\mathcal{A}}\, Q[ok_0/ok]$

This definition resembles relational composition with the notable difference that instead of conjunction we use the operator $;_{\mathcal{A}}$ that handles the non-homogeneous alphabet of the relations. In Section 2.4.4 we previously discussed its definition as found in [38]. Since in our theory we have a different alphabet, we redefine the operator $;_{\mathcal{A}}$ in terms of the input state s as follows.

Definition 96 ($;_{\mathcal{A}}$-sequence) $P \,;_{\mathcal{A}}\, Q \mathrel{\widehat=} P[\{s : State \mid Q\}/ac']$

This is the definition adopted throughout this thesis. Just like before, this sequential composition can be understood as follows: a final state of $P \,;_{\mathcal{A}}\, Q$ is a final state of Q that can be reached from a set of input states s of Q that is available to P as a set ac' of angelic choices.

In Appendix F we explore and prove properties observed by the $;_{\mathcal{A}}$ operator. Based on those results, and the fact that *ok* and *ok′* are not free in neither the pre nor postcondition, it is possible to characterise the sequential composition of two angelic designs as follows.

Theorem T.4.5.1 *Provided ok and ok′ are not free in P, Q, R and S, and that*

$\neg P$ *and* Q *are* **PBMH**-*healthy,*

$$(P \vdash Q) \mathbin{;_{\mathcal{D}ac}} (R \vdash S) = (\neg (\neg P \mathbin{;_{\mathcal{A}}} \mathit{true}) \land \neg (Q \mathbin{;_{\mathcal{A}}} \neg R) \vdash Q \mathbin{;_{\mathcal{A}}} (R \Rightarrow S))$$

The result obtained is very similar to that of sequential composition for the original theory of designs [39, 51], except for the postcondition and the fact that we use the operator $;_{\mathcal{A}}$ instead of the sequential composition operator for relations [39]. While the precondition guarantees that it is not the case that Q establishes $\neg R$, the implication in the postcondition acts as a filter that removes final states available for angelic choice in Q that fail to satisfy R. We consider the following Example 24.

Example 24

$(\mathit{true} \vdash \{x \mapsto 1\} \in ac' \land \{x \mapsto 2\} \in ac') \mathbin{;_{\mathcal{D}ac}} (s.x \neq 1 \vdash s \in ac')$

{Theorem T.4.5.1}

$$= \left(\begin{array}{l} \neg (\neg \mathit{true} \mathbin{;_{\mathcal{A}}} \mathit{true}) \land \neg ((\{x \mapsto 1\} \in ac' \land \{x \mapsto 2\} \in ac') \mathbin{;_{\mathcal{A}}} s.x = 1) \\ \vdash \\ (\{x \mapsto 1\} \in ac' \land \{x \mapsto 2\} \in ac') \mathbin{;_{\mathcal{A}}} (s.x \neq 1 \Rightarrow s \in ac') \end{array}\right)$$

{Predicate calculus}

$$= \left(\begin{array}{l} \neg (\mathit{false} \mathbin{;_{\mathcal{A}}} \mathit{true}) \land \neg ((\{x \mapsto 1\} \in ac' \land \{x \mapsto 2\} \in ac') \mathbin{;_{\mathcal{A}}} s.x = 1) \\ \vdash \\ (\{x \mapsto 1\} \in ac' \land \{x \mapsto 2\} \in ac') \mathbin{;_{\mathcal{A}}} (s.x \neq 1 \Rightarrow s \in ac') \end{array}\right)$$

{Property of $;_{\mathcal{A}}$}

$$= \left(\begin{array}{l} \neg \mathit{false} \land \neg ((\{x \mapsto 1\} \in ac' \land \{x \mapsto 2\} \in ac') \mathbin{;_{\mathcal{A}}} s.x = 1) \\ \vdash \\ (\{x \mapsto 1\} \in ac' \land \{x \mapsto 2\} \in ac') \mathbin{;_{\mathcal{A}}} (s.x \neq 1 \Rightarrow s \in ac') \end{array}\right)$$

{Predicate calculus}

$$= \left(\begin{array}{l} \neg ((\{x \mapsto 1\} \in ac' \land \{x \mapsto 2\} \in ac') \mathbin{;_{\mathcal{A}}} s.x = 1) \\ \vdash \\ (\{x \mapsto 1\} \in ac' \land \{x \mapsto 2\} \in ac') \mathbin{;_{\mathcal{A}}} (s.x \neq 1 \Rightarrow s \in ac') \end{array}\right)$$

{Definition of $;_{\mathcal{A}}$ and substitution}

$$= \left(\begin{array}{l} \neg (\{x \mapsto 1\} \in \{s \mid s.x = 1\} \land \{x \mapsto 2\} \in \{s \mid s.x = 1\}) \\ \vdash \\ (\{x \mapsto 1\} \in \{s \mid s.x \neq 1 \Rightarrow s \in ac'\} \land \{x \mapsto 2\} \in \{s \mid s.x \neq 1 \Rightarrow s \in ac'\}) \end{array}\right)$$

{Property of sets}

$$= \left(\begin{array}{l} \neg\,(\{x \mapsto 1\}.x = 1 \land \{x \mapsto 2\}.x = 1) \\ \vdash \\ (\{x \mapsto 1\}.x \neq 1 \Rightarrow \{x \mapsto 1\} \in ac') \land (\{x \mapsto 2\}.x \neq 1 \Rightarrow \{x \mapsto 2\} \in ac') \end{array} \right)$$

{Value of component x}

$$= \left(\begin{array}{l} \neg\,(1 = 1 \land 2 = 1) \\ \vdash \\ (1 \neq 1 \Rightarrow \{x \mapsto 1\} \in ac') \land (2 \neq 1 \Rightarrow \{x \mapsto 2\} \in ac') \end{array} \right)$$

{Predicate calculus}

$$= \left(\begin{array}{l} \mathit{true} \\ \vdash \\ (\mathit{false} \Rightarrow \{x \mapsto 1\} \in ac') \land (\mathit{true} \Rightarrow \{x \mapsto 2\} \in ac') \end{array} \right) \qquad \text{\{Predicate calculus\}}$$

$$= (\mathit{true} \vdash \{x \mapsto 2\} \in ac')$$

In this case, there is an angelic choice between the assignment of the value 1 and 2 to the program variable x, sequentially composed with the program that aborts if x is 1 and that otherwise behaves as *Skip*. The resulting design is just the assignment of 2 to x that avoids aborting. In this case, the implication in the postcondition of Theorem T.4.5.1 is discarding the angelic choice where x is 1.

If we consider designs that observe **H3**, we can simplify the result further as there are no dashed variables in the precondition as established by Theorem T.4.5.2.

Theorem T.4.5.2 *Provided ok and ok′ are not free in P, Q, R and S, and that ¬ P and Q are* **PBMH**-*healthy, and that ac′ is not free in P,*

$$(P \vdash Q) \;;_{\mathcal{D}ac}\; (R \vdash S) = (P \land \neg\,(Q \;;_{\mathcal{A}}\; \neg R) \vdash Q \;;_{\mathcal{A}}\; (R \Rightarrow S))$$

This is similar to the definition of sequential composition for designs where the precondition is a condition [51], except for the use of the operator $;_{\mathcal{A}}$.

Closure

It is important that we establish closure of sequential composition ($;_{\mathcal{D}ac}$) with respect to **A**. The proof of the following closure theorem relies on results established in Appendices E and F.

Theorem T.4.5.3 *Provided P and Q are* **A**-*healthy and ok, ok′ are not free in P and Q,*

$$\mathbf{A}(P \;;_{\mathcal{D}ac}\; Q) = P \;;_{\mathcal{D}ac}\; Q$$

This result establishes that $;_{\mathcal{D}ac}$ is closed with respect to **A** provided both operands are also **A**-healthy.

Sequential Composition in Extended Binary Multirelations

The following Theorem T.4.5.4 establishes that for designs that are **A**-healthy, the definition of sequential composition corresponds to that in the isomorphic model of extended binary multirelations.

Theorem T.4.5.4 *Provided P and Q are* **A***-healthy designs,*

$$bmb2d(d2bmb(P)\ ;_{BM_\perp}\ d2bmb(Q)) = P\ ;_{\mathcal{D}ac}\ Q$$

Together with the closure of $;_{\mathcal{D}ac}$, this result enables us to ascertain the closure of $;_{BM_\perp}$.

In what follows, we concentrate our attention on important properties observed by the sequential composition operator.

Skip

Similarly to the original theory of designs, we identify the **Skip** of the theory. We denote it by $I\!I_{\mathcal{D}ac}$ and define it as follows.

Definition 97 (Skip) $I\!I_{\mathcal{D}ac} \mathrel{\widehat=} (true \vdash s \in ac')$

This is a design whose precondition is *true*, thus it is always applicable, and upon terminating it establishes that the input state s is in all sets of angelic choices ac'. The only results that can be guaranteed by the angel are those that are available in all demonic choices of the value of ac' that can be made. In this case, s is the only guarantee that we have, so the behaviour of $I\!I_{\mathcal{D}ac}$ is to maintain the current state. The following Theorems T.4.5.5 and T.4.5.6 establish that $I\!I_{\mathcal{D}ac}$ is **A**-healthy and that it is the left-unit for sequential composition ($;_{\mathcal{D}ac}$).

Theorem T.4.5.5 $\mathbf{A}(I\!I_{\mathcal{D}ac}) = I\!I_{\mathcal{D}ac}$

Theorem T.4.5.6 *Provided P is a design,* $I\!I_{\mathcal{D}ac}\ ;_{\mathcal{D}ac}\ P = P$

These results confirm that $I\!I_{\mathcal{D}ac}$ is indeed a suitable definition for the identity. We observe that $I\!I_{\mathcal{D}ac}$ is only a right-identity for angelic designs that are **H3**-healthy. This is the motivation for the following discussion.

In what follows we establish that an **H3**-design in our theory requires the precondition not to mention dashed variables, as expected [39]. We first show the result of sequentially composing an **A**-healthy design P with $\mathit{II}_{\mathcal{D}ac}$ in Theorem T.4.5.7.

Theorem T.4.5.7 *Provided P is an* **A***-healthy design,*

$$P \mathbin{;_{\mathcal{D}ac}} \mathit{II}_{\mathcal{D}ac} = ((\neg \exists ac' \bullet P^f) \vdash P^t)$$

Finally Theorem T.4.5.8 establishes that $P \mathbin{;_{\mathcal{D}ac}} \mathit{II}_{\mathcal{D}ac} = P$ restricts the precondition to a condition.

Theorem T.4.5.8 *Provided P is an* **A***-healthy design, it is* **H3***-healthy if, and only if, its precondition does not mention ac',*

$$(P \mathbin{;_{\mathcal{D}ac}} \mathit{II}_{\mathcal{D}ac}) = P \Leftrightarrow ((\exists ac' \bullet \neg P^f) = \neg P^f)$$

Sequential Composition and the Extreme Points

We now explore the consequences of sequentially composing a program with the extreme points of the lattice. As expected, we establish the same left-zero laws that hold in the original theory of designs [39].

The following Theorem T.4.5.9 shows that it is impossible to recover from an aborting program. Theorem T.4.5.10 establishes that if a design is miraculous then sequentially composing it with another design does not change its behaviour.

Theorem T.4.5.9 $\bot_{\mathcal{D}} \mathbin{;_{\mathcal{D}ac}} P = \bot_{\mathcal{D}}$

Theorem T.4.5.10 $\top_{\mathcal{D}} \mathbin{;_{\mathcal{D}ac}} P = \top_{\mathcal{D}}$

Both of these results are expected of a theory of designs [39].

This concludes our discussion of sequential composition. In the following Sections 4.5.3 and 4.5.4 we concentrate our attention on nondeterminism.

4.5.3 Demonic Choice

The intuition for the demonic choice in our theory is related to the possible ways of choosing a value for ac'. In general, this can be described using disjunction like in the original theory of designs [39].

Definition 98 $P \sqcap_{\mathcal{D}ac} Q \mathrel{\widehat{=}} P \lor Q$

This corresponds to the greatest lower bound of the lattice. We consider the following example, where $\oplus$ is the overriding operator [9].

Example 25

$$
\begin{aligned}
&(x := 1) \sqcap_{\mathcal{D}ac} (x := 2) && \{\text{Definition of assignment}\}\\
&= (\mathit{true} \vdash s \oplus (x \mapsto 1) \in ac') \sqcap_{\mathcal{D}ac} (\mathit{true} \vdash s \oplus (x \mapsto 2) \in ac')\\
& && \{\text{Definition of } \sqcap_{\mathcal{D}ac} \text{ and disjunction of designs}\}\\
&= (\mathit{true} \vdash s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto 2) \in ac')
\end{aligned}
$$

In this example we have at least two choices for the final value of ac': one has a state where x is 1 and the other has a state where x is 2. The demon can choose any set ac' satisfying either predicate. In this case, the angel is not guaranteed to be able to choose a particular final value for x, since there are no choices in the intersection of all possible choices of ac'.

Closure Properties

The demonic choice operator is closed with respect to **A**, provided that both operands are also **A**-healthy. This result follows from the distributive property of **A** with respect to disjunction, as established by the following Theorem T.4.5.11.

Theorem T.4.5.11 *Provided P and Q are designs,*

$$\mathbf{A}(P \lor Q) = \mathbf{A}(P) \lor \mathbf{A}(Q)$$

Theorem T.4.5.12 *Provided P and Q are* **A***-healthy designs,*

$$\mathbf{A}(P \sqcap_{\mathcal{D}ac} Q) = P \sqcap_{\mathcal{D}ac} Q$$

Relationship with Extended Binary Multirelations

The demonic choice operator ($\sqcap_{\mathcal{D}ac}$) corresponds exactly to the demonic choice operator ($\sqcap_{BM_\perp}$) of the binary multirelational model. This result is established by the following Theorem T.4.5.13.

Theorem T.4.5.13 $bmb2p(B_0 \sqcap_{BM_\perp} B_1) = bmb2p(B_0) \sqcap_{\mathcal{D}ac} bmb2p(B_1)$

This result confirms the correspondence of demonic choice in both models. In what follows we focus our attention on its properties.

Properties

In general, and since demonic choice is the greatest lower bound, if presented with the possibility to abort ($\perp_{\mathcal{D}}$), we expect the demon to choose the worst possible outcome as shown by the following Theorem T.4.5.14.

Theorem T.4.5.14 $P \sqcap_{\mathcal{D}ac} \perp_{\mathcal{D}} = \perp_{\mathcal{D}}$

As observed in the original theory of designs [39], the sequential composition operator distributes through demonic choice, but only from the right as established by Theorem T.4.5.15.

Theorem T.4.5.15 $(P \sqcap_{\mathcal{D}ac} Q) \mathbin{;_{\mathcal{D}ac}} R = (P \mathbin{;_{\mathcal{D}ac}} R) \sqcap_{\mathcal{D}ac} (Q \mathbin{;_{\mathcal{D}ac}} R)$

These results conclude our discussion regarding the demonic choice operator and its properties. In the following section we focus our attention on the angelic choice operator and its respective properties.

4.5.4 Angelic Choice

Similarly to other models, angelic choice is defined as the least upper bound, which in this case is conjunction.

Definition 99 $P \sqcup_{\mathcal{D}ac} Q \mathrel{\widehat{=}} P \wedge Q$

This definition is justified by the correspondence with the angelic choice operator of the binary multirelational model of Chapter 3.

To provide the intuition for this definition we consider the following Example 26.

Example 26

$$
\begin{aligned}
&((x \mapsto 1) \notin ac' \vdash (x \mapsto 1) \in ac') \sqcup_{\mathcal{D}ac} (\mathit{true} \vdash (x \mapsto 2) \in ac') && \{\text{Definition of } \sqcup_{\mathcal{D}ac}\}\\
&= \left(\begin{array}{l}
(x \mapsto 1) \notin ac' \vee \mathit{true} \\
\vdash \\
\left(\begin{array}{l}
(x \mapsto 1) \notin ac' \Rightarrow (x \mapsto 1) \in ac' \\
\wedge \\
\mathit{true} \Rightarrow (x \mapsto 2) \in ac'
\end{array}\right)
\end{array}\right) && \{\text{Predicate calculus}\}\\
&= (\mathit{true} \vdash (x \mapsto 1) \in ac' \wedge (x \mapsto 2) \in ac')
\end{aligned}
$$

It considers the angelic choice between a design that assigns 1 to the only program variable x, but does not necessarily terminate, and a design that assigns 2 to x, but terminates. The result is a program that always terminates and, for every set of final states, there is the possibility to choose angelically the assignment of the value 1 or 2 to x.

Closure Properties

Having defined angelic choice as the least upper bound operator, in the following Theorem T.4.5.16 we prove that it is closed under **A**, provided that both operands are **A**-healthy.

Theorem T.4.5.16 *Provided P and Q are* **A***-healthy,*

$$\mathbf{A}(P \sqcup_{\mathcal{D}ac} Q) = P \sqcup_{\mathcal{D}ac} Q$$

The proof for this theorem relies on the closure of **PBMH** for conjunction.

Relationship with Extended Binary Multirelations

Theorem T.4.5.17 establishes that the angelic choice operator of the designs and the binary multirelations models are in correspondence. This requires the operands to be **BMH1**-healthy. This is satisfied by every binary multirelation that is **BMH0-BMH2**.

Theorem T.4.5.17 *Provided B_0 and B_1 are* **BMH1***-healthy,*

$$bmb2p(B_0 \sqcup_{BM_\perp} B_1) = bmb2p(B_0) \sqcup_{\mathcal{D}ac} bmb2p(B_1)$$

Having established the correspondence of the angelic choice operator in both models, in the following section we focus on its properties.

Properties

In general, and since angelic choice is the least upper bound, the angelic choice of a design P and the top of the lattice ($\top_{\mathcal{D}}$) is also $\top_{\mathcal{D}}$.

Theorem T.4.5.18 *Provided P is a design, $P \sqcup_{\mathcal{D}ac} \top_{\mathcal{D}} = \top_{\mathcal{D}}$.*

In this model, sequential composition does not necessarily distribute from the right nor from the left. In order to explain the intuition behind this we present the following Counter-example 2 for distribution from the left.

Counter-example 2 *Assuming $;_{\mathcal{D}ac}$ distributes over $\sqcap_{\mathcal{D}ac}$ from the left,*

$$\left(\begin{array}{l}(true \vdash s \oplus (x \mapsto 1) \in ac') \\ \sqcap_{\mathcal{D}ac} \\ (true \vdash s \oplus (x \mapsto -1) \in ac')\end{array}\right) ;_{\mathcal{D}ac} \left(\begin{array}{l}(s.x = 1 \vdash false) \\ \sqcup \\ (s.x = -1 \vdash false)\end{array}\right) \qquad \{\text{Assumption}\}$$

$$= \left(\begin{array}{l}\left(\begin{array}{l}(true \vdash s \oplus (x \mapsto 1) \in ac') \\ \sqcap_{\mathcal{D}ac} \\ (true \vdash s \oplus (x \mapsto -1) \in ac')\end{array}\right) ;_{\mathcal{D}ac} (s.x = 1 \vdash false) \\ \sqcup_{\mathcal{D}ac} \\ \left(\begin{array}{l}(true \vdash s \oplus (x \mapsto 1) \in ac') \\ \sqcap_{\mathcal{D}ac} \\ (true \vdash s \oplus (x \mapsto -1) \in ac')\end{array}\right) ;_{\mathcal{D}ac} (s.x = -1 \vdash false)\end{array}\right)$$

$$\{\text{Definition of } \sqcap\}$$

$$= \left(\begin{array}{l}((true \vdash s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') ;_{\mathcal{D}ac} (s.x = 1 \vdash false)) \\ \sqcup_{\mathcal{D}ac} \\ ((true \vdash s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') ;_{\mathcal{D}ac} (s.x = -1 \vdash false))\end{array}\right)$$

$$\{\text{Theorem T.4.5.1}\}$$

$$= \left(\begin{array}{l}\left(\begin{array}{l}(true ;_{\mathcal{A}} true) \land \\ \neg ((s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') ;_{\mathcal{A}} s.x \neq 1) \\ \vdash \\ (s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') ;_{\mathcal{A}} (s.x = 1 \Rightarrow false)\end{array}\right) \\ \sqcup_{\mathcal{D}ac} \\ \left(\begin{array}{l}(true ;_{\mathcal{A}} true) \land \\ \neg ((s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') ;_{\mathcal{A}} s.x \neq -1) \\ \vdash \\ (s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') ;_{\mathcal{A}} (s.x = -1 \Rightarrow false)\end{array}\right)\end{array}\right)$$

$$\{\text{Predicate calculus}\}$$

$$= \left(\begin{array}{l} \left(\begin{array}{l} (\mathit{true} \mathbin{;_\mathcal{A}} \mathit{true}) \land \\ \neg\,((s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') \mathbin{;_\mathcal{A}} s.x \neq 1) \\ \vdash \\ (s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') \mathbin{;_\mathcal{A}} s.x \neq 1 \end{array}\right) \\ \sqcup_{\mathcal{D}ac} \\ \left(\begin{array}{l} (\mathit{true} \mathbin{;_\mathcal{A}} \mathit{true}) \land \\ \neg\,((s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') \mathbin{;_\mathcal{A}} s.x \neq -1) \\ \vdash \\ (s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') \mathbin{;_\mathcal{A}} s.x \neq -1 \end{array}\right) \end{array}\right)$$

{Property of $;_\mathcal{A}$ and propositional calculus}

$$= \left(\begin{array}{l} \left(\begin{array}{l} \neg\,((s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') \mathbin{;_\mathcal{A}} s.x \neq 1) \\ \vdash \\ (s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') \mathbin{;_\mathcal{A}} s.x \neq 1 \end{array}\right) \\ \sqcup_{\mathcal{D}ac} \\ \left(\begin{array}{l} \neg\,((s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') \mathbin{;_\mathcal{A}} s.x \neq -1) \\ \vdash \\ (s \oplus (x \mapsto 1) \in ac' \lor s \oplus (x \mapsto -1) \in ac') \mathbin{;_\mathcal{A}} s.x \neq -1 \end{array}\right) \end{array}\right)$$

{Definition of $;_\mathcal{A}$ and subsitution}

$$= \left(\begin{array}{l} \left(\begin{array}{l} \neg\,(s \oplus (x \mapsto 1) \in \{z \mid z.x \neq 1\} \lor s \oplus (x \mapsto -1) \in \{z \mid z.x \neq 1\}) \\ \vdash \\ (s \oplus (x \mapsto 1) \in \{s \mid s.x \neq 1\} \lor s \oplus (x \mapsto -1) \in \{s \mid s.x \neq 1\}) \end{array}\right) \\ \sqcup_{\mathcal{D}ac} \\ \left(\begin{array}{l} \neg\,(s \oplus (x \mapsto 1) \in \{z \mid z.x \neq -1\} \lor s \oplus (x \mapsto -1) \in \{z \mid z.x \neq -1\}) \\ \vdash \\ (s \oplus (x \mapsto 1) \in \{s \mid s.x \neq -1\} \lor s \oplus (x \mapsto -1) \in \{s \mid s.x \neq -1\}) \end{array}\right) \end{array}\right)$$

{Property of sets and predicate calculus}

$$= \left(\begin{array}{l} \left(\begin{array}{l} \neg\,(\neg\,(s \oplus (x \mapsto 1).x \neq 1) \lor \neg\,(s \oplus (x \mapsto -1).x \neq 1)) \\ \vdash \\ \mathit{true} \end{array}\right) \\ \sqcup_{\mathcal{D}ac} \\ \left(\begin{array}{l} \neg\,(\neg\,(s \oplus (x \mapsto 1).x \neq -1) \lor \neg\,(s \oplus (x \mapsto -1).x \neq -1)) \\ \vdash \\ \mathit{true} \end{array}\right) \end{array}\right)$$

{Property of $\oplus$}

$$= \left(\begin{array}{l} (\neg\,(\neg\, \mathit{false} \vee \neg\, \mathit{true}) \vdash \mathit{true}) \\ \sqcup_{\mathcal{D}ac} \\ (\neg\,(\neg\, \mathit{true} \vee \neg\, \mathit{false}) \vdash \mathit{true}) \end{array}\right) \qquad \{\text{Propositional calculus}\}$$

$$= (\mathit{false} \vdash \mathit{true}) \sqcup_{\mathcal{D}ac} (\mathit{false} \vdash \mathit{true}) \qquad \{\text{Property of } \sqcup_{\mathcal{D}ac}\}$$

$$= (\mathit{false} \vdash \mathit{true}) \qquad \{\text{Definition of design and propositional calculus}\}$$

$$= \mathit{true} \qquad \{\text{Definitionf of } \bot_{\mathcal{D}}\}$$

$$= \bot_{\mathcal{D}}$$

This is a sequential composition. In the first program the precondition always holds and the program presents a choice to the demon. In this case, the demon can choose the set of final states, ac', by guaranteeing that either x is set to 1 or -1 in the final set of states ac'. The second program presents an angelic choice, but the precondition makes a restriction on the value of x in the initial state s: in either case, if the precondition is satisfied the program is $\top_{\mathcal{D}}$, otherwise if no precondition can be satisfied, the program behaves as $\bot_{\mathcal{D}}$.

It is expected that the angel will avoid $\bot_{\mathcal{D}}$ if possible. In this case, it is expected, since the angel can avoid aborting irrespective of the choice the demon makes before the angel. However, if we assume that the sequential composition operator $;_{\mathcal{D}ac}$ left-distributes over angelic choice we get a different result as shown above.

In addition, sequential composition does not distribute from the right. We illustrate this in Counter-example 3. It is the sequential composition of two designs. The first design is the angelic choice between the program that assigns 2 to x, but may not terminate, and the program that always terminates but whose final set of states ac' is unrestricted, except that it cannot be the empty set. The second design is miraculous for $s.x = 2$ and for every other value of $s.x$ it aborts.

Counter-example 3

$$\left(\begin{array}{l} ((x \mapsto 2) \notin ac' \vdash (x \mapsto 2) \in ac') \\ \sqcup_{\mathcal{D}ac} \\ (\mathit{true} \vdash ac' \neq \emptyset) \end{array}\right) ;_{\mathcal{D}ac} \left(\begin{array}{l} s.x = 2 \\ \vdash \\ s.x \neq 2 \wedge ac' \neq \emptyset \end{array}\right)$$

$$\{\text{Definition of } \sqcup_{\mathcal{D}ac}\}$$

$$= \left(\begin{array}{l} (x \mapsto 2) \notin ac' \lor true \\ \vdash \\ \left(\begin{array}{l} (x \mapsto 2) \notin ac' \Rightarrow (x \mapsto 2) \in ac' \\ \land \\ true \Rightarrow ac' \neq \emptyset \end{array} \right) \end{array} \right) ;_{\mathcal{D}ac} \left(\begin{array}{l} s.x = 2 \\ \vdash \\ s.x \neq 2 \land ac' \neq \emptyset \end{array} \right)$$

{Predicate calculus}

$= (true \vdash (x \mapsto 2) \in ac' \land ac' \neq \emptyset) ;_{\mathcal{D}ac} (s.x = 2 \vdash s.x \neq 2 \land ac' \neq \emptyset)$

{Property of sets and predicate calculus}

$= (true \vdash (x \mapsto 2) \in ac') ;_{\mathcal{D}ac} (s.x = 2 \vdash s.x \neq 2 \land ac' \neq \emptyset)$ {Theorem T.4.5.1}

$$= \left(\begin{array}{l} \neg\, (false ;_{\mathcal{A}} true) \land \neg\, ((x \mapsto 2) \in ac' ;_{\mathcal{A}} s.x \neq 2) \\ \vdash \\ (x \mapsto 2) \in ac' ;_{\mathcal{A}} (s.x = 2 \Rightarrow (s.x \neq 2 \land ac' \neq \emptyset)) \end{array} \right)$$ {Predicate calculus}

$$= \left(\begin{array}{l} \neg\, (false ;_{\mathcal{A}} true) \land \neg\, ((x \mapsto 2) \in ac' ;_{\mathcal{A}} s.x \neq 2) \\ \vdash \\ (x \mapsto 2) \in ac' ;_{\mathcal{A}} s.x \neq 2) \end{array} \right)$$

{Definition of $;_{\mathcal{A}}$ and substitution}

$$= \left(\begin{array}{l} \neg\, false \land \neg\, ((x \mapsto 2) \in \{z \mid z.x \neq 2\}) \\ \vdash \\ (x \mapsto 2) \in \{z \mid z.x \neq 2\} \end{array} \right)$$ {Property of sets}

$$= \left(\begin{array}{l} \neg\, false \land \neg\, ((x \mapsto 2).x \neq 2) \\ \vdash \\ (x \mapsto 2).x \neq 2 \end{array} \right)$$ {Predicate calculus}

$= (\neg\, (2 \neq 2) \vdash 2 \neq 2)$ {Predicate calculus}

$= (true \vdash false)$ {Predicate calculus and definition of $\top_{\mathcal{D}}$}

$= \top_{\mathcal{D}}$

$\neq$

$$\left(\begin{array}{l} ((x \mapsto 2) \notin ac' \vdash (x \mapsto 2) \in ac') ;_{\mathcal{D}ac} (s.x = 2 \vdash s.x \neq 2 \land ac' \neq \emptyset) \\ \sqcup_{\mathcal{D}ac} \\ (true \vdash ac' \neq \emptyset) ;_{\mathcal{D}ac} (s.x = 2 \vdash s.x \neq 2 \land ac' \neq \emptyset) \end{array} \right)$$

{Theorem T.4.5.1}

$$= \left(\begin{array}{l} \left(\begin{array}{l} \neg\, ((x \mapsto 2) \in ac' \,;_{\mathcal{A}}\, \mathit{true}) \land \neg\, ((x \mapsto 2) \in ac' \,;_{\mathcal{A}}\, s.x \neq 2) \\ \vdash \\ (x \mapsto 2) \in ac' \,;_{\mathcal{A}}\, (s.x = 2 \Rightarrow (s.x \neq 2 \land ac' \neq \emptyset)) \end{array} \right) \\ \sqcup_{\mathcal{D}ac} \\ \left(\begin{array}{l} \neg\, (\mathit{false} \,;_{\mathcal{A}}\, \mathit{true}) \land \neg\, (ac' \neq \emptyset \,;_{\mathcal{A}}\, s.x \neq 2) \\ \vdash \\ ac' \neq \emptyset \,;_{\mathcal{A}}\, (s.x = 2 \Rightarrow (s.x \neq 2 \land ac' \neq \emptyset)) \end{array} \right) \end{array} \right)$$

{Predicate calculus}

$$= \left(\begin{array}{l} \left(\begin{array}{l} \neg\, ((x \mapsto 2) \in ac' \,;_{\mathcal{A}}\, \mathit{true}) \land \neg\, ((x \mapsto 2) \in ac' \,;_{\mathcal{A}}\, s.x \neq 2) \\ \vdash \\ (x \mapsto 2) \in ac' \,;_{\mathcal{A}}\, s.x \neq 2 \end{array} \right) \\ \sqcup_{\mathcal{D}ac} \\ \left(\begin{array}{l} \neg\, (\mathit{false} \,;_{\mathcal{A}}\, \mathit{true}) \land \neg\, (ac' \neq \emptyset \,;_{\mathcal{A}}\, s.x \neq 2) \\ \vdash \\ ac' \neq \emptyset \,;_{\mathcal{A}}\, s.x \neq 2 \end{array} \right) \end{array} \right)$$

{Definition of $;_{\mathcal{A}}$ and substitution}

$$= \left(\begin{array}{l} \left(\begin{array}{l} \neg\, ((x \mapsto 2) \in \{z \mid \mathit{true}\}) \land \neg\, ((x \mapsto 2) \in \{z \mid z.x \neq 2\}) \\ \vdash \\ (x \mapsto 2) \in \{z \mid z.x \neq 2\} \end{array} \right) \\ \sqcup_{\mathcal{D}ac} \\ \left(\begin{array}{l} \neg\, \mathit{false} \land \neg\, (\{z \mid z.x \neq 2\} \neq \emptyset) \\ \vdash \\ \{z \mid z.x \neq 2\} \neq \emptyset \end{array} \right) \end{array} \right)$$

{Predicate calculus and property of sets}

$$= \left(\begin{array}{l} \left(\begin{array}{l} \neg\, \mathit{true} \land \neg\, (x \mapsto 2).x \neq 2 \\ \vdash \\ (x \mapsto 2).x \neq 2 \end{array} \right) \\ \sqcup_{\mathcal{D}ac} \\ \left(\begin{array}{l} \neg\, \mathit{false} \land \neg\, \mathit{true} \\ \vdash \\ \mathit{true} \end{array} \right) \end{array} \right)$$

{Predicate calculus}

$= (\mathit{false} \vdash \mathit{false}) \sqcup_{\mathcal{D}ac} (\mathit{false} \vdash \mathit{true})$ {Predicate calculus and definition of $\bot_{\mathcal{D}}$}

$= \bot_{\mathcal{D}} \sqcup_{\mathcal{D}ac} \bot_{\mathcal{D}}$ {Definition of $\sqcup_{\mathcal{D}ac}$, $\bot_{\mathcal{D}}$ and predicate calculus}

$= \bot_{\mathcal{D}}$

When the angelic choice is resolved first the result is the program that always terminates and whose set of final states ac' has a state where x is assigned the value 2. Sequentially composing this with the second design results in a miracle ($\top_{\mathcal{D}}$) as the only state available for angelic choice is where x has the value 2. And this is precisely the case in which the design behaves miraculously.

If we distribute the sequential composition through the angelic choice, in the resulting angelic choice there are two sequential compositions. In the first one, the result is $\bot_{\mathcal{D}}$ as the first design may not terminate. In the second, termination is guaranteed but any final set of states ($ac' \neq \emptyset$) may fail to satisfy the precondition $s.x = 2$, in which case the design aborts. In conclusion, angelic choice does not distribute through sequential composition at all.

4.6 Relationship with Designs

In this section we study the relationship between the model of **A**-designs and the original theory of homogeneous designs of Hoare and He [39]. As we depict in Figures 1.1 and 1.4, this is achieved by defining a pair of linking functions: $d2ac$, which maps from designs into angelic designs, and $ac2p$, which maps in the opposite direction.

In the following Section 4.6.1 we introduce the definition of $d2ac$. In Section 4.6.2 we define $ac2p$ and discuss how the angelic nondeterminism of a theory can be removed. Finally in Section 4.6.3 we establish that there is a Galois connection between the theory of **A**-designs and the original theory of designs, and that there is an isomorphism when we consider the subset of **A2**-healthy angelic designs.

4.6.1 From Designs to Angelic Designs ($d2ac$ and $p2ac$)

The main concern when mapping a design into an angelic design pertains to encoding both the pre and postcondition in terms of a single initial state s and a set of final states ac'. Since the model of **A**-designs is also a theory of designs, ok and ok' retain the same meaning. The function $d2ac$ is defined as follows.

Definition 100 $d2ac(P) \mathrel{\widehat=} (\neg\, p2ac(P^f) \land (\neg\, P^f[\mathbf{s}/in\alpha_{-ok}] \mathrel{;} true) \vdash p2ac(P^t))$

The negation of the precondition P^f and the postcondition are mapped using the auxiliary function $p2ac$, while the second conjunct in the precondition of the angelic design requires that whenever $\neg\, P^f$ holds, then there is some final observation of the values of the variables in $out\alpha$. The predicate $\neg\, P^f[\mathbf{s}/in\alpha_{-ok}] \mathrel{;} true$ can be restated as $\exists\, out\alpha \bullet \neg\, P^f[\mathbf{s}/in\alpha_{-ok}]$. Essentially this allows the value of ac' to be

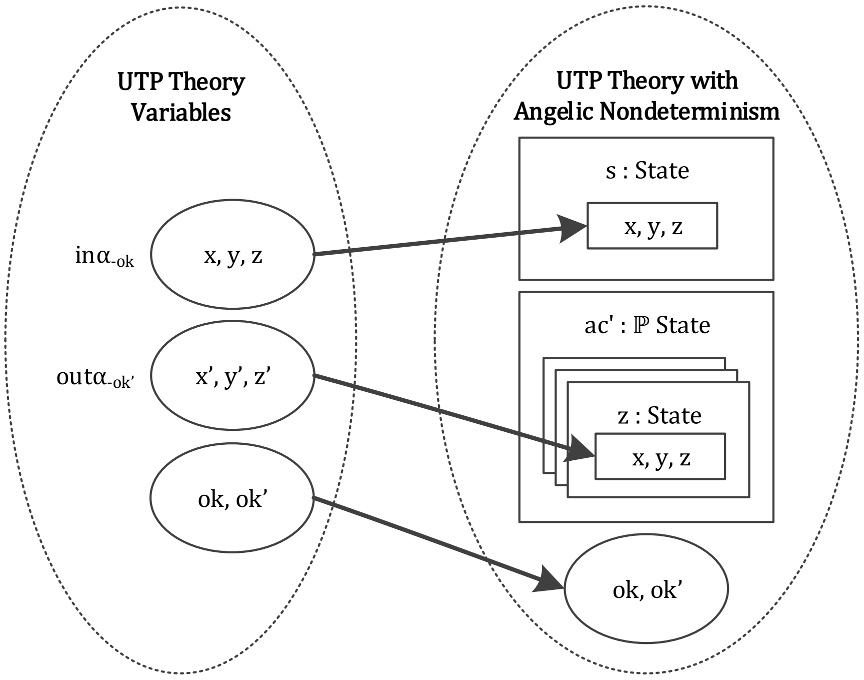


Figure 4.1: Encoding variables in a theory of angelic designs using $p2ac$

unspecified when the precondition $\neg\ P^f$ is not satisfied. This is defined using the substitution operator $[\mathbf{s}/S\alpha]$, where the boldface indicates that s is a record, and so the substitution is not simply s for $S\alpha$. Instead, for an arbitrary set of variables $S\alpha$, the substitution operator needed is defined as follows.

Definition 101 $P[\mathbf{z}/S\alpha] \mathrel{\widehat=} P[z.s_0, \ldots, z.s_n/s_0, \ldots, s_n]$

Each variable s_i in $S\alpha$ is replaced with $z.s_i$. As an example, we consider the substitution $(x' = 2 \land ok')[\mathbf{s}, \mathbf{z}/in\alpha_{-ok}, out\alpha_{-ok'}]$, whose result is $z.x' = 2 \land ok'$. The substitution $[\mathbf{z}/S\alpha]$ is well-formed whenever $S\alpha$ is a subset of the record components of z. In Appendix D we establish properties satisfied by this operator.

The main purpose of $p2ac$ is to encode predicates in terms of s and ac'. For a given predicate P whose input and output alphabets are $in\alpha$ and $out\alpha$, respectively, its encoding in a theory with angelic nondeterminism is given by the following function $p2ac$, which we illustrate in Figure 4.1.

Definition 102 $p2ac(P) \mathrel{\widehat=} \exists\, z \bullet P[\mathbf{s}, \mathbf{z}/in\alpha_{-ok}, out\alpha_{-ok'}] \land undash(z) \in ac'$

First, each variable in the set of input and output variables, other than ok and ok', is replaced with the corresponding component of the initial state s and a final state z from the set of final states available for angelic choice. Since in our encoding states have undashed components, we require $undash(z)$ to be in ac'.

The result of $p2ac$ is upward-closed, that is, the predicates in the range of $p2ac$ are fixed points of **PBMH** as established by the following Lemma L.4.6.1.

Lemma L.4.6.1 $\mathbf{PBMH} \circ p2ac(P) = p2ac(P)$

As previously discussed, this property is essential for a theory of angelic nondeterminism. The function $p2ac$ distributes through disjunction as established by the following Theorem T.4.6.1

Theorem T.4.6.1 $p2ac(P \lor Q) = p2ac(P) \lor p2ac(Q)$

In the case of conjunction there is an implication as established by Theorem T.4.6.2, rather than an equality, as $p2ac$ is defined using an existential quantifier.

Theorem T.4.6.2 $p2ac(P \land Q) \Rightarrow p2ac(P) \land p2ac(Q)$

More importantly, the result of $p2ac$ is **A2**-healthy as established by Theorem T.4.6.3.

Theorem T.4.6.3 $\mathbf{A2} \circ p2ac(P) = p2ac(P)$

This is expected since the original predicates mapped by $p2ac$ do not have angelic nondeterminism.

A consequence of the definition of $p2ac$ is that it requires ac' not to be empty, unless P is itself *false*. In the following Theorem T.4.6.4, we consider the application of $p2ac$ to a design P when ac' is not empty.

Theorem T.4.6.4

$$ac' \neq \emptyset \land p2ac(\neg\, P^f \vdash P^t) = ac' \neq \emptyset \land (\neg\, p2ac(P^f) \vdash p2ac(P^t))$$

In this case $p2ac$ can be applied directly to the negation of the precondition P^f and the postcondition P^t of a design P. This result sheds light on the relationship between $p2ac$ and $d2ac$ as established by Theorem T.4.6.5.

Theorem T.4.6.5 *Provided P is a design,*

$$ac' \neq \emptyset \land p2ac(P) = ac' \neq \emptyset \land d2ac(P)$$

When we consider the case of a design whose set of final states ac' is not empty, then $d2ac$ is simply $p2ac$.

Finally, we establish that $d2ac$ yields an **A**-healthy design, that is, the designs in the range of $d2ac$ are fixed points of the healthiness condition **A**.

Theorem T.4.6.6 $\mathbf{A} \circ d2ac(P) = d2ac(P)$

This concludes our discussion regarding the definition of $d2ac$ and its most important properties.

4.6.2 Removing Angelic Nondeterminism ($ac2p$)

The mapping from angelic to non-angelic predicates is defined by $ac2p$, whose goal is to collapse the set of final states ac' into a single state, and, introduce the input and output variables as used in other theories. Its definition is presented below.

Definition 103

$$ac2p(P) \mathrel{\widehat{=}} \mathbf{PBMH}(P)[State_{II}(in\alpha_{-ok})/s] \;;_{\mathcal{A}} \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x$$

First, for a predicate P, $ac2p$ takes the result of applying **PBMH** to P to achieve upward closure of ac'. This is followed by the replacement of s to introduce the corresponding input variables of the set $in\alpha_{-ok}$, which excludes ok. As already mentioned, the observational variables ok and ok' retain the same meaning in the theories considered. Finally, the resulting predicate is sequentially composed, using $;_{\mathcal{A}}$, with a predicate that introduces the corresponding output variables of the resulting final state, except for ok'. For a set of variables $S\alpha$, $State_{II}(S\alpha)$ is an identity record, whose components s_i are mapped to the respective variables s_i.

Definition 104 $State_{II}(S\alpha) \mathrel{\widehat{=}} \{s_0 \mapsto s_0, \ldots, s_n \mapsto s_n\}$

As an example, we consider the substitution $(s.x = 1 \land ok)[State_{II_{-ok}}(in\alpha)/s]$, whose result is $x = 1 \land ok$. If we consider the definition of **PBMH** and $;_{\mathcal{A}}$, then $ac2p$ can be rewritten as established by the following Lemma L.4.6.2.

Lemma L.4.6.2

$$ac2p(P) = \exists\, ac' \bullet \left(\begin{array}{l} P[State_{II}(in\alpha)/s] \\ \land \\ \forall\, z \bullet z \in ac' \Rightarrow (\bigwedge x : out\alpha \bullet dash(z).x = x) \end{array}\right)$$

That is, the variable ac' is quantified away, and for each state z in the set ac', the output variables in $out\alpha$, except for ok', are introduced and set to the respective values of the components of z. Since in our encoding the components of a state are always undashed, we apply the function $dash(z)$ to z. If there is more than one state in ac', then $ac2p$ yields *false* as no x variable can take on more than one value.

4.6.3 Isomorphism and Galois Connection

Having defined a pair of linking functions between the theory of angelic designs and designs, in this section we show that, in general, there is a Galois connection between the two theories. In addition, when we consider the subset of **A2**-healthy designs these two theories can be shown to be isomorphic.

From Designs

The mapping of a design P through $d2ac$ and then $ac2p$ yields the same design P as established by the following Theorem T.4.6.7.

Theorem T.4.6.7 *Provided that P is a design, $ac2p \circ d2ac(P) = P$.*

That is, in the theory of angelic designs we can model the original designs of Hoare and He [39] without angelic nondeterminism. This is a reassuring result which confirms the suitability of our model.

From Angelic Designs

When the linking functions are applied in the reverse order, however, we do not obtain the same design P. This result is established by Theorem T.4.6.8.

Theorem T.4.6.8 *Provided P is an* **A**-*healthy design, $d2ac \circ ac2p(P) \sqsupseteq P$.*

In general, the result of the application of $ac2p$ followed by $d2ac$ to an **A**-healthy design P is stronger than P. This is because the angelic nondeterminism is removed. For instance, the mapping of an angelic choice over two assignments $x := 1$ and $x := 2$ yields the top of the lattice $\top_{\mathcal{D}}$.

Example 27

$d2ac \circ ac2p(x := 1 \sqcup x := 2)$ {Definition of assignment and $\sqcup$}

$= d2ac \circ ac2p(true \vdash s \oplus \{x \mapsto 1\} \in ac' \land s \oplus \{x \mapsto 2\} \in ac')$

{Lemma L.C.5.47}

$$= \left(\begin{array}{l} \neg\ p2ac(ac2p(\mathit{false})) \land (\exists\ out\alpha \bullet \neg\ ac2p(\mathit{false})[\mathbf{s}/in\alpha]) \\ \vdash \\ p2ac(ac2p(s \oplus \{x \mapsto 1\} \in ac' \land s \oplus \{x \mapsto 2\} \in ac')) \end{array}\right)$$

{Lemma L.C.5.27}

$$= \left(\begin{array}{l} \neg\ p2ac(\mathit{false}) \land (\exists\ out\alpha \bullet \neg\ \mathit{false}[\mathbf{s}/in\alpha]) \\ \vdash \\ p2ac(ac2p(s \oplus \{x \mapsto 1\} \in ac' \land s \oplus \{x \mapsto 2\} \in ac')) \end{array}\right)$$

{Predicate calculus and Lemma L.C.5.3}

$$= \left(\begin{array}{l} \mathit{true} \\ \vdash \\ p2ac(ac2p(s \oplus \{x \mapsto 1\} \in ac' \land s \oplus \{x \mapsto 2\} \in ac')) \end{array}\right) \qquad \{\text{Lemma L.5.3.1}\}$$

$$= \left(\begin{array}{l} \mathit{true} \\ \vdash \\ \exists\ ac_0, y \bullet \left(\begin{array}{l} s \oplus \{x \mapsto 1\} \in ac' \\ \land \\ s \oplus \{x \mapsto 2\} \in ac' \end{array}\right) [ac_0/ac'] \land ac_0 \subseteq \{y\} \land y \in ac' \end{array}\right)$$

{Substitution and property of sets}

$$= (\mathit{true} \vdash \mathit{false}) \qquad \{\text{Definition of } \top_{\mathcal{D}}\}$$

$$= \top_{\mathcal{D}}$$

The results of Theorems T.4.6.8 and T.4.6.7 establish that we have a Galois connection between the two theories.

From A2-healthy Angelic Designs

If we consider the subset of **A**-healthy designs that is in addition **A2**-healthy, then we can prove the reverse implication of Theorem T.4.6.8 as established by the following Theorem T.4.6.9.

Theorem T.4.6.9 *Provided P is an* **A0**-**A2**-*healthy design,* $d2ac \circ ac2p(P) \sqsubseteq P$.

Together these results allow us to prove that there is a bijection for the subset of **A2**-healthy designs.

Theorem T.4.6.10 *Provided P is a design that is* **A0**-**A2**-*healthy,*

$$d2ac \circ ac2p(P) = P$$

Proof. Follows from Theorems T.4.6.8 and T.4.6.9. □

This result confirms that these models are isomorphic as depicted in Figure 1.1.

This concludes our discussion on the relationship between the original theory of designs and the model of angelic designs. In the following Section 4.7 we focus our attention on the relationship with the **PBMH** theory [38].

4.7 Relationship with the PBMH Theory

The final link that we study in this chapter pertains to the relationship between the model of **A**-designs and the theory of angelic nondeterminism of Cavalcanti et al. [38]. As previously discussed in Section 2.4.4, in that theory the alphabet consists of the input program variables, and a single output variable ac', which is a record whose components range over the dashed output program variables. In addition, termination is captured without considering ok and ok'.

When establishing a link between the theories of interest, the first concern is their alphabets. As we discussed in Section 4.1, the ac' of both theories can be related through the functions *undashset* and *dashset*, which *undash* and *dash* the components of every state in a set, respectively.

In order to relate both theories, we introduce a pair of linking functions, $d2pbmh$, which maps **A**-healthy designs to **PBMH** predicates, and $pbmh2d$, which maps predicates in the opposite direction. We introduce their definitions in the following Sections 4.7.1 and 4.7.2. Finally in Section 4.7.3 we show that there is a Galois connection between the theories, and that in general, the subset of angelic designs that is **H3**-healthy is isomorphic to the theory of [38].

4.7.1 From Angelic Designs to PBMH ($d2pbmh$)

In order to map angelic designs into the theory of **PBMH**, it is necessary to hide the variables ok and ok', introduce the input variables in $in\alpha$, and appropriately *dash* the set of final states ac'. This is captured by the function $d2pbmh$ as follows.

Definition 105

$$\begin{aligned}
&d2pbmh : \mathbf{A} \to \mathbf{PBMH}\\
&d2pbmh(P) \mathrel{\widehat=} (\neg\, P^f \Rightarrow P^t)[true/ok][undashset(ac')/ac'][State_{II}(in\alpha_{-ok})/s]
\end{aligned}$$

First we consider the implication between the precondition $\neg\, P^f$ and postcondition P^t of a design P. We require that ok is $true$ and perform the following substitutions.

Since the new variable ac' considers dashed components, the old variable ac' is replaced with an *undashed* version of ac'. Finally, the input variables in $in\alpha_{-ok}$, which excludes ok, are introduced via the substitution of $State_{\mathit{II}}(in\alpha_{-ok})$ for s.

We consider Example 28, where *d2pbmh* is applied to the assignment $x := 1$.

Example 28

$$\begin{aligned}
&d2pbmh(x := 1) && \{\text{Definition of assignment}\}\\
&= d2pbmh(true \vdash s \oplus \{x \mapsto 1\} \in ac') && \{\text{Definition of } d2pbmh\}\\
&= (true \Rightarrow s \oplus \{x \mapsto 1\} \in ac')[true/ok][undashset(ac')/ac'][State_{\mathit{II}}(in\alpha_{-ok})/s]\\
& && \{\text{Substitution}\}\\
&= true \Rightarrow State_{\mathit{II}}(in\alpha_{-ok}) \oplus \{x \mapsto 1\} \in undashset(ac') && \{\text{Predicate calculus}\}\\
&= State_{\mathit{II}}(in\alpha_{-ok}) \oplus \{x \mapsto 1\} \in undashset(ac') && \{\text{Definition of } State_{\mathit{II}}\}\\
&= \{x_0 \mapsto x_0, \ldots, x_n \mapsto x_n\} \oplus \{x \mapsto 1\} \in undashset(ac') && \{\text{Definition of } \theta in\alpha\}\\
&= \theta in\alpha \oplus \{x \mapsto 1\} \in undashset(ac') && \{\text{Property of sets, } dash \text{ and } dashsset\}\\
&= (\theta in\alpha)' \oplus \{x' \mapsto 1\} \in ac'
\end{aligned}$$

The result is the corresponding assignment in the **PBMH** theory [38], where the state obtained by dashing every component of the initial state $\theta in\alpha$ is overridden so that the component x' takes the value of 1. The following Theorem T.4.7.1 establishes that *d2pbmh* yields predicates that are **PBMH**-healthy.

Theorem T.4.7.1 *Provided P is* **PBMH***-healthy,*

$$\mathbf{PBMH} \circ d2pbmh(P) = d2pbmh(P)$$

That is, when *d2pbmh* is applied to an angelic design that is **A**-healthy, then it is also **PBMH**-healthy. Therefore the application of *d2pbmh* yields a **PBMH**-healthy predicate as required.

4.7.2 From PBMH to Angelic Designs (*pbmh2d*)

In order to define a mapping in the opposite direction, we need to consider how to express a precondition in the theory of [38]. In that model, successful termination is guaranteed whenever ac' is not empty. The definition of the mapping from **PBMH** into angelic designs, *pbmh2d*, is defined below.

Definition 106

$$\begin{aligned} &pbmh2d : \mathbf{PBMH} \rightarrow \mathbf{A} \\ &pbmh2d(P) \mathrel{\widehat=} (\neg\ P[\emptyset/ac'] \vdash P[dashset(ac')/ac'])[\mathbf{s}/in\alpha_{-ok}] \end{aligned}$$

The precondition of the corresponding **A**-design requires that ac' is not empty. In the postcondition we substitute the existing set of final states ac' with a dashed version $dashset(ac')$. Finally, we require that the initial variables of P are components of the initial state s. In the following Theorem T.4.7.2 we prove that $pbmh2d$ yields designs that are **A** and **H3**-healthy.

Theorem T.4.7.2 *Provided P is* **PBMH***-healthy,*

$$\mathbf{A} \circ \mathbf{H3} \circ pbmh2d(P) = pbmh2d(P)$$

Similarly to the definition of $d2pbmh$, the proviso of Theorem T.4.7.2 ensures that the function is only applied to predicates that are **PBMH**-healthy.

4.7.3 Galois Connection and Isomorphism

In general, the model of angelic designs can express every existing program of the theory of [38]. That is, those programs can be specified as angelic designs, where the precondition may not refer to the final set of states ac'. This is formally established by the following Theorem T.4.7.3.

Theorem T.4.7.3 *Provided P is* **PBMH***-healthy, $d2pbmh \circ pbmh2d(P) = P$.*

Its only requirement is that the predicate must be **PBMH**-healthy.

However, when we consider the reverse functional composition of $d2pbmh$ and $pbmh2d$, we obtain a different result as established by Theorem T.4.7.4.

Theorem T.4.7.4 *Provided P is an* **A***-healthy design,*

$$pbmh2d \circ d2pbmh(P) \sqsubseteq P$$

This is because the theory of [38] cannot model sets of final states where termination is not guaranteed, as is the case for angelic designs which are not **H3**-healthy. Hence, these two results establish that the two adjoints form a Galois connection.

If we consider the subset of angelic designs that are, in addition, **H3**-healthy, then we obtain a bijection via the functions $d2pbmh$ and $pbmh2d$, as established by the following Theorem T.4.7.5.

Theorem T.4.7.5 *Provided P is design that is* **A** *and* **H3***-healthy,*

$$pbmh2d \circ d2pbmh(P) = P$$

While this is an expected result, it is reassuring that the subset of our theory that is **H3**-healthy is in exact correspondence with the UTP theory of [38].

We observe that the subset of the binary multirelational model of Chapter 3 that is **BMH3**-healthy is isomorphic to the original theory of binary multirelations. Since binary multirelations are also isomorphic to the UTP theory of [38], the result presented in this section is also in agreement.

4.8 Final Considerations

In this chapter we have presented a new theory of designs where both angelic and demonic nondeterminism can be modelled. This consists of an extension of the binary multirelational encoding of [38] to include the auxiliary variables *ok* and *ok′* of the theory of designs [39]. Our angelic designs are not necessarily **H3**-healthy as required for a treatment of processes.

The healthiness conditions of the theory have been presented and their main properties proved. Through the development of the extended theory of binary multirelations of Chapter 3, and the subsequent isomorphism, we have been able to justify and explore the definition of the operators and the refinement order. It is reassuring to know that the usual refinement order defined by universal reverse implication corresponds to subset inclusion in the binary multirelational model.

Perhaps the most challenging aspect of the theory is that it relies on non-homogeneous relations. As a consequence, sequential composition cannot be defined as relational composition. While the definition may not be immediately obvious, it is more intuitive when considered in the equivalent binary multirelational model of Chapter 3. We have taken advantage of this correspondence to define an operator with the expected properties.

In addition, we have established that every design can be expressed in the theory of angelic designs. Moreover, the subset of **A2**-healthy designs is isomorphic to the original theory of homogeneous designs of Hoare and He [39].

Finally, we have also studied the relationship between angelic designs and the UTP theory of [38]. This is a complementary result to the link between the model of $BM_{\perp}$ relations and that of the original theory of binary multirelations. This gives us further assurance as to the capability to express the existing theories as a subset of our own correctly.

Chapter 5

Reactive Angelic Designs

Based on the theory of angelic designs and the principles underlying the theory of reactive processes, in this chapter we propose a natural extension to the UTP theory of CSP where both angelic and demonic nondeterminism can be modelled. In Section 5.1 we introduce the principles underlying our approach and justify the encoding proposed for CSP. In Section 5.2 the healthiness conditions of the theory are presented. Section 5.3 discusses the relationship between the new theory and the existing model of CSP. The operators of the theory are discussed in Section 5.4 and, for each operator, we discuss the relationship with their respective counterpart in the original CSP theory. In Section 5.5 we characterise the important subset of non-divergent reactive angelic designs. Finally, we summarize our results in Section 5.6.

5.1 Introduction

As discussed earlier in Section 2.5.4 the observational variables of the UTP theory of CSP are ok and ok' to record stability, and the additional variables $wait$, tr and ref, and their respective dashed counterparts. Based on the concept of states originally introduced in Section 2.3, we consider a model where the observational variables of the theory of reactive processes are encoded as components of a $State$. We define the alphabet as follows.

Definition 107 (Alphabet)

$$ok, ok' : \{true, false\}, s : State(\{tr, ref, wait\}), ac' : \mathbb{P}\, State(\{tr, ref, wait\})$$

In addition to a single initial state s, a set of final states ac', and the observational variables ok and ok' that record stability, we require that every $State$ has record components of name tr, $wait$ and ref. This enables the angelic choice over the final or intermediate observations of tr, ref and $wait$.

We next show how we can express every healthiness condition of the original theory of reactive processes, and ultimately CSP, in this new encoding. We then propose linking functions between the theories so that we can reason about the correspondence of the healthiness conditions and operators of both models. These are important aspects for establishing the validity of the model.

5.2 Healthiness Conditions

Since this is a theory with angelic nondeterminism, the set of final states ac' must be upward-closed, so relations in this theory need to satisfy **PBMH**. As previously discussed in Section 2.5.4, in the UTP, CSP processes are characterised as the image of designs through the function **R**. In order to preserve the existing semantics, we propose a corresponding construction; in the following Sections 5.2.1 to 5.2.5 we restate all the properties enforced by **R** in this new model. Namely, we define healthiness conditions **RA1**, **RA2** and **RA3**, whose functional composition is named **RA**, and, **CSPA1** and **CSPA2**. All the healthiness conditions discussed in this chapter are monotonic and idempotent. In Section 5.2.6 we show how this construction allows CSP processes with angelic nondeterminism to be expressed as the image of angelic designs through **RA**, the counterpart to **R**.

5.2.1 RA1

The first property of interest that underpins the theory of reactive processes is the notion that the history of events observed cannot be undone. In general, for any initial state x, the set of all final states that satisfy this property is given by $States_{tr \leq tr'}(x)$ as defined below.

Definition 108 $States_{tr \leq tr'}(x) \mathrel{\widehat{=}} \{z : State(\{tr, ref, wait\}) \mid x.tr \leq z.tr\}$

This definition is used for introducing the first healthiness condition, **RA1**, that not only enforces this notion for final states in ac', but also requires that there is some final state satisfying this property available for angelic choice.

Definition 109 $\mathbf{RA1}(P) \mathrel{\widehat{=}} (P \wedge ac' \neq \emptyset)[States_{tr \leq tr'}(s) \cap ac'/ac']$

A consequence of the definition of **RA1** is that it also enforces **A0**.

Theorem T.5.2.1 $\mathbf{RA1} \circ \mathbf{A0}(P) = \mathbf{RA1}(P)$

Although **A0** only requires ac' not to be empty in the postcondition of an angelic design, **RA1** requires this under all circumstances. Proof of this and other results not explicitly included in the body of this document can be found in Appendix G of the extended version of this thesis [74].

The function **RA1** distributes through both conjunction and disjunction as established by the following Theorems T.5.2.2 and T.5.2.3.

Theorem T.5.2.2 $\mathbf{RA1}(P \land Q) = \mathbf{RA1}(P) \land \mathbf{RA1}(Q)$

Theorem T.5.2.3 $\mathbf{RA1}(P \lor Q) = \mathbf{RA1}(P) \lor \mathbf{RA1}(Q)$

Since **RA1** is also idempotent, consequently both conjunction and disjunction are also closed under **RA1**.

Similarly to the theory of angelic designs, in this model, the definition of sequential composition is also based on $;_{\mathcal{A}}$. In Theorem T.5.2.4 we establish that this operator is closed under **RA1**.

Theorem T.5.2.4 *Provided P and Q are* **RA1***-healthy and Q is* **PBMH***-healthy,*

$$\mathbf{RA1}(P \mathbin{;_{\mathcal{A}}} Q) = P \mathbin{;_{\mathcal{A}}} Q$$

For every healthiness condition of the theory, the upward-closure enforced by **PBMH** must be maintained. Theorem T.5.2.5 establishes this for **RA1**.

Theorem T.5.2.5 $\mathbf{PBMH} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P) = \mathbf{RA1} \circ \mathbf{PBMH}(P)$

However, **PBMH** and **RA1** do not commute in general. We consider the following Counter-example 4 where the healthiness conditions are applied to the relation $ac' = \emptyset$, which is not **PBMH**-healthy.

Counter-example 4

$$\begin{aligned}
&\mathbf{RA1} \circ \mathbf{PBMH}(ac' = \emptyset) && \{\text{Definition of } \mathbf{PBMH} \text{ (Lemma L.4.2.1)}\}\\
&= \mathbf{RA1}(\exists\, ac_0 \bullet ac_0 = \emptyset \land ac_0 \subseteq ac') && \{\text{One-point rule and property of sets}\}\\
&= \mathbf{RA1}(\mathit{true}) && \{\text{Lemma L.G.1.11}\}\\
&= \mathit{States}_{tr \le tr'}(s) \cap ac' \neq \emptyset
\end{aligned}$$

$$\begin{aligned}
&\mathbf{PBMH} \circ \mathbf{RA1}(ac' = \emptyset) && \{\text{Definition of } \mathbf{RA1}\}\\
&= \mathbf{PBMH}((ac' = \emptyset \land ac' \neq \emptyset)[\mathit{States}_{tr \le tr'}(s) \cap ac'/ac']) && \{\text{Predicate calculus}\}
\end{aligned}$$

$= \mathbf{PBMH}(\mathit{false})$ {Definition of **PBMH** (Lemma L.4.2.1)}

$= \mathit{false}$

In the first case, the application of **PBMH** yields *true*. The result of the functional composition is then **RA1**(*true*). On the other hand, in the second case, there is a contradiction arising from the application of **RA1**, which leaves us with the result *false*.

5.2.2 RA2

The next healthiness condition of interest is **RA2**, which requires a process to be insensitive to the initial trace of events $s.tr$. It is the counterpart to **R2** of the original theory of reactive processes, and is also defined using substitution.

Definition 110

$$\mathbf{RA2}(P) \mathrel{\widehat=} P\left[s \oplus \{tr \mapsto \langle\rangle\}, \left\{ z \;\middle|\; \begin{array}{l} z \in ac' \land s.tr \leq z.tr \\ \bullet\; z \oplus \{tr \mapsto z.tr - s.tr\} \end{array} \right\} \middle/ s, ac' \right]$$

It defines the component tr in the initial state s to be the empty sequence, and consequently the set of final states ac' is restricted by considering those states z whose traces are a suffix of $s.tr$, and furthermore, defining their trace to be the difference with respect to the initial trace $s.tr$.

Since substitution distributes through conjunction and disjunction, so does the function **RA2** as established by the following Theorems T.5.2.6 and T.5.2.7.

Theorem T.5.2.6 $\mathbf{RA2}(P \land Q) = \mathbf{RA2}(P) \land \mathbf{RA2}(Q)$

Theorem T.5.2.7 $\mathbf{RA2}(P \lor Q) = \mathbf{RA2}(P) \lor \mathbf{RA2}(Q)$

As **RA2** is idempotent, both conjunction and disjunction are closed under **RA2**.

Similarly to the case for **RA1**, the operator $;_{\mathcal{A}}$ is also closed under **RA2**.

Theorem T.5.2.8 *Provided P and Q are* **RA2***-healthy,*

$$\mathbf{RA2}(P \;;_{\mathcal{A}}\; Q) = P \;;_{\mathcal{A}}\; Q$$

A consequence of the definition of **RA2** is that applying it to the predicate that requires ac' not to be empty is equivalent to applying **RA2** to the relation *true*.

Theorem T.5.2.9 $\mathbf{RA2}(ac' \neq \emptyset) = \mathbf{RA1}(\mathit{true})$

Proof.

$$\begin{aligned}
&\mathbf{RA2}(ac' \neq \emptyset) && \{\text{Definition of } \mathbf{RA2}\}\\
&= (ac' \neq \emptyset)[s \oplus \{tr \mapsto \langle\rangle, \{z \mid z \in ac' \land s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\}/s, ac'] && \{\text{Substitution}\}\\
&= \{z \mid z \in ac' \land s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\} \neq \emptyset && \{\text{Property of sets}\}\\
&= \exists\, y \bullet y \in \{z \mid z \in ac' \land s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\} && \{\text{Property of sets}\}\\
&= \exists\, y, z \bullet z \in ac' \land s.tr \leq z.tr \land y = z \oplus \{tr \mapsto z.tr - s.tr\} && \{\text{One-point rule}\}\\
&= \exists\, z \bullet z \in ac' \land s.tr \leq z.tr && \{\text{Lemma L.G.1.10}\}\\
&= \mathbf{RA1}(true)
\end{aligned}$$

□

This result sheds light on the relationship between **RA2** and **RA1**, as in fact, these functions are commutative as established by Theorem T.5.2.10.

Theorem T.5.2.10 $\mathbf{RA2} \circ \mathbf{RA1}(P) = \mathbf{RA1} \circ \mathbf{RA2}(P)$

Finally, **RA2** preserves the upward closure of **PBMH**.

Theorem T.5.2.11 $\mathbf{PBMH} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P) = \mathbf{RA2} \circ \mathbf{PBMH}(P)$

These results conclude our discussion of **RA2** and its most important properties.

5.2.3 RA3

Similarly to the theory of reactive processes, we must ensure that a process cannot be started before the previous process has finished interacting with the environment. The counterpart to **R3** in this new theory is **RA3**. Before exploring its definition, we introduce the identity $I\!I_{\mathbf{RAD}}$ of our theory.

Definition 111 $I\!I_{\mathbf{RAD}} \mathrel{\widehat=} (\mathbf{RA1}(\neg\, ok) \lor (ok' \land s \in ac'))$

Similarly to the reactive identity $I\!I_{rea}$, the behaviour for an unstable state $\neg\, ok$ is given by **RA1**, that is, there must be at least one final state in ac' whose trace is a suffix of the initial trace $s.tr$. Otherwise, the process is stable, ok' is $true$, and the initial state s is in the set of final states ac'.

Having defined the identity, we introduce the definition of **RA3** below.

Definition 112 $\mathbf{RA3}(P) \mathrel{\widehat=} I\!I_{\mathbf{RAD}} \lhd s.wait \rhd P$

This definition is similar to that of the original theory, except that we use $I\!I_{\mathbf{RAD}}$ as the identity and use $s.wait$ instead of $wait$ as a condition since in our theory $wait$ is a component of the initial state s. Using Leibniz's substitution, it is possible to establish the following Lemma L.5.2.1, where $P_w = P[s \oplus \{wait \mapsto w\}/s]$.

Lemma L.5.2.1 $\mathbf{RA3}(P) = \mathbf{RA3}(P_f)$

This result is in correspondence with a similar property of **R3** in the original theory of CSP that is essential in the characterisation of CSP processes via reactive designs.

Similarly to the previous healthiness conditions, **RA3** also distributes through both conjunction and disjunction as established by Theorems T.5.2.12 and T.5.2.13.

Theorem T.5.2.12 $\mathbf{RA3}(P \land Q) = \mathbf{RA3}(P) \land \mathbf{RA3}(Q)$

Theorem T.5.2.13 $\mathbf{RA3}(P \lor Q) = \mathbf{RA3}(P) \lor \mathbf{RA3}(Q)$

Consequently, these operators are closed under **RA3**.

The operator $;_{\mathcal{A}}$ is also closed under **RA3** provided that the second operand is also **RA1**-healthy as established by Theorem T.5.2.14.

Theorem T.5.2.14 *Provided P and Q are* **RA3***-healthy and Q is* **RA1***-healthy,*

$$\mathbf{RA3}(P \mathbin{;_{\mathcal{A}}} Q) = P \mathbin{;_{\mathcal{A}}} Q$$

The proviso is similar to that observed for the closure of $;$ under **R3** in the original theory of reactive processes [44]. The extra restriction on Q, which needs to be **RA1**-healthy, is not a problem since the theory of interest is characterised by the functional composition of all healthiness conditions.

Furthermore, as required, the function **RA3** also preserves the upward-closure.

Theorem T.5.2.15 $\mathbf{PBMH} \circ \mathbf{RA3} \circ \mathbf{PBMH}(P) = \mathbf{RA3} \circ \mathbf{PBMH}(P)$

The identity $I\!I_{\mathbf{RAD}}$ is a fixed point of every healthiness condition, including **RA1**, **RA2**, **RA3** and **PBMH** as established by Theorems T.G.3.1 to T.G.3.4. Finally, **RA3** commutes with both **RA1** and **RA2** as established by Theorems T.5.2.16 and T.5.2.17.

Theorem T.5.2.16 $\mathbf{RA3} \circ \mathbf{RA1}(P) = \mathbf{RA3} \circ \mathbf{RA1}(P)$

Theorem T.5.2.17 $\mathbf{RA2} \circ \mathbf{RA3}(P) = \mathbf{RA3} \circ \mathbf{RA2}(P)$

This concludes our discussion of the most important properties of **RA3**.

5.2.4 RA

The healthiness conditions that we have considered so far in this chapter are counterparts to those of the original model of reactive processes. Hence this is a theory that is similarly characterised by the functional composition of the healthiness conditions **RA1**, **RA2**, **RA3**, besides **PBMH**. In order to provide a parallel with the original theory of reactive processes, we define part of this composition as **RA**.

Definition 113 $\mathbf{RA}(P) \mathrel{\widehat=} \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{RA3}(P)$

The order of the functional composition is not important since these functions commute, except for **PBMH** that does not necessarily commute with every function. So when considering the counterpart theory to reactive processes, but with angelic nondeterminism, **PBMH** needs to be applied before **RA**.

As previously stated, every healthiness condition considered in this chapter is idempotent and monotonic. Theorems T.G.1.1, T.G.2.1 and T.G.3.5 in Appendix G establish that **RA1**, **RA2** and **RA3** are idempotent. Similarly monotonicity is established for these functions by Theorems T.G.1.2, T.G.2.2 and T.G.3.6. As a consequence the functional composition **RA** is also idempotent and monotonic.

In addition, since all of the **RA** functions distribute through conjunction and disjunction so does the functional composition **RA**. Finally, **RA** maintains the upward-closure enforced by **PBMH** since all of the **RA** healthiness conditions do so as well. This concludes our discussion of the most important properties of **RA**.

5.2.5 CSP Processes with Angelic Nondeterminism

In the original theory of CSP, another two healthiness conditions, **CSP1** and **CSP2**, are required, in addition to **R**, to characterise CSP processes. In order to consider a theory of CSP with angelic nondeterminism we follow a similar approach by defining a counterpart to these functions in what follows.

CSPA1

The first healthiness condition of interest is **CSPA1**, which is the counterpart to **CSP1** in the new theory. Its definition is presented below.

Definition 114 $\mathbf{CSPA1}(P) \mathrel{\widehat=} P \lor \mathbf{RA1}(\neg\, ok)$

A CSP process with angelic nondeterminism P is required to observe **RA1** when in an unstable state. For a **RA**-healthy process, this property is already enforced by **RA1** under all circumstances. Similarly to the original theory of CSP [44] the following Theorem T.5.2.18 establishes that this behaviour can also be described as the functional composition of **RA1** after **H1**.

Theorem T.5.2.18 $\mathbf{CSPA1} \circ \mathbf{RA1}(P) = \mathbf{RA1} \circ \mathbf{H1}(P)$

Proof.

$$\begin{array}{lr}
\mathbf{CSPA1} \circ \mathbf{RA1}(P) & \{\text{Definition of } \mathbf{CSPA1}\} \\
= \mathbf{RA1}(P) \lor \mathbf{RA1}(\neg\, ok) & \{\text{Theorem T.5.2.3}\} \\
= \mathbf{RA1}(P \lor \neg\, ok) & \{\text{Predicate calculus}\} \\
= \mathbf{RA1}(ok \Rightarrow P) & \{\text{Definition of } \mathbf{H1}\} \\
= \mathbf{RA1} \circ \mathbf{H1}(P) &
\end{array}$$

□

The function **CSPA1** is idempotent and monotonic. Furthermore, it preserves the upward-closure as required by **PBMH**.

Theorem T.5.2.19 *Provided P is* **PBMH***-healthy,*

$$\mathbf{PBMH} \circ \mathbf{CSPA1}(P) = \mathbf{CSPA1}(P)$$

This concludes the discussion of the properties of **CSPA1**.

CSPA2

The last healthiness condition of interest is the counterpart to **CSP2**. It is defined as **H2** with the extended alphabet that includes s and ac'.

Definition 115 $\mathbf{CSPA2}(P) \mathrel{\widehat=} \mathbf{H2}(P)$

This healthiness condition satisfies the same properties as **H2**, including, for example, those established by Theorems T.4.2.10 and T.E.6.1. It can alternatively be defined using the J-split of Woodcock and Cavalcanti [51].

5.2.6 Reactive Angelic Designs (RAD)

The theory of CSP processes in the new model is defined by **RAD**, which is the functional composition of all the healthiness conditions of interest.

Definition 116 $\mathbf{RAD}(P) \mathrel{\widehat=} \mathbf{RA} \circ \mathbf{CSPA1} \circ \mathbf{CSPA2} \circ \mathbf{PBMH}(P)$

Since **PBMH** and **RA1** do not commute, **PBMH** is applied first. The fixed points of **RAD** are the reactive angelic designs. Every such process P can be expressed as $\mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f)$ as established by the following Theorem T.5.2.20, where $P^o_w = P[o, s \oplus \{wait \mapsto w\}/ok', s]$

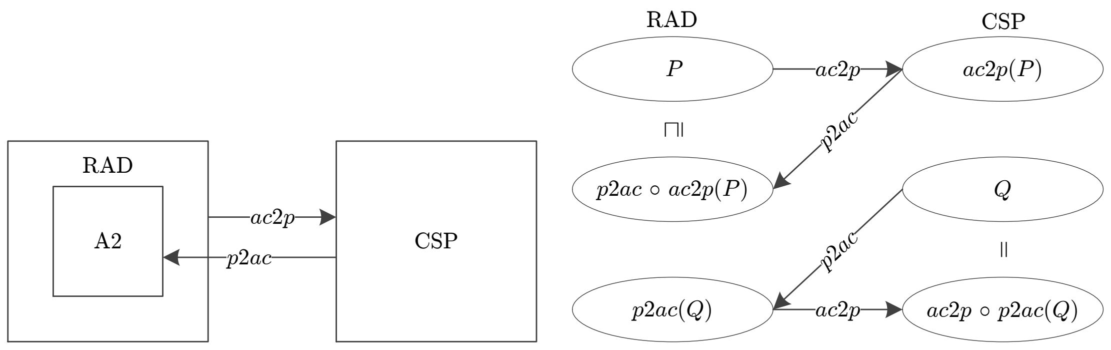
Theorem T.5.2.20 $\mathbf{RAD}(P) = \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f)$

Proof.

$$
\begin{array}{lr}
\mathbf{RAD}(P) & \{\text{Definition of } \mathbf{RAD}\}\\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{CSPA1} \circ \mathbf{CSPA2} \circ \mathbf{PBMH}(P) & \{\text{Theorem T.G.5.3}\}\\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{CSPA2} \circ \mathbf{PBMH}(P) & \{\mathbf{CSPA2} \text{ is } \mathbf{H2}\}\\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{H2} \circ \mathbf{PBMH}(P) & \{\text{Theorem T.5.2.1}\}\\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{A0} \circ \mathbf{H1} \circ \mathbf{H2} \circ \mathbf{PBMH}(P) & \\
 & \{\text{Theorems T.E.6.1 and T.E.6.2}\}\\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{A0} \circ \mathbf{PBMH} \circ \mathbf{H1} \circ \mathbf{H2}(P) & \{\text{Definition of design}\}\\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{A0} \circ \mathbf{PBMH}(\neg\, P^f \vdash P^t) & \{\text{Definition of } \mathbf{A}\}\\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{A}(\neg\, P^f \vdash P^t) & \\
 & \{\text{Theorems T.5.2.10, T.5.2.17 and T.5.2.16}\}\\
= \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{RA3} \circ \mathbf{A}(\neg\, P^f \vdash P^t) & \{\text{Lemmas L.C.1.5 and L.5.2.1}\}\\
= \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{RA3} \circ \mathbf{A}((\neg\, P^f \vdash P^t)_f) & \{\text{Substitution}\}\\
= \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{RA3} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) & \{\text{Definition of } \mathbf{RA}\}\\
= \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) &
\end{array}
$$

□

That is, such processes can be specified as the image of an **A**-healthy design through the function **RA**. This is a result similar to that obtained for CSP processes as the image of designs through **R** [39, 44]. Since both **RA** and **A** are monotonic and idempotent, and the theory of designs is a complete lattice [39], so is the theory of reactive angelic designs.



(a) Theories and links (b) Predicates and links

Figure 5.1: Relationship between theories

Since **PBMH** is just **A1**, and **RA1** enforces **A0**, a fixed point P of **RAD** can alternatively be expressed as shown in the following Lemma L.5.2.2.

Lemma L.5.2.2 $\mathbf{RAD}(P) = \mathbf{RA}(\neg\, \mathbf{PBMH}(P)^f_f \vdash \mathbf{PBMH}(P)^t_f)$

That is, an angelic design, with **PBMH** applied to the negation of the precondition and postcondition. Furthermore, it is possible to infer that if P is a reactive angelic design, then it is also **PBMH**-healthy.

Theorem T.5.2.21 *Provided P is* **RAD**-*healthy,* $\mathbf{PBMH}(P) = P$.

This concludes our discussion of the healthiness condition of the theory of reactive angelic designs, **RAD**, and its respective properties.

5.3 Relationship with CSP

The theory of reactive angelic designs can be related to the original UTP theory of CSP through the pair of linking functions $ac2p$ and $p2ac$ previously introduced in Section 4.6 and reproduced below.

$$ac2p(P) \mathrel{\widehat=} \mathbf{PBMH}(P)[State_{II}(in\alpha_{-ok})/s] \;;_{\mathcal{A}} \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x$$

$$p2ac(P) \mathrel{\widehat=} \exists\, z \bullet P[\mathbf{s}, \mathbf{z}/in\alpha_{-ok}, out\alpha_{-ok'}] \land undash(z) \in ac'$$

We employ $ac2p$ by considering the set of variables $in\alpha$ to be $\{tr, ref, wait\}$, and a corresponding set of variables $out\alpha$ with dashed counterparts; $State$, therefore, has components ranging over $in\alpha$. Similarly, for the mapping in the opposite direction, from reactive angelic designs to CSP processes we employ $p2ac$ with the same sets of variables $in\alpha$ and $out\alpha$.

The relationship between the models has previously been illustrated in the context of all theories in Figure 1.1. Here we focus our attention on the relationship with CSP. In Figure 5.1(a) each theory is labelled according to its healthiness conditions. The subset of reactive angelic designs that corresponds exactly to CSP processes is characterised by **A2**, the healthiness condition which we previously discussed in Section 4.2.4 that characterises predicates with no angelic nondeterminism.

In Figure 5.1(b) the relationship between the predicates of each theory is illustrated. For a predicate P of the theory of reactive angelic designs, the functional composition $p2ac \circ ac2p(P)$ yields a stronger predicate since any angelic nondeterminism in P is virtually collapsed into a single final state, while for a predicate Q of the CSP theory, the composition $ac2p \circ p2ac(Q)$ yields exactly the same predicate Q. Thus a Galois connection exists between the theories.

5.3.1 From Reactive Angelic Designs to CSP ($ac2p$)

As already stated, the mapping from reactive angelic designs to CSP processes achieved through $ac2p$ defines a Galois connection. Application of this function to a predicate P that is both **RA**-healthy and **PBMH**-healthy yields a healthy counterpart in the original theory as established by the following Theorem T.5.3.1.

Theorem T.5.3.1 *Provided P is* **PBMH***-healthy,* $ac2p \circ \mathbf{RA}(P) = \mathbf{R} \circ ac2p(P)$

If we consider P to be a reactive angelic design, then we can show that the application of $ac2p$ yields a reactive design as established by Theorem T.5.3.2

Theorem T.5.3.2 $ac2p \circ \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) = \mathbf{R}(\neg\, ac2p(P^f_f) \vdash ac2p(P^t_f))$

Proof.

$$\begin{aligned}
& ac2p \circ \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) && \{\text{Theorem T.G.1.6}\}\\
&= ac2p \circ \mathbf{RA} \circ \mathbf{PBMH}(\neg\, P^f_f \vdash P^t_f) && \{\text{Theorem T.5.3.1}\}\\
&= \mathbf{R} \circ ac2p \circ \mathbf{PBMH}(\neg\, P^f_f \vdash P^t_f) && \{\text{Lemma L.C.5.36}\}\\
&= \mathbf{R} \circ ac2p(\neg\, P^f_f \vdash P^t_f) && \{\text{Lemma L.C.5.28}\}\\
&= \mathbf{R}(\neg\, ac2p(P^f_f) \vdash ac2p(P^t_f))
\end{aligned}$$

□

This is a pleasing result that supports the reuse of results across the theories. We consider the following Example 29, where $ac2p$ is applied to the angelic choice between a prefixing on the event a followed by deadlock, and on the event b followed by deadlock. The operators of the theory of reactive angelic designs have subscript $_{\mathbf{RAD}}$ in order to distinguish them from those of the original theory of CSP which have subscript $_{\mathbf{R}}$.

Example 29

$$\begin{aligned}&ac2p(a \to_{\mathbf{RAD}} Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \to_{\mathbf{RAD}} Stop_{\mathbf{RAD}})\\&=\\&a \to_{\mathbf{R}} Stop_{\mathbf{R}} \sqcup_{\mathbf{R}} b \to_{\mathbf{R}} Stop_{\mathbf{R}}\end{aligned}$$

Proof. Lemma L.G.8.2 □

The result is the least upper bound of the corresponding CSP process, where $\sqcup_{\mathbf{R}}$ is also defined using conjunction. This is a process that cannot be expressed using the standard operators of CSP. The conjunction of non-divergent CSP processes requires the conjunction of their respective postconditions, and thus an agreement. In this case, both processes can only agree on the trace of events remaining unchanged, and not refusing events a and b, while waiting.

5.3.2 From CSP to Reactive Angelic Designs ($p2ac$)

The mapping in the opposite direction, from CSP processes to reactive angelic designs, is achieved through the function $p2ac$. As discussed in Section 4.6 the result of applying $p2ac$ is upward-closed as established by Lemma L.4.6.1. The application of $p2ac$ to a process P that is $\mathbf{R}$-healthy, can be described by the functional composition of $\mathbf{RA}$ after $p2ac$ to the original process P, as established by the following Theorem T.5.3.3.

Theorem T.5.3.3 $p2ac \circ \mathbf{R}(P) = \mathbf{RA} \circ p2ac(P)$

The result of applying $p2ac$ to a reactive design is established in Theorem T.5.3.4: $p2ac$ can be directly applied to the pre and postconditions separately, followed by $\mathbf{A}$ and $\mathbf{RA}$.

Theorem T.5.3.4 $p2ac \circ \mathbf{R}(\neg\, P^f_f \vdash P^t_f) = \mathbf{RA} \circ \mathbf{A}(\neg\, p2ac(P^f_f) \vdash p2ac(P^t_f))$

Proof.

$p2ac \circ \mathbf{R}(\neg\, P^f_f \vdash P^t_f)$ {Theorem T.5.3.3 and definition of $\mathbf{RA}$}

$$
\begin{aligned}
&= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ p2ac(\neg P^f_f \vdash P^t_f) && \{\text{Definition of } \mathbf{RA1}\}\\
&= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1}(p2ac(\neg P^f_f \vdash P^t_f) \land ac' \neq \emptyset) && \{\text{Theorem T.4.6.4}\}\\
&= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1}((\neg p2ac(P^f_f) \vdash p2ac(P^t_f)) \land ac' \neq \emptyset) && \{\mathbf{RA1} \text{ and } \mathbf{RA}\}\\
&= \mathbf{RA}(\neg p2ac(P^f_f) \vdash p2ac(P^t_f)) && \{\text{Lemma L.4.6.1}\}\\
&= \mathbf{RA}(\neg \mathbf{PBMH} \circ p2ac(P^f_f) \vdash \mathbf{PBMH} \circ p2ac(P^t_f)) && \{\text{Definition of } \mathbf{A1}\}\\
&= \mathbf{RA} \circ \mathbf{A1}(\neg p2ac(P^f_f) \vdash p2ac(P^t)) && \{\text{Definition of } \mathbf{RA} \text{ and Theorem T.5.2.1}\}\\
&= \mathbf{RA} \circ \mathbf{A0} \circ \mathbf{A1}(\neg p2ac(P^f_f) \vdash p2ac(P^t)) && \{\text{Definition of } \mathbf{A}\}\\
&= \mathbf{RA} \circ \mathbf{A}(\neg p2ac(P^f_f) \vdash p2ac(P^t))
\end{aligned}
$$

□

This result enables CSP processes to be easily mapped into the theory of reactive angelic designs by considering the mapping of the pre and postconditions of CSP processes directly.

We consider the following example, where the terminating process $Skip_{\mathbf{R}}$ is mapped through $p2ac$ into the theory of reactive angelic designs.

Example 30

$$p2ac(Skip_{\mathbf{R}}) = \mathbf{RA} \circ \mathbf{A}(true \vdash \exists\, y \bullet \neg\, y.wait \land y.tr = s.tr \land y \in ac')$$

Proof. Theorem T.5.4.19 □

The reactive angelic design also has *true* as its precondition, while the postcondition asserts that there is a final state y in the set of angelic choices ac' where the trace of events $s.tr$ is kept unchanged and the value of the component *wait* is *false*, that is, the process has finished interacting with the environment.

5.3.3 Galois Connection and Isomorphism

As already mentioned, the pair of linking functions we have considered establish a Galois connection between the theory of CSP and that of reactive angelic designs. When considering the mapping from the original theory of reactive processes, followed by the mapping in the opposite direction, we obtain an exact correspondence as shown in the following Theorem T.5.3.5.

Theorem T.5.3.5 $ac2p \circ p2ac(P) = P$

Proof.

$$
\begin{aligned}
& ac2p \circ p2ac(P) && \{\text{Definition of } ac2p\}\\
&= (\mathbf{PBMH} \circ p2ac(P))[State_{II}(in\alpha_{-ok})/s] \;;_{\mathcal{A}} \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x \\
& && \{\text{Lemma L.4.6.1}\}\\
&= p2ac(P)[State_{II}(in\alpha_{-ok})/s] \;;_{\mathcal{A}} \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x \\
& && \{\text{Definition of } p2ac\}\\
&= \left(\begin{array}{l} (\exists z \bullet P[\mathbf{s}, \mathbf{z}/in\alpha_{-ok}, out\alpha_{-ok'}] \land undash(z) \in ac')[State_{II}(in\alpha_{-ok})/s] \\ ;_{\mathcal{A}} \\ \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x \end{array}\right) \\
& && \{\text{Substitution}\}\\
&= \left(\begin{array}{l} (\exists z \bullet P[\mathbf{s}, \mathbf{z}/in\alpha_{-ok}, out\alpha_{-ok'}][State_{II}(in\alpha_{-ok})/s] \land undash(z) \in ac') \\ ;_{\mathcal{A}} \\ \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x \end{array}\right) \\
& && \{\text{Lemma L.D.1.10}\}\\
&= (\exists z \bullet P[\mathbf{z}/out\alpha_{-ok'}] \land undash(z) \in ac') \;;_{\mathcal{A}} \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x \\
& && \{\text{Definition of } ;_{\mathcal{A}} \text{ and substitution}\}\\
&= \exists z \bullet P[\mathbf{z}/out\alpha_{-ok'}] \land undash(z) \in \{s \mid \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x\} \\
& && \{\text{Property of sets}\}\\
&= \exists z \bullet P[\mathbf{z}/out\alpha_{-ok'}] \land \bigwedge x : out\alpha_{-ok'} \bullet dash(undash(z)).x = x \\
& && \{\text{Property of } dash \text{ and } undash\}\\
&= \exists z \bullet P[\mathbf{z}/out\alpha_{-ok'}] \land \bigwedge x : out\alpha_{-ok'} \bullet z.x = x && \{\text{Lemma L.D.1.9}\}\\
&= P[\mathbf{z}/out\alpha_{-ok'}][State_{II}(out\alpha_{-ok'})/z] && \{\text{Lemma L.D.1.10}\}\\
&= P
\end{aligned}
$$

□

This results establishes that our theory can accommodate the existing CSP processes appropriately, that is, those without angelic nondeterminism.

When considering the mapping in the opposite direction we obtain the following result in Lemma L.5.3.1.

Lemma L.5.3.1 $p2ac \circ ac2p(P) = \exists ac_0, y \bullet P[ac_0/ac'] \land ac_0 \subseteq \{y\} \land y \in ac'$

If the set of final states ac_0 in P has more than one state, then the result of $p2ac \circ ac2p(P)$ is *false*, otherwise, ac_0 is either a singleton, in which case ac' is any set containing its element, or empty, in which case ac' is arbitrary. Most importantly, the functional composition only preserves predicates whose set of angelic choices is either empty or a singleton, otherwise the result is *false*.

We consider the following Example 31, where Lemma L.5.3.1 is applied to the angelic choice between events a or b followed by deadlock.

Example 31

$$\begin{array}{l}
p2ac \circ ac2p(a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}} \sqcup b \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}) \\
= \\
\mathbf{RA} \circ \mathbf{A} \, (true \vdash \exists \, y \bullet y \in ac' \land y.wait \land y.tr = s.tr \land a \notin y.ref \land b \notin y.ref)
\end{array}$$

Proof. Lemmas L.G.8.2 and L.G.8.3 □

This process corresponds to the application of $p2ac$ to the result obtained in the previous Example 29. In this case, the process is always waiting for the environment and keeps the trace of events unchanged, however it requires that neither event a nor b are refused. This is a process whose behaviour cannot be described using the standard operators of CSP.

If we consider the result of Lemma L.5.3.1 in the context of the predicates of our theory, that is, those which are **PBMH**-healthy, then we obtain an inequality as shown in the following Theorem T.5.3.6.

Theorem T.5.3.6 *Provided P is* **PBMH**-*healthy,* $p2ac \circ ac2p(P) \sqsupseteq P$.

Proof.

$$\begin{array}{ll}
p2ac \circ ac2p(P) & \{\text{Lemma L.5.3.1}\} \\
= \exists \, ac_0, y \bullet P[ac_0/ac'] \land ac_0 \subseteq \{y\} \land y \in ac' & \{\text{Property of sets}\} \\
= \exists \, ac_0, y \bullet P[ac_0/ac'] \land ac_0 \subseteq \{y\} \land \{y\} \subseteq ac' & \{\text{Predicate calculus}\} \\
\Rightarrow \exists \, ac_0 \bullet P[ac_0/ac'] \land ac_0 \subseteq ac' & \{\text{Definition of } \mathbf{PBMH} \text{ (Lemma L.4.2.1)}\} \\
= \mathbf{PBMH}(P) & \{\text{Assumption: } P \text{ is } \mathbf{PBMH}\text{-healthy}\} \\
= P &
\end{array}$$

□

This theorem, together with Theorem T.5.3.5, establishes the existence of a Galois connection between the theories. In particular, these results also hold between reactive processes, characterised by **R**, and the reactive angelic designs, characterised by **RAD**, that is, in general, the Galois connection is not restricted to CSP processes. This is because the proviso of Theorem T.5.3.5 only requires P to be **PBMH**-healthy.

The result of Theorem T.5.3.6 can be strengthened into an equality by considering the subset of reactive angelic designs that are **A2**-healthy. These are reactive processes that do not exhibit angelic nondeterminism. If we consider the application of **A2** to the process $a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}$, we obtain exactly the same result as in Example 31. In other words, for reactive angelic designs, **A2** characterises the same fixed points as $p2ac \circ ac2p(P)$. We observe, however, that in general, **A2** permits an empty set of final states, whereas in the theory of reactive angelic designs, both **RA1** and the mapping $p2ac$ require the set of final states not to be empty. For example, in the theory of angelic designs the bottom $\bot_{\mathcal{D}}$ of the lattice, which is *true*, is a fixed point of **A2** (Lemma L.C.1.13).

Finally, Theorem T.5.3.7 establishes that the result $p2ac \circ ac2p(P)$ for a reactive angelic design P that is **A2**-healthy yields exactly the same reactive angelic design P.

Theorem T.5.3.7 *Provided P^f_f and P^t_f are* **A2***-healthy,*

$$p2ac \circ ac2p \circ \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) = \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f)$$

In summary, when we consider the theory of reactive angelic designs that are **A2**-healthy, then we find that there is a bijection with the original theory of reactive designs. Thus this subset is isomorphic to the theory of CSP.

5.4 Operators

Having discussed the healthiness conditions of our theory, and the relationship with the original model of CSP, in this section we present the definition of some important operators of CSP in the new model. For each of the operators we show how they relate to their original CSP counterparts.

5.4.1 Angelic Choice

The first operator of interest is angelic choice. Similarly to the theory of angelic designs, it is also defined as the least upper bound of the lattice, which is conjunction.

Definition 117 $P \sqcup_{\mathbf{RAD}} Q \mathrel{\widehat{=}} P \wedge Q$

For reactive angelic designs P and Q, this result can be restated as shown in the following Theorem T.5.4.1.

Theorem T.5.4.1 *Provided P and Q are reactive angelic designs,*

$$P \sqcup Q = \mathbf{RA} \circ \mathbf{A}(\neg P^f_f \lor \neg Q^f_f \vdash (\neg P^f_f \Rightarrow P^t_f) \land (\neg Q^f_f \Rightarrow Q^t_f))$$

The precondition of the resulting process is the disjunction of the preconditions of P and Q, while the postcondition is the conjunction of two implications. In both cases, if either the precondition of P or Q holds, then the corresponding postcondition is established. This is a result that is similar to that observed for the least upper bound of designs [39, 51].

The least upper bound of this theory can be related with that of CSP as follows. If we consider two CSP processes P and Q, apply $p2ac$ followed by the least upper bound $\sqcup_{\mathbf{RAD}}$ and then $ac2p$, then we obtain the same result defined by the original least upper bound operator $\sqcup_{\mathbf{R}}$ of CSP as shown in Theorem T.5.4.2.

Theorem T.5.4.2 $ac2p(p2ac(P) \sqcup_{\mathbf{RAD}} p2ac(Q)) = P \sqcup_{\mathbf{R}} Q$

Proof.

$$\begin{array}{lr}
ac2p(p2ac(P) \sqcup_{\mathbf{RAD}} p2ac(Q)) & \{\text{Definition of } \sqcup_{\mathbf{RAD}}\} \\
= ac2p(p2ac(P) \land p2ac(Q)) & \{\text{Theorem T.C.5.2}\} \\
= ac2p \circ p2ac(P) \land ac2p \circ p2ac(Q) & \{\text{Theorem T.5.3.5}\} \\
= P \land Q & \{\text{Definition of } \sqcup_{\mathbf{R}}\} \\
= P \sqcup_{\mathbf{R}} Q &
\end{array}$$

□

This is expected since we can express every existing CSP process in the new theory. The result in the opposite direction, however, is an inequality as shown in the following Theorem T.5.4.3.

Theorem T.5.4.3 *Provided that P and Q are reactive angelic designs,*

$$p2ac(ac2p(P) \sqcup_{\mathbf{R}} ac2p(Q)) \sqsupseteq P \sqcup_{\mathbf{RAD}} Q$$

Proof.

$$\begin{array}{lr}
p2ac(ac2p(P) \sqcup_{\mathbf{R}} ac2p(Q)) & \{\text{Definition of } \sqcup_{\mathbf{R}}\}
\end{array}$$

$= p2ac(ac2p(P) \land ac2p(Q))$ {Theorem T.4.6.2}

$\sqsupseteq p2ac \circ ac2p(P) \land p2ac \circ ac2p(Q)$ {Theorem T.G.7.13}

$\sqsupseteq \mathbf{PBMH}(P) \land \mathbf{PBMH}(Q)$ {P and Q are **RAD**-healthy and Theorem T.5.2.21}

$= P \land Q$ {Definition of $\sqcup_{\mathbf{RAD}}$}

$= P \sqcup_{\mathbf{RAD}} Q$

□

That is, there is a strengthening of the resulting predicate. This is expected, as in general the application of $ac2p$ collapses the angelic nondeterminism, and $p2ac$ cannot undo such effect completely.

This concludes our discussion of the basic properties of angelic choice. In the following sections, and as we present the definition of the CSP operators, we revisit angelic choice and explore its role when applied together with other operators.

5.4.2 Demonic Choice

Similarly to the definition of internal choice in CSP, in our theory, this operator is also defined using the greatest lower bound of the lattice, disjunction.

Definition 118 $P \sqcap_{\mathbf{RAD}} Q \mathrel{\widehat=} P \lor Q$

For any two reactive angelic designs P and Q, their demonic choice can be described as a reactive angelic design as stated as in Theorem T.5.4.4.

Theorem T.5.4.4 *Provided P and Q are reactive angelic processes,*

$$P \sqcap_{\mathbf{RAD}} Q = \mathbf{RA} \circ \mathbf{A}(\neg P^f_f \land \neg Q^f_f \vdash P^t_f \lor Q^t_f)$$

That is, the resulting precondition is the conjunction of the respective preconditions of P and Q, while the postcondition is the disjunction of the respective postconditions of P and Q. Intuitively, in a demonic choice both preconditions need to be satisfied, while either the postcondition of P or Q may be observed.

The greatest lower bound of both theories can be related through the pair of linking functions $p2ac$ and $ac2p$. Since $p2ac$ distributes through disjunction we can establish the following general result in Theorem T.5.4.5.

Theorem T.5.4.5

$$p2ac(ac2p(P) \sqcap_{\mathbf{R}} ac2p(Q)) = p2ac \circ ac2p(P) \sqcap_{\mathbf{RAD}} p2ac \circ ac2p(Q)$$

Proof.

$$\begin{aligned}
& p2ac(ac2p(P) \sqcap_{\mathbf{R}} ac2p(Q)) && \{\text{Definition of } \sqcap\} \\
&= p2ac(ac2p(P) \lor ac2p(Q)) && \{\text{Theorem T.4.6.1}\} \\
&= p2ac \circ ac2p(P) \lor p2ac \circ ac2p(Q) && \{\text{Definition of } \sqcap\} \\
&= p2ac \circ ac2p(P) \sqcap_{\mathbf{RAD}} p2ac \circ ac2p(Q)
\end{aligned}$$

□

If we consider two reactive angelic designs P and Q and apply $ac2p$, followed by the greatest lower bound $\sqcap_{\mathbf{R}}$ and then $p2ac$, then this result can be directly obtained by applying $p2ac \circ ac2p$ followed by the greatest lower bound $\sqcap_{\mathbf{RAD}}$. When P and Q are **A2**-healthy (Theorem T.5.3.7) we obtain the result shown in Lemma L.5.4.1.

Lemma L.5.4.1 *Provided P and Q are reactive angelic designs and* **A2***-healthy,*

$$p2ac(ac2p(P) \sqcap_{\mathbf{R}} ac2p(Q)) = P \sqcap_{\mathbf{RAD}} Q$$

That is, for reactive angelic designs with no angelic nondeterminism, the demonic choice of both theories is in correspondence. Similarly, since $ac2p$ also distributes through disjunction, we can establish the following result in the opposite direction, as shown in Theorem T.5.4.6.

Theorem T.5.4.6 $ac2p(p2ac(P) \sqcap_{\mathbf{RAD}} p2ac(Q)) = P \sqcap_{\mathbf{R}} Q$

That is, the greatest lower bound of both theories is in correspondence. Finally, since the least upper bound is conjunction, and the greatest lower bound is disjunction, angelic and demonic choice distribute over each other.

5.4.3 Chaos

The following operator of interest is $Chaos_{\mathbf{RAD}}$, which is the bottom of the lattice of reactive angelic designs.

Definition 119 $Chaos_{\mathbf{RAD}} \mathrel{\widehat=} \mathbf{RA} \circ \mathbf{A}(false \vdash ac' \neq \emptyset)$

Its precondition is *false* while the postcondition requires that ac' is not empty. The postcondition can alternatively be specified as *true* since both **A** and **RA1** ensure that the design is **A0**-healthy. This process is a zero for demonic choice as established by Theorem T.5.4.7.

Theorem T.5.4.7 *Provided P is a reactive angelic design,*

$$Chaos_{\mathbf{RAD}} \sqcap_{\mathbf{RAD}} P = Chaos_{\mathbf{RAD}}$$

Similarly to the original theory, if a process may diverge immediately in a demonic choice, then this is the only possibility. The dual of this property is the unit law for angelic choice as shown in the following Theorem T.5.4.8.

Theorem T.5.4.8 *Provided P is a reactive angelic design,*

$$Chaos_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P = P$$

Proof.

$$
\begin{array}{ll}
Chaos_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P & \{\text{Assumption: } P \text{ is } \mathbf{RAD}\text{-healthy}\} \\
Chaos \sqcup \mathbf{RA} \circ \mathbf{A}(\neg P^f_f \vdash P^t_f) & \{\text{Definition of } Chaos\} \\
= \mathbf{RA} \circ \mathbf{A}(false \vdash ac' \neq \emptyset) \sqcup \mathbf{RA} \circ \mathbf{A}(\neg P^f_f \vdash P^t_f) & \{\text{Theorem T.5.4.1}\} \\
= \mathbf{RA} \circ \mathbf{A}(false \lor \neg P^f_f \vdash (false \Rightarrow ac' \neq \emptyset) \land (\neg P^f_f \Rightarrow P^t_f)) & \{\text{Predicate calculus}\} \\
= \mathbf{RA} \circ \mathbf{A}(\neg P^f_f \vdash (\neg P^f_f \Rightarrow P^t_f)) & \{\text{Definition of design and predicate calculus}\} \\
= \mathbf{RA} \circ \mathbf{A}(\neg P^f_f \vdash P^t_f) & \{\text{Assumption: } P \text{ is } \mathbf{RAD}\text{-healthy}\} \\
= P &
\end{array}
$$

□

When the angel is given the choice between diverging immediately or behaving as P, then the choice is resolved in favour of P. This is one of the fundamental properties underlying an angelic choice, in that, if possible, the angel can avoid divergence.

The bottom of the lattice is also in direct correspondence with that of the original theory of CSP as Theorems T.5.4.9 and T.5.4.10 establish.

Theorem T.5.4.9 $ac2p(Chaos_{\mathbf{RAD}}) = Chaos_{\mathbf{R}}$

Theorem T.5.4.10 $p2ac(Chaos_{\mathbf{R}}) = Chaos_{\mathbf{RAD}}$

This is a reassuring result in that the bottom of the lattice of CSP also maps into the bottom of the lattice of reactive angelic designs and vice versa.

5.4.4 Choice

The next operator we introduce in this section corresponds to *Chaos* in Roscoe's original presentation [17] of CSP, where it is the most nondeterministic process that does not diverge. In our model, this behaviour is given by $Choice_{\mathbf{RAD}}$.

Definition 120 $Choice_{\mathbf{RAD}} \mathrel{\widehat=} \mathbf{RA} \circ \mathbf{A}(true \vdash ac' \neq \emptyset)$

The precondition is *true* while the postcondition allows any non-empty set of final states ac'. Similarly to the definition of $Chaos_{\mathbf{RAD}}$, and every other reactive angelic design, we observe that the complete behaviour of a process is constrained by **RA** and thus the final states in ac' must observe the properties enforced by **RA**, notably that the traces are suffixes of the initial trace $s.tr$.

If we consider the design $Choice = (true \vdash true)$, then we can obtain a similar process in the theory of CSP by applying **R** as $Choice_{\mathbf{R}} = \mathbf{R}(true \vdash true)$. The application of $p2ac$ to this process yields $Choice_{\mathbf{RAD}}$ as shown in Theorem T.5.4.11.

Theorem T.5.4.11 $p2ac(Choice_{\mathbf{R}}) = Choice_{\mathbf{RAD}}$

Likewise, Theorem T.5.4.12 shows that applying $ac2p$ to $Choice_{\mathbf{RAD}}$ yields exactly the process $Choice_{\mathbf{R}}$ of the CSP model.

Theorem T.5.4.12 $ac2p(Choice_{\mathbf{RAD}}) = Choice_{\mathbf{R}}$

As is discussed later in Section 5.5 the process $Choice_{\mathbf{RAD}}$ plays an important role in the characterisation of the subset of non-divergent processes. The intuition is that for non-divergent processes, the addition of more choices does not change those that are actually available for angelic choice, which are those in the distributed intersection over all permitted values of ac'. Consider the general result of the least upper bound and $Choice_{\mathbf{RAD}}$ in Theorem T.5.4.13.

Theorem T.5.4.13 *Provided P is* **RAD***-healthy,*

$$Choice_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P = \mathbf{RA} \circ \mathbf{A}(true \vdash P^t_f)$$

The precondition is *true*, while the postcondition P^t_f is that of P. In other words, if P could diverge, this is no longer possible in an angelic choice with $Choice_{\mathbf{RAD}}$.

Finally, when considering the greatest lower bound $\sqcap_{\mathbf{RAD}}$ and $\mathit{Choice}_{\mathbf{RAD}}$ we obtain the following result.

Theorem T.5.4.14 *Provided P is* **RAD***-healthy,*

$$\mathit{Choice}_{\mathbf{RAD}} \sqcap_{\mathbf{RAD}} P = \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash ac' \neq \emptyset)$$

Proof.

$$\begin{aligned}
&\mathit{Choice}_{\mathbf{RAD}} \sqcap_{\mathbf{RAD}} P && \{\text{Definition of } \mathit{Choice}_{\mathbf{RAD}}\}\\
&= \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash ac' \neq \emptyset) \sqcap_{\mathbf{RAD}} P && \{\text{Assumption: } P \text{ is } \mathbf{RAD}\text{-healthy}\}\\
&= \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash ac' \neq \emptyset) \sqcap_{\mathbf{RAD}} \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) && \{\text{Theorem T.5.4.4}\}\\
&= \mathbf{RA} \circ \mathbf{A}(\mathit{true} \wedge \neg\, P^f_f \vdash ac' \neq \emptyset \vee P^t_f) && \{\text{Predicate calculus}\}\\
&= \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash ac' \neq \emptyset \vee P^t_f) && \{\text{Definition of } \mathbf{A}, \mathbf{A0} \text{ and predicate calculus}\}\\
&= \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash ac' \neq \emptyset)
\end{aligned}$$

□

The precondition of P is maintained, while the postcondition requires a non-empty set of final states ac'. In other words, if there was a possibility to diverge in P, this is still the case. However, if the precondition $\neg\, P^f_f$ is satisfied then the process behaves nondeterministically like $\mathit{Choice}_{\mathbf{RAD}}$.

5.4.5 Stop

Similarly to CSP, the notion of deadlock is captured by $\mathit{Stop}_{\mathbf{RAD}}$.

Definition 121 $\mathit{Stop}_{\mathbf{RAD}} \mathrel{\widehat=} \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash \circledin^{y}_{ac'}(y.tr = s.tr \wedge y.wait))$

The precondition is *true* while the postcondition requires the process to always be waiting for the environment and keep the trace of events unchanged. In this definition and others to follow, we introduce the following auxiliary predicate.

Definition 122 $\circledin^{y}_{ac'}(P) \mathrel{\widehat=} \exists\, y \bullet y \in ac' \wedge P[\{y\}/ac']$

This definition requires that P admits a state y as a single option for angelic choice. In general, this predicate allows the definition of CSP operators to be lifted into the theory of reactive angelic designs. It can be further extrapolated to other CSP operators, such as external choice.

An angelic choice between a process P and $Stop_{\mathbf{RAD}}$ is, in general, not resolved in favour of either process as shown in Theorem T.5.4.15.

Theorem T.5.4.15 *Provided P is* **RAD***-healthy,*

$$
\begin{array}{l}
Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P \\
= \\
\mathbf{RA} \circ \mathbf{A}(true \vdash (\neg\, P^f_f \Rightarrow P^t_f) \wedge \circledin^y_{ac'}(y.tr = s.tr \wedge y.wait))
\end{array}
$$

Proof.

$$
\begin{array}{lr}
Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P & \{\text{Definition of } Stop_{\mathbf{RAD}}\} \\
= \mathbf{RA} \circ \mathbf{A}(true \vdash \circledin^y_{ac'}(y.tr = s.tr \wedge y.wait)) \sqcup_{\mathbf{RAD}} P & \\
& \{\text{Assumption: } P \text{ is } \mathbf{RAD}\text{-healthy}\} \\
= \left(\begin{array}{l} \mathbf{RA} \circ \mathbf{A}(true \vdash \circledin^y_{ac'}(y.tr = s.tr \wedge y.wait)) \\ \sqcup_{\mathbf{RAD}} \\ \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) \end{array}\right) & \{\text{Theorem T.5.4.1}\} \\
= \mathbf{RA} \circ \mathbf{A}(true \vee \neg\, P^f_f \vdash (\neg\, P^f_f \Rightarrow P^t_f) \wedge \circledin^y_{ac'}(y.tr = s.tr \wedge y.wait)) & \\
& \{\text{Predicate calculus}\} \\
= \mathbf{RA} \circ \mathbf{A}(true \vdash (\neg\, P^f_f \Rightarrow P^t_f) \wedge \circledin^y_{ac'}(y.tr = s.tr \wedge y.wait)) &
\end{array}
$$

□

However, the possibility for divergence is avoided, since the precondition becomes *true*. If P diverges, then the process behaves as $Stop_{\mathbf{RAD}}$, otherwise there is an angelic choice between P or $Stop_{\mathbf{RAD}}$ which corresponds to the conjunction of their respective postconditions.

Finally, we can establish that the definition of $Stop_{\mathbf{RAD}}$ is in correspondence with $Stop_{\mathbf{R}}$ of CSP as established by Theorems T.5.4.16 and T.5.4.17.

Theorem T.5.4.16 $p2ac(Stop_{\mathbf{R}}) = Stop_{\mathbf{RAD}}$

Theorem T.5.4.17 $ac2p(Stop_{\mathbf{RAD}}) = Stop_{\mathbf{R}}$

This is a reassuring result that follows our intuition on using the auxiliary predicate $\circledin^y_{ac'}$ to capture the definition of CSP operators in our new model.

5.4.6 Skip

The process that always terminates successfully is defined as $Skip_{\mathbf{RAD}}$.

Definition 123 $Skip_{\mathbf{RAD}} \mathrel{\widehat=} \mathbf{RA} \circ \mathbf{A}(true \vdash \textcircled{\in}_{ac'}^{y}(\neg\, y.wait \land y.tr = s.tr))$

Its precondition is *true* while the postcondition requires that there is a final state in ac' such that the trace of events $s.tr$ is unchanged and that it terminates by requiring the component *wait* to be *false*.

Similarly to the case with $Stop_{\mathbf{RAD}}$, the angelic choice between a process P and $Skip_{\mathbf{RAD}}$ does not resolve in favour of either as Theorem T.5.4.18 shows.

Theorem T.5.4.18 *Provided P is* **RAD***-healthy,*

$$
\begin{array}{l}
Skip_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P \\
= \\
\mathbf{RA} \circ \mathbf{A}(true \vdash \textcircled{\in}_{ac'}^{y}(\neg\, y.wait \land y.tr = s.tr)) \land (\neg\, P_f^f \Rightarrow P_t^t))
\end{array}
$$

However, the possibility for any divergence in P is avoided. If P diverges, then the angelic choice behaves as $Skip_{\mathbf{RAD}}$, otherwise the behaviour is given by the conjunction of the postconditions of P and $Skip_{\mathbf{RAD}}$. We consider in Example 32 an angelic choice between terminating and deadlocking.

Example 32

$$
\begin{array}{ll}
Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} Skip_{\mathbf{RAD}} & \{\text{Definition of } Stop_{\mathbf{RAD}} \text{ and } Skip_{\mathbf{RAD}}\} \\
= \left(\begin{array}{l}
\mathbf{RA} \circ \mathbf{A}(true \vdash \textcircled{\in}_{ac'}^{y}(y.tr = s.tr \land y.wait)) \\
\sqcup_{\mathbf{RAD}} \\
\mathbf{RA} \circ \mathbf{A}(true \vdash \textcircled{\in}_{ac'}^{y}(\neg\, y.wait \land y.tr = s.tr))
\end{array}\right) & \{\text{Theorem T.5.4.1}\} \\
= \mathbf{RA} \circ \mathbf{A}\left(\begin{array}{l}
true \lor true \\
\vdash \\
\left(\begin{array}{l}
(true \Rightarrow \textcircled{\in}_{ac'}^{y}(y.tr = s.tr \land y.wait)) \\
\land \\
(true \Rightarrow \textcircled{\in}_{ac'}^{y}(\neg\, y.wait \land y.tr = s.tr))
\end{array}\right)
\end{array}\right) & \\
& \{\text{Predicate calculus}\} \\
= \mathbf{RA} \circ \mathbf{A}(true \vdash \textcircled{\in}_{ac'}^{y}(y.tr = s.tr \land y.wait) \land \textcircled{\in}_{ac'}^{y}(\neg\, y.wait \land y.tr = s.tr)) &
\end{array}
$$

In this case, the choice is not resolved by either process. If we map this example into the original theory of CSP, then we obtain the top $\top_{\mathbf{R}}$ of that lattice, defined by $\top_{\mathbf{R}} = \mathbf{R}(true \vdash false)$, as Lemma L.5.4.2 establishes.

Lemma L.5.4.2 $ac2p(Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} Skip_{\mathbf{RAD}}) = \top_{\mathbf{R}}$

This is because the result of mapping $Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$ through $ac2p$ insists on both waiting for an interaction and terminating. Likewise, if we map $\top_{\mathbf{R}}$ through $p2ac$, the top of the lattice of reactive angelic designs is obtained. Thus, this is an instance of the general strengthening indicated by Theorem T.5.4.3. Although the miraculous process $\top_{\mathbf{R}}$ is not part of the standard CSP semantics [17, 18] it plays an important role, for example, in the characterisation of deadline operators in the context of timed versions of process calculi [75–78].

Finally, the definition of $Skip_{\mathbf{RAD}}$ can be be related with the original $Skip_{\mathbf{R}}$ process of CSP by applying $p2ac$ and $p2ac$ as established by Theorems T.5.4.19 and T.5.4.20.

Theorem T.5.4.19 $p2ac(Skip_{\mathbf{R}}) = Skip_{\mathbf{RAD}}$

Theorem T.5.4.20 $ac2p(Skip_{\mathbf{RAD}}) = Skip_{\mathbf{R}}$

In other words, as expected the two processes are in correspondence.

5.4.7 Sequential Composition

The definition of sequential composition is exactly $;_{\mathcal{D}ac}$ from the theory of angelic designs, which is itself layered upon $;_{\mathcal{A}}$. When considering reactive angelic designs, we obtain the following closure result.

Theorem T.5.4.21 *Provided P and Q are reactive angelic designs,*

$$\begin{array}{l}
P \;;_{\mathcal{D}ac}\; Q \\
= \\
\mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l}
\left(\begin{array}{l}
\neg\, (\mathbf{RA1}(P^f_f) \;;_{\mathcal{A}}\; \mathbf{RA1}(true)) \\
\wedge \\
\neg\, (\mathbf{RA1}(P^t_f) \;;_{\mathcal{A}}\; (\neg\, s.wait \wedge \mathbf{RA2} \circ \mathbf{RA1}(Q^f_f)))
\end{array} \right) \\
\vdash \\
\mathbf{RA1}(P^t_f) \;;_{\mathcal{A}}\; (s \in ac' \lhd s.wait \rhd (\mathbf{RA2} \circ \mathbf{RA1}(\neg\, Q^f_f \Rightarrow Q^t_f)))
\end{array} \right)
\end{array}$$

This is a result that resembles that for CSP, apart from the postcondition of the design. When $s.wait$ is $false$, and hence P^t_f has finished its interaction with the environment, the behaviour is given by $\mathbf{RA2} \circ \mathbf{RA1}(\neg\, Q^f_f \Rightarrow Q^t_f)$. In contrast with the result in CSP (Section 2.5.4), this is an implication between the pre and postcondition of Q, instead of its postcondition.

As previously discussed in Section 4.5.2, in the theory of angelic designs, the sequential composition operator also has a similar implication in the postcondition that acts as a filter by eliminating final states of P that fail to satisfy the precondition of Q. For example, we consider the result established in Lemma L.5.4.3.

Lemma L.5.4.3 $(\mathit{Stop}_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) \mathbin{;_{\mathcal{D}ac}} \mathit{Chaos}_{\mathbf{RAD}} = \mathit{Stop}_{\mathbf{RAD}}$

In this case there is an angelic choice between deadlocking and terminating, followed by divergence. The angel avoids the divergence by choosing to deadlock. The precondition of $\mathit{Chaos}_{\mathbf{RAD}}$ is unsatisfiable since it is *false*. Once the preceding process of the sequential composition terminates, that is the component *wait* is *false*, then the composition diverges. However, because the angel can choose the non-terminating process $\mathit{Stop}_{\mathbf{RAD}}$, the divergence can be avoided.

In general, when considering the result of applying the sequential composition of CSP to two processes P and Q mapped through $ac2p$, followed by $p2ac$, a strengthening is obtained as established by the following Theorem T.5.4.22.

Theorem T.5.4.22 *Provided P and Q are reactive angelic designs,*

$$p2ac(ac2p(P) \mathbin{;} ac2p(Q)) \sqsupseteq P \mathbin{;_{\mathcal{D}ac}} Q$$

Proof.

$$\begin{aligned}
&p2ac(ac2p(P) \mathbin{;} ac2p(Q)) && \{\text{Theorem T.G.7.11}\}\\
&= p2ac \circ ac2p(P) \mathbin{;_{\mathcal{D}ac}} p2ac \circ ac2p(Q)\\
& && \{\text{Theorem T.G.7.13 and Lemmas L.C.4.2 and L.C.4.3}\}\\
&\sqsupseteq \mathbf{PBMH}(P) \mathbin{;_{\mathcal{D}ac}} \mathbf{PBMH}(Q)\\
& && \{\text{Assumption: } P \text{ and } Q \text{ are } \mathbf{RAD}\text{-healthy and Theorem T.5.2.21}\}\\
&= P \mathbin{;_{\mathcal{D}ac}} Q
\end{aligned}$$

□

We consider, for example, the case of the processes of Lemma L.5.4.3. As previously discussed in Section 5.4.6, the result of $ac2p(\mathit{Skip}_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} \mathit{Stop}_{\mathbf{RAD}})$ is the top $\top_{\mathbf{R}}$ of the lattice of reactive designs (Lemma L.5.4.2). The result of applying $ac2p(\mathit{Chaos}_{\mathbf{RAD}})$ is the bottom $\mathit{Chaos}_{\mathbf{R}}$ as established by Theorem T.5.4.9. The sequential composition of $\top_{\mathbf{R}}$ followed by $\mathit{Chaos}_{\mathbf{R}}$ is also $\top_{\mathbf{R}}$. Applying $p2ac(\top_{\mathbf{R}})$

yields the top of the lattice of reactive angelic designs $\top_{\mathbf{RAD}} \mathrel{\widehat{=}} \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash \mathit{false})$. This is a trivial refinement of any process, including $\mathit{Stop}_{\mathbf{RAD}}$.

If we strengthen the assumption of Theorem T.5.4.22 by considering the case where both P and Q are, in addition, **A2**-healthy, then an equality is obtained instead as established by Theorem T.5.4.23.

Theorem T.5.4.23 *Provided P and Q are* **RAD***-healthy and* **A2***-healthy,*

$$p2ac(ac2p(P) \ ; \ ac2p(Q)) = P \ ;_{\mathcal{D}ac} \ Q$$

This is because **A2**-healthy processes do not have angelic nondeterminism, and so the result obtained in both models is exactly the same.

When considering two CSP processes P and Q, we also obtain an equality as shown in the following Theorem T.5.4.24.

Theorem T.5.4.24 $ac2p(p2ac(P) \ ;_{\mathcal{D}ac} \ p2ac(Q)) = P \ ; \ Q$

This result confirms the correspondence of sequential composition in both models. In particular, the result of sequentially composing two CSP processes with no angelic nondeterminism can be directly calculated in the new model.

Finally, the sequential composition operator is closed under **A2** for reactive angelic designs as shown in the following Theorem T.5.4.25.

Theorem T.5.4.25 *Provided P and Q are reactive angelic designs and* **A2***-healthy,* $\mathbf{A2}(P \ ;_{\mathcal{D}ac} \ Q) = P \ ;_{\mathcal{D}ac} \ Q$

Therefore, given any two reactive angelic designs P and Q with no angelic nondeterminism, their sequential composition does not introduce any angelic choices. This concludes our discussion of the sequential composition operator.

5.4.8 Prefixing

Having discussed the definition of sequential composition, in this section we introduce the definition of event prefixing, which is similar to that of CSP.

Definition 124

$$a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}} \mathrel{\widehat{=}} \mathbf{RA} \circ \mathbf{A} \left(\mathit{true} \vdash \textcircled{\in}_{ac'}^{y} \left(\begin{array}{l} (y.tr = s.tr \land a \notin y.ref) \\ \lhd y.wait \rhd \\ (y.tr = s.tr \frown \langle a \rangle) \end{array} \right) \right)$$

The precondition is *true*, while the postcondition is split into two cases. When the process is waiting for an interaction from the environment, that is, $y.wait$ is *true*, then a is not in the set of refusals and the trace $s.tr$ is kept unchanged. While in the second case, the process has interacted with the environment, and so the only guarantee is that the event a is part of the final trace $y.tr$.

Like for $Stop_{\mathbf{RAD}}$ and $Skip_{\mathbf{RAD}}$, an angelic choice between a process P and $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$ avoids divergence as established by Theorem T.5.4.26.

Theorem T.5.4.26 *Provided P is a reactive angelic design,*

$$\begin{array}{l} a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P \\ = \\ \mathbf{RA} \circ \mathbf{A} \left(true \vdash \circledin_{ac'}^{y} \left(\begin{array}{l} (y.tr = s.tr \land a \notin y.ref) \\ \lhd y.wait \rhd \\ (y.tr = s.tr \frown \langle a \rangle) \end{array} \right) \land (\neg\, P^f_f \Rightarrow P^t_f) \right) \end{array}$$

The complete behaviour of this process depends on that of P as well. If P diverges, then the process behaves as $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$, otherwise there is an angelic choice between the behaviour of $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$ and P.

Event prefixing in both theories is in exact correspondence as established by the following Theorems T.5.4.27 and T.5.4.28.

Theorem T.5.4.27 $ac2p(a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}) = a \rightarrow_{\mathbf{R}} Skip_{\mathbf{R}}$

Theorem T.5.4.28 $p2ac(a \rightarrow_{\mathbf{R}} Skip_{\mathbf{R}}) = a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$

This is expected since event prefixing, even in the presence of angelic nondeterminism, does not behave differently to prefixing in the original theory of CSP.

In order to illustrate the behaviour of angelic choice we consider the following examples. In Example 33 we have a choice between terminating and deadlocking following event a, sequentially composed with $Chaos_{\mathbf{RAD}}$. In general, the process $a \rightarrow_{\mathbf{RAD}} P$ denotes the compound process $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}} \mathbin{;_{\mathcal{D}ac}} P$, whose result as a reactive angelic design is established by Theorem T.5.4.29.

Theorem T.5.4.29 *Provided P is* **RAD**-*healthy,*

$$\begin{array}{l} a \rightarrow_{\mathbf{RAD}} P \\ = \end{array}$$

$$\mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} \neg\ \exists\, y \bullet y.tr = s.tr \frown \langle a \rangle \wedge \neg\ y.wait \wedge (\mathbf{RA2} \circ \mathbf{RA1}(P^f_f))[y/s] \\ \vdash \\ \exists\, y \bullet \left(\begin{array}{l} (y \in ac' \wedge y.tr = s.tr \wedge a \notin y.ref) \\ \lhd y.wait \rhd \\ (y.tr = s.tr \frown \langle a \rangle \wedge (\mathbf{RA2} \circ \mathbf{RA1}(P^t_f))[y/s]) \end{array} \right) \end{array} \right)$$

The precondition states that it is not the case that once event a occurs the precondition of P fails to be satisfied. While the postcondition considers two cases: when the process is waiting for the environment the trace of events is kept unchanged and event a is not refused; when he process does event a, then the result is that of the postcondition of P with initial state y, where the trace $y.tr$ includes event a.

Example 33

$$\begin{array}{l} ((a \rightarrow_{\mathbf{RAD}} \mathit{Stop}_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}})\ ;_{\mathcal{D}ac}\ \mathit{Chaos}_{\mathbf{RAD}} \\ = \\ a \rightarrow_{\mathbf{RAD}} \mathit{Stop}_{\mathbf{RAD}} \end{array}$$

Proof. Lemma L.G.8.13 □

In the case of Example 33, the angel avoids divergence by choosing non termination by allowing the environment to perform the event a and then deadlocking. In Example 34 there is a choice between terminating or diverging upon performing the event a.

Example 34

$(a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} (a \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}})$

{Definition of prefixing and Theorem T.G.8.8}

$$= \left(\begin{array}{l} \mathbf{RA} \circ \mathbf{A} \left(\mathit{true} \vdash \left(\begin{array}{l} \circledin^{y}_{ac'}(y.wait \land y.tr = s.tr \land a \notin y.ref) \\ \lor \\ \circledin^{y}_{ac'}(\neg\, y.wait \land y.tr = s.tr \frown \langle a \rangle) \end{array} \right) \right) \\ \sqcup \\ \mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} \neg \circledin^{y}_{ac'}(s.tr \frown \langle a \rangle \le y.tr) \\ \vdash \\ \circledin^{y}_{ac'}(y.wait \land y.tr = s.tr \land a \notin y.ref) \end{array} \right) \end{array}\right)$$

{Theorem T.5.4.1 and predicate calculus}

$$= \mathbf{RA} \circ \mathbf{A} \left(\mathit{true} \vdash \left(\begin{array}{l} \left(\begin{array}{l} \circledin^{y}_{ac'}(y.wait \land y.tr = s.tr \land a \notin y.ref) \\ \lor \\ \circledin^{y}_{ac'}(\neg\, y.wait \land y.tr = s.tr \frown \langle a \rangle) \end{array} \right) \\ \land \\ \left(\begin{array}{l} \circledin^{y}_{ac'}(s.tr \frown \langle a \rangle \le y.tr) \\ \lor \\ \circledin^{y}_{ac'}(y.wait \land y.tr = s.tr \land a \notin y.ref) \end{array} \right) \end{array} \right) \right)$$

{Predicate calculus}

$$= \mathbf{RA} \circ \mathbf{A} \left(\mathit{true} \vdash \left(\begin{array}{l} \circledin^{y}_{ac'}(y.wait \land y.tr = s.tr \land a \notin y.ref) \\ \lor \\ \circledin^{y}_{ac'}(\neg\, y.wait \land y.tr = s.tr \frown \langle a \rangle) \end{array} \right) \right)$$

{Definition of prefixing}

$$= a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}$$

The result is a process that following event a can only terminate, and thus avoids divergence. This property illustrates that our angelic choice operator is a counterpart to that of the refinement calculus. It resolves choices to avoid divergence but here we have choices over interactions.

However, if we consider the processes of Example 34 to be prefixes on different events, the result of the angelic choice is rather different as shown in Example 35.

Example 35

$$\begin{array}{l} (a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} (b \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}}) \\ = \\ (a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} (b \rightarrow_{\mathbf{RAD}} \mathit{Choice}_{\mathbf{RAD}}) \end{array}$$

Proof. Lemma L.G.8.9 □

In this case, the possibility of diverging after the event a is avoided by turning $\mathit{Chaos}_{\mathbf{RAD}}$ into $\mathit{Choice}_{\mathbf{RAD}}$. The possibility for engaging in the event a cannot be avoided by the angel, since **RA1** requires that under all circumstances no trace of events may be undone. Ideally for a counterpart to the angelic choice of the refinement calculus, it should be possible to discard any trace of events that lead to divergence. This is the motivation for the theory of angelic processes that we introduce in the following Chapter 6.

5.4.9 External Choice

External choice, which offers the environment the choice over the events initially offered by processes P and Q, is similarly (Section 2.5.4) defined in our theory as follows.

Definition 125

$$\begin{array}{l} P \mathbin{\Box_{\mathbf{RAD}}} Q \\ \widehat{=} \\ \mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} (\neg P^f_f \land \neg Q^f_f) \\ \vdash \\ \circledin^y_{ac'} ((P^t_f \land Q^t_f) \lhd y.tr = s.tr \land y.wait \rhd (P^t_f \lor Q^t_f)) \end{array} \right) \end{array}$$

The precondition is the conjunction of the preconditions of the processes P and Q, while the postcondition is split into two cases. When the process is waiting and the trace of events $s.tr$ is unchanged, then the behaviour is given by the conjunction of both postconditions, otherwise it is given by their disjunction. In other words, before the process performs any event, P and Q must be in agreement. In particular, if there is angelic nondeterminism in either P or Q, there must be an agreement on a single common state in ac'.

Once the process has finished interacting with the environment or performed an event, there is a choice between P and Q. Even if there is angelic nondeterminism in either P or Q, then there is also a requirement for there to be an agreement on a final state, as enforced by the lifting $\circledin^y_{ac'}$. We consider, for example, the following result on the external choice between a reactive angelic design and $\mathit{Stop}_{\mathbf{RAD}}$.

Theorem T.5.4.30 *Provided P is a reactive angelic design,*

$$P \mathbin{\Box_{\mathbf{RAD}}} \mathit{Stop}_{\mathbf{RAD}} = \mathbf{RA} \circ \mathbf{A}(\neg P^f_f \vdash \exists y \bullet (P^t_f)[\{y\}/ac'] \land y \in ac')$$

That is, the angelic nondeterminism of P is collapsed. Unlike in the original theory of CSP, $\mathit{Stop}_{\mathbf{RAD}}$ is not necessarily a unit for external choice. However, when considering the subset of reactive angelic designs corresponding to CSP processes, which are the **A2**-healthy, then $\mathit{Stop}_{\mathbf{RAD}}$ is a unit as expected.

Theorem T.5.4.31 *Provided P is a reactive angelic design and* **A2***-healthy,*

$$P \mathrel{\Box_{\mathbf{RAD}}} \mathit{Stop}_{\mathbf{RAD}} = P$$

Theorem T.5.4.31 follows from the correspondence of the operator in both models, which we discuss below, and the proviso which ensures that there is no angelic nondeterminism in P.

As established by the following Theorem T.5.4.32 the result of mapping two CSP processes P and Q through $p2ac$ and composing them with the external choice operator $\Box_{\mathbf{RAD}}$ of reactive angelic designs, followed by the mapping $ac2p$ in the opposite direction is exactly the same as applying $\Box_{\mathbf{R}}$ to the original processes.

Theorem T.5.4.32 *Provided that P and Q are CSP processes,*

$$ac2p(p2ac(P) \mathrel{\Box_{\mathbf{RAD}}} p2ac(Q)) = P \mathrel{\Box_{\mathbf{R}}} Q$$

However, if we consider the application in the opposite direction in the following Theorem T.5.4.33, the result obtained is not an equality.

Theorem T.5.4.33 *Provided P and Q are reactive angelic designs,*

$$p2ac(ac2p(P) \mathrel{\Box_{\mathbf{R}}} ac2p(Q)) \sqsupseteq P \mathrel{\Box_{\mathbf{RAD}}} Q$$

This establishes that by considering two reactive angelic designs, applying $ac2p$ to both, composing the result with the external choice operator of CSP, and then mapping back through $p2ac$, the result obtained is stronger than the respective composition using $\Box_{\mathbf{RAD}}$. This follows from the fact that, since P and Q can be nondeterministic, and external choice is monotonic with respect to refinement, the application of $ac2p$ may yield stronger processes.

We consider the following Example 36 in the context of Theorem T.5.4.33. Here we have an angelic choice between engaging in an event a or an event b followed by divergence, with $\mathit{Stop}_{\mathbf{RAD}}$ in an external choice.

Example 36

$$
\begin{array}{l}
(a \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}}) \Box_{\mathbf{RAD}} \mathit{Stop}_{\mathbf{RAD}} \\
= \\
\mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l}
\neg\, (\circledin_{ac'}^{y} (s.tr \frown \langle a \rangle \leq y.tr) \land \circledin_{ac'}^{y} (s.tr \frown \langle b \rangle \leq y.tr)) \\
\vdash \\
\circledin_{ac'}^{y} (y.wait \land y.tr = s.tr \land a \notin y.ref \land b \notin y.ref)
\end{array} \right)
\end{array}
$$

Proof. Lemma L.G.8.7 □

The precondition requires that there is not a final state where the trace includes the event a or the event b. The postcondition states that the process is always waiting for the environment, while keeping the trace of events unchanged and not refusing either a or b. The mapping through $ac2p$ of the left-hand side of Example 36 yields a CSP process whose precondition is *true* as shown in the following Example 37.

Example 37

$$
\begin{array}{l}
ac2p(a \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}}) \\
= \\
\mathbf{R}(\mathit{true} \vdash tr' = tr \land wait' \land a \notin ref' \land b \notin ref')
\end{array}
$$

Proof. Lemma L.G.8.1 □

The postcondition, expressed in the theory of reactive designs, is similar to that of Example 36. The mapping of Example 37 through $p2ac$ yields a refinement of the reactive angelic design of Example 36. This is an expected result, which follows from the general result of Theorem T.5.4.33.

If we consider reactive angelic designs that are in addition **A2**-healthy, an equality is obtained as established by Theorem T.5.4.34.

Theorem T.5.4.34 *Provided P and Q are* **RAD***-healthy and* **A2***-healthy,*

$$p2ac(ac2p(P) \Box_{\mathbf{R}} ac2p(Q)) = P \Box_{\mathbf{RAD}} Q$$

Furthermore, the external choice operator is also closed under **A2** as established by Theorem T.5.4.35.

Theorem T.5.4.35 *Provided P and Q are reactive angelic designs and* **A2**-

healthy,

$$\mathbf{A2}(P \Box_{\mathbf{RAD}} Q) = P \Box_{\mathbf{RAD}} Q$$

In other words, the definition of external choice is in correspondence between both models for processes with no angelic nondeterminism.

5.5 Non-divergent Reactive Angelic Designs

As previously discussed in Chapter 1, and as part of our approach to studying the relationship between theories, it is useful to identify the subset of non-divergent reactive angelic designs. These are processes that satisfy the following healthiness condition $\mathbf{ND_{RAD}}$.

Definition 126 $\mathbf{ND_{RAD}}(P) = P \sqcup_{\mathbf{RAD}} \mathit{Choice}_{\mathbf{RAD}}$

This function is defined using the least upper bound of the lattice $\sqcup_{\mathbf{RAD}}$ and the most nondeterministic process $\mathit{Choice}_{\mathbf{RAD}}$ that does not diverge. The intuition underlying $\mathbf{ND_{RAD}}$ is that, for a given process P, increasing the number of final states available for angelic choice, does not actually add any new choices, unless the process P could itself diverge. We consider the following Example 38 where the function $\mathbf{ND_{RAD}}$ is applied to the bottom of the lattice $\mathit{Chaos}_{\mathbf{RAD}}$.

Example 38 $\mathbf{ND_{RAD}}(\mathit{Chaos}_{\mathbf{RAD}}) = \mathit{Choice}_{\mathbf{RAD}}$

Proof. Lemma L.G.6.1 □

The divergence is avoided and the result is the process $\mathit{Choice}_{\mathbf{RAD}}$. If instead we consider a process that is not divergent, such as $\mathit{Skip}_{\mathbf{RAD}}$, the result is as follows.

Example 39 $\mathbf{ND_{RAD}}(a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) = a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}$

Proof. Lemma L.G.6.2 □

The process is a fixed point of $\mathbf{ND_{RAD}}$.

The function $\mathbf{ND_{RAD}}$ is idempotent as shown in the following Theorem T.5.5.1.

Theorem T.5.5.1 $\mathbf{ND_{RAD}} \circ \mathbf{ND_{RAD}}(P) = \mathbf{ND_{RAD}}(P)$

Proof.

$\mathbf{ND_{RAP}} \circ \mathbf{ND_{RAP}}(P)$ {Definition of $\mathbf{ND_{RAP}}$}

$$
\begin{aligned}
&= \mathbf{ND_{RAP}}(P) \sqcup \mathit{Choice}_{\mathbf{RAP}} && \{\text{Definition of } \mathbf{ND_{RAP}}\}\\
&= P \sqcup \mathit{Choice}_{\mathbf{RAP}} \sqcup \mathit{Choice}_{\mathbf{RAP}} && \{\text{Predicate calculus}\}\\
&= P \sqcup \mathit{Choice}_{\mathbf{RAP}} && \{\text{Definition of } \mathbf{ND_{RAP}}\}\\
&= \mathbf{ND_{RAP}}(P)
\end{aligned}
$$

□

More importantly, when considering a reactive angelic design P, Theorem T.5.5.2 establishes that the application of $\mathbf{ND_{RAD}}$ to a reactive angelic design P requires the precondition of the design to be *true*.

Theorem T.5.5.2 *Provided P is* **RAD***-healthy,*

$$\mathbf{ND_{RAD}}(P) = \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash P^t_f)$$

Furthermore, if we consider the fixed points of $\mathbf{ND_{RAD}}$ then we obtain the following result in Theorem T.5.5.3.

Theorem T.5.5.3 *Provided P is* **RAD***-healthy,*

$$\mathbf{ND_{RAD}}(P) = P \Leftrightarrow \forall\, s, ac' \bullet \neg\, P^f_f$$

That is, it must be the case that the precondition $\neg\, P^f_f$ of the reactive angelic design P is satisfied for every possible initial state s and set of final states ac'. These complementary results confirm our intuition about the definition of $\mathbf{ND_{RAD}}$.

5.6 Final Considerations

Based on the underlying principles of the theory of CSP [39, 44] and the model of angelic designs presented in Chapter 4, in this chapter we have presented a model for CSP where both angelic and demonic nondeterminism can be expressed. The approach we have followed consists of a natural extension to the existing CSP model. First we have encoded the observational variables of the theory of reactive processes and enforced all of the healthiness conditions of the original model in this new theory. Similarly to the original theory of CSP we have shown how CSP processes can be specified through reactive angelic designs. We have then established links with the original theory and studied this relationship.

We have established that there is a Galois connection between the theory of reactive angelic designs and CSP. In addition, when considering the subset of processes that are **A2**-healthy, this relationship can be strengthened into a bijection. We have studied the most important operators of the theory and shown that they are in correspondence with their CSP counterparts. Furthermore, we have also proposed a natural way for specifying existing CSP operators in this new theory, including, for example, the external choice operator. While the definition of the external choice operator preserves the semantics of CSP, it is not the only one possible. Indeed, we hypothesize that there are other plausible semantic-preserving definitions for external choice with different algebraic properties. For example, when considering an external choice which includes angelic choices it may be desirable to allow the environment to choose any of those choices.

Finally, a number of examples have been presented to illustrate the role of angelic choice in a theory of CSP. In particular, we have shown that whenever possible, angelic choice avoids divergence. This behaviour is closer in spirit to that of the original choice operator of the refinement calculus than that of any other notion of angelic choice for CSP which we are aware. However, this avoidance still preserves any potential sequence of observable events. Ideally, the counterpart to the angelic choice of the refinement calculus should avoid any divergent behaviour altogether. For example, in the case of Example 35 the angelic choice should be resolved in favour of $a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}$. This is the motivation for the theory of angelic processes which we discuss in the next Chapter 6.

Chapter 6

Angelic Processes

Following from the impossibility for the angel to completely avoid divergent processes in the theory of reactive angelic designs, and based on its underlying principles, in this chapter we present a different approach to characterising CSP processes with angelic nondeterminism. The result is a theory which better accommodates the angelic choice over divergent processes, in that the resulting algebraic properties are closer in spirit to the angelic choice of the refinement calculus. In Section 6.1 we revisit the motivation for this theory and discuss our approach. Section 6.2 introduces the healthiness conditions of the theory and discusses their relationship with the theory of reactive angelic designs. In Section 6.3 we study the relationship between the two models and establish that the subsets of non-divergent processes are isomorphic. In Section 6.4 we present operators of this model and discuss some of their properties as well as their relationship with counterparts in the theory of reactive angelic designs. Finally, the chapter ends with a summary of the results in Section 6.5.

6.1 Introduction

As previously discussed in Chapter 5, in the theory of reactive angelic designs, healthy processes, as required by **RA1**, must never undo the history of events. For example, the definition of $Chaos_{\mathbf{RAD}}$, which diverges immediately, guarantees that there is always a final state in ac' where the trace of events is a suffix of the initial trace $s.tr$. This behaviour is as expected for a theory of processes.

Since angelic choice is defined as the least upper bound, and $Chaos_{\mathbf{RAD}}$ is the bottom of the lattice of reactive angelic designs, it follows that immediate divergence is avoided, if possible, by the angel. However, once there is the possibility for interacting with the environment, such as in the case of Example 33, the possibility for performing an event followed by divergence cannot be eliminated completely, as doing so would violate **RA1**. This is unlike the angelic choice of the refinement calculus and the theory of angelic designs, where angelic choices leading to divergence are pruned altogether.

In this chapter we propose a theory like **RAD**, but which does not necessarily enforce **RA1** when a process diverges. This is a departure from the norm for a theory of CSP. The main consequence of this approach is that divergent processes have a different semantics to standard CSP. However, the subset of non-divergent processes preserves the existing semantics defined by **RAD**, and by extension, the semantics of non-divergent CSP processes.

6.2 Healthiness Conditions

The alphabet of angelic processes is exactly the same as that of reactive angelic designs. Namely, we have variables ok, ok', s and ac', where a *State* is defined with components tr, ref and $wait$.

As with every UTP theory, we define the healthiness conditions. Since we aim to define a theory like **RAD**, but without necessarily enforcing **RA1**, we focus our attention on the definition of **RAD**, which we reproduce below.

$$\mathbf{RAD}(P) \mathrel{\widehat=} \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{RA3} \circ \mathbf{CSPA1} \circ \mathbf{CSPA2} \circ \mathbf{PBMH}(P)$$

If we simply remove **RA1** from the functional composition, then **A0** is not necessarily enforced any more, and thus successful termination does not guarantee that ac' is not empty. Furthermore, **CSPA1** is also stronger than required, since when in an unstable state, that is $\neg\, ok$, **RA1** should not be enforced. Equally, the identity $I\!I_{\mathbf{RAD}}$ and, therefore, **RA3** also need to be changed, so that divergence no longer requires **RA1**. This leads us to the following healthiness condition **AP**.

Definition 127 $\mathbf{AP}(P) \mathrel{\widehat=} \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A} \circ \mathbf{H1} \circ \mathbf{CSPA2}(P)$

The healthiness condition **RA3** is replaced with $\mathbf{RA3_{AP}}$, which does not require **RA1**. The function **A** is included in the functional composition since it enforces both **A0** and **A1** (itself **PBMH** as previously discussed in Section 4.2.2) as required. The function **CSPA1** is replaced with **H1**, since in an unstable state, that is when $\neg\, ok$ is *true*, **RA1** is no longer enforced. Finally **CSPA2** is enforced like in **RAD**.

The definition of $\mathbf{RA3_{AP}}$ is introduced in the following Section 6.2.1. In Section 6.2.2 the definition of **AP** is explored in more detail. Finally in Section 6.2.3

the subset of non-divergent angelic processes is characterised by another healthiness condition $\mathbf{ND_{AP}}$.

6.2.1 Redefining RA3 as $\mathbf{RA3_{AP}}$

Similarly to the theory of reactive angelic designs, we define a new identity $I\!I_{\mathbf{AP}}$ as follows.

Definition 128 $I\!I_{\mathbf{AP}} \mathrel{\widehat=} \mathbf{H1}(ok' \land s \in ac')$

In contrast with the definition for $I\!I_{\mathbf{RAD}}$, there is no longer a requirement for **RA1** to be enforced when the process is unstable and *ok* is *false*. Instead, the only guarantee in this case is that if the process is stable, and *ok* is *true*, then stability is maintained and the state is kept unchanged, by requiring the initial state s to be in the set of final states ac'.

The definition of $\mathbf{RA3_{AP}}$ is similar to **RA3** except that we use the identity $I\!I_{\mathbf{AP}}$, which does not enforce **RA1**, instead of $I\!I_{\mathbf{RAD}}$.

Definition 129 $\mathbf{RA3_{AP}}(P) \mathrel{\widehat=} I\!I_{\mathbf{AP}} \lhd s.wait \rhd P$

The function $\mathbf{RA3_{AP}}$ is idempotent and monotonic as established by the following Theorems T.6.2.1 and T.6.2.2. Proof of these and other theorems to follow, which are not included explicitly in the body of this thesis, can be found in Appendix H of the extended version [74].

Theorem T.6.2.1 $\mathbf{RA3_{AP}} \circ \mathbf{RA3_{AP}}(P) = \mathbf{RA3_{AP}}(P)$

Theorem T.6.2.2 $P \sqsubseteq Q \Rightarrow \mathbf{RA3_{AP}}(P) \sqsubseteq \mathbf{RA3_{AP}}(Q)$

Furthermore, it distributes through both conjunction and disjunction.

Theorem T.6.2.3 $\mathbf{RA3_{AP}}(P \land Q) = \mathbf{RA3_{AP}}(P) \land \mathbf{RA3_{AP}}(Q)$

Theorem T.6.2.4 $\mathbf{RA3_{AP}}(P \lor Q) = \mathbf{RA3_{AP}}(P) \lor \mathbf{RA3_{AP}}(Q)$

Since $\mathbf{RA3_{AP}}$ is idempotent and distributes through both conjunction and disjunction, conjunction and disjunction are closed under $\mathbf{RA3_{AP}}$. More importantly, the operator $;_{\mathcal{A}}$ is closed under $\mathbf{RA3_{AP}}$.

Theorem T.6.2.5 *Provided P and Q are $\mathbf{RA3_{AP}}$-healthy,*

$$\mathbf{RA3_{AP}}(P \mathbin{;_{\mathcal{A}}} Q) = P \mathbin{;_{\mathcal{A}}} Q$$

Finally, $\mathbf{RA3_{AP}}$ commutes with **PBMH**, and **RA2** as established by the following Theorems T.6.2.6 and T.6.2.7

Theorem T.6.2.6 $\mathbf{RA3_{AP}} \circ \mathbf{PBMH}(P) = \mathbf{PBMH} \circ \mathbf{RA3_{AP}}(P)$

Proof.

$$
\begin{array}{lr}
\mathbf{RA3_{AP}} \circ \mathbf{PBMH}(P) & \{\text{Definition of } \mathbf{RA3_{AP}}\} \\
= \mathbf{H1}(ok' \land s \in ac') \lhd s.wait \rhd \mathbf{PBMH}(P) & \{\text{Lemma L.E.4.3}\} \\
= \mathbf{H1}(ok' \land \mathbf{PBMH}(s \in ac')) \lhd s.wait \rhd \mathbf{PBMH}(P) & \{\text{Lemma L.E.4.8}\} \\
= \mathbf{H1} \circ \mathbf{PBMH}(ok' \land s \in ac') \lhd s.wait \rhd \mathbf{PBMH}(P) & \{\text{Theorem T.E.6.2}\} \\
= \mathbf{PBMH} \circ \mathbf{H1}(ok' \land s \in ac') \lhd s.wait \rhd \mathbf{PBMH}(P) & \{\text{Lemma L.E.4.9}\} \\
= \mathbf{PBMH}(\mathbf{H1}(ok' \land s \in ac') \lhd s.wait \rhd P) & \{\text{Definition of } \mathbf{RA3_{AP}}\} \\
= \mathbf{PBMH} \circ \mathbf{RA3_{AP}}(P) &
\end{array}
$$

□

Theorem T.6.2.7 $\mathbf{RA2} \circ \mathbf{RA3_{AP}}(P) = \mathbf{RA3_{AP}} \circ \mathbf{RA2}(P)$

Theorem T.6.2.6 is important in establishing that $\mathbf{RA3_{AP}}$ preserves the upward-closure of **PBMH**. This is established by Lemma L.6.2.1.

Lemma L.6.2.1 $\mathbf{PBMH} \circ \mathbf{RA3_{AP}} \circ \mathbf{PBMH}(P) = \mathbf{RA3_{AP}} \circ \mathbf{PBMH}(P)$

This concludes our discussion of the most important properties of $\mathbf{RA3_{AP}}$.

6.2.2 Angelic Processes (AP)

As already mentioned, the theory of angelic processes is characterised by the functional composition of $\mathbf{RA3_{AP}}$, **RA2**, **A**, **H1** and **CSPA2**. A parallel result to that of the theory of reactive angelic designs (Theorem T.5.2.20) can be obtained as established by the following Theorem T.6.2.8: **AP** processes can also be expressed in terms of a design.

Theorem T.6.2.8 $\mathbf{AP}(P) = \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f)$

Proof.

$$
\begin{array}{lr}
\mathbf{AP}(P) & \{\text{Definition of } \mathbf{AP}\} \\
= \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A} \circ \mathbf{H1} \circ \mathbf{CSPA2}(P) & \{\text{Definition of } \mathbf{CSPA2}\}
\end{array}
$$

$= \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A} \circ \mathbf{H1} \circ \mathbf{H2}(P)$ {Property of designs}

$= \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A}(\neg P^f \vdash P^t)$ {Theorem T.6.2.7}

$= \mathbf{RA2} \circ \mathbf{RA3_{AP}} \circ \mathbf{A}(\neg P^f \vdash P^t)$ {Lemma L.5.2.1}

$= \mathbf{RA2} \circ \mathbf{RA3_{AP}} \circ \mathbf{A}(\neg P^f \vdash P^t)_f$ {Lemma L.C.1.5}

$= \mathbf{RA2} \circ \mathbf{RA3_{AP}} \circ \mathbf{A}((\neg P^f \vdash P^t)_f)$ {Substitution}

$= \mathbf{RA2} \circ \mathbf{RA3_{AP}} \circ \mathbf{A}(\neg P^f_f \vdash P^t_f)$ {Theorem T.6.2.7}

$= \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A}(\neg P^f_f \vdash P^t_f)$

□

This result establishes that an angelic process can also be specified in terms of pre and postconditions, as the image of a design through the functions $\mathbf{RA3_{AP}}$, **RA2** and **A**. Since these functions are all idempotent and monotonic, and the theory of designs is a complete lattice [39], so is the theory of angelic processes.

The original theory of CSP is not a theory of designs, since when *ok* is *false*, **R1** must hold, unlike in the theory of designs, where **H1** requires that no meaningful observations can be made about a design unless it is started, that is, unless *ok* is *true*. Here, since we have dropped **RA1**, in fact the theory we propose is a theory of angelic designs as established by the following Theorem T.6.2.9.

Theorem T.6.2.9

$$\mathbf{AP}(P) = \left(\begin{array}{l} true \lhd s.wait \rhd \neg\, \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f) \\ \vdash \\ s \in ac' \lhd s.wait \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f) \end{array}\right)$$

Proof.

$\mathbf{AP}(P)$ {Theorem T.6.2.8}

$= \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A}(\neg P^f_f \vdash P^t_f)$ {Definition of **A**}

$= \mathbf{RA3_{AP}} \circ \mathbf{RA2}(\neg\, \mathbf{PBMH}(P^f_f) \vdash \mathbf{PBMH}(P^t_f) \land ac' \neq \emptyset)$ {Lemma L.G.2.15}

$= \mathbf{RA3_{AP}}(\neg\, \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f) \vdash \mathbf{RA2}(\mathbf{PBMH}(P^t_f) \land ac' \neq \emptyset))$

{Lemma L.G.2.9}

$= \mathbf{RA3_{AP}}(\neg\, \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f) \vdash \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f))$

{Lemma L.H.1.4}

$$= \left(\begin{array}{l} \mathit{true} \lhd s.\mathit{wait} \rhd \neg\, \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f) \\ \vdash \\ s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f) \end{array} \right)$$

□

The precondition of the design has a conditional on *s.wait*. If the previous process has not terminated interacting with the environment, then this is simply *true*. Otherwise, the original precondition of P must be satisfied, and its negation must be **PBMH** and **RA2**-healthy. We recall that in a non-**H3** design it is actually the negation of the precondition that is established irrespective of termination.

The postcondition of an angelic process also has a conditional on *s.wait*. When the previous process has not terminated its interactions with the environment, then the state is kept unchanged by making sure that the initial state s is in the set of final states ac'. Otherwise, the original postcondition of P holds and must be **PBMH**, **RA2** and **RA1**-healthy.

Although we have dropped **RA1** because the postcondition requires that the set of final states ac' is not empty, and since we enforce **RA2**, this means that **RA1** is enforced in the postcondition (Theorem T.5.2.9). Similarly, if the negation of the precondition imposes any particular set of final states ac', because it must also be **RA2**-healthy, it will also enforce **RA1**.

6.2.3 Non-divergent Angelic Processes ($\mathbf{ND_{AP}}$)

Like in the theory of reactive angelic designs, it is possible to identify the subset of non-divergent angelic processes. These are angelic processes that satisfy the following healthiness condition $\mathbf{ND_{AP}}$. As depicted in Figures 1.1 and 1.6 we show that the subsets of non-divergent processes of the theory of angelic processes and reactive angelic designs are isomorphic. This is a key result that supports our hypothesis on the preservation of the semantics of a subset of CSP.

Definition 130 $\mathbf{ND_{AP}}(P) \mathrel{\widehat=} \mathit{Choice}_{\mathbf{AP}} \sqcup_{\mathbf{AP}} P$

The definition of $\mathbf{ND_{AP}}$ is similar to that of $\mathbf{ND_{RAD}}$, except that here we use the corresponding least upper bound $\sqcup_{\mathbf{AP}}$ and $\mathit{Choice}_{\mathbf{AP}}$ operators of the theory of angelic processes. An angelic process that is non-divergent can be characterised as established by the following Theorem T.6.2.10.

Theorem T.6.2.10 *Provided P is* **AP***-healthy.*

$$\mathit{Choice}_{\mathbf{AP}} \sqcup P = (\mathit{true} \vdash s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f))$$

The precondition is *true*, while the postcondition corresponds to that of P. If P could diverge, then by applying $\mathbf{ND_{AP}}$ this is no longer the case. Since in **H3**-healthy designs the precondition cannot have any free dashed variables, every non-divergent angelic process is also **H3**-healthy. However, not every **H3**-healthy angelic process is necessarily non-divergent. For example, the angelic process $(s.\mathit{wait} \vdash s \in ac')$ is **H3**-healthy, however, it diverges when $s.\mathit{wait}$ is *false*.

6.3 Relationship with Reactive Angelic Designs

As part of our approach for validating the theories we propose, in this section we study the relationship between the theory of angelic processes and reactive angelic designs. Through the links previously discussed in Section 5.3 between the theory of reactive angelic designs and CSP these results also link this new theory to that of CSP.

In Section 6.3.1 we discuss how reactive angelic designs can be mapped into the theory of angelic processes. In Section 6.3.2 we present the reverse mapping between angelic processes and reactive angelic designs. Finally in Section 6.3.3 we show that the subsets of non-divergent processes of both theories are isomorphic.

6.3.1 From Reactive Angelic Designs to Angelic Processes

As already mentioned, in defining **AP** we have dropped **RA1** and thus the theory of angelic processes is a theory of designs that satisfies both **H1** and **H2**. Therefore, a reactive angelic design, can be turned into an angelic process by applying **H1**. Since **CSPA2** is equally enforced in both models, **H2** is also satisfied.

The following result characterises the designs obtained when we apply **H1** to a reactive angelic design **RAD**.

Theorem T.6.3.1

$$\mathbf{H1} \circ \mathbf{RAD}(P) = \left(\begin{array}{l} \mathit{true} \lhd s.\mathit{wait} \rhd \neg\, \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f) \\ \vdash \\ s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P^t_f) \end{array}\right)$$

In words, and considering the general result for angelic processes established by

Theorem T.6.2.9, the postcondition is exactly the same as that of any other angelic process, while the precondition requires, in addition, that P^f_f is **RA1**-healthy. This is a property carried over from the theory of reactive angelic designs, where the negation of the precondition must also be **RA1**-healthy (Lemma L.G.1.23).

We consider the following Example 40 where **H1** is applied to $\mathit{Chaos}_{\mathbf{RAD}}$.

Example 40 $\mathbf{H1}(\mathit{Chaos}_{\mathbf{RAD}}) = (s.\mathit{wait} \lor \neg\, \mathbf{RA1}(\mathit{true}) \vdash s.\mathit{wait} \land s \in ac')$

Proof. Theorem T.6.4.10 □

In this case, if the previous process is still waiting for the environment, and $s.\mathit{wait}$ is *true*, then the state is kept unchanged by requiring s to be in the set of final states ac'. Otherwise, once the process starts, and $s.\mathit{wait}$ is *false*, the design can be restated as $ok \Rightarrow \mathbf{RA1}(\mathit{true})$.

Non-divergent Processes

The application of **H1** to a reactive angelic design that is non-divergent, that is $\mathbf{ND_{RAD}}$-healthy, is established by Lemma L.6.3.1.

Lemma L.6.3.1

$$\begin{array}{l}\mathbf{H1} \circ \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash P^t_f)\\ =\\ (\mathit{true} \vdash s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f))\end{array}$$

The precondition is *true*, similarly to the original reactive angelic design, while the postcondition is that corresponding to the mapping through **H1**, which follows the general result of Theorem T.6.3.1. We consider, for example, the mapping of the process $\mathit{Skip}_{\mathbf{RAD}}$ through **H1**.

Example 41

$$\begin{array}{l}\mathbf{H1}(\mathit{Skip}_{\mathbf{RAD}})\\ =\\ (\mathit{true} \vdash s \in ac' \lhd s.\mathit{wait} \rhd \circledin^{y}_{ac'}(\neg\, y.\mathit{wait} \land y.tr = s.tr))\end{array}$$

Proof. Theorem T.6.4.16 and Lemma L.H.1.9 □

The original postcondition of $\mathit{Skip}_{\mathbf{RAD}}$ is kept intact on the right-handside of the conditional on $s.wait$.

6.3.2 From Angelic Processes to Reactive Angelic Designs

When considering the mapping in the opposite direction, from angelic processes to reactive angelic designs, we must ensure that **RA1** is observed under all circumstances. Therefore, the mapping we need is **RA1** itself. The result of applying **RA1** to an angelic process is established by Theorem T.6.3.2.

Theorem T.6.3.2 $\mathbf{RA1} \circ \mathbf{AP}(P) = \mathbf{RA} \circ \mathbf{A}(\neg P^f_f \vdash P^t_f)$

Proof.

$$
\begin{aligned}
& \mathbf{RA1} \circ \mathbf{AP}(P) && \{\text{Theorem T.6.2.9}\} \\
&= \mathbf{RA1}\left(\begin{array}{l} \mathit{true} \lhd s.\mathit{wait} \rhd \neg \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f) \\ \vdash \\ s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f) \end{array}\right) && \{\text{Lemma L.G.4.1}\} \\
&= \mathbf{RA1} \circ \mathbf{RA3}\left(\begin{array}{l} \neg \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f) \\ \vdash \\ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f) \end{array}\right) && \{\text{Lemma L.G.2.15}\} \\
&= \mathbf{RA1} \circ \mathbf{RA3} \circ \mathbf{RA2}\left(\begin{array}{l} \neg \mathbf{PBMH}(P^f_f) \\ \vdash \\ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f) \end{array}\right) \\
& && \{\text{Theorems T.5.2.10 and T.5.2.16}\} \\
&= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1}\left(\begin{array}{l} \neg \mathbf{PBMH}(P^f_f) \\ \vdash \\ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f) \end{array}\right) && \{\text{Lemma L.G.1.20}\} \\
&= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1}(\neg \mathbf{PBMH}(P^f_f) \vdash \mathbf{PBMH}(P^t_f)) && \{\text{Lemma L.4.2.2}\} \\
&= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(\neg P^f_f \vdash P^t_f) && \{\text{Definition of } \mathbf{RA}\} \\
&= \mathbf{RA} \circ \mathbf{PBMH}(\neg P^f_f \vdash P^t_f) && \{\text{Theorem T.G.1.6}\} \\
&= \mathbf{RA} \circ \mathbf{A}(\neg P^f_f \vdash P^t_f)
\end{aligned}
$$

□

The reactive angelic design ensures that **RA1** applies to the whole angelic design, which by extension also includes the negation of the precondition (Lemma L.G.1.23). We consider the following Example 42, where we apply **RA1** to the design of Example 40.

Example 42 $\mathbf{RA1}(s.wait \lor \neg \mathbf{RA1}(true) \vdash s.wait \land s \in ac') = Chaos_{\mathbf{RAD}}$

Proof. Theorems T.6.4.10 and T.6.4.11. □

This result shows that it is possible to recover the original $Chaos_{\mathbf{RAD}}$ of reactive angelic designs. In fact, as we discuss in the next Section 6.3.3 this is the case for every reactive angelic design.

6.3.3 Galois Connection and Isomorphism

The results of the previous section suggest that every reactive angelic design can be expressed as an angelic process. If we consider the application of **H1** to a reactive angelic design followed by the application of **RA1**, then we obtain the same reactive angelic design as established by the following Theorem T.6.3.3.

Theorem T.6.3.3 $\mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{RAD}(P) = \mathbf{RAD}(P)$

Proof.

$$
\begin{aligned}
&\mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{RAD}(P) && \{\text{Lemma L.H.2.4}\}\\
&= \mathbf{RA1} \circ \mathbf{AP}(\neg \mathbf{RA1} \circ \mathbf{PBMH}(P^f_f) \vdash P^t_f) \\
& && \{\text{Theorem T.6.3.2 and Lemmas L.A.2.5 and L.A.2.6}\}\\
&= \mathbf{RA} \circ \mathbf{A}(\neg \mathbf{RA1} \circ \mathbf{PBMH}(P^f_f) \vdash P^t_f) && \{\text{Theorem T.G.1.6}\}\\
&= \mathbf{RA} \circ \mathbf{PBMH}(\neg \mathbf{RA1} \circ \mathbf{PBMH}(P^f_f) \vdash P^t_f) && \{\text{Lemma L.4.2.2}\}\\
&= \mathbf{RA}(\neg \mathbf{PBMH} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^f_f) \vdash \mathbf{PBMH}(P^t_f)) && \{\text{Theorem T.5.2.5}\}\\
&= \mathbf{RA}(\neg \mathbf{RA1} \circ \mathbf{PBMH}(P^f_f) \vdash \mathbf{PBMH}(P^t_f)) && \{\text{Definition of } \mathbf{RA}\}\\
&= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1}(\neg \mathbf{RA1} \circ \mathbf{PBMH}(P^f_f) \vdash \mathbf{PBMH}(P^t_f)) && \{\text{Lemma L.G.1.23}\}\\
&= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1}(\neg \mathbf{PBMH}(P^f_f) \vdash \mathbf{PBMH}(P^t_f)) && \{\text{Definition of } \mathbf{RA}\}\\
&= \mathbf{RA}(\neg \mathbf{PBMH}(P^f_f) \vdash \mathbf{PBMH}(P^t_f)) && \{\text{Lemma L.4.2.2}\}\\
&= \mathbf{RA} \circ \mathbf{PBMH}(\neg P^f_f \vdash P^t_f) && \{\text{Theorem T.G.1.6}\}\\
&= \mathbf{RA} \circ \mathbf{A}(\neg P^f_f \vdash P^t_f) && \{\text{Theorem T.5.2.20}\}\\
&= \mathbf{RAD}(P)
\end{aligned}
$$

□

This is a fundamental result, which together with the links between the theory of reactive angelic designs and CSP, establishes that every CSP process can also be modelled in this theory, following results on the composition of Galois connections (Theorem 4.2.5 in [39]).

When we consider the mapping in the opposite direction, however, an inequality is obtained, as established by Theorem T.6.3.4.

Theorem T.6.3.4 $\mathbf{H1} \circ \mathbf{RA1} \circ \mathbf{AP}(P) \sqsupseteq \mathbf{AP}(P)$

Proof.

$$
\begin{array}{ll}
\mathbf{H1} \circ \mathbf{RA1} \circ \mathbf{AP}(P) & \{\text{Theorem T.6.3.2}\} \\
= \mathbf{H1} \circ \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) & \{\text{Theorem T.5.2.20 and Lemma L.H.2.4}\} \\
= \mathbf{AP}(\neg\, \mathbf{RA1} \circ \mathbf{PBMH}(P^f_f) \vdash P^t_f) & \\
& \{\text{Lemma L.G.1.21 and strengthen precondition}\} \\
\sqsupseteq \mathbf{AP}(\neg\, \mathbf{PBMH}(P^f_f) \vdash P^t_f) & \{\text{Lemma L.H.1.11}\} \\
= \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A}(\neg\, \mathbf{PBMH}(P^f_f) \vdash P^t_f) & \\
& \{\text{Definition of } \mathbf{A} \text{ and Lemma L.4.2.2 and Theorem T.E.2.1}\} \\
= \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) & \{\text{Lemma L.H.1.11}\} \\
= \mathbf{AP}(P) &
\end{array}
$$

□

This is expected, since reactive angelic designs require **RA1** to be enforced under all circumstances, whereas angelic processes do not necessarily enforce **RA1**. Thus there is a Galois connection between the theory of reactive angelic designs and angelic processes. We consider the following example, where **RA1** and **H1** are applied to the bottom of the lattice $\bot_{\mathbf{AP}} = (s.wait \vdash s \in ac')$ of angelic processes.

Example 43

$$
\begin{array}{l}
\mathbf{H1} \circ \mathbf{RA1}(s.wait \vdash s \in ac') \\
= \\
(s.wait \lor \neg\, \mathbf{RA1}(true) \vdash s.wait \land s \in ac')
\end{array}
$$

Proof. Theorems T.6.4.11 and T.6.4.10. □

The result is exactly the same as the result of applying **H1** to $Chaos_{\mathbf{RAD}}$. This angelic process has a weaker precondition than that of the bottom $\bot_{\mathbf{AP}}$ and is therefore a refinement of $\bot_{\mathbf{AP}}$.

If we restrict our attention to the subset of angelic processes that are non-divergent, then Theorem T.6.3.4 can be strengthened into an equality as the established by the following Theorem T.6.3.5.

Theorem T.6.3.5 $\mathbf{H1} \circ \mathbf{RA1} \circ \mathbf{ND_{AP}} \circ \mathbf{AP}(P) = \mathbf{ND_{AP}} \circ \mathbf{AP}(P)$

Therefore, the subsets of non-divergent processes of the theories of angelic processes and of reactive angelic designs are isomorphic. In addition, if we consider the links between CSP and the theory of reactive angelic designs, and in particular, the subset characterised by **A2** and $\mathbf{ND_{RAD}}$, then we can also ascertain that there is a subset corresponding exactly to non-divergent CSP processes in our model.

6.4 Operators

In this section we present the definition of some important operators of the theory of angelic processes. Similarly to the approach in Section 5.4 we study the relationship between these operators and their counterparts as reactive angelic designs.

6.4.1 Angelic Choice

The angelic choice operator of this theory is also defined through the least upper bound of the lattice of angelic processes, which is conjunction.

Definition 131 $P \sqcup_{\mathbf{AP}} Q \mathrel{\widehat{=}} P \wedge Q$

This operator is closed under **AP** as established by Theorem T.6.4.1.

Theorem T.6.4.1 *Provided P and Q are* **AP***-healthy,*

$$\mathbf{AP}(P \sqcup_{\mathbf{AP}} Q) = P \sqcup_{\mathbf{AP}} Q$$

It is also closed under the subset of non-divergent angelic processes, characterised by $\mathbf{ND_{AP}}$, as established by Theorem T.6.4.2.

Theorem T.6.4.2 *Provided P and Q are* $\mathbf{ND_{AP}}$*-healthy,*

$$\mathbf{ND_{AP}}(P \sqcup_{\mathbf{AP}} Q) = P \sqcup_{\mathbf{AP}} Q$$

The angelic choice of two reactive angelic designs can be equally obtained through the least upper bound of the lattice of angelic processes as established by the following Theorem T.6.4.3.

Theorem T.6.4.3 *Provided P and Q are* **RAD***-healthy,*

$$\mathbf{RA1}(\mathbf{H1}(P) \sqcup_{\mathbf{AP}} \mathbf{H1}(Q)) = P \sqcup_{\mathbf{RAD}} Q$$

Proof.

$$\begin{aligned}
&\mathbf{RA1}(\mathbf{H1}(P) \sqcup \mathbf{H1}(Q)) && \{\text{Definition of } \sqcup\}\\
&= \mathbf{RA1}(\mathbf{H1}(P) \land \mathbf{H1}(Q)) && \{\text{Theorem T.5.2.2}\}\\
&= \mathbf{RA1} \circ \mathbf{H1}(P) \land \mathbf{RA1} \circ \mathbf{H1}(Q) && \{\text{Assumption: } P \text{ and } Q \text{ are } \mathbf{RAD}\text{-healthy}\}\\
&= \mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{RAD}(P) \land \mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{RAD}(Q) && \{\text{Theorem T.6.3.3}\}\\
&= \mathbf{RAD}(P) \land \mathbf{RAD}(Q) && \{\text{Assumption: } P \text{ and } Q \text{ are } \mathbf{RAD}\text{-healthy}\}\\
&= P \land Q && \{\text{Definition of } \sqcup\}\\
&= P \sqcup Q
\end{aligned}$$

□

In words, if we consider two reactive angelic designs P and Q, and after mapping them through the function **H1** we take the least upper bound $\sqcup_{\mathbf{AP}}$, followed by **RA1**, then we obtain the same result as the least upper bound $\sqcup_{\mathbf{RAD}}$ of P and Q. Together with the result of Theorem T.6.4.2 this establishes that the angelic choice operator for the subset of non-divergent processes is in correspondence with that of the theory of reactive angelic designs.

However, when we consider the result in the opposite direction, that is, by considering two angelic processes P and Q mapped through **RA1**, followed by the application of **H1**, then the result is not an equality.

Theorem T.6.4.4 *Provided P and Q are* **AP***-healthy,*

$$\mathbf{H1}(\mathbf{RA1}(P) \sqcup_{\mathbf{RAD}} \mathbf{RA1}(Q)) \sqsupseteq P \sqcup_{\mathbf{AP}} Q$$

This is expected since the theory of angelic processes is less strict with regards to enforcing **RA1**.

6.4.2 Demonic Choice

Like in the theory of reactive angelic designs, demonic choice is also defined using the greatest lower bound, which is disjunction.

Definition 132 $P \sqcap_{\mathbf{AP}} Q \mathrel{\widehat=} P \lor Q$

This operator is closed under **AP** as established by Theorem T.6.4.5, and is also closed under the subset of non-divergent processes as established by Theorem T.6.4.6.

Theorem T.6.4.5 *Provided P and Q are* **AP***-healthy,* $\mathbf{AP}(P \sqcap Q) = P \sqcap Q$.

Theorem T.6.4.6 *Provided P and Q are* $\mathbf{ND_{AP}}$*-healthy,*

$$\mathbf{ND_{AP}}(P \sqcap_{\mathbf{AP}} Q) = P \sqcap_{\mathbf{AP}} Q$$

The demonic choice of two reactive angelic designs P and Q can be equally obtained through the greatest lower bound of the lattice of angelic processes as the following Theorem T.6.4.7 establishes.

Theorem T.6.4.7 *Provided P and Q* **RAD***-healthy,*

$$\mathbf{RA1}(\mathbf{H1}(P) \sqcap_{\mathbf{AP}} \mathbf{H1}(Q)) = P \sqcap_{\mathbf{RAD}} Q$$

Proof.

$$\begin{aligned}
&\mathbf{RA1}(\mathbf{H1}(P) \sqcap_{\mathbf{AP}} \mathbf{H1}(Q)) && \{\text{Definition of } \sqcap_{\mathbf{AP}}\}\\
&= \mathbf{RA1}(\mathbf{H1}(P) \lor \mathbf{H1}(Q)) && \{\text{Theorem T.5.2.3}\}\\
&= \mathbf{RA1} \circ \mathbf{H1}(P) \lor \mathbf{RA1} \circ \mathbf{H1}(Q) && \{\text{Assumption: } P \text{ and } Q \text{ are } \mathbf{RAD}\text{-healthy}\}\\
&= \mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{RAD}(P) \lor \mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{RAD}(Q) && \{\text{Theorem T.6.3.3}\}\\
&= \mathbf{RAD}(P) \lor \mathbf{RAD}(Q) && \{\text{Assumption: } P \text{ and } Q \text{ are } \mathbf{RAD}\text{-healthy}\}\\
&= P \lor Q && \{\text{Definition of } \sqcap_{\mathbf{RAD}}\}\\
&= P \sqcap_{\mathbf{RAD}} Q
\end{aligned}$$

□

If we map P and Q through **H1**, take the greatest lower bound $\sqcap_{\mathbf{AP}}$, and then apply **RA1**, then the same result can be obtained by taking the greatest lower bound of reactive angelic designs $\sqcap_{\mathbf{RAD}}$. With this result, together with the closure of $\sqcap_{\mathbf{AP}}$ under $\mathbf{ND_{AP}}$ (Theorem T.6.4.6) it is possible to ascertain that the demonic choice for non-divergent processes is in correspondence in both models.

In general, the greatest lower bound of the theory of angelic processes cannot be replicated in the theory of reactive angelic designs, as established by the following Theorem T.6.4.8.

Theorem T.6.4.8 *Provided P and Q are* **AP***-healthy,*

$$\mathbf{H1}(\mathbf{RA1}(P) \sqcap_{\mathbf{RAD}} \mathbf{RA1}(Q)) \sqsupseteq P \sqcap_{\mathbf{AP}} Q$$

This inequality is expected, since the model of angelic processes does not necessarily enforce **RA1** under all circumstances, while in the theory of reactive angelic designs this is always the case.

6.4.3 Divergence: Chaos and Chaos of CSP

In our theory of angelic processes, the bottom of the lattice is defined by $Chaos_{\mathbf{AP}}$, whose definition can be given in terms of the bottom of designs as follows.

Definition 133 $Chaos_{\mathbf{AP}} \mathrel{\widehat{=}} \mathbf{AP}(false \vdash true)$

This result can be expanded into a design as established by Lemma L.6.4.1.

Lemma L.6.4.1 $Chaos_{\mathbf{AP}} = (s.wait \vdash s \in ac')$

The precondition requires the component *wait* of the initial state s to be *true*, while the postcondition keeps the state unchanged by requiring s to be in the set of final states ac'. In other words, as long as the environment is waiting for an interaction, the state is kept unchanged. However, once the environment is no longer waiting, then $Chaos_{\mathbf{AP}}$ diverges and the behaviour is described by *true*. $Chaos_{\mathbf{AP}}$ is a unit for angelic choice as established by Theorem T.6.4.9.

Theorem T.6.4.9 *Provided P is* **AP***-healthy,* $P \sqcup_{\mathbf{AP}} Chaos_{\mathbf{AP}} = P$

In other words, if possible, the angel can avoid divergence.

In this theory, the process that corresponds to $Chaos_{\mathbf{RAD}}$ is $ChaosCSP_{\mathbf{AP}}$, which is defined through a design as follows.

Definition 134 $ChaosCSP_{\mathbf{AP}} \mathrel{\widehat{=}} \mathbf{AP}(\neg\, \mathbf{RA1}(true) \vdash true)$

Instead of *false*, the precondition requires $\neg\, \mathbf{RA1}(true)$. As already discussed, it is the negation of the precondition of a design that gives the behaviour in case of possible non-termination. This design can be expanded as established by the following Lemma L.6.4.2.

Lemma L.6.4.2 $ChaosCSP_{\mathbf{AP}} = (s.wait \lor \neg\, \mathbf{RA1}(true) \vdash s.wait \land s \in ac')$

In words, when the environment is waiting for an interaction, the state is kept unchanged. Otherwise, the design diverges, but still requires that **RA1** holds, unlike $Chaos_{\mathbf{AP}}$. This corresponds exactly to the mapping of $Chaos_{\mathbf{RAD}}$ through the linking function **H1** as established by Theorem T.6.4.10.

Theorem T.6.4.10 $\mathbf{H1}(Chaos_{\mathbf{RAD}}) = ChaosCSP_{\mathbf{AP}}$

Similarly, if we map $ChaosCSP_{\mathbf{AP}}$ through **RA1** we obtain the bottom of the lattice of reactive angelic designs $Chaos_{\mathbf{RAD}}$.

Theorem T.6.4.11 $\mathbf{RA1}(ChaosCSP_{\mathbf{AP}}) = Chaos_{\mathbf{RAD}}$

This follows from the general result of Theorem T.6.3.3.

6.4.4 Choice

The most nondeterministic process that does not diverge is defined as $Choice_{\mathbf{AP}}$ and can be defined through a design as follows.

Definition 135 $Choice_{\mathbf{AP}} \mathrel{\widehat{=}} \mathbf{AP}(true \vdash ac' \neq \emptyset)$

The precondition is *true*, while any set of final states ac' is acceptable. The resulting behaviour, constrained by **AP**, is established through the following Lemma L.6.4.3.

Lemma L.6.4.3 $\mathbf{AP}(true \vdash ac' \neq \emptyset) = (true \vdash s \in ac' \lhd s.wait \rhd \mathbf{RA1}(true))$

The precondition is also *true*, while the postcondition has a conditional on $s.wait$. As is the case for every angelic process, when the process is waiting for the environment, and $s.wait$ is *true*, the state is kept unchanged. Otherwise, the only guarantee is that there is a final state in ac' satisfying **RA1**.

As previously discussed, the operator $Choice_{\mathbf{AP}}$ is used to characterise algebraically the subset of angelic processes that are non-divergent. Therefore, it is closed under $\mathbf{ND_{AP}}$, and by definition, equally closed under **AP**. It is the counterpart to $Choice_{\mathbf{RAD}}$ of the theory of reactive angelic designs as established by the following Theorems T.6.4.12 and T.6.4.13.

Theorem T.6.4.12 $\mathbf{H1}(Choice_{\mathbf{RAD}}) = Choice_{\mathbf{AP}}$

Theorem T.6.4.13 $\mathbf{RA1}(Choice_{\mathbf{AP}}) = Choice_{\mathbf{RAD}}$

The result of Theorem T.6.4.13 follows directly from Theorem T.6.4.12 and the general result of Theorem T.6.3.3.

6.4.5 Stop

In this theory, deadlock is modelled by $\mathit{Stop}_{\mathbf{AP}}$, whose definition is similar to that of the reactive angelic design $\mathit{Stop}_{\mathbf{RAD}}$.

Definition 136 $\mathit{Stop}_{\mathbf{AP}} \mathrel{\widehat=} \mathbf{AP}(\mathit{true} \vdash \circledin_{ac'}^{y}(y.tr = s.tr \land y.\mathit{wait}))$

The precondition is *true*, while the postcondition states that there is a final state y in the set of final states ac' where the trace is kept unchanged and the process is always waiting for the environment. This definition can be directly obtained by applying **H1** to $\mathit{Stop}_{\mathbf{RAD}}$ as established by Theorem T.6.4.14.

Theorem T.6.4.14 $\mathbf{H1}(\mathit{Stop}_{\mathbf{RAD}}) = \mathit{Stop}_{\mathbf{AP}}$

Similarly, $\mathit{Stop}_{\mathbf{RAD}}$ can be obtained by applying **RA1** to $\mathit{Stop}_{\mathbf{AP}}$ as established by the following Theorem T.6.4.15.

Theorem T.6.4.15 $\mathbf{RA1}(\mathit{Stop}_{\mathbf{AP}}) = \mathit{Stop}_{\mathbf{RAD}}$

This is expected since $\mathit{Stop}_{\mathbf{AP}}$ is a non-divergent angelic process, and so it is in direct correspondence with a reactive angelic design.

6.4.6 Skip

The process that always terminates successfully is characterised by $\mathit{Skip}_{\mathbf{AP}}$. Its definition as a design is presented below.

Definition 137 $\mathit{Skip}_{\mathbf{AP}} \mathrel{\widehat=} \mathbf{AP}(\mathit{true} \vdash \circledin_{ac'}^{y}(y.tr = s.tr \land \neg\, y.\mathit{wait}))$

The precondition is *true*, while the postcondition states that there is a final state y in ac' where the trace of events is kept unchanged and the component *wait* is *false*. $\mathit{Skip}_{\mathbf{AP}}$ is in correspondence with $\mathit{Skip}_{\mathbf{RAD}}$ of the theory of reactive angelic designs as established by the following Theorems T.6.4.16 and T.6.4.17.

Theorem T.6.4.16 $\mathbf{H1}(\mathit{Skip}_{\mathbf{RAD}}) = \mathit{Skip}_{\mathbf{AP}}$

Theorem T.6.4.17 $\mathbf{RA1}(\mathit{Skip}_{\mathbf{AP}}) = \mathit{Skip}_{\mathbf{RAD}}$

These results are expected since $\mathit{Skip}_{\mathbf{AP}}$ and $\mathit{Skip}_{\mathbf{RAD}}$ are both non-divergent processes.

6.4.7 Sequential Composition

In our theory of angelic processes, the definition of sequential composition is also $;_{\mathcal{D}ac}$ from the theory of angelic designs. When we consider two angelic processes P and Q, the following closure result is obtained.

Theorem T.6.4.18 *Provided P and Q are* **AP***-healthy,*

$$\begin{array}{l} P \; ;_{\mathcal{D}ac} \; Q \\ = \\ \mathbf{AP}\left(\begin{array}{l} \neg\,(P^f_f \; ;_{\mathcal{A}} \; true) \land \neg\,(\mathbf{RA1}(P^t_f) \; ;_{\mathcal{A}} \; (\neg\, s.wait \land \mathbf{RA2}(Q^f_f))) \\ \vdash \\ \mathbf{RA1}(P^t_f) \; ;_{\mathcal{A}} \; (s \in ac' \lhd s.wait \rhd \mathbf{RA2}(\neg\, Q^f_f \Rightarrow \mathbf{RA1}(Q^t_f))) \end{array}\right) \end{array}$$

This result is similar to that obtained in the theory of reactive angelic designs (Theorem T.5.4.21). The differences are in that **RA1** is no longer applied to P^f_f and Q^f_f, the negation of the preconditions of P and Q, respectively. If P may diverge, then the result is the bottom of the lattice $Chaos_{\mathbf{AP}}$. Similarly, since the precondition of Q does not need to observe **RA1**, if Q diverges, then the sequential composition also behaves like $Chaos_{\mathbf{AP}}$ once P has finished interacting with the environment.

Thus, in our theory of angelic processes, $;_{\mathcal{D}ac}$ is a sequential composition operator that behaves differently to that of CSP, in that it can back propagate the divergence of Q through P, irrespective of other interactions that happen in P, as long as, eventually the environment may terminate its interactions with P and behave as Q. We consider the following example Example 44.

Example 44 $(Stop_{\mathbf{AP}} \sqcup_{\mathbf{AP}} Skip_{\mathbf{AP}}) \; ;_{\mathcal{D}ac} \; Chaos_{\mathbf{AP}} = Stop_{\mathbf{AP}}$

Proof. Lemma L.H.3.6. □

In this case, the angel avoids the divergence of $Chaos_{\mathbf{AP}}$ by resolving the choice in favour of deadlock. This is similar to the behaviour in the theory of reactive angelic designs, since $Stop_{\mathbf{AP}}$ can prevent $Chaos_{\mathbf{AP}}$ from ever being reached.

In general, the result of applying **RA1** to the sequential composition of two reactive angelic designs P and Q mapped through **H1** is not equivalent to sequentially composing these two processes in the theory of reactive angelic designs as established by Theorem T.6.4.19.

Theorem T.6.4.19 *Provided P and Q are reactive angelic designs,*

$$\mathbf{RA1}(\mathbf{H1}(P) \mathbin{;_{\mathcal{D}ac}} \mathbf{H1}(Q)) \sqsubseteq P \mathbin{;_{\mathcal{D}ac}} Q$$

This is because the possibility to diverge in P, in the theory of angelic processes, can lead to immediate divergence, as already discussed. Thus, when the sequential composition of $\mathbf{H1}(P)$ and $\mathbf{H1}(Q)$ is mapped back through $\mathbf{RA1}$, there is a weakening.

Similarly, the reverse mapping through $\mathbf{H1}$ of the sequential composition of two angelic processes P and Q mapped through $\mathbf{RA1}$ is also an inequality as established by Theorem T.6.4.20.

Theorem T.6.4.20 *Provided P and Q are* $\mathbf{AP}$*-healthy,*

$$\mathbf{H1}(\mathbf{RA1}(P) \mathbin{;_{\mathcal{D}ac}} \mathbf{RA1}(Q)) \sqsupseteq P \mathbin{;_{\mathcal{D}ac}} Q$$

This is due to the fact that the notion of divergence is different. In a sequential composition of P and the bottom of the lattice $Chaos_{\mathbf{AP}}$, the result is also $Chaos_{\mathbf{AP}}$. If we map $Chaos_{\mathbf{AP}}$ through $\mathbf{RA1}$ the result is $Chaos_{\mathbf{RAD}}$ (Theorem T.6.4.11), which when sequentially composed after the process $\mathbf{RA1}(P)$, still preserves the history of events in P, whereas the corresponding process in the theory of angelic processes does not. Hence, there is a strengthening.

However, if we consider the subset of non-divergent reactive angelic designs, characterised by $\mathbf{ND_{RAD}}$, then Theorem T.6.4.19 can be strengthened into an equality as established by Theorem T.6.4.21.

Theorem T.6.4.21 *Provided P and Q are reactive angelic designs and* $\mathbf{ND_{RAD}}$*-healthy,*

$$\mathbf{RA1}(\mathbf{H1}(P) \mathbin{;_{\mathcal{D}ac}} \mathbf{H1}(Q)) = P \mathbin{;_{\mathcal{D}ac}} Q$$

In addition, the operator $;_{\mathcal{D}ac}$ is closed under $\mathbf{ND_{AP}}$ as established by the following Theorem T.6.4.22.

Theorem T.6.4.22 *Provided P and Q are angelic processes and* $\mathbf{ND_{AP}}$*-healthy,*

$$\mathbf{ND_{AP}}(P \mathbin{;_{\mathcal{D}ac}} Q) = P \mathbin{;_{\mathcal{D}ac}} Q$$

Thus, as long as P and Q are non-divergent, $;_{\mathcal{D}ac}$ behaves exactly in the same way as in the theory of reactive angelic designs. By extension, this also applies to the subset of **A2** processes, which do not exhibit angelic nondeterminism. Therefore, it also applies to the subset of non-divergent CSP processes.

6.4.8 Prefixing

Similarly to the previous non-divergent processes, event prefixing has a definition similar to that of $a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}$ in the theory of reactive angelic designs.

Definition 138

$$a \rightarrow_{\mathbf{AP}} \mathit{Skip}_{\mathbf{AP}} \mathrel{\widehat=} \mathbf{AP}\left(\mathit{true} \vdash \circledin^{y}_{ac'} \left(\begin{array}{l} (y.tr = s.tr \land a \notin y.\mathit{ref}) \\ \lhd y.\mathit{wait} \rhd \\ (y.tr = s.tr \frown \langle a \rangle) \end{array}\right)\right)$$

The precondition is *true*, while the postcondition is exactly like that of the corresponding reactive angelic design $a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}$ (Section 5.4.8).

The event prefixing of both theories is in correspondence as established by the following Lemmas L.6.4.4 and L.6.4.5.

Lemma L.6.4.4 $\mathbf{H1}(a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) = a \rightarrow_{\mathbf{AP}} \mathit{Skip}_{\mathbf{AP}}$

Lemma L.6.4.5 $\mathbf{RA1}(a \rightarrow_{\mathbf{AP}} \mathit{Skip}_{\mathbf{AP}}) = a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}$

Similarly to the theory of reactive angelic designs, in general, the process $a \rightarrow_{\mathbf{AP}} P$ denotes the compound process $a \rightarrow_{\mathbf{AP}} \mathit{Skip}_{\mathbf{AP}} \mathbin{;_{\mathcal{D}ac}} P$, whose result as an angelic process is established by Theorem T.6.4.23.

Theorem T.6.4.23 *Provided P is* **AP***-healthy,*

$$\begin{array}{l} a \rightarrow P \\ = \\ \mathbf{AP}\left(\begin{array}{l} \neg\, (\exists\, y \bullet \neg\, y.\mathit{wait} \land y.tr = s.tr \frown \langle a \rangle \land (\mathbf{RA2} \circ \mathbf{PBMH}(P^f_f))[y/s]) \\ \vdash \\ \exists\, y \bullet \left(\begin{array}{l} (y.tr = s.tr \land a \notin y.\mathit{ref} \land y \in ac') \\ \lhd y.\mathit{wait} \rhd \\ (y.tr = s.tr \frown \langle a \rangle \land \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f)[y/s]) \end{array}\right) \end{array}\right) \end{array}$$

This result is a counterpart to that of Theorem T.5.4.29. The difference lies in the precondition of the design: the negation of the precondition of P is not necessarily required to observe **RA1**. In addition, the application of **PBMH** can be simplified by taking into account that every **AP**-healthy process is also **PBMH**-healthy.

In order to illustrate the behaviour of prefixing in the presence of divergence, we consider the following Example 45.

Example 45 $a \rightarrow_{\mathbf{AP}} Chaos_{\mathbf{AP}} = Chaos_{\mathbf{AP}}$

Proof. Lemma L.H.3.8. □

In this case, the potential for divergence after performing event a leads to immediate divergence. If instead we sequentially compose prefixing on the event a with $ChaosCSP_{\mathbf{AP}}$, the behaviour is different as established by Lemma L.6.4.6.

Lemma L.6.4.6

$$\begin{aligned}
& a \rightarrow_{\mathbf{AP}} ChaosCSP_{\mathbf{AP}} \\
& = \\
& \mathbf{AP}(\neg \circledcirc_{ac'}^{y}(s.tr \frown \langle a \rangle \leq y.tr) \vdash \circledcirc_{ac'}^{y}(y.wait \land y.tr = s.tr \land a \notin y.ref))
\end{aligned}$$

This result mirrors the behaviour of $a \rightarrow_{\mathbf{RAD}} Chaos_{\mathbf{RAD}}$ of the theory of reactive angelic designs (Theorem T.G.8.8).

We revisit Example 35, by restating it in the theory of angelic processes as Example 46.

Example 46 $a \rightarrow_{\mathbf{AP}} Chaos_{\mathbf{AP}} \sqcup_{\mathbf{AP}} b \rightarrow_{\mathbf{AP}} Skip_{\mathbf{AP}} = b \rightarrow_{\mathbf{AP}} Skip_{\mathbf{AP}}$

Proof. Lemma L.H.3.8 and Theorem T.6.4.9. □

Now, in the context of the theory of angelic processes, the possibility for divergence is avoided altogether, and the result is the prefixing on the event b. As required, the angel can avoid processes that may lead to divergence altogether, a property that is not observed in the theory of reactive angelic designs.

6.5 Final Considerations

The motivation for the theory of angelic processes stems from the limitations of the angelic choice of reactive angelic designs, which is unable to avoid divergence completely, as in the case of Example 35. The possibility to avoid divergence is a desirable property that is much closer in spirit to the refinement calculus. In order to tackle this aspect, we have pursued a theory that drops **RA1**, and thus, is able to undo the history of events if necessary. The result is a theory of angelic designs, whose pre and postconditions observe a subset of the healthiness conditions of the theory of reactive angelic designs, such as **RA2** and **PBMH**.

We have studied the relationship between the theories and established that there is a Galois connection between them. As illustrated in Figures 1.1 and 1.6, reactive angelic designs can be mapped into this theory by turning them into designs, through **H1**, while angelic processes can be mapped in the opposite direction by applying **RA1**. We have found that the subset of non-divergent angelic processes, characterised by $\mathbf{ND_{AP}}$, is isomorphic to the subset of non-divergent reactive angelic designs characterised by $\mathbf{ND_{RAD}}$. Together with the linking results from Chapter 5 between **RAD** and CSP, this implies that the subset of non-divergent CSP processes has exactly the same semantics in this model.

Since every reactive angelic design can be mapped into the model of angelic processes and back, we can ascertain that there is a subset in **AP** that characterises all reactive angelic designs. This is essentially a subset whose negated preconditions satisfy **RA1**. If we consider the subset of **RAD** that is isomorphic to CSP (characterised by **A2**), it is possible to postulate that there is also a subset in **AP** characterising every CSP process.

However, since we allow the history of events to be undone when *ok* is *false*, not all operators are necessarily in correspondence, as is the case, for example, with sequential composition. A parallel can be drawn in the theory of CSP, where this problem corresponds to the possibility of characterising CSP processes as designs, rather than reactive designs. The difference between these two can clearly be seen from the fact that **H1** and **R1** are not commutative. While such a theory of designs could possibly characterise CSP processes, this would mean that the definition of the operators would need to change in order to accommodate such a model, thus negating the benefits of unification in the UTP.

Chapter 7

Conclusions

In this chapter we conclude this thesis by summarizing our contributions. In addition, we discuss lines for future work.

7.1 Contributions

As previously discussed, angelic nondeterminism has been used in a variety of different contexts, such as in problems whose solutions may involve a combination of search and backtracking. This is the case, for example, when modelling game-like scenarios, theorem-proving tactics, or constraint satisfaction problems. In general, angelic nondeterminism enables a great degree of abstraction in the context of formal models and specifications. Its characterisation in the context of process algebras, such as CSP, however, has to the best of our knowledge, been elusive. The existing approaches have either considered notions of angelic nondeterminism [18] different from that of refinement calculi, or different CSP semantics [43].

Angelic nondeterminism has traditionally been studied in the context of theories of correctness for sequential computations, such as in the refinement calculus [29, 31, 32], where it is characterised as the least upper bound of the lattice of monotonic predicate transformers. Isomorphic models include Rewitzky's theory of binary multirelations [35], which is the foundation of our approach.

Our first contribution in Chapter 3 is an extended model of binary multirelations that caters for possibly non-terminating computations. This model provides a complementary view of our theory of angelic designs, which allows for preconditions that refer to the later or final values of a computation, as required for characterising CSP processes. Unlike purely sequential computations, in a reactive system, there is a rich sequence of interactions, whose history cannot be undone even in the case of divergence, such as in the case of the process $a \rightarrow \mathit{Chaos}$.

Our work is based on the UTP of Hoare and He [39], a relational framework suitable for characterising different programming paradigms. As such, our results are applicable not only to CSP, but also to any other algebra of (state-rich) reactive systems whose semantics is or can be described in the UTP. Our theories are complete lattices and angelic and demonic choice are modelled as the meet and join, respectively. Each and every one of them is appropriately justified by studying its relationship with the established theories, which is central to the unification of theories in the UTP.

Our theory of angelic designs generalises the theory of Cavalcanti et al. [38] to include the variables *ok* and ok' for capturing termination. It caters for non-**H3** designs, as required for specifying CSP processes like *Chaos*, whose precondition, as a reactive design, refers to the after value of the trace of events. Its relationship with the theories of [38] and of extended binary multirelations sheds light on the definition of less trivial operators. Sequential composition, for instance, due to the use of non-homogeneous relations, is not relational composition like in other UTP theories. Apart from the relational characterisation of *ok* and ok', this suggests itself as a form of a Kleisli composition through the results established between the theory of angelic designs and binary multirelations, and its respective characterisation as the category of multirelations or multifunctions [79]. The result obtained for the sequential composition of angelic designs is pleasing, in that, using the operators $;_{\mathcal{D}ac}$ and $;_{\mathcal{A}}$, we have a definition similar to that in the original theory of designs.

The theory of reactive angelic designs considers the encoding of the observational variables *ref*, *tr* and *wait* of CSP as state components. This enables angelic choice over the value of these components in final or after states. Rather pleasingly, like the processes in the theory of CSP [39, 44], every **RAD** process can be specified in terms of designs, that is, pre and postcondition pairs, but now we use angelic designs. Unlike other attempts [18, 43], our approach consists of a natural extension of the concept of angelic nondeterminism from a theory of sequential correctness to a model of processes. This approach is strongly justified by the relationship between the theories, their isomorphic subsets, and by the correspondence of operators in both theories. We have a theory of CSP that preserves its existing semantics and that can be used to describe both angelic and demonic nondeterminism.

An important result obtained in the theory of reactive angelic designs pertains to the capability of the angel to avoid divergence. However, unlike in a theory of correctness for sequential computations, the history of interactions, as recorded by traces, cannot simply be undone, even in the presence of divergence. The healthiness condition **RA1**, the counterpart to **R1** of CSP in the model of reactive angelic designs, ensures that this is the case under all circumstances.

Our final theory does not adopt **RA1** as a healthiness condition and as such allows the angel to discard traces of events leading to divergence. It is a theory of angelic designs: a complete lattice whose bottom $Chaos_{\mathbf{AP}}$ is not the $Chaos$ of CSP. It is a process that once executed behaves arbitrarily, and may even undo the history of interactions. More importantly, in an angelic choice involving other interactions, it becomes possible for the angel to undo the history of events, if necessary, and avoid divergence. This is a property much closer in spirit to the angelic choice of the refinement calculus.

As a consequence not every operator preserves the original semantics of CSP. That is the case of the sequential composition operator, for instance. However, the subset of non-divergent angelic processes is isomorphic to the subset of non-divergent reactive angelic designs. Moreover, each of the operators studied is closed within this subset.

In summary, we have two closely related theories for characterising angelic nondeterminism in CSP whose algebraic properties are clearly distinct. The theory of reactive angelic designs is a natural extension of CSP, where the angelic choice cannot undo the history of events, but which preserves the semantics of CSP. On the other hand, the theory of angelic processes possesses algebraic properties closer to those of the refinement calculus, but does not necessarily preserve the semantics of all CSP processes. Nevertheless, the semantics of the subset of non-divergent processes is maintained, and so our initial hypothesis is satisfied.

7.2 Future Work

The work presented in this thesis lays the foundation for the complete development of process algebras with angelic nondeterminism in the wider context of state-rich reactive systems. Our approach has focused mainly on CSP, however due to the UTP basis of our work, our results are equally applicable to other process calculi, including, for example, *Circus*, which is a combination of CSP and Z, and whose semantics [22] is also given using the UTP. Depending on the desired properties of the algebra, a future approach to incorporating our results in *Circus* needs to consider the implications of the treatment of divergence, which in the case of our model of angelic processes, is rather different from the CSP theory.

A practical application of angelic nondeterminism in *Circus* can be found, for instance, in the modelling strategy of [80], which uses *Circus Time*, a timed version of *Circus*. Therefore, an interesting avenue for future work includes studying the role of angelic nondeterminism in timed versions of process calculi, such as Timed CSP [81] and *Circus Time* [50, 76–78]. A concern that is likely to surface is whether the angel should be allowed to change time in order to avoid divergence, an issue similar to the problem posed by **RA1**. Such a construction would enable angelic nondeterminism to be employed as a specification abstraction in a theory that also includes time.

While we have studied a number of CSP operators, a complete theory of angelic nondeterminism for CSP requires other important operators to be considered, such as hiding and parallel composition. Recursion can be treated in a similar way to other UTP theories as the weakest fixed point. For many of these, the use of our lifting operator $\circledin^{y}_{ac'}$ is likely to be useful and give rise to definitions similar to those in the original theory of CSP, however, some operators, such as parallel composition, require further work. For instance, in the CSP theory, parallel composition is defined using the parallel by merge technique [39] which, in the context of our theory, requires further support for renaming and changing the fields of records.

Furthermore, the algebraic properties of many of the operators have yet to be fully explored. For example, in the case of the external choice operator, there are other alternative and plausible definitions that preserve the CSP semantics, whose algebraic properties, in the context of processes with angelic nondeterminism, are different. In the case of hiding, and similarly to the case of sequential composition, we hypothesize that angelic choice is likely not to be distributive, however future work is necessary in order to propose and establish further laws. A related, and interesting, path for future work is the study of the encoding of additional healthiness conditions [39, 44] of CSP and whether the addition of angelic choice may be needed to enable or simplify the algebraic specification of these.

Even in the context of the theory of angelic designs there is a wide scope for further work. While we have established links between that theory, the extended model of binary multirelations and the **PBMH** theory, it would also be beneficial to have a direct link with the weakest precondition model. The model of extended binary multirelations is also ameanable to further study. For instance, recently Guttmann [65] has proposed a model of binary multirelations in the context of general correctness. A link could be established with this theory, and perhaps, with other models of binary multirelations [36]. The links with the $\mathbf{BMH}_{\perp}$ theory open the door for our theories to be studied in the context of multirelations.

From a practitioner's point of view a theory becomes significantly more useful once there is a toolkit. There may be different approaches for tackling this aspect. For instance, one approach could involve the mechanisation of our theories using a theorem prover, which would not only help practitioners, but also help further validate our theories, proofs and examples. Approaches for mechanising UTP theories include those of Foster et al. [82] and Feliachi et al. [83] using Isabelle/HOL, Zeyda et al. [84] and Oliveira et al. [22] using ProofPower/Z, and others [85, 86]. Particular issues that would need to be considered include reasoning about families of theories and encoding record types, with the capability to change and rename fields as well as type check them, as required to appropriately model sets of final states.

Finally, since the concept of angelic nondeterminism has been used in a variety of different contexts, it would be useful to conduct case studies. For example, in [80] angelic nondeterminism is employed to facilitate the faithful characterisation of idealised time models of control systems using *Circus Time*. In that context, the specification models are constructed from Simulink counterparts which, embody a notion of infinitely fast computations, while the respective implementation models capture the constraints of actual real-time computers. The link between these two is established through an assertion that requires the values output by the implementation to be in agreement with the values of the simulation model. Angelic nondeterminism is employed as an abstract specification mechanism, which, through back propagation enforces the correct choices in the model. A necessary prerequisite for such a case study is the treatment of parallel composition which features prominently.

We envision that many problems that have traditionally been tackled using angelic nondeterminism could be just as easily modelled using our theories, with the added benefit that they can be modelled in the context of process algebras. It remains to be seen how the inclusion of angelic nondeterminism can be fully exploited in the development of refinement strategies for the formal specification and verification of complex state-rich reactive systems. An example to be considered is the refinement of a specification with angelic nondeterminism to an algorithm which uses explicit backtracking. Related to this construction is the relationship between our theories and that of concurrent logic programming [39], which has yet to be explored.

In summary, we have now presented the first extension of CSP that includes a notion of angelic nondeterminism compatible with that of refinement calculi. It is a solid foundation for the extension of state-rich process algebra for refinement. As such, it provides a basis for further work on theory, so as to explore the algebra, techniques, and applications.

Appendix A

UTP: Relations, Designs and CSP

A.1 Theory of Relations

A.1.1 Conditional

Lemma L.A.1.1 $P \lhd c \rhd (Q \Rightarrow R) = (\mathit{true} \lhd c \rhd Q) \Rightarrow (P \lhd c \rhd R)$

Lemma L.A.1.2 *Provided ac' is not free in c,*

$$(P \lhd c \rhd Q) \mathbin{;_\mathcal{A}} R = (P \mathbin{;_\mathcal{A}} R) \lhd c \rhd (Q \mathbin{;_\mathcal{A}} R)$$

Lemma L.A.1.3 $\neg\, (P \lhd c \rhd Q) = (\neg\, P \lhd c \rhd \neg\, Q)$

Lemma L.A.1.4 $P \lhd c \rhd (Q \lor R) = (P \lhd c \rhd Q) \lor (P \lhd c \rhd R)$

Lemma L.A.1.5 $\neg\, (\mathit{false} \lhd c \rhd Q) = \mathit{true} \lhd c \rhd \neg\, Q$

Lemma L.A.1.6 $\neg\, (\mathit{true} \lhd c \rhd Q) = \mathit{false} \lhd c \rhd \neg\, Q$

A.1.2 Predicate Calculus

Lemma L.A.1.7 $(P \land Q) \Leftrightarrow P = P \Rightarrow Q$

Lemma L.A.1.8 $(P \lor Q) \Leftrightarrow (P \lor R) = P \lor (Q \Leftrightarrow R)$

A.2 Theory of Designs

A.2.1 Healthiness Conditions

H1

Lemma L.A.2.1 $\mathbf{H1}(P \lhd c \rhd Q) = \mathbf{H1}(P) \lhd c \rhd \mathbf{H1}(Q)$

Lemma L.A.2.2 $\mathbf{H1}(P \land Q) = \mathbf{H1}(P) \land \mathbf{H1}(Q)$

Lemma L.A.2.3 $\mathbf{H1}(P \lor Q) = \mathbf{H1}(P) \lor \mathbf{H1}(Q)$

H2

Definition 139 $\mathbf{H2A}(P) \mathrel{\widehat{=}} \neg\, P^f \Rightarrow (P^t \land ok')$

Lemma L.A.2.4 (H2A $\Leftrightarrow$ H2) *The definition of* **H2A** *implies that the fixed points are the same as those of* **H2***,*

A.2.2 Lemmas

Lemma L.A.2.5 *Provided $ok \land P$ and ok' is not free in P, $(P \vdash Q)^t = Q$.*

Lemma L.A.2.6 *Provided ok' is not free in P, $ok \land \neg\,(P \vdash Q)^f = ok \land P$.*

Lemma L.A.2.7 $\exists\, ok' \bullet (P \vdash Q) = (ok \land P) \Rightarrow Q$

Lemma L.A.2.8

$$\begin{aligned}&(\neg\, P^f \vdash P^t) \sqcup (\neg\, Q^f \vdash Q^t)\\ &=\\ &(\neg\, P^f \lor \neg\, Q^f \vdash (\neg\, P^f \Rightarrow P^t) \land (\neg\, Q^f \Rightarrow Q^t))\end{aligned}$$

Lemma L.A.2.9 *Provided P and Q are designs,*

$$\exists\, ok' \bullet (P \land Q) = (\exists\, ok' \bullet P) \land (\exists\, ok' \bullet Q)$$

Lemma L.A.2.10

$$\begin{aligned}&(\neg\, P^f \vdash P^t) \sqcup (\neg\, Q^f \vdash Q^t)\\ &=\\ &(\neg\, P^f \lor \neg\, Q^f \vdash (P^f \land Q^t) \lor (P^t \land Q^f) \lor (P^t \land Q^t))\end{aligned}$$

Lemma L.A.2.11 $(P \vdash Q)^f = ok \Rightarrow \neg\, P^f$

Lemma L.A.2.12 $(P \vdash Q)^t = (ok \wedge P^t) \Rightarrow Q^t$

Lemma L.A.2.13 $ok \wedge \neg\, \exists\, ac' \bullet (P \vdash Q)^f = ok \wedge \neg\, \exists\, ac' \bullet \neg\, P^f$

Lemma L.A.2.14 *Provided ok is not free in P and Q,*

$$((P \vdash Q)^f \vdash (P \vdash Q)^t) = (P \vdash Q)$$

Lemma L.A.2.15 *Provided ok′ is not free in P and Q,*

$$(\neg\, \exists\, ac' \bullet (P \vdash Q)^f \vdash (P \vdash Q)^t) = (\neg\, \exists\, ac' \bullet \neg\, P \vdash Q)$$

Lemma L.A.2.16 *Provided ok′ is not free in P and Q,*

$$(\neg\, (P \vdash Q)_f^f \vdash (P \vdash Q)_f^t) = (P_f \vdash Q_f)$$

A.3 Theory of CSP

A.3.1 Operators

Lemma L.A.3.1 $\top_{\mathbf{R}} \,\Box_{\mathbf{R}}\, Skip_{\mathbf{R}} = Skip_{\mathbf{R}}$

Lemma L.A.3.2 *Provided P is a CSP process,*

$$P \,\Box_{\mathbf{R}}\, Stop_{\mathbf{R}} = P$$

Lemma L.A.3.3

$$a \rightarrow_{\mathbf{R}} Stop_{\mathbf{R}} = \mathbf{R}(true \vdash wait' \wedge ((a \notin ref' \wedge tr' = tr) \vee (tr' = tr \frown \langle a \rangle)))$$

Lemma L.A.3.4

$$a \rightarrow_{\mathbf{R}} Choice_{\mathbf{R}} = \mathbf{R}\left(\begin{array}{l} true \\ \vdash \\ (tr' = tr \wedge a \notin ref' \wedge wait') \vee (tr \frown \langle a \rangle \leq tr') \end{array}\right)$$

Lemma L.A.3.5

$$a \rightarrow_{\mathbf{R}} Chaos_{\mathbf{R}} = \mathbf{R}(\neg\, (tr \frown \langle a \rangle \leq tr') \vdash wait' \wedge tr' = tr \wedge a \notin ref')$$

Appendix B

Extended Binary Multirelations

B.1 Healthiness Conditions

B.1.1 BMH0

Definition 12 $\mathbf{BMH} \mathrel{\widehat=} \forall s, ss_0, ss_1 \bullet ((s, ss_0) \in B \land ss_0 \subseteq ss_1) \Rightarrow (s, ss_1) \in B$

Lemma L.3.2.1

$$\begin{array}{l}
\mathbf{BMH0} \\
\Leftrightarrow \\
\left(\begin{array}{l}
\left(\begin{array}{l}
\forall s, ss_0, ss_1 \bullet \\
((s, ss_0) \in B \land ss_0 \subseteq ss_1 \land \bot \in ss_0 \land \bot \in ss_1) \Rightarrow (s, ss_1) \in B
\end{array}\right) \\
\land \\
\mathbf{BMH}
\end{array}\right)
\end{array}$$

Lemma L.B.1.1 *Provided B is* **BMH0**-*healthy,*

$$\begin{array}{l}
\left(\begin{array}{l}
\exists s_0 : State, ss_0, ss_1 : \mathbb{P}\, State_\bot \\
\bullet ((s_0, ss_0) \in B \land ss_0 \subseteq ss_1 \land \bot \in ss_0 \land \bot \in ss_1)
\end{array}\right) \\
= \\
(\exists s_0 : State, ss_1 : \mathbb{P}\, State_\bot \bullet (s_0, ss_1) \in B \land \bot \in ss_1)
\end{array}$$

B.1.2 BMH1

Lemma L.B.1.2

$\mathbf{BMH1}$

$$\Leftrightarrow$$

$$\forall s : State, ss : \mathbb{P}\, State_{\perp} \bullet (s, ss \cup \{\perp\}) \in B \land \perp \notin ss \Rightarrow (s, ss) \in B$$

B.2 Healthiness Conditions as Fixed Points

B.2.1 $\mathbf{bmh_0}$

Lemma L.3.3.1 $\mathbf{BMH0} \Leftrightarrow \mathbf{bmh_0}(B) = B$

Lemma L.3.3.5 $\mathbf{bmh_0} \circ \mathbf{bmh_0}(B) = \mathbf{bmh_0}(B)$

B.2.2 $\mathbf{bmh_1}$

Lemma L.3.3.2 $\mathbf{BMH1} \Leftrightarrow \mathbf{bmh_1}(B) = B$

Lemma L.3.3.6 $\mathbf{bmh_1} \circ \mathbf{bmh_1}(B) = \mathbf{bmh_1}(B)$

B.2.3 $\mathbf{bmh_2}$

Lemma L.3.3.3 $\mathbf{BMH2} \Leftrightarrow \mathbf{bmh_2}(B) = B$

Lemma L.3.3.7 $\mathbf{bmh_2} \circ \mathbf{bmh_2}(B) = \mathbf{bmh_2}(B)$

B.2.4 $\mathbf{bmh_3}$

Lemma L.3.3.4 $\mathbf{BMH3} \Leftrightarrow \mathbf{bmh_3}(B) = B$

Lemma L.3.3.8 $\mathbf{bmh_3} \circ \mathbf{bmh_3}(B) = \mathbf{bmh_3}(B)$

B.2.5 $\mathbf{bmh_0}$ and $\mathbf{bmh_1}$

Lemma L.B.2.1

$$\mathbf{bmh_0} \circ \mathbf{bmh_1}(B)$$

$$=$$

$$\left\{ \begin{array}{l|l} \multicolumn{2}{l}{s : State, ss : \mathbb{P}\, State_{\perp}} \\ & \exists\, ss_0 \bullet ((s, ss_0) \in B \lor (s, ss_0 \cup \{\perp\}) \in B) \\ & \land\, ss_0 \subseteq ss \land (\perp \in ss_0 \Leftrightarrow \perp \in ss)) \end{array} \right\}$$

Properties

Lemma L.B.2.2 $\mathbf{bmh_0} \circ \mathbf{bmh_1}(B) = \mathbf{bmh_1} \circ \mathbf{bmh_0}(B)$

B.2.6 bmh₁ and bmh₂

Lemma L.B.2.3

$$\mathbf{bmh_1} \circ \mathbf{bmh_2}(B)$$
$$= \left\{ \begin{array}{l} s : State, ss : \mathbb{P}\, State_\perp \\ \mid ((s, \{\perp\}) \in B \Leftrightarrow (s, \emptyset) \in B) \land ((s, ss \cup \{\perp\}) \in B \lor (s, ss) \in B) \end{array} \right\}$$

Lemma L.B.2.4

$$\mathbf{bmh_2} \circ \mathbf{bmh_1}(B)$$
$$= \left\{ \begin{array}{l} s : State, ss : \mathbb{P}\, State_\perp \\ \mid ((s, ss \cup \{\perp\}) \in B \lor (s, ss) \in B) \land ((s, \emptyset) \in B) \Rightarrow (s, \{\perp\}) \in B) \end{array} \right\}$$

It can be conclued from Lemma L.B.2.4 and Lemma L.B.2.3 that the functional application of $\mathbf{bmh_1} \circ \mathbf{bmh_2}$ is stronger than that of $\mathbf{bmh_2} \circ \mathbf{bmh_1}$. The order in which these two healthiness conditions are functionally composed is important, since they are not necessarily commutative. The following counter-example illustrates the issue for a relation that is not **BMH2**-healthy.

Counter-example 5

$\mathbf{bmh_2} \circ \mathbf{bmh_1}(\{s : State, ss : \mathbb{P}\, State_\perp \mid ss = \{\perp\}\})$ {Lemma L.B.2.4}
$= \{s : State, ss : \mathbb{P}\, State_\perp \mid ss = \{\perp\} \lor ss = \emptyset\}$

$\mathbf{bmh_1} \circ \mathbf{bmh_2}(\{s : State, ss : \mathbb{P}\, State_\perp \mid ss = \{\perp\}\})$ {Lemma L.B.2.3}
$= \emptyset$

B.2.7 bmh₂ and bmh₃

Lemma L.B.2.5

$$\mathbf{bmh_2} \circ \mathbf{bmh_3}(B)$$
$$=$$

$$\left\{\begin{array}{l} s : State, ss : \mathbb{P}\, State_\perp \\ \mid ((s, \emptyset) \in B \lor \perp \notin ss) \land (s, ss) \in B \land ((s, \emptyset) \in B \Rightarrow (s, \{\perp\}) \in B) \end{array}\right\}$$

Lemma L.B.2.6

$$\mathbf{bmh_3} \circ \mathbf{bmh_2}(B)$$

$$=$$

$$\left\{\begin{array}{l} s : State, ss : \mathbb{P}\, State_\perp \\ \mid ((s, \emptyset) \in B \lor \perp \notin ss) \land (s, ss) \in B \land ((s, \{\perp\}) \in B \Leftrightarrow (s, \emptyset) \in B) \end{array}\right\}$$

The functions **bmh₂** and **bmh₃** are not in general commutative. The following counter-example illustrates the issue for a relation that is not **BMH2**-healthy.

Counter-example 6

$\mathbf{bmh_2} \circ \mathbf{bmh_3}(\{s : State, ss : \mathbb{P}\, State_\perp \mid ss = \{\perp\} \lor ss = \{s\}\})$ {Lemma L.B.2.5}
$= \{s : State, ss : \mathbb{P}\, State_\perp \mid ss = \{s\}\}$

$\mathbf{bmh_3} \circ \mathbf{bmh_2}(\{s : State, ss : \mathbb{P}\, State_\perp \mid ss = \{\perp\} \lor ss = \{s\}\})$ {Lemma L.B.2.6}
$= \emptyset$

B.2.8 bmh₁ and bmh₃

Lemma L.B.2.7

$$\mathbf{bmh_3} \circ \mathbf{bmh_1}(B)$$

$$=$$

$$\left\{\begin{array}{l} s : State, ss : \mathbb{P}\, State_\perp \\ \mid ((s, \{\perp\}) \in B \lor (s, \emptyset) \in B \lor \perp \notin ss) \\ \mid \land \\ \mid ((s, ss \cup \{\perp\}) \in B \lor (s, ss) \in B) \end{array}\right\}$$

Lemma L.B.2.8

$$\mathbf{bmh_1} \circ \mathbf{bmh_3}(B)$$

$$=$$

$$\left\{ \begin{array}{l|l} \multicolumn{2}{l}{s : State, ss : \mathbb{P}\, State_\perp} \\ & ((s, \emptyset) \in B \land ((s, ss \cup \{\perp\}) \in B \lor (s, ss) \in B)) \\ & \lor \\ & (\perp \notin ss \land (s, ss) \in B) \end{array} \right\}$$

The functions $\mathbf{bmh_3}$ and $\mathbf{bmh_1}$ do not necessarily commute. The following counter-example shows this for a relation that is not **BMH3**-healthy. In fact, the functional application $\mathbf{bmh_3} \circ \mathbf{bmh_1}$ is not suitable as the counter-example shows that we have a fixed point.

Counter-example 7

$\mathbf{bmh_3} \circ \mathbf{bmh_1}(\{s : State, ss : \mathbb{P}\, State_\perp \mid ss = \{\perp, s\} \lor ss = \{\perp\}\})$ {Lemma L.B.2.7}
$= \{s : State, ss : \mathbb{P}\, State_\perp \mid ss = \{\perp, s\} \lor ss = \{\perp\}\}$

$\mathbf{bmh_1} \circ \mathbf{bmh_3}(\{s : State, ss : \mathbb{P}\, State_\perp \mid ss = \{\perp, s\} \lor ss = \{\perp\}\})$ {Lemma L.B.2.8}
$= \emptyset$

B.2.9 $\mathbf{bmh_{0,1,2}}$

Lemma L.3.3.9

$$\mathbf{bmh_{0,1,2}}(B) = \left\{ s, ss \left| \begin{array}{l} \exists ss_0 \bullet ((s, ss_0) \in B \lor (s, ss_0 \cup \{\perp\}) \in B) \\ \land ((s, \{\perp\}) \in B \Leftrightarrow (s, \emptyset) \in B) \\ \land ss_0 \subseteq ss \land (\perp \in ss_0 \Leftrightarrow \perp \in ss) \end{array} \right. \right\}$$

Theorem T.3.3.1 $\mathbf{BMH0} \land \mathbf{BMH1} \land \mathbf{BMH2} \Leftrightarrow \mathbf{bmh_{0,1,2}}(B) = B$

Lemma L.3.3.10 $(\mathbf{bmh_{0,1,2}}(B) = B) \Rightarrow \mathbf{BMH0}$

Lemma L.3.3.11 $(\mathbf{bmh_{0,1,2}}(B) = B) \Rightarrow \mathbf{BMH1}$

Lemma L.3.3.12 $(\mathbf{bmh_{0,1,2}}(B) = B) \Rightarrow \mathbf{BMH2}$

Lemma L.3.3.13 *Provided B is* $\mathbf{BMH0} - \mathbf{BMH2}$*-healthy,* $\mathbf{bmh_{0,1,2}}(B) = B$.

Lemma L.B.2.9 $\mathbf{bmh_{0,1,2}} \circ \mathbf{bmh_{0,1,2}}(B) = \mathbf{bmh_{0,1,2}}(B)$

Lemma L.B.2.10

$\mathbf{bmh}_{0,1,2}(B)$

$$=$$

$$\left\{\begin{array}{l} s : State, ss : \mathbb{P}\, State_\perp \\ \left|\begin{array}{l} ((s, \{\perp\}) \in B \land (s, \emptyset) \in B) \\ \lor \\ \left(\begin{array}{l} ((s, \{\perp\}) \notin B \land (s, \emptyset) \notin B) \\ \land \\ \left(\begin{array}{l} (((s, ac') \in B \ ; ac \subseteq ss) \land \perp \notin ss) \\ \lor \\ ((s, ac' \cup \{\perp\}) \ ; ac \subseteq ss) \end{array}\right) \end{array}\right) \end{array}\right. \end{array}\right\}$$

Lemma L.B.2.11

$$(s, ss) \in \mathbf{bmh_{0,1,2}}(B)$$

$$=$$

$$\left(\begin{array}{l} ((s, \{\perp\}) \in B \Leftrightarrow (s, \emptyset) \in B) \\ \land \\ \exists\, ss_0 \bullet \left(\begin{array}{l} ((s, ss_0) \in B \lor (s, ss_0 \cup \{\perp\}) \in B) \\ \land\ ss_0 \subseteq ss \land (\perp \in ss_0 \Leftrightarrow \perp \in ss) \end{array}\right) \end{array}\right)$$

Lemma L.B.2.12

$$\exists\, ss_1 \bullet (s, ss_1) \in \mathbf{bmh_{0,1,2}}(B) \land ss_1 \subseteq ss \land (\perp \in ss_1 \Leftrightarrow \perp \in ss)$$

$$=$$

$$\left(\begin{array}{l} ((s, \{\perp\}) \in B \Leftrightarrow (s, \emptyset) \in B) \\ \land \\ \exists\, ss_0 \bullet ((s, ss_0) \in B \lor (s, ss_0 \cup \{\perp\}) \in B) \land ss_0 \subseteq ss \land (\perp \in ss_0 \Leftrightarrow \perp \in ss) \end{array}\right)$$

Lemma L.B.2.13 $(s, \emptyset) \in \mathbf{bmh_{0,1,2}}(B) = (s, \emptyset) \in B \land (s, \{\perp\}) \in B$

Lemma L.B.2.14 $(s, \{\perp\}) \in \mathbf{bmh_{0,1,2}}(B) = (s, \emptyset) \in B \land (s, \{\perp\}) \in B$

Lemma L.B.2.15

$$B_1 \subseteq B_0$$

$$\Leftrightarrow$$

$$\forall\, s : State, ss : \mathbb{P}\, State \bullet \left(\begin{array}{l} (s, ss) \in B_1 \Rightarrow (s, ss) \in B_0 \\ \land \\ (s, ss \cup \{\perp\}) \in B_1 \Rightarrow (s, ss \cup \{\perp\}) \in B_0 \end{array}\right)$$

B.2.10 $\mathbf{bmh_{0,1,3}}$

Lemma L.B.2.16

$$
\begin{array}{l}
\mathbf{bmh_0} \circ \mathbf{bmh_1} \circ \mathbf{bmh_3}(B) \\
= \\
\left\{ \begin{array}{l}
s : State, ss : \mathbb{P}\, State_\perp \\
\left| \begin{array}{l}
\exists\, ss_0 \bullet \left(\begin{array}{l}
((s, ss_0) \in B \lor (s, ss_0 \cup \{\perp\}) \in B) \\
\land \\
(s, \emptyset) \in B \land ss_0 \subseteq ss \land (\perp \in ss_0 \Leftrightarrow \perp \in ss)
\end{array} \right) \\
\lor \\
\exists\, ss_0 \bullet ((s, ss_0) \in B \land ss_0 \subseteq ss \land \perp \notin ss_0 \land \perp \notin ss)
\end{array} \right.
\end{array} \right\}
\end{array}
$$

Lemma L.B.2.17

$$
\begin{array}{l}
\exists\, ss_0 \bullet \left(\begin{array}{l}
((s, ss_0) \in B \lor (s, ss_0 \cup \{\perp\}) \in B) \\
\land \\
ss_0 \subseteq ss \land (\perp \in ss_0 \Leftrightarrow \perp \in ss)
\end{array} \right) \\
= \\
\exists\, ss_0 \bullet \left(\begin{array}{l}
((s, ss_0) \in B \lor (s, ss_0 \cup \{\perp\}) \in B) \\
\land \\
ss_0 \subseteq ss \land (\perp \in ss_0 \Leftrightarrow \perp \in ss)
\end{array} \right) \lor (s, \{\perp\}) \in B
\end{array}
$$

B.2.11 $\mathbf{bmh_{0,1,3,2}}$

Lemma L.3.3.14

$$
\begin{array}{l}
\mathbf{bmh_0} \circ \mathbf{bmh_1} \circ \mathbf{bmh_3} \circ \mathbf{bmh_2}(B) \\
= \\
\left\{ s, ss \left| \begin{array}{l}
((s, \emptyset) \in B \land (s, \{\perp\}) \in B) \\
\lor \\
\left(\begin{array}{l}
(s, \{\perp\}) \notin B \land (s, \emptyset) \notin B \\
\land \\
(\exists\, ss_0 \bullet (s, ss_0) \in B \land ss_0 \subseteq ss \land \perp \notin ss_0 \land \perp \notin ss)
\end{array} \right)
\end{array} \right. \right\}
\end{array}
$$

Theorem T.3.3.2 $\mathbf{BMH0} \land \mathbf{BMH1} \land \mathbf{BMH2} \land \mathbf{BMH3} \Leftrightarrow \mathbf{bmh_{0,1,3,2}}(B) = B$

Lemma L.3.3.15 $\mathbf{BMH0} \land \mathbf{BMH1} \land \mathbf{BMH2} \land \mathbf{BMH3} \Rightarrow \mathbf{bmh_{0,1,3,2}}(B) = B$

Lemma L.3.3.16 $\mathbf{bmh_{0,1,2}} \circ \mathbf{bmh_{0,1,3,2}}(B) = \mathbf{bmh_{0,1,3,2}}(B)$

Lemma L.3.3.17 $(\mathbf{bmh_{0,1,3,2}}(B) = B) \Rightarrow \mathbf{BMH3}$

Lemma L.B.2.18 $\mathbf{bmh_{0,1,3,2}} \circ \mathbf{bmh_{0,1,3,2}}(B) = \mathbf{bmh_{0,1,3,2}}(B)$

Lemma L.B.2.19

$$\begin{array}{l}(s, ss) \in \mathbf{bmh_{0,1,3,2}}(B)\\ =\\ \left(\begin{array}{l}((s, \emptyset) \in B \land (s, \{\bot\}) \in B)\\ \lor\\ \left(\begin{array}{l}(s, \{\bot\}) \notin B \land (s, \emptyset) \notin B\\ \land\\ \exists\, ss_0 \bullet ((s, ss_0) \in B \land ss_0 \subseteq ss \land \bot \notin ss_0 \land \bot \notin ss)\end{array}\right)\end{array}\right)\end{array}$$

Lemma L.B.2.20

$$\begin{array}{l}\exists\, ss_1 : \mathbb{P}\, State_\bot \bullet (s, ss_1 \cup \{\bot\}) \in \mathbf{bmh_{0,1,3,2}}(B) \land ss_1 \subseteq ss \land (\bot \in ss_1 \Leftrightarrow \bot \in ss)\\ \Leftrightarrow\\ ((s, \emptyset) \in B \land (s, \{\bot\}) \in B)\end{array}$$

Lemma L.B.2.21

$$\begin{array}{l}\exists\, ss_1 : \mathbb{P}\, State_\bot \bullet (s, ss_1) \in \mathbf{bmh_{0,1,3,2}}(B) \land ss_1 \subseteq ss \land (\bot \in ss_1 \Leftrightarrow \bot \in ss)\\ \Leftrightarrow\\ \left(\begin{array}{l}((s, \emptyset) \in B \land (s, \{\bot\}) \in B)\\ \lor\\ \left(\begin{array}{l}(s, \{\bot\}) \notin B \land (s, \emptyset) \notin B\\ \land\\ \exists\, ss_0 : \mathbb{P}\, State_\bot \bullet (s, ss_0) \in B \land ss_0 \subseteq ss \land \bot \notin ss_0 \land \bot \notin ss\end{array}\right)\end{array}\right)\end{array}$$

Lemma L.B.2.22 $(s, \emptyset) \in \mathbf{bmh_{0,1,3,2}}(B) = (s, \emptyset) \in B \land (s, \{\bot\}) \in B$

Lemma L.B.2.23 $(s, \{\bot\}) \in \mathbf{bmh_{0,1,3,2}}(B) = (s, \emptyset) \in B \land (s, \{\bot\}) \in B$

Lemma L.B.2.24 *Provided B is* **BMH0** *and* **BMH2**-*healthy,*

$$\begin{array}{l}B = (B \rhd \{ss : \mathbb{P}\, State_\bot \mid \bot \in ss\}) \cup \{s_0 : State, ss : \mathbb{P}\, State_\bot \mid (s_0, \emptyset) \in B\}\\ \Leftrightarrow\\ \mathbf{BMH3}\end{array}$$

B.3 Operators

B.3.1 Angelic Choice

Lemma L.3.5.1 $(x :=_{BM_\perp} e) \sqcup_{BM_\perp} (x :=_{BM} e) = (x :=_{BM} e)$

Lemma L.3.5.2 $\top_{BM_\perp} \sqcup_{BM_\perp} B = \top_{BM_\perp}$

Lemma L.3.5.3 $\perp_{BM_\perp} \sqcup_{BM_\perp} B = B$

B.3.2 Demonic Choice

Lemma L.3.5.4 $(x :=_{BM} e) \sqcap_{BM_\perp} (x :=_{BM_\perp} e) = (x :=_{BM_\perp} e)$

Lemma L.3.5.5 $\perp_{BM_\perp} \sqcap_{BM_\perp} B = \perp_{BM_\perp}$

Lemma L.3.5.6 $\top_{BM_\perp} \sqcap_{BM_\perp} B = B$

B.3.3 Sequential Composition

Theorem T.3.5.1 *Provided B_0 is* **BMH0***-healthy,*

$$B_0 \mathbin{;_{BM_\perp}} B_1 = \left(\begin{array}{l} \{s_0, ss_0 \mid (s_0, State_\perp) \in B_0\} \\ \cup \\ \{s_0, ss_0 \mid (s_0, \{s_1 \mid (s_1, ss_0) \in B_1\}) \in B_0\} \end{array}\right)$$

Lemma L.3.5.7 $\top_{BM_\perp} \mathbin{;_{BM_\perp}} B = \top_{BM_\perp}$

Lemma L.3.5.8 $\perp_{BM_\perp} \mathbin{;_{BM_\perp}} B = \perp_{BM_\perp}$

B.4 Relationship with Binary Multirelations

B.4.1 *bmb2bm*

Theorem T.3.6.1 (bmb2bm-is-bmh$_{\mathbf{up}}$)

$$\mathbf{bmh_{up}} \circ bmb2bm(\mathbf{bmh_{0,1,3,2}}(B)) = bmb2bm(\mathbf{bmh_{0,1,3,2}}(B))$$

Lemma L.3.6.1 $\mathbf{BMH} \Leftrightarrow \mathbf{bmh_{up}}(B) = B$

Lemma L.B.4.1

$$
\begin{array}{l}
bmb2bm(\mathbf{bmh_{0,1,3,2}}(B)) \\
= \\
\left\{ \begin{array}{l|l}
s : State, ss : \mathbb{P}\, State_{\perp} & \\
& ((s, \emptyset) \in B \land (s, \{\perp\}) \in B) \land \perp \notin ss \\
& \lor \\
& \left(\begin{array}{l} (s, \{\perp\}) \notin B \land (s, \emptyset) \notin B \\ \land \\ \exists\, ss_0 \bullet \left((s, ss_0) \in B \land ss_0 \subseteq ss \land \perp \notin ss_0 \land \perp \notin ss \right) \end{array} \right)
\end{array} \right\}
\end{array}
$$

Lemma L.B.4.2 $(s, \emptyset) \in bmb2bm(\mathbf{bmh_{up}}) = (s, \emptyset) \in B$

Lemma L.B.4.3 $(s, \{\perp\}) \in bmb2bm(\mathbf{bmh_{up}}) = (s, \emptyset) \in B$

Theorem T.B.4.1 *Provided B is $\mathbf{BMH_{0,1,2,3}}$-healthy,*

$$\mathbf{bmh_{up}} \circ bmb2bm(B) = bmb2bm(B)$$

Lemma L.B.4.4

$$
\begin{array}{l}
bmb2bm(\mathbf{bmh_{0,1,3,2}}(B)) \\
= \\
\left\{ \begin{array}{l|l}
s : State, ss : \mathbb{P}\, State_{\perp} & \\
& ((s, \emptyset) \in B \land (s, \{\perp\}) \in B) \land \perp \notin ss \\
& \lor \\
& \left(\begin{array}{l} (s, \{\perp\}) \notin B \land (s, \emptyset) \notin B \\ \land \\ \exists\, ss_0 \bullet \left((s, ss_0) \in B \land ss_0 \subseteq ss \land \perp \notin ss_0 \land \perp \notin ss \right) \end{array} \right)
\end{array} \right\}
\end{array}
$$

B.4.2 *bm2bmb*

Theorem T.3.6.2

$$\mathbf{bmh_{0,1,3,2}} \circ bm2bmb(\mathbf{bmh_{up}}(B)) = bm2bmb(\mathbf{bmh_{up}}(B))$$

Theorem T.3.6.3 *Provided B is $\mathbf{BMH_{0,1,2,3}}$-healthy, $bm2bmb \circ bmb2bm(B) = B$,*

Theorem T.3.6.4 *Provided B is $\mathbf{BMH}$-healthy, $bmb2bm \circ bm2bmb(B) = B$,*

Lemma L.B.4.5

$$\begin{array}{l} bm2bmb(\mathbf{bmh_{up}}(B)) \\ = \\ \left\{ \begin{array}{l} s : State, ss : \mathbb{P}\, State_\perp \\ \left| \begin{array}{l} \exists\, ss_0 \bullet (s, ss_0) \in B \land \bot \notin ss_0 \land ss_0 \subseteq ss \land \bot \notin ss \\ \lor \\ (s, \emptyset) \in B \end{array} \right. \end{array} \right\} \end{array}$$

Lemma L.B.4.6

$$\begin{array}{l} bm2bmb(\mathbf{bmh_{up}}(B)) \\ = \\ \left\{ \begin{array}{l} s : State, ss : \mathbb{P}\, State_\perp \\ \left| \begin{array}{l} \exists\, ss_0 \bullet (s, ss_0) \in B \land \bot \notin ss_0 \land ss_0 \subseteq ss \land \bot \notin ss \\ \lor \\ (s, \emptyset) \in B \end{array} \right. \end{array} \right\} \end{array}$$

B.5 Set Theory

Lemma L.B.5.1

$$\begin{array}{l} \exists\, ss_0 \bullet (s, ss_0 \cup \{\bot\}) \in B \land ss_0 \subseteq ss \land (\bot \in ss_0 \Leftrightarrow \bot \in ss) \\ \Leftrightarrow \\ \exists\, ss_0 \bullet (s, ss_0) \in B \land ss_0 \subseteq (ss \cup \{\bot\}) \land \bot \in ss_0 \end{array}$$

Lemma L.B.5.2 $(A = B \cup \{x\} \land x \notin B) \Leftrightarrow (A \setminus \{x\} = B \land x \in A)$

Lemma L.B.5.3 $\{x\} \subseteq A \Leftrightarrow x \in A$

Lemma L.B.5.4 $x \notin A \Leftrightarrow (\forall\, y \bullet y \in A \Rightarrow y \notin \{x\})$

Lemma L.B.5.5 $(A = (B \cup \{x\}) \land x \in B) \Leftrightarrow (A = B \land x \in B)$

Lemma L.B.5.6

$$((A \cup \{x\}) \subseteq (B \cup \{x\}) \land x \notin A \land x \notin B) \Leftrightarrow (A \subseteq B \land x \notin A \land x \notin B)$$

Appendix C

Angelic Designs (A)

C.1 Healthiness Conditions

C.1.1 A0

Definition 87 $\mathbf{A0}(P) \mathrel{\widehat=} P \land ((ok \land \neg P^f) \Rightarrow (ok' \Rightarrow ac' \neq \emptyset))$

Theorem T.4.2.1 $\mathbf{A0} \circ \mathbf{A0}(P) = \mathbf{A0}(P)$

Theorem T.4.2.2 $(P \sqsubseteq Q) \Rightarrow (\mathbf{A0}(P) \sqsubseteq \mathbf{A0}(Q))$

Theorem T.4.2.3 *If P is a design so is $\mathbf{A0}(P)$.*

$$\mathbf{A0}(P) = (\neg P^f \vdash P^t \land ac' \neq \emptyset)$$

Theorem T.4.2.4 *Provided P and Q are $\mathbf{A0}$-healthy,*

$$\mathbf{A0}(P \land Q) = P \land Q$$

Theorem T.4.2.5 *Provided P and Q are $\mathbf{A0}$-healthy designs,*

$$\mathbf{A0}(P \lor Q) = P \lor Q$$

Theorem T.C.1.1 $\mathbf{A0}(P \land Q) = \mathbf{A0}(P) \land \mathbf{A0}(Q)$

Theorem T.C.1.2 $\mathbf{A0} \circ \mathbf{H1} \circ \mathbf{H2}(P) = (\neg P^f \vdash P^t \land ac' \neq \emptyset)$

Theorem T.C.1.3 $\mathbf{H1} \circ \mathbf{H2} \circ \mathbf{A0}(P) = \mathbf{A0} \circ \mathbf{H1} \circ \mathbf{H2}(P)$

Lemma L.C.1.1 *Provided ok' not free in e,* $\mathbf{A0}(P)[e/s] = \mathbf{A0}(P[e/s])$.

Lemma L.C.1.2 $\mathbf{A0}(P)^o = P^o \land ((ok \land \neg\, P^f) \Rightarrow (o \Rightarrow ac' \neq \emptyset))$

Lemma L.C.1.3 $\mathbf{A0}(P)^f = P^f$

Lemma L.C.1.4 $\mathbf{A0}(P)^t = P^t \land ((ok \land \neg\, P^f) \Rightarrow ac' \neq \emptyset)$

C.1.2 A1

Theorem T.4.2.6 $\mathbf{A1} \circ \mathbf{A1}(P_0 \vdash P_1) = \mathbf{A1}(P_0 \vdash P_1)$

Theorem T.4.2.7 $(P \sqsubseteq Q) \Rightarrow \mathbf{A1}(P) \sqsubseteq \mathbf{A1}(Q)$

C.1.3 A

Theorem T.4.2.8 *Provided P^t satisfies* **PBMH**, $\mathbf{A0} \circ \mathbf{A1}(P) = \mathbf{A1} \circ \mathbf{A0}(P)$

Theorem T.4.2.9 $\mathbf{A} \circ \mathbf{A}(P) = \mathbf{A}(P)$

Theorem T.4.2.10 $\mathbf{H1} \circ \mathbf{H2} \circ \mathbf{A}(P) = \mathbf{A} \circ \mathbf{H1} \circ \mathbf{H2}(P)$

Theorem T.C.1.4 $P \sqsubseteq Q \Rightarrow \mathbf{A}(P) \sqsubseteq \mathbf{A}(Q)$

Lemma L.C.1.5 *Provided ok' is not free in e,* $\mathbf{A}(P)[e/s] = \mathbf{A}(P[e/s])$

Lemma L.C.1.6 $s.x = v \land P \Leftrightarrow s.x = v \land P[s \oplus \{x \mapsto v\}/s]$

Lemma L.C.1.7 *Provided P is an* **A***-healthy design,* $P^f = ok \Rightarrow P^f$.

Lemma L.C.1.8 *Provided P is an* **A***-healthy design,*

$$P^t = ((ok \land \neg\, P^f) \Rightarrow (P^t \land ac' \neq \emptyset))$$

Lemma L.C.1.9 *Provided P is an* **A***-healthy design,*

$$\begin{array}{l}(\neg\, \exists\, ac' \bullet \mathbf{PBMH}(P^f) \vdash \mathbf{PBMH}(P^t) \land ac' \neq \emptyset)\\ =\\ (\neg\, \exists\, ac' \bullet P^f \vdash P^t)\end{array}$$

Theorem T.C.1.5 *Provided P is an* **A***-healthy design,*

$$\mathbf{H3}_{\mathcal{D}\mathbf{ac}}(P) = (\neg \exists ac' \bullet P^f \vdash P^t)$$

C.1.4 A2

Theorem T.4.2.11 $\mathbf{A2}(P) = P[\emptyset/ac'] \lor (\exists y \bullet P[\{y\}/ac'] \land y \in ac')$

Theorem T.4.2.12 $\mathbf{A2} \circ \mathbf{A2}(P) = \mathbf{A2}(P)$

Theorem T.4.2.13 $P \sqsubseteq Q \Rightarrow \mathbf{A2}(P) \sqsubseteq \mathbf{A2}(Q)$

Theorem T.4.2.14 $\mathbf{A2}(P \lor Q) = \mathbf{A2}(P) \lor \mathbf{A2}(Q)$

Theorem T.C.1.6 (A2-idempotent) *Provided P is* **PBMH***-healthy,*

$$\mathbf{A2} \circ \mathbf{A2}(P) = \mathbf{A2}(P)$$

Lemmas

Lemma L.4.2.3 $\mathbf{A2}(P \vdash Q) = (\neg \mathbf{A2}(\neg P) \vdash \mathbf{A2}(Q))$

Lemma L.C.1.10 $\mathbf{A2}(P) = \exists ac_0 \bullet P[\{s \mid \{s\} = ac_0\}/ac'] \land ac_0 \subseteq ac'$

Lemma L.C.1.11

$$\begin{aligned}&\mathbf{A2} \circ \mathbf{A}(\neg P^f \vdash P^t)\\ &=\\ &(\neg \mathbf{A2} \circ \mathbf{PBMH}(P^f) \vdash \mathbf{A2}(\mathbf{PBMH}(P^t) \land ac' \neq \emptyset))\end{aligned}$$

Lemma L.C.1.12 $\mathbf{A2}(\mathit{false}) = \mathit{false}$

Lemma L.C.1.13 $\mathbf{A2}(\mathit{true}) = \mathit{true}$

Lemma L.C.1.14 *Provided ac' is not free in P,*

$$\mathbf{A2}(\exists y \bullet y \in ac' \land P) = \exists y \bullet y \in ac' \land P$$

Properties

Lemma L.C.1.15 *Provided ac' is not free in P,* $\mathbf{A2}(P \land Q) = P \land \mathbf{A2}(Q)$.

Lemma L.C.1.16 *Provided ac' not free in P, $\mathbf{A2}(P) = P$.*

Lemma L.C.1.17 $\mathbf{A2}(P \land ac' \neq \emptyset) = \exists z \bullet P[\{z\}/ac'] \land z \in ac'$

Lemma L.C.1.18 $\mathbf{A2}(P \land ac' = \emptyset) = P[\emptyset/ac']$

Lemma L.C.1.19 $\mathbf{A2}(P)[\emptyset/ac'] = P[\emptyset/ac']$

Lemma L.C.1.20 *Provided ac' is not free in c,*

$$\mathbf{A2}(P \lhd c \rhd Q) = \mathbf{A2}(P) \lhd c \rhd \mathbf{A2}(Q)$$

Lemma L.C.1.21 $\mathbf{A2}(x \in ac') = x \in ac'$

Lemma L.C.1.22 $\mathbf{A2}(P)^o_w = \mathbf{A2}(P^o_w)$

Lemma L.C.1.23 *Provided ac' is not free in o, $\mathbf{A2}(P)[o/ok] = \mathbf{A2}(P[o/ok])$.*

Lemma L.C.1.24 *Provided that x is not ac', $\mathbf{A2}(\exists x \bullet P) = \exists x \bullet \mathbf{A2}(P)$*

Properties with respect to PBMH

Theorem T.C.1.7 $\mathbf{A2} \circ \mathbf{PBMH}(P) = \mathbf{A2}(P)$

Lemma L.C.1.25 *Provided P is **PBMH**-healthy,*

$$\begin{aligned}&\mathbf{PBMH}(P \mathbin{;_\mathcal{A}} \{s \mid \{s\} = ac'\}) \mathbin{;_\mathcal{A}} \{s \mid \{s\} = ac'\}\\ &=\\ &P \mathbin{;_\mathcal{A}} \{s \mid \{s\} = ac'\}\end{aligned}$$

Lemma L.C.1.26 $\mathbf{PBMH} \circ \mathbf{A2}(P) = \mathbf{A2}(P)$

Properties with respect to $;_\mathcal{A}$

Theorem T.C.1.8 *Provided P and Q are **A2**-healthy, $\mathbf{A2}(P \mathbin{;_\mathcal{A}} Q) = P \mathbin{;_\mathcal{A}} Q$*

Lemma L.C.1.27

$$\begin{aligned}&\mathbf{A2}(P) \mathbin{;_\mathcal{A}} \mathbf{A2}(Q)\\ &=\end{aligned}$$

$$\left(\begin{array}{l} P[\emptyset/ac'] \lor (\exists\, y \bullet P[\{y\}/ac'] \land Q[\emptyset/ac'][y/s]) \\ \lor \\ (\exists\, y \bullet P[\{y\}/ac'] \land (\exists\, y \bullet Q[\{y\}/ac'][y/s] \land y \in ac')) \end{array}\right)$$

Lemma L.C.1.28 $\mathbf{A2}(\mathbf{A2}(P) \mathbin{;_{\mathcal{A}}} \mathbf{A2}(Q)) = \mathbf{A2}(P) \mathbin{;_{\mathcal{A}}} \mathbf{A2}(Q)$

Properties with respect to links (*p*2*ac* and *ac*2*p*)

Lemma L.C.1.29 $p2ac \circ ac2p \circ \mathbf{A2}(P) = \mathbf{A2}(P) \land ac' \neq \emptyset$

Lemma L.C.1.30 $p2ac \circ ac2p \circ \mathbf{PBMH}(P) = p2ac \circ ac2p(P)$

Lemma L.C.1.31 $p2ac \circ ac2p \circ \mathbf{A2}(P) = p2ac \circ ac2p(P \mathbin{;_{\mathcal{A}}} \{s\} = ac')$

Lemma L.C.1.32

$$\begin{array}{l} p2ac \circ ac2p \circ \mathbf{A2}(P) \\ = \\ (P[\emptyset/ac'] \land ac' \neq \emptyset) \lor (\exists\, y \bullet P[\{y\}/ac'] \land y \in ac') \end{array}$$

C.2 Relationship with Extended Binary Multirelations

C.2.1 *d*2*bmb*

Theorem T.4.3.1 *Provided P is a design,*

$$\mathbf{bmh_{0,1,2}} \circ d2bmb(\mathbf{A}(P)) = d2bmb(\mathbf{A}(P))$$

Lemma L.C.2.1 (*d*2*bmb*-A-healthy) *Provided P is a design,*

$$\begin{array}{l} d2bmb(\mathbf{A}(P)) \\ = \\ \left\{\begin{array}{l} s : State, ss : \mathbb{P}\, State_{\perp} \\ \left|\ \left(\begin{array}{l} \exists\, ac_0 : \mathbb{P}\, State \bullet \\ (P^f[ac_0/ac'] \lor (P^t[ac_0/ac'] \land \perp \notin ss \land ss \neq \emptyset)) \land ac_0 \subseteq ss \end{array}\right)\right. \end{array}\right\} \end{array}$$

Lemma L.C.2.2 *Provided P is a design,*

$$\exists\, ss_0 : \mathbb{P}\, State_\perp \bullet \left(\begin{array}{l} (s, ss_0 \cup \{\perp\}) \in d2bmb(\mathbf{A}(P)) \\ \wedge \\ ss_0 \subseteq ss \wedge (\perp \in ss_0 \Leftrightarrow \perp \in ss) \end{array}\right)$$
$$=$$
$$\exists\, ac_0 : \mathbb{P}\, State \bullet P^f[ac_0/ac'] \wedge ac_0 \subseteq ss$$

Lemma L.C.2.3 *Provided P is a design,*

$$\exists\, ss_0 : \mathbb{P}\, State_\perp \bullet (s, ss_0) \in d2bmb(\mathbf{A}(P)) \wedge ss_0 \subseteq ss \wedge (\perp \in ss_0 \Leftrightarrow \perp \in ss)$$
$$=$$
$$\exists\, ac_0 : \mathbb{P}\, State \bullet (P^f[ac_0/ac'] \vee (P^t[ac_0/ac'] \wedge ss \neq \emptyset \wedge \perp \notin ss)) \wedge ac_0 \subseteq ss$$

Lemma L.C.2.4 *Provided P is a design,*

$$(s, \{\perp\}) \in d2bmb(\mathbf{A}(P)) \Leftrightarrow (s, \emptyset) \in d2bmb(\mathbf{A}(P))$$

Lemma L.C.2.5 *Provided P is a design,*

$$(s, \{\perp\}) \in d2bmb(\mathbf{A}(P)) = P^f[\emptyset/ac']$$

Lemma L.C.2.6 *Provided P is a design,*

$$(s, \emptyset) \in d2bmb(\mathbf{A}(P)) = P^f[\emptyset/ac']$$

Lemma L.C.2.7 *Provided P is a design,*

$$(s, \emptyset) \in d2bmb(\mathbf{A}(P)) \Leftrightarrow (s, \{\perp\}) \in d2bmb(\mathbf{A}(P)) = true$$

Lemma L.C.2.8 *Provided ok and ok' are not free in P and Q,*

$$d2bmb(P \vdash Q) = \left\{ s, ss \,\middle|\, \begin{array}{l} ((P \Rightarrow Q)[ss/ac'] \wedge \perp \notin ss) \\ \vee \\ ((\neg\, P)[(ss \setminus \{\perp\})/ac'] \wedge \perp \in ss) \end{array} \right\}$$

C.2.2 *bmb2d*

Lemma L.4.3.1 $bmb2d(B) = ok \Rightarrow \left(\begin{array}{l} ((s, ac') \in B \land \bot \notin ac' \land ok') \\ \lor \\ (s, ac' \cup \{\bot\}) \in B \end{array}\right)$

Theorem T.4.3.2 *Provided B satisfies* $\mathbf{bmh_{0,1,2}}$, $\mathbf{A} \circ bmb2d(B) = bmb2d(B)$.

Lemma L.C.2.9

$$\begin{array}{l} ((s, ac') \in B \ ; \ ac \subseteq ac') \land (s, \emptyset) \notin B \\ \Leftrightarrow \\ ((s, ac') \in B \ ; \ ac \subseteq ac') \land ac' \neq \emptyset \land (s, \emptyset) \notin B \end{array}$$

Lemma L.C.2.10 *Provided B satisfies* $\mathbf{bmh_{0,1,2}}$,

$$bmb2d(B) = \left(\begin{array}{l} \neg\, ((s, ac' \cup \{\bot\}) \in B \ ; \ ac \subseteq ac') \\ \vdash \\ ((s, ac') \in B \ ; \ ac \subseteq ac') \land (s, \emptyset) \notin B \end{array}\right)$$

Lemma L.C.2.11

$$\begin{array}{l} bmb2d(\mathbf{bmh_{0,1,2}}(B)) \\ = \\ \left(\begin{array}{l} \left(\begin{array}{l} \neg\, ((s, \{\bot\}) \in B \land (s, \emptyset) \in B) \\ \land \\ \neg\, (((s, ac' \cup \{\bot\}) \in B \ ; \ ac \subseteq ac') \land (s, \{\bot\}) \notin B \land (s, \emptyset) \notin B) \end{array}\right) \\ \vdash \\ ((s, ac') \in B \ ; \ ac \subseteq ac') \land (s, \{\bot\}) \notin B \land (s, \emptyset) \notin B \end{array}\right) \end{array}$$

Lemma L.C.2.12 *Provided P is a design,*

$$(s, \{s_1 : State_\bot \mid true\}) \in d2bmb(P) = P^f[\{s_1 : State \mid true\}/ac']$$

Lemma L.C.2.13 *Provided $\bot \notin ac'$ and P is a design,*

$$\{s : State \mid (s, ac' \cup \{\bot\}) \in d2bmb(P)\} = \{s : State \mid P^f\}$$

Lemma L.C.2.14 *Provided $\bot \notin ac'$ and P is a design,*

$$\{s : State \mid (s, ac') \in d2bmb(P)\} = \{s : State \mid (\neg\, P^f \Rightarrow P^t)\}$$

Lemma L.C.2.15 *Provided P and Q are designs,*

$$\begin{array}{l} (s, \{s : State \mid (s, ac' \cup \{\bot\}) \in d2bmb(P)\}) \in d2bmb(Q) \\ = \\ (\neg\, Q^f \Rightarrow Q^t)[\{s : State \mid P^f\}/ac'] \end{array}$$

Lemma L.C.2.16 *Provided P and Q are designs,*

$$\begin{array}{l} (s, \{s : State \mid (s, ac') \in d2bmb(P)\}) \in d2bmb(Q) \\ = \\ (\neg\, Q^f \Rightarrow Q^t)[\{s : State \mid (\neg\, P^f \Rightarrow P^t)\}/ac'] \end{array}$$

Lemma L.C.2.17

$$\begin{array}{l} bmb2d(B_0 \;;\; B_1) \\ = \\ ok \Rightarrow \left(\begin{array}{l} ((s, \{s_1 : State \mid (s_1, ac') \in B_1\}) \in B_0 \land \bot \notin ac' \land ok') \\ \lor \\ ((s, \{s_1 : State_\bot \mid true\}) \in B_0 \land \bot \notin ac') \\ \lor \\ ((s, \{s_1 : State \mid (s_1, ac' \cup \{\bot\}) \in B_1\}) \in B_0 \land \bot \notin ac') \end{array} \right) \end{array}$$

C.2.3 Isomorphism: $d2bmb$ and $bmb2d$

Theorem T.4.3.3 *Provided B is* **BMH0**-**BMH2***-healthy,*

$$d2bmb \circ bmb2d(B) = B$$

Theorem T.4.3.4 *Provided P is an* **A***-healthy design,*

$$bmb2d \circ d2bmb(P) = P$$

C.3 Refinement and Extreme Points

Theorem T.4.4.1 $\mathbf{A}(\bot_{\mathcal{D}}) = \bot_{\mathcal{D}}$

Theorem T.4.4.2 $\mathbf{A}(\top_{\mathcal{D}}) = \top_{\mathcal{D}}$

Theorem T.4.4.3 *Provided B_0 and B_1 are* **BMH0**-**BMH2**-*healthy,*

$$bmb2d(B_0) \sqsubseteq_{\mathcal{D}} bmb2d(B_1) \Leftrightarrow B_0 \sqsubseteq_{BM_\perp} B_1$$

Theorem T.C.3.1 *Provided that P is an angelic design, $\perp_{\mathcal{D}ac} \sqsubseteq_{\mathcal{D}} P \sqsubseteq_{\mathcal{D}} \top_{\mathcal{D}ac}$*

Lemma L.C.3.1 $[(\exists\, ac' \bullet P^f) = P^f] \Leftrightarrow [(\exists\, ac' \bullet \neg\, P^f) = \neg\, P^f]$

Lemma L.C.3.2 *Provided B_0 and B_1 are of type $BM_\perp$,*

$$\left[\begin{array}{l} (s, ac') \in B_1 \Rightarrow (s, ac') \in B_0 \\ \wedge \\ (s, ac' \cup \{\perp\}) \in B_1 \Rightarrow (s, ac' \cup \{\perp\}) \in B_0 \end{array}\right] \Leftrightarrow B_1 \subseteq B_0$$

C.4 Operators

C.4.1 Sequential Composition

Theorem T.4.5.1 *Provided ok and ok' are not free in P, Q, R and S, and that $\neg\, P$ and Q are* **PBMH**-*healthy,*

$$(P \vdash Q) ;_{\mathcal{D}ac} (R \vdash S) = (\neg\, (\neg\, P ;_{\mathcal{A}} true) \wedge \neg\, (Q ;_{\mathcal{A}} \neg\, R) \vdash Q ;_{\mathcal{A}} (R \Rightarrow S))$$

Theorem T.4.5.2 *Provided ok and ok' are not free in P, Q, R and S, and that $\neg\, P$ and Q are* **PBMH**-*healthy, and that ac' is not free in P,*

$$(P \vdash Q) ;_{\mathcal{D}ac} (R \vdash S) = (P \wedge \neg\, (Q ;_{\mathcal{A}} \neg\, R) \vdash Q ;_{\mathcal{A}} (R \Rightarrow S))$$

Theorem T.4.5.3 ($;_{\mathcal{D}ac}$-A-closure) *Provided P and Q are* **A**-*healthy and ok, ok' are not free in P and Q,*

$$\mathbf{A}(P ;_{\mathcal{D}ac} Q) = P ;_{\mathcal{D}ac} Q$$

Relationship with Extended Binary Multirelations

Theorem T.4.5.4 *Provided P and Q are* **A**-*healthy designs,*

$$bmb2d(d2bmb(P) ;_{BM_\perp} d2bmb(Q)) = P ;_{\mathcal{D}ac} Q$$

Skip

Theorem T.4.5.5 $\mathbf{A}(I\!I_{\mathcal{D}ac}) = I\!I_{\mathcal{D}ac}$

Theorem T.4.5.6 *Provided P is a design,* $I\!I_{\mathcal{D}ac} ;_{\mathcal{D}ac} P = P$

Theorem T.4.5.7 *Provided P is an* **A***-healthy design,*

$$P ;_{\mathcal{D}ac} I\!I_{\mathcal{D}ac} = ((\neg \exists ac' \bullet P^f) \vdash P^t)$$

Theorem T.4.5.8 *Provided P is an* **A***-healthy design, it is* **H3***-healthy if, and only if, its precondition does not mention ac',*

$$(P ;_{\mathcal{D}ac} I\!I_{\mathcal{D}ac}) = P \Leftrightarrow ((\exists ac' \bullet \neg P^f) = \neg P^f)$$

Properties with respect to the Extreme Points

Theorem T.4.5.9 $\bot_{\mathcal{D}} ;_{\mathcal{D}ac} P = \bot_{\mathcal{D}}$

Theorem T.4.5.10 $\top_{\mathcal{D}} ;_{\mathcal{D}ac} P = \top_{\mathcal{D}}$

Properties with respect to A2

Theorem T.C.4.1 *Provided P and Q are* **A2***-healthy,* $\mathbf{A2}(P ;_{\mathcal{D}ac} Q) = P ;_{\mathcal{D}ac} Q$

Other Properties

Lemma L.C.4.1 *Provided P is* **PBMH***-healthy and ok' is not free in P.*

$$P ;_{\mathcal{D}ac} Q \Rightarrow P ;_{\mathcal{A}} (\exists ok \bullet Q)$$

C.4.2 Demonic Choice

Properties

Theorem T.4.5.11 *Provided P and Q are designs,*

$$\mathbf{A}(P \vee Q) = \mathbf{A}(P) \vee \mathbf{A}(Q)$$

Theorem T.4.5.12 *Provided P and Q are* **A***-healthy designs,*

$$\mathbf{A}(P \sqcap_{\mathcal{D}ac} Q) = P \sqcap_{\mathcal{D}ac} Q$$

Relationship with Extended Binary Multirelations

Theorem T.4.5.13 $bmb2p(B_0 \sqcap_{BM_\perp} B_1) = bmb2p(B_0) \sqcap_{\mathcal{D}ac} bmb2p(B_1)$

Other Properties

Theorem T.4.5.14 $P \sqcap_{\mathcal{D}ac} \perp_{\mathcal{D}} = \perp_{\mathcal{D}}$

Theorem T.4.5.15 $(P \sqcap_{\mathcal{D}ac} Q) \mathbin{;_{\mathcal{D}ac}} R = (P \mathbin{;_{\mathcal{D}ac}} R) \sqcap_{\mathcal{D}ac} (Q \mathbin{;_{\mathcal{D}ac}} R)$

Other Properties

Lemma L.C.4.2 *Provided* $P \Rightarrow R$, $P \mathbin{;_{\mathcal{D}ac}} Q \Rightarrow R \mathbin{;_{\mathcal{D}ac}} Q$.

Lemma L.C.4.3 *Provided* $Q \Rightarrow R$, $P \mathbin{;_{\mathcal{D}ac}} Q \Rightarrow P \mathbin{;_{\mathcal{D}ac}} R$.

Lemma L.C.4.4 *Provided ok' is not free in P and ok is not free in Q,*

$$P \mathbin{;_{\mathcal{D}ac}} Q = P \mathbin{;_{\mathcal{A}}} Q$$

C.4.3 Angelic Choice

Closure

Theorem T.4.5.16 *Provided P and Q are* **A***-healthy,*

$$\mathbf{A}(P \sqcup_{\mathcal{D}ac} Q) = P \sqcup_{\mathcal{D}ac} Q$$

Relationship with Extended Binary Multirelations

Theorem T.4.5.17 *Provided B_0 and B_1 are* **BMH1***-healthy,*

$$bmb2p(B_0 \sqcup_{BM_\perp} B_1) = bmb2p(B_0) \sqcup_{\mathcal{D}ac} bmb2p(B_1)$$

Properties with respect to the Extreme Points

Theorem T.4.5.18 *Provided P is a design,* $P \sqcup_{\mathcal{D}ac} \top_{\mathcal{D}} = \top_{\mathcal{D}}$.

C.5 Relationship with Angelic Designs

C.5.1 *d2ac*

Theorem T.4.6.6 $\mathbf{A} \circ d2ac(P) = d2ac(P)$

C.5.2 $p2ac$

Properties

Lemma L.4.6.1 $\mathbf{PBMH} \circ p2ac(P) = p2ac(P)$

Theorem T.4.6.1 $p2ac(P \lor Q) = p2ac(P) \lor p2ac(Q)$

Theorem T.4.6.2 $p2ac(P \land Q) \Rightarrow p2ac(P) \land p2ac(Q)$

Theorem T.4.6.3 $\mathbf{A2} \circ p2ac(P) = p2ac(P)$

Theorem T.4.6.4

$$ac' \neq \emptyset \land p2ac(\neg P^f \vdash P^t) = ac' \neq \emptyset \land (\neg p2ac(P^f) \vdash p2ac(P^t))$$

Theorem T.4.6.5 *Provided P is a design,*

$$ac' \neq \emptyset \land p2ac(P) = ac' \neq \emptyset \land d2ac(P)$$

Lemmas

Lemma L.C.5.1 *Provided c is a condition.*

$$p2ac(P \lhd c \rhd Q) = p2ac(P) \lhd s.c \rhd p2ac(Q)$$

Lemma L.C.5.2 $p2ac(true) = ac' \neq \emptyset$

Lemma L.C.5.3 $p2ac(false) = false$

Lemma L.C.5.4 $\exists\, out\alpha_{-ok'} \bullet P = \exists\, z \bullet P[\mathbf{z}/out\alpha_{-ok'}]$

Lemma L.C.5.5 *Provided that no variable in $in\alpha_{-ok} \cup out\alpha_{-ok'}$ is free in P,*

$$p2ac(P \land Q) = P \land p2ac(Q)$$

Lemma L.C.5.6 *Provided that no variable in $in\alpha_{-ok} \cup out\alpha_{-ok'}$ is free in P,*

$$p2ac(P) = P \land ac' \neq \emptyset$$

Lemma L.C.5.7 *Provided that no dashed variable in $out\alpha_{-ok}$ is free in P,*

$$p2ac(P) = P[\mathbf{s}/in\alpha] \land ac' \neq \emptyset$$

Lemma L.5.3.1 $p2ac \circ ac2p(P) = \exists\, ac_0, y \bullet P[ac_0/ac'] \land ac_0 \subseteq \{y\} \land y \in ac'$

Lemma L.C.5.8 $p2ac(P)^o_w = p2ac(P^o_w)$

Lemma L.C.5.9 $p2ac(P) \Rightarrow ac' \neq \emptyset$

Lemma L.C.5.10 *Provided ac' is not free in P nor Q,*

$$p2ac(P \land Q)[\{y\} \cap ac'/ac'] = (p2ac(P) \land p2ac(Q))[\{y\} \cap ac'/ac']$$

Lemma L.C.5.11

$$\begin{array}{l} p2ac(P)[\{undash(State_{II}(out\alpha_{-ok'}))\} \cap ac'/ac'] \\ = \\ P[\mathbf{s}/in\alpha_{-ok}] \land undash(State_{II}(out\alpha_{-ok'})) \in ac' \end{array}$$

Lemma L.C.5.12 *Provided ac' is not free in P,*

$$p2ac(P)[\{y \mid e\} \cap ac'/ac'] = p2ac(P \land e[z/y])$$

Lemma L.C.5.13 *Provided ac' is not free in P nor in Q,*

$$p2ac(P \land Q) = \exists\, x \bullet p2ac(P)[\{x\}/ac'] \land p2ac(Q)[\{x\}/ac'] \land x \in ac'$$

Lemma L.C.5.14 *Provided that ac' is not free in P,*

$$p2ac(P \land Q) = \exists\, z \bullet \left(\begin{array}{l} P[\mathbf{s}, \mathbf{z}/in\alpha_{-ok}, out\alpha_{-ok'}] \\ \land \\ p2ac(Q)[\{undash(z)\}/ac'] \land undash(z) \in ac' \end{array} \right)$$

Lemma L.C.5.15 *Provided z is not ac', $\exists\, x \bullet p2ac(P) = p2ac(\exists\, x \bullet p2ac(P))$.*

Lemma L.C.5.16 $p2ac(P)[o/ok] = p2ac([o/ok])$

Lemma L.C.5.17

$$p2ac(P \,;\, Q) = \exists\, z \bullet (P[\mathbf{s}/in\alpha_{-ok}] \,;\, Q[\mathbf{z}/out\alpha_{-ok'}]) \land undash(z) \in ac'$$

Lemma L.C.5.18 *Provided ac' is not free in P,*

$$p2ac(P \,;\, Q) = P[\mathbf{s}/in\alpha_{-ok}] \,;\, (\exists\, z \bullet Q[\mathbf{z}/out\alpha_{-ok'}] \land undash(z) \in ac')$$

C.5.3 *ac2p*

Properties

Theorem T.C.5.1 $ac2p(P \lor Q) = ac2p(P) \lor ac2p(Q)$

Theorem T.C.5.2 *Provided P and Q are* **PBMH**-*healthy,*

$$ac2p(P \land Q) = ac2p(P) \land ac2p(Q)$$

Lemmas

Lemma L.4.6.2 (*ac2p*-alternative-1)

$$ac2p(P) = \exists\, ac' \bullet \left(\begin{array}{l} P[State_{II}(in\alpha)/s] \\ \land \\ \forall\, z \bullet z \in ac' \Rightarrow (\bigwedge x : out\alpha \bullet dash(z).x = x) \end{array}\right)$$

Lemma L.C.5.19 (*ac2p*-alternative-2)

$$ac2p(P) \mathrel{\widehat{=}} \left(\begin{array}{l} \exists\, ac', s \bullet P \land (\forall\, z \bullet z \in ac' \Rightarrow \bigwedge x : out\alpha_{-ok'} \bullet dash(z).x = x) \\ \land \\ (\bigwedge x : in\alpha_{-ok} \bullet s.x = x) \end{array}\right)$$

Lemma L.C.5.20 (*ac2p*-alternative-3)

$$\begin{array}{l} ac2p(P) \\ = \\ \exists\, ac' \bullet P[State_{II}(in\alpha_{-ok})/s] \land ac' \subseteq \{s \mid \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x\} \end{array}$$

Lemma L.C.5.21 *Provided ac' is not free in e,*

$$ac2p(\exists\, y \bullet y \in ac' \land e) = e[State_{II}(in\alpha_{-ok}), undash(State_{II}(out\alpha_{-ok'}))/s, y]$$

Lemma L.C.5.22 *Provided P is* **A2**-*healthy,*

$$ac2p(P) = \left(\begin{array}{l} \exists\, ac_0 \bullet P[\{s \mid \{s\} = ac_0\}/ac'][State_{II}(in\alpha_{-ok})/s] \\ \land \\ ac_0 \subseteq \{s \mid \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x\} \end{array}\right)$$

Lemma L.C.5.23

$$\exists\, out\alpha \bullet \neg\, ac2p(P)[\mathbf{s}/in\alpha] \Rightarrow \neg\, P[\emptyset/ac']$$

The following lemma can be restated in a few different ways. Namely it can also imply:

$$\exists\, out\alpha \bullet (\neg\, P[State_{\mathit{II}}(in\alpha)/s] \;;_{\mathcal{A}} \bigwedge x : out\alpha \bullet dash(s).x = x)$$

Lemma L.C.5.24 *Provided P is* **PBMH***-healthy,*

$$\exists\, out\alpha \bullet \neg\, ac2p(P) \Rightarrow \exists\, out\alpha \bullet ac2p(\neg\, P)$$

Lemma L.C.5.25 *Provided none of the variables in $out\alpha$ are free in P,*

$$\exists\, out\alpha \bullet ac2p(P) \Rightarrow \exists\, ac' \bullet P[State_{\mathit{II}}(in\alpha)/s]$$

Lemma L.C.5.26 *Provided that s and ac' are not free in P,*

$$ac2p(P \wedge Q) = P \wedge ac2p(Q)$$

Lemma L.C.5.27 *Provided that s and ac' are not free in P,*

$$ac2p(P) = P$$

Lemma L.C.5.28 *Provided P is a design,*

$$ac2p(P) = (\neg\, ac2p(P^f) \vdash ac2p(P^t))$$

Lemma L.C.5.29 $ac2p(P) \Rightarrow \exists\, ac' \bullet P[State_{\mathit{II}}(in\alpha)/s]$

Lemma L.C.5.30 *Provided ac' is not free in P,*

$$ac2p(P) = P[State_{II}(in\alpha)/s]$$

Lemma L.C.5.31 $ac2p(P)^o_w = ac2p(P^o_w)$

Lemma L.C.5.32 *Provided ac' is not free in c,*

$$ac2p(P \lhd c \rhd Q) = ac2p(P) \lhd c[State_{\mathit{II}}(in\alpha_{-ok})/s] \rhd ac2p(Q)$$

Lemma L.C.5.33 *Provided ac' is not free in P,*

$$ac2p(P \land Q) = P[State_{II}(in\alpha_{-ok})/s] \land ac2p(Q)$$

Lemma L.C.5.34 *Provided $in\alpha_{-ok} = \{x_0, \ldots, x_i\}$ and $in\alpha'_{-ok} = out\alpha_{-ok'}$,*

$$ac2p(s \in ac') = x_0 = x'_0 \land \ldots \land x_i = x'_i$$

Lemma L.C.5.35 *Provided P is* **PBMH***-healthy,*

$$ac2p(P \land ac' \neq \emptyset) = ac2p(P)$$

Lemma L.C.5.36 $ac2p \circ \mathbf{PBMH}(P) = ac2p(P)$

Lemma L.C.5.37 *Provided that x is not s nor ac', $ac2p(\exists\, x \bullet P) = \exists\, x \bullet ac2p(P)$*

Lemma L.C.5.38 $ac2p(y \in ac') = \bigwedge x : out\alpha_{-ok'} \bullet dash(y[State_{II}(in\alpha_{-ok})/s]).x = x$

Lemma L.C.5.39 *Provided y is not s, $ac2p(y \in ac') = \bigwedge x : out\alpha_{-ok'} \bullet dash(y).x = x$*

Lemma L.C.5.40 *Provided P is* **PBMH***-healthy and y is not s,*

$$\exists\, y \bullet ac2p(P \land y \in ac') = ac2p(P)[undash(State_{II}(out\alpha_{-ok'})/y]$$

Lemma L.C.5.41 *Provided P is* **PBMH***-healthy,*

$$ac2p(\circledin^{y}_{ac'}(P)) = ac2p(P[\{y\} \cap ac'/ac'])[undash(State_{II}(out\alpha_{-ok'})/y]$$

Lemma L.C.5.42 *Provided P is* **PBMH***-healthy,*

$$\begin{array}{l} ac2p(\circledin^{y}_{ac'}(P)) \\ = \\ ac2p(P[undash(State_{II}(out\alpha_{-ok'}))/y][\{undash(State_{II}(out\alpha_{-ok'}))\} \cap ac'/ac']) \end{array}$$

Lemma L.C.5.43 *Provided P and Q are* **PBMH***-healthy, y is not free in P and ac' is not free in Q,*

$$\begin{array}{l} ac2p(\circledin^{y}_{ac'}(P \land Q)) \\ = \end{array}$$

$$\left(\begin{array}{l} ac2p(P[\{undash(State_{II}(out\alpha_{-ok'}))\} \cap ac'/ac']) \\ \land \\ Q[undash(State_{II}(out\alpha_{-ok'}))/y][State_{II}(in\alpha_{-ok})/s] \end{array}\right)$$

Lemma L.C.5.44 *Provided that ac' is not free in P, and s and ac' are not free in e, and that y is not ac' nor s,*

$$ac2p(P)[e/y] = ac2p(P[e/y])$$

Lemma L.C.5.45 *Provided ac' is not free in P,*

$$ac2p(P[\mathbf{s}/in\alpha_{-ok}] \land undash(State_{II}(out\alpha_{-ok'})) \in ac') = P$$

Lemma L.C.5.46 $ac2p(undash(State_{II}(out\alpha_{-ok'})) \in ac') = true$

Properties with respect to Angelic Designs

Theorem T.C.5.3 *Provided that P is a design,*

$$ac2p \circ \mathbf{A}(P) = (\neg\ ac2p(P^f) \vdash ac2p(P^t))$$

C.5.4 Isomorphism and Galois Connection ($d2ac$ and $ac2p$)

Theorem T.4.6.7 *Provided that P is a design, $ac2p \circ d2ac(P) = P$.*

Theorem T.4.6.8 *Provided P is an $\mathbf{A}$-healthy design, $d2ac \circ ac2p(P) \sqsupseteq P$.*

Theorem T.4.6.9 *Provided P is an $\mathbf{A0}$-$\mathbf{A2}$-healthy design, $d2ac \circ ac2p(P) \sqsubseteq P$.*

Theorem T.4.6.10 *Provided P is a design that is $\mathbf{A0}$-$\mathbf{A2}$-healthy,*

$$d2ac \circ ac2p(P) = P$$

Lemma L.C.5.47

$$\begin{array}{l} d2ac \circ ac2p(P) \\ = \\ (\neg\ p2ac(ac2p(P^f)) \land (\exists\ out\alpha \bullet \neg\ ac2p(P^f)[\mathbf{s}/in\alpha]) \vdash p2ac(ac2p(P^t))) \end{array}$$

C.6 Relationship with the PBMH Theory

C.6.1 *d2pbmh*

Theorem T.4.7.1 *Provided P is* **PBMH**-*healthy,*

$$\mathbf{PBMH} \circ d2pbmh(P) = d2pbmh(P)$$

Lemma L.C.6.1

$$\begin{array}{l} d2pbmh \circ \mathbf{PBMH}(P) \\ = \\ \exists\, ac_0 \bullet \left(\begin{array}{l} (\neg\, P^f \Rightarrow P^t)[true/ok][State_{\mathcal{II}}(in\alpha_{-ok})/s][ac_0/ac'] \\ \land \\ ac_0 \subseteq undashset(ac') \end{array}\right) \end{array}$$

C.6.2 *pbmh2d*

Theorem T.4.7.2 *Provided P is* **PBMH**-*healthy,*

$$\mathbf{A} \circ \mathbf{H3} \circ pbmh2d(P) = pbmh2d(P)$$

C.6.3 Galois Connection and Isomorphism (*d2pbmh* and *pbmh2d*)

Theorem T.4.7.3 *Provided P is* **PBMH**-*healthy, $d2pbmh \circ pbmh2d(P) = P$.*

Theorem T.4.7.4 *Provided P is an* **A**-*healthy design,*

$$pbmh2d \circ d2pbmh(P) \sqsubseteq P$$

Theorem T.4.7.5 *Provided P is design that is* **A** *and* **H3**-*healthy,*

$$pbmh2d \circ d2pbmh(P) = P$$

Lemma L.C.6.2 *Provided f is bijective,*

$$\mathbf{PBMH}(P)[f(ac')/ac'] = \mathbf{PBMH}(P[f(ac')/ac'])$$

Lemma L.C.6.3 $P \land ac' = \emptyset = P[\emptyset/ac'] \land ac' = \emptyset$

Lemma L.C.6.4

$$pbmh2d \circ d2pbmh(P) = (\neg\ P^f[\emptyset/ac'] \land \neg\ P^t[\emptyset/ac'] \vdash (\neg\ P^f \Rightarrow P^t))$$

Appendix D

State Substitution Rules

D.1 State Substitution

The substitution operator $[\mathbf{s}/S\alpha]$, where the boldface indicates that s is a record, is defined for an arbitrary set of variables $S\alpha$ as follows.

Definition 101 $P[\mathbf{z}/S\alpha] \mathrel{\widehat{=}} P[z.s_0, \ldots, z.s_n/s_0, \ldots, s_n]$

Each variable s_i in $S\alpha$ is replaced with $z.s_i$. As an example, we consider the substitution $(x' = 2 \wedge ok')[\mathbf{s}, \mathbf{z}/in\alpha_{-ok}, out\alpha_{-ok'}]$, whose result is $z.x' = 2 \wedge ok'$. The substitution $[\mathbf{z}/S\alpha]$ is well-formed whenever $S\alpha$ is a subset of the record components of z.

Lemma L.D.1.1 *Provided that $A\alpha \cap B\alpha = \emptyset$, $A\alpha \subseteq S\alpha$ and $B\alpha \subseteq S\alpha$,*

$$P[\mathbf{z}/S\alpha] = P[\mathbf{z}/A\alpha][\mathbf{z}/B\alpha]$$

Lemma L.D.1.2 *Provided that $A\alpha \cap B\alpha = \emptyset$, $A\alpha \subseteq S\alpha$ and $B\alpha \subseteq S\alpha$,*

$$P[\mathbf{z}/S\alpha] = \left(\begin{array}{l} \exists\, z_A, z_B \bullet P[\mathbf{z}_A/A\alpha][\mathbf{z}_B/B\alpha] \\ \wedge \\ (\bigwedge x : A\alpha \bullet z_A.x = z.x) \wedge (\bigwedge x : B\alpha \bullet z_B.x = z.x) \end{array}\right)$$

Lemma L.D.1.3 *Provided $z, y : State(S\alpha)$,*

$$P[\mathbf{z}/S\alpha][y \oplus \{s_i \mapsto e\}/z] = P[\mathbf{y}/(S\alpha \setminus \{s_i\})][e/s_i]$$

Lemma L.D.1.4 *Provided $z, y : State(S\alpha)$ and s_i not free in e,*

$$P[\mathbf{z}/S\alpha][y \oplus \{s_i \mapsto e\}/z] = P[e/s_i][\mathbf{y}/(S\alpha)]$$

Lemma L.D.1.5 *Provided $s_i \in S\alpha$,*

$$P[e/s_i][\mathbf{z}/S\alpha] = P[\mathbf{z}/S\alpha \setminus \{s_i\}][e[\mathbf{z}/S\alpha]/s_i]$$

Lemma L.D.1.6 $P[z/(S\alpha \cup T\alpha)] = P[z/S\alpha][z/T\alpha]$

Lemma L.D.1.7

$$\begin{array}{l} P[e_0, \ldots, e_n/x_0, \ldots, x_n][z/S\alpha] \\ = \\ P[z/(S\alpha \setminus T\alpha)][e_0[z/T\alpha], \ldots, e_n[z/T\alpha]/x_0, \ldots, x_n] \end{array}$$

Provided that:

1. $T\alpha \subseteq S\alpha$
2. $T\alpha = \{x_0, \ldots, x_n\}$
3. $\forall\, y \bullet y \in (S\alpha \setminus T\alpha) \Rightarrow y \notin fv(e_0, \ldots, e_n)$

Definition 140 *For $S\alpha = \{x_0, \ldots, x_n\}$,*

$$State_{II}(S\alpha) \mathrel{\widehat=} \{x_0 \mapsto x_0, \ldots, x_n \mapsto x_n\}$$

Lemma L.D.1.8 $State_{II}(S\alpha)' = \{x_0' \mapsto x_0, \ldots, x_n' \mapsto x_n\}$

Lemma L.D.1.9

$$\exists\, z : State(S\alpha) \bullet P \wedge (\bigwedge x : S\alpha \bullet z.x = x) = P[State_{II}(S\alpha)/z]$$

Lemma L.D.1.10 *Provided z is not free in P,*

$$P[\mathbf{z}/S\alpha][State_{II}(S\alpha)/z] = P$$

Lemma L.D.1.11 *Provided none of the varibles in $S\alpha$ are free in P,*

$$P[State_{II}(S\alpha)/z][\mathbf{z}/S\alpha] = P$$

Lemma L.D.1.12 *Provided $x_i \in S\alpha$ and x_i is not free in P nor in e,*

$$P[State_{II}(S\alpha)/z][e/x_i] = P[z \oplus x_i \mapsto e\}/z][State_{II}(S\alpha)/z]$$

D.2 *dash* and *undash*

Definition 141

$$\begin{aligned} dash(z) &\mathrel{\widehat=} \{x : S\alpha, e \mid (x \mapsto e) \in z \bullet x' \mapsto e\} \\ undash(z) &\mathrel{\widehat=} \{x : S\alpha, e \mid (x' \mapsto e) \in z \bullet x \mapsto e\} \end{aligned}$$

The function *dash* considers every pair (x, e) in z, where x is a variable name and e the corresponding expression or value associated with x, and dashes the name of x into x'. Function *undash* is similar except for the undash of x' to x.

Lemma L.D.2.1 $dash(z).x' = z.x$

Lemma L.D.2.2 $undash(z).x = z.x'$

Lemma L.D.2.3 $undash \circ dash(z) = z$

Lemma L.D.2.4 $dash \circ undash(z) = z$

Lemma L.D.2.5 *Provided y is fresh,*

$$\exists z \bullet P \land undash(z) \in ac' = \exists y \bullet P[dash(y)/z] \land y \in ac'$$

Appendix E

PBMH

E.1 Definition

Definition 88 $\mathbf{PBMH}(P) \mathrel{\widehat=} P \mathbin{;} ac \subseteq ac' \land ok' = ok$

E.2 Properties

Lemma L.E.2.1 $P \Rightarrow \mathbf{PBMH}(P)$

Theorem T.E.2.1 $\mathbf{PBMH} \circ \mathbf{PBMH}(P) = \mathbf{PBMH}(P)$

Theorem T.E.2.2 $\mathbf{PBMH}(P \lor Q) = \mathbf{PBMH}(P) \lor \mathbf{PBMH}(Q)$

Lemma L.E.2.2 *Provided P satisfies* **PBMH***,* $P[\emptyset/ac'] \lor P = P$

E.3 Closure Properties

Lemma L.E.3.1 *Provided P and Q satisfy* **PBMH***,*

$$\mathbf{PBMH}(P \land Q) = \mathbf{PBMH}(P) \land \mathbf{PBMH}(Q)$$

Lemma L.E.3.2 $\mathbf{PBMH}(P \land Q) \Rightarrow \mathbf{PBMH}(P) \land \mathbf{PBMH}(Q)$

Theorem T.E.3.1 *Provided P and Q are* **PBMH***-healthy,*

$$\mathbf{PBMH}(P \land Q) = P \land Q$$

Theorem T.E.3.2 *Provided P and Q satisfy* **PBMH***,*

$$\mathbf{PBMH}(P \lor Q) = P \lor Q$$

E.4 Lemmas

Lemma L.4.2.1 $\mathbf{PBMH}(P) = \exists\, ac_0 \bullet P[ac_0/ac'] \land ac_0 \subseteq ac'$

Lemma L.E.4.1 $\mathbf{PBMH}(\mathit{true}) = \mathit{true}$

Lemma L.E.4.2 $\mathbf{PBMH}(\mathit{false}) = \mathit{false}$

Lemma L.E.4.3 $\mathbf{PBMH}(s \in ac') = s \in ac'$

Lemma L.E.4.4 $\mathbf{PBMH}(ac' \neq \emptyset) = ac' \neq \emptyset$

Lemma L.E.4.5 *Provided ac' is not free in P,* $\mathbf{PBMH}(P) = P$.

Lemma L.E.4.6 *Provided c is a condition,* $\mathbf{PBMH}(c) = c$.

Lemma L.E.4.7 $\mathbf{PBMH}(x \in ac') = x \in ac'$

Lemma L.E.4.8 *Provided ac' is not free in c,* $\mathbf{PBMH}(c \land P) = c \land \mathbf{PBMH}(P)$

Lemma L.E.4.9 *Provided ac' is not free in c,*

$$\mathbf{PBMH}(P \lhd c \rhd Q) = \mathbf{PBMH}(P) \lhd c \rhd \mathbf{PBMH}(Q)$$

Lemma L.E.4.10 *Provided ac' is not free in e,*

$$\mathbf{PBMH}(\exists\, y \bullet y \in ac' \land e) = \exists\, y \bullet y \in ac' \land e$$

Lemma L.E.4.11

$$\begin{aligned}
&(P \land ac' \neq \emptyset) \mathbin{;_{\mathcal{A}}} (Q \land ac' \neq \emptyset)\\
&=\\
&(P \land ac' \neq \emptyset) \mathbin{;_{\mathcal{A}}} (Q \land ac' \neq \emptyset)) \land ac' \neq \emptyset
\end{aligned}$$

Lemma L.E.4.12 $\mathbf{PBMH}(P \mathbin{;} ac = \emptyset) = P \mathbin{;} ac = \emptyset$

Lemma L.E.4.13 *Provided ac_1 is not free in $F(x)$,*

$$\exists\, ac_1 \bullet (\forall\, x \bullet x \in ac_0 \Rightarrow F(x) \in ac_1) \land ac_1 \subseteq ac'$$
$$\Leftrightarrow$$
$$\forall\, x \bullet x \in ac_0 \Rightarrow F(x) \in ac'$$

Lemma L.E.4.14 $P \sqsubseteq Q \Leftrightarrow [\{ac' \mid Q\} \subseteq \{ac' \mid P\}]$

Lemma L.E.4.15 $\mathbf{PBMH}(P) \Rightarrow \exists\, ac' \bullet P$

Lemma L.E.4.16 $\mathbf{PBMH}(P) \mathbin{;_{\mathcal{A}}} \mathit{true} = \exists\, ac' \bullet P$

E.5 Substitution Lemmas

Lemma L.E.5.1 $\mathbf{PBMH}(P)^o_w = \mathbf{PBMH}(P^o_w)$

Lemma L.E.5.2 *Provided ac' is not free in e,*

$$\mathbf{PBMH}(P)[e/s] = \mathbf{PBMH}(P[e/s])$$

Lemma L.E.5.3 *Provided x is not ac',* $\mathbf{PBMH}(\exists\, x \bullet P) = \exists\, x \bullet \mathbf{PBMH}(x)$

Lemma L.E.5.4 *Provided P is* **PBMH***-healthy,*

$$\mathbf{PBMH}(P[\{y\} \cap ac'/ac']) = \mathbf{PBMH}(P)[\{y\} \cap ac'/ac']$$

Lemma L.E.5.5

$$\mathbf{PBMH}(P)[o/ok] = \mathbf{PBMH}(P[o/ok])$$

E.6 Properties with respect to Designs

Lemma L.4.2.2 $\mathbf{PBMH}(P \vdash Q) = (\neg\, \mathbf{PBMH}(\neg\, P) \vdash \mathbf{PBMH}(Q))$

Lemma L.E.6.1 $J \mathbin{;} (ac \subseteq ac' \land ok' = ok) = (ac \subseteq ac' \land ok' = ok) \mathbin{;} J$

Lemma L.E.6.2 $\mathbf{PBMH}(\neg\, \mathbf{PBMH}(\neg\, P) \vdash Q) = \mathbf{PBMH}(P \vdash Q)$

Theorem T.E.6.1 $\mathbf{H2} \circ \mathbf{PBMH}(P) = \mathbf{PBMH} \circ \mathbf{H2}(P)$

Theorem T.E.6.2 $\mathbf{H1} \circ \mathbf{PBMH}(P) = \mathbf{PBMH} \circ \mathbf{H1}(P)$

E.7 Properties with respect to A2

Lemma L.E.7.1 *Provided P is* **PBMH**-*healthy.*

$$
\begin{array}{l}
\mathbf{PBMH}(P \mathbin{;_{\mathcal{A}}} \{s \mid \{s\} = ac'\}) \\
= \\
\exists\, ac_1, ac_0 \bullet P[ac_0/ac'] \land ac_0 \subseteq \{s \mid \{s\} = ac_1\} \land ac_1 \subseteq ac'
\end{array}
$$

Theorem T.E.7.1 *Provided P is* **PBMH**-*healthy and v is not free in P,*

$$
\exists\, v \bullet (P \mathbin{;_{\mathcal{A}}} Q) \Rightarrow P \mathbin{;_{\mathcal{A}}} (\exists\, v \bullet Q)
$$

Appendix F

Sequential Composition ($\mathcal{A}$)

F.1 Properties

Lemma L.F.1.1 *Provided ac' is not free in P, $P \mathbin{;_{\mathcal{A}}} Q = P$.*

Lemma L.F.1.2 $\neg (P \mathbin{;_{\mathcal{A}}} Q) = (\neg P \mathbin{;_{\mathcal{A}}} Q)$

Lemma L.F.1.3 *Provided P and Q satisfy* **PBMH***,*

$$P \mathbin{;_{\mathcal{A}}} (Q \mathbin{;_{\mathcal{A}}} R) = (P \mathbin{;_{\mathcal{A}}} Q) \mathbin{;_{\mathcal{A}}} R$$

Lemma L.F.1.4 $(P \lor Q) \mathbin{;_{\mathcal{A}}} R = (P \mathbin{;_{\mathcal{A}}} R) \lor (Q \mathbin{;_{\mathcal{A}}} R)$

Lemma L.F.1.5 $(P \land Q) \mathbin{;_{\mathcal{A}}} R = (P \mathbin{;_{\mathcal{A}}} R) \land (Q \mathbin{;_{\mathcal{A}}} R)$

Lemma L.F.1.6 *Provided P is* **PBMH***-healthy,*

$$P \mathbin{;_{\mathcal{A}}} (Q \land R) \Rightarrow (P \mathbin{;_{\mathcal{A}}} Q) \land (P \mathbin{;_{\mathcal{A}}} R)$$

F.2 Lemmas

Lemma L.F.2.1 *Provided P is* **PBMH***-healthy,*

$$(P \mathbin{;_{\mathcal{A}}} Q) \lor (P \mathbin{;_{\mathcal{A}}} R) \Rightarrow (P \mathbin{;_{\mathcal{A}}} (Q \lor R))$$

Lemma L.F.2.2 *Provided P is* **PBMH***-healthy,*

$$(P \mathbin{;_{\mathcal{A}}} Q) \lor (P \mathbin{;_{\mathcal{A}}} true) = P \mathbin{;_{\mathcal{A}}} true$$

Lemma L.F.2.3 *Provided P is* **PBMH**-*healthy,*

$$(P \mathbin{;_{\mathcal{A}}} Q) \lor (P \mathbin{;_{\mathcal{A}}} \mathit{false}) = P \mathbin{;_{\mathcal{A}}} Q$$

Lemma L.F.2.4 *Provided P is* **PBMH**-*healthy,*

$$P \mathbin{;_{\mathcal{A}}} (Q \Rightarrow (R \land ok')) = (P \mathbin{;_{\mathcal{A}}} \lnot Q) \lor ((P \mathbin{;_{\mathcal{A}}} (Q \Rightarrow R)) \land ok')$$

Lemma L.F.2.5 *Provided x is not free in e,*

$$\begin{array}{l} \forall x \bullet P \Rightarrow (Q \Rightarrow (R \land e)) \\ = \\ (\forall x \bullet P \Rightarrow \lnot Q) \lor ((\forall x \bullet P \Rightarrow (Q \Rightarrow R)) \land e) \end{array}$$

Lemma L.F.2.6 *Provided P is* **PBMH**-*healthy,*

$$P \mathbin{;_{\mathcal{A}}} (Q \land ok') = (P \mathbin{;_{\mathcal{A}}} \mathit{false}) \lor ((P \mathbin{;_{\mathcal{A}}} Q) \land ok')$$

Lemma L.F.2.7 *Provided s is not free in R and P is* **PBMH**-*healthy,*

$$(P \mathbin{;_{\mathcal{A}}} (Q \land R)) \land R = (P \mathbin{;_{\mathcal{A}}} Q) \land R$$

Lemma L.F.2.8 *Provided ac' is not free in P,*

$$(P \land Q) \mathbin{;_{\mathcal{A}}} R = P \land (Q \mathbin{;_{\mathcal{A}}} R)$$

Lemma L.F.2.9 *Provided P is* **PBMH**-*healthy and s is not free in e,*

$$P \mathbin{;_{\mathcal{A}}} (Q \Rightarrow (R \land e)) = (P \mathbin{;_{\mathcal{A}}} \lnot Q) \lor ((P \mathbin{;_{\mathcal{A}}} (Q \Rightarrow R)) \land e)$$

F.3 Closure Properties

Theorem T.F.3.1 *Provided P and Q are* **PBMH**-*healthy,*

$$\mathbf{PBMH}(P \mathbin{;_{\mathcal{A}}} Q) = P \mathbin{;_{\mathcal{A}}} Q$$

F.4 Extreme Points

Lemma L.F.4.1 *Provided P is* **PBMH***-healthy,*

$$P \mathbin{;_{\mathcal{A}}} \mathbf{false} = P[\emptyset/ac']$$

Lemma L.F.4.2 *Provided P is* **PBMH***-healthy,*

$$P \mathbin{;_{\mathcal{A}}} \mathbf{true} = \exists\, ac' \bullet P$$

F.5 Algebraic Properties and Sequential Composition

Lemma L.F.5.1 *Provided ok and ac are not free in R,*

$$(P \mathbin{;} Q) \mathbin{;_{\mathcal{A}}} R = P \mathbin{;} (Q \mathbin{;_{\mathcal{A}}} R)$$

F.6 Skip

Definition 142 $I\!I_{\mathcal{A}} \mathrel{\widehat=} s \in ac'$

Lemma L.F.6.1 *$I\!I_{\mathcal{A}}$ is a fixed point of* **PBMH***,* $\mathbf{PBMH}(I\!I_{\mathcal{A}}) = I\!I_{\mathcal{A}}$.

Lemma L.F.6.2 $I\!I_{\mathcal{A}} \mathbin{;_{\mathcal{A}}} P = P$

Lemma L.F.6.3 *Provided P is* **PBMH***-healthy, $P \mathbin{;_{\mathcal{A}}} I\!I_{\mathcal{A}}$.*

Appendix G

Reactive Angelic Designs (RAD)

G.1 RA1

G.1.1 Definition

Definition 109 $\mathbf{RA1}(P) \mathrel{\widehat{=}} (P \land ac' \neq \emptyset)[States_{tr \leq tr'}(s) \cap ac'/ac']$

G.1.2 Properties

Theorem T.5.2.1 $\mathbf{RA1} \circ \mathbf{A0}(P) = \mathbf{RA1}(P)$

Theorem T.5.2.2 $\mathbf{RA1}(P \land Q) = \mathbf{RA1}(P) \land \mathbf{RA1}(Q)$

Theorem T.5.2.3 $\mathbf{RA1}(P \lor Q) = \mathbf{RA1}(P) \lor \mathbf{RA1}(Q)$

Theorem T.5.2.4 *Provided P and Q are* **RA1***-healthy and Q is* **PBMH***-healthy,*

$$\mathbf{RA1}(P \mathbin{;_{\mathcal{A}}} Q) = P \mathbin{;_{\mathcal{A}}} Q$$

Theorem T.5.2.5 $\mathbf{PBMH} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P) = \mathbf{RA1} \circ \mathbf{PBMH}(P)$

Theorem T.G.1.1 $\mathbf{RA1} \circ \mathbf{RA1}(P) = \mathbf{RA1}(P)$

Theorem T.G.1.2 $P \sqsubseteq Q \Rightarrow \mathbf{RA1}(P) \sqsubseteq \mathbf{RA1}(Q)$

G.1.3 Lemmas

Lemma L.G.1.1

$\mathbf{RA1}(P)$

$$=$$

$$P[\{z \mid z \in ac' \land s.tr \leq z.tr\}/ac'] \land \exists\, z \bullet s.tr \leq z.tr \land z \in ac'$$

Lemma L.G.1.2

$$\mathbf{RA1}(P) = (P \land ac' \neq \emptyset)[\{z \mid z \in ac' \land z \in \{z \mid s.tr \leq z.tr\}\}/ac']$$

Lemma L.G.1.3 $\mathbf{RA1}(P)[\emptyset/ac'] = \mathit{false}$

Lemma L.G.1.4 $\mathbf{RA1}(\mathit{true})[\{y\}/ac'] = s.tr \leq y.tr$

Lemma L.G.1.5 *Provided y is not s and not ac',*

$$\mathbf{RA1}(\exists\, y \bullet P[\{y\}/ac'] \land y \in ac')$$

$$=$$

$$\exists\, y \bullet P[\{y\}/ac'] \land s.tr \leq y.tr \land y \in ac'$$

Lemma L.G.1.6 $\mathbf{RA1}(P) \Rightarrow ac' \neq \emptyset$

Lemma L.G.1.7 $s \in ac' \Rightarrow \exists\, z \bullet s.tr \leq z.tr \land z \in ac'$

Lemma L.G.1.8

$$\exists\, z \bullet z \in ac' \land tr_0 \leq z.tr \land x = z \oplus \{tr \mapsto z.tr - tr_0\}$$

$$\Leftrightarrow$$

$$x \oplus \{tr \mapsto tr_0 \frown x.tr\} \in ac'$$

Lemma L.G.1.9 $\mathbf{RA1}(\mathit{false}) = \mathit{false}$

Lemma L.G.1.10

$$\mathbf{RA1}(\mathit{true}) = \exists\, z \bullet s.tr \leq z.tr \land z \in ac'$$

Lemma L.G.1.11

$$\mathbf{RA1}(\mathit{true}) = \mathit{States}_{tr \leq tr'}(s) \cap ac' \neq \emptyset$$

Lemma L.G.1.12 *Provided x is not in the set $\{s, ac'\}$,*

$$\mathbf{RA1}(\exists\, x \bullet P) = \exists\, x \bullet \mathbf{RA1}(P)$$

Lemma L.G.1.13 $\mathbf{RA1}(x \in ac') = s.tr \leq x.tr \land x \in ac'$

Lemma L.G.1.14

$$\mathbf{RA1}(s \in ac') = s \in ac'$$

Lemma L.G.1.15 *Provided c is a condition,*

$$\mathbf{RA1}(P \lhd c \rhd Q) = \mathbf{RA1}(P) \lhd c \rhd \mathbf{RA1}(Q)$$

Lemma L.G.1.16 *Provided ac' is not free in P,*

$$\mathbf{RA1}(P \land Q) = P \land \mathbf{RA1}(Q)$$

Lemma L.G.1.17 $\mathbf{RA1}(\neg\, ok) = \neg\, ok \land \mathbf{RA1}(true)$

Lemma L.G.1.18

$$\mathbf{RA1}(\neg\, P^f_f \vdash P^t_f) = \mathbf{RA1}(\neg\, (P^f_f \land ac' \neq \emptyset) \vdash P^t_f \land ac' \neq \emptyset)$$

Lemma L.G.1.19 *Provided ac' is not free in P,*

$$\mathbf{RA1}(P) = P \land \mathbf{RA1}(true)$$

Lemma L.G.1.20 $\mathbf{RA1}(P \vdash Q) = \mathbf{RA1}(P \vdash \mathbf{RA1}(Q))$

Lemma L.G.1.21 *Provided P is* **PBMH**-*healthy,*

$$\mathbf{RA1}(P) \Rightarrow P$$

Lemma L.G.1.22 $\mathbf{RA1}(ac' \neq \emptyset) = \mathbf{RA1}(true)$

Lemma L.G.1.23 $\mathbf{RA1}(P \vdash Q) = \mathbf{RA1}(\neg\, \mathbf{RA1}(\neg\, P) \vdash Q)$

G.1.4 Substitution Properties

Lemma L.G.1.24 $\mathbf{RA1}(P)^o_w = \mathbf{RA1}(P^o_w)$

G.1.5 Properties with respect to $;_\mathcal{A}$

Theorem T.G.1.3

$$\mathbf{RA1}(\mathit{true}) ;_\mathcal{A} (P \lor Q) = (\mathbf{RA1}(\mathit{true}) ;_\mathcal{A} P) \lor (\mathbf{RA1}(\mathit{true}) ;_\mathcal{A} Q)$$

Theorem T.G.1.4 *Provided ac' is not free in P,*

$$\mathbf{RA1}(P) ;_\mathcal{A} (Q \lor R) = (\mathbf{RA1}(P) ;_\mathcal{A} Q) \lor (\mathbf{RA1}(P) ;_\mathcal{A} R)$$

Theorem T.G.1.5 *Provided P is* **PBMH**-*healthy,*

$$\begin{array}{l}(P ;_\mathcal{A} \mathbf{RA1}(\mathit{true})) \lor (P ;_\mathcal{A} \mathbf{RA1}(Q)) \\ = \\ (P ;_\mathcal{A} \mathbf{RA1}(\mathit{true}))\end{array}$$

Lemma L.G.1.25 $\mathbf{RA1}(\mathit{true}) ;_\mathcal{A} \mathit{true}$

Lemma L.G.1.26

$$\mathbf{RA1}(\mathit{true}) ;_\mathcal{A} (s.\mathit{wait} \land \neg\, \mathit{ok} \land \mathbf{RA1}(\mathit{true})) = \neg\, \mathit{ok} \land \mathbf{RA1}(\mathit{true})$$

Lemma L.G.1.27 *Provided P is* **RA3** *and* **RA1**-*healthy,*

$$\mathbf{RA1}(\neg\, \mathit{ok}) ;_\mathcal{A} P = \mathbf{RA1}(\neg\, \mathit{ok})$$

Lemma L.G.1.28 $\mathbf{RA1}(\mathit{true}) ;_\mathcal{A} \mathbf{RA1}(\mathit{true}) = \mathbf{RA1}(\mathit{true})$

Lemma L.G.1.29 *Provided ac' is not free in P,*

$$\mathbf{RA1}(P) ;_\mathcal{A} \mathbf{RA1}(\mathit{true}) = \mathbf{RA1}(P)$$

Lemma L.G.1.30 *Provided P is* **PBMH**-*healthy,*

$$\mathbf{RA1}(P) ;_\mathcal{A} \mathbf{RA1}(\mathit{true}) \Rightarrow \mathbf{RA1}(P) ;_\mathcal{A} \mathit{true}$$

Lemma L.G.1.31 *Provided P is* **PBMH**-*healthy,*

$$P ;_\mathcal{A} Q \Rightarrow P ;_\mathcal{A} \mathit{true}$$

Lemma L.G.1.32 $\mathbf{RA1}(\mathit{true}) ;_\mathcal{A} \mathbf{RA1}(P) \Rightarrow \mathbf{RA1}(\mathit{true})$

G.1.6 Properties with respect to RA2

Lemma L.G.1.33

$$\mathbf{RA1} \circ \mathbf{RA2}(P)$$
$$=$$
$$\mathbf{RA2}(P) \land \exists z \bullet s.tr \le z.tr \land z \in ac'$$

Lemma L.G.1.34

$$\mathbf{RA2}(P)[\{z \mid z \in ac' \land s.tr \le z.tr\}/ac'] = \mathbf{RA2}(P)$$

Lemma L.G.1.35 $\mathbf{RA1}(P) \Rightarrow \mathbf{RA1}(true)$

Lemma L.G.1.36 $\mathbf{RA1} \circ \mathbf{RA2}(P) \Rightarrow \mathbf{RA1}(true)$

G.1.7 Properties with respect to PBMH

Theorem T.G.1.6 $\mathbf{RA} \circ \mathbf{A}(P) = \mathbf{RA} \circ \mathbf{PBMH}(P)$

Lemma L.G.1.37 *Provided P is* **PBMH***-healthy,*

$$\mathbf{RA1}(P) = \mathbf{PBMH}(P \land ac' \neq \emptyset \land ac' \subseteq States_{tr \le tr'}(s))$$

Lemma L.G.1.38

$$\mathbf{PBMH}(P \land ac' \neq \emptyset \land ac' \subseteq States_{tr \le tr'}(s)) \Rightarrow ac' \cap States_{tr \le tr'}(s) \neq \emptyset$$

Lemma L.G.1.39

$$ac' \cap States_{tr \le tr'}(s) \neq \emptyset \mathbin{;_{\mathcal{A}}} \mathbf{PBMH}(P \land ac' \neq \emptyset \land ac' \subseteq States_{tr \le tr'}(s))$$
$$\Rightarrow$$
$$ac' \cap States_{tr \le tr'}(s) \neq \emptyset$$

G.1.8 Properties with respect to A2

Lemma L.G.1.40

$$\mathbf{RA1} \circ \mathbf{A2}(P)$$
$$=$$

$$\mathbf{RA1}(true) \land \left(\begin{array}{l} (P[\emptyset/ac']) \\ \lor \\ (\exists\, y \bullet P[\{y\}/ac'] \land s.tr \leq y.tr \land y \in ac') \end{array}\right)$$

Theorem T.G.1.7 $\mathbf{A2} \circ \mathbf{RA1} \circ \mathbf{A2}(P) = \mathbf{RA1} \circ \mathbf{A2}(P)$

G.2 RA2

G.2.1 Definition

Definition 110

$$\mathbf{RA2}(P) \mathrel{\widehat=} P\left[s \oplus \{tr \mapsto \langle\rangle\}, \left\{ z \,\middle|\, \begin{array}{l} z \in ac' \land s.tr \leq z.tr \\ \bullet\, z \oplus \{tr \mapsto z.tr - s.tr\} \end{array} \right\} \middle/ s, ac'\right]$$

G.2.2 Properties

Theorem T.5.2.6 $\mathbf{RA2}(P \land Q) = \mathbf{RA2}(P) \land \mathbf{RA2}(Q)$

Theorem T.5.2.7 $\mathbf{RA2}(P \lor Q) = \mathbf{RA2}(P) \lor \mathbf{RA2}(Q)$

Theorem T.5.2.8 *Provided P and Q are* **RA2***-healthy,*

$$\mathbf{RA2}(P \,;_{\mathcal{A}}\, Q) = P \,;_{\mathcal{A}}\, Q$$

Theorem T.5.2.9 $\mathbf{RA2}(ac' \neq \emptyset) = \mathbf{RA1}(true)$

Theorem T.5.2.10 $\mathbf{RA2} \circ \mathbf{RA1}(P) = \mathbf{RA1} \circ \mathbf{RA2}(P)$

Theorem T.5.2.11 $\mathbf{PBMH} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P) = \mathbf{RA2} \circ \mathbf{PBMH}(P)$

Theorem T.G.2.1 $\mathbf{RA2} \circ \mathbf{RA2}(P) = \mathbf{RA2}(P)$

Theorem T.G.2.2 $P \sqsubseteq Q \Rightarrow \mathbf{RA2}(P) \sqsubseteq \mathbf{RA2}(Q)$

G.2.3 Lemmas

Lemma L.G.2.1

$$\mathbf{RA2}(P) = P[s \oplus \{tr \mapsto \langle\rangle\}, \{y \mid y \oplus \{tr \mapsto s.tr \frown y.tr\} \in ac'\}/s, ac']$$

Lemma L.G.2.2 $\mathbf{RA2}(true) = true$

Lemma L.G.2.3 $\mathbf{RA2}(s \in ac') = s \in ac'$

Lemma L.G.2.4 *Provided s and ac' are not free in P, $\mathbf{RA2}(P) = P$.*

Lemma L.G.2.5

$$\mathbf{RA2}(P \lhd c \rhd Q) = \mathbf{RA2}(P) \lhd \mathbf{RA2}(c) \rhd \mathbf{RA2}(Q)$$

Lemma L.G.2.6 *Provided c is* **RA2***-healthy,*

$$\mathbf{RA2}(P \lhd c \rhd Q) = \mathbf{RA2}(P) \lhd c \rhd \mathbf{RA2}(Q)$$

Lemma L.G.2.7 $\mathbf{RA2}(\neg P) = \neg\, \mathbf{RA2}(P)$

Lemma L.G.2.8 *Where c is not tr,* $\mathbf{RA2}(s.c) = s.c$

Lemma L.G.2.9 $\mathbf{RA2}(P \land ac' \neq \emptyset) = \mathbf{RA2} \circ \mathbf{RA1}(P)$

Lemma L.G.2.10

$$\begin{aligned}&\mathbf{RA2}(P)[\{y\}/ac'] \land s.tr \leq y.tr\\ &=\\ &P[s \oplus \{tr \mapsto \langle\rangle\}, \{y \oplus \{tr \mapsto y.tr - s.tr\}\}/s, ac'] \land s.tr \leq y.tr\end{aligned}$$

Lemma L.G.2.11 *Provided ac' is not free in Q and P is* **PBMH***-healthy,*

$$\begin{aligned}&\circledin_{ac'}^{y}(\mathbf{RA1} \circ \mathbf{RA2}(P) \land Q)\\ &=\\ &\exists\, y \bullet \left(\begin{array}{l} P[s \oplus \{tr \mapsto \langle\rangle\}, \{y \oplus \{tr \mapsto y.tr - s.tr\}\}/s, ac'] \\ \land\, s.tr \leq y.tr \land Q \land y \in ac' \end{array}\right)\end{aligned}$$

Lemma L.G.2.12

$$\begin{aligned}&\mathbf{RA2}(x \in ac')\\ &=\\ &\exists\, z \bullet z \in ac' \land s.tr \leq z.tr \land x = z \oplus \{tr \mapsto z.tr - s.tr\}\end{aligned}$$

Lemma L.G.2.13 *Provided ac' is not free in Q and P is* **PBMH***-healthy,*

$$\mathbf{RA2}(\circledin_{ac'}^{y}(P \land Q))$$

$$= \\ \exists y \bullet \left(\begin{array}{l} P[s \oplus \{tr \mapsto \langle\rangle\}/s][\{y \oplus \{tr \mapsto y.tr - s.tr\}\}/ac'] \\ \land \\ Q[s \oplus \{tr \mapsto \langle\rangle\}/s][y \oplus \{tr \mapsto y.tr - s.tr\}/y] \\ \land \\ y \in ac' \land s.tr \leq y.tr \end{array}\right)$$

Theorem T.G.2.3 *Provided ac' is not free in Q, P is* **PBMH**-*healthy, and* $Q = [s \oplus \{tr \mapsto \langle\rangle\}/s][y \oplus \{tr \mapsto y.tr - s.tr\}/y]$,

$$\mathbf{RA2}(\circledin^{y}_{ac'}(P \land Q)) \\ = \\ \circledin^{y}_{ac'}(\mathbf{RA1} \circ \mathbf{RA2}(P) \land Q)$$

G.2.4 Substitution Properties

Lemma L.G.2.14 $\mathbf{RA2}(P)^o_w = \mathbf{RA2}(P^o_w)$

G.2.5 Properties with respect to Designs

Lemma L.G.2.15 $\mathbf{RA2}(P \vdash Q) = (\neg\, \mathbf{RA2}(\neg\, P) \vdash \mathbf{RA2}(Q))$

Lemma L.G.2.16 $\mathbf{RA2}(P \vdash Q) = \mathbf{RA2}(P \vdash \mathbf{RA2}(Q))$

G.2.6 Properties with respect to $;_\mathcal{A}$

Theorem T.G.2.4 $\mathbf{RA2}(P \,;_\mathcal{A} \mathbf{RA2}(Q)) = \mathbf{RA2}(P) \,;_\mathcal{A} \mathbf{RA2}(Q)$

Lemma L.G.2.17

$$\mathbf{RA2}(P) \,;_\mathcal{A} \mathbf{RA2}(Q) \\ = \\ \left(P\right)\begin{array}{l}[s \oplus \{tr \mapsto \langle\rangle\}/s] \\ \left[\left\{t \,\middle|\, \left(Q\right)\begin{array}{l}[(t \oplus \{tr \mapsto \langle\rangle\}/s] \\ [\{y \mid y \oplus \{tr \mapsto s.tr \frown t.tr \frown y.tr\} \in ac'\}/ac']\end{array}\right\}\Big/ ac'\right]\end{array}$$

Lemma L.G.2.18 $\mathbf{RA2}(P) \,;_\mathcal{A}\, true = P[s \oplus \{tr \mapsto \langle\rangle\}/s] \,;_\mathcal{A}\, true$

G.2.7 Properties with respect to A2

Theorem T.G.2.5 $\mathbf{A2} \circ \mathbf{RA2} \circ \mathbf{A2}(P) = \mathbf{RA2} \circ \mathbf{A2}(P)$

Lemma L.G.2.19

$$\mathbf{RA2} \circ \mathbf{A2}(P)$$
$$=$$
$$\left(\begin{array}{l} P[\emptyset/ac'][s \oplus \{tr \mapsto \langle\rangle\}/s] \\ \vee \\ (\exists\, y \bullet P[\{y\}/ac'][s \oplus \{tr \mapsto \langle\rangle\}/s] \wedge y \oplus \{tr \mapsto s.tr \frown y.tr\} \in ac') \end{array}\right)$$

G.3 RA3

G.3.1 Definition

Definition 112 $\mathbf{RA3}(P) \mathrel{\widehat=} I\!I_{\mathbf{RAD}} \lhd s.wait \rhd P$

G.3.2 Properties

Theorem T.5.2.12 $\mathbf{RA3}(P \wedge Q) = \mathbf{RA3}(P) \wedge \mathbf{RA3}(Q)$

Theorem T.5.2.13 $\mathbf{RA3}(P \vee Q) = \mathbf{RA3}(P) \vee \mathbf{RA3}(Q)$

Theorem T.5.2.14 *Provided P and Q are* **RA3***-healthy and Q is* **RA1***-healthy,*

$$\mathbf{RA3}(P \mathbin{;_{\mathcal{A}}} Q) = P \mathbin{;_{\mathcal{A}}} Q$$

Theorem T.5.2.15 $\mathbf{PBMH} \circ \mathbf{RA3} \circ \mathbf{PBMH}(P) = \mathbf{RA3} \circ \mathbf{PBMH}(P)$

Theorem T.5.2.16 $\mathbf{RA3} \circ \mathbf{RA1}(P) = \mathbf{RA3} \circ \mathbf{RA1}(P)$

Theorem T.5.2.17 $\mathbf{RA2} \circ \mathbf{RA3}(P) = \mathbf{RA3} \circ \mathbf{RA2}(P)$

Theorem T.G.3.1 $\mathbf{RA1}(I\!I_{\mathbf{RAD}}) = I\!I_{\mathbf{RAD}}$

Theorem T.G.3.2 $\mathbf{RA2}(I\!I_{\mathbf{RAD}}) = I\!I_{\mathbf{RAD}}$

Theorem T.G.3.3 $\mathbf{RA3}(I\!I_{\mathbf{RAD}}) = I\!I_{\mathbf{RAD}}$

Theorem T.G.3.4 $\mathbf{PBMH}(I\!I_{\mathbf{RAD}}) = I\!I_{\mathbf{RAD}}$

Theorem T.G.3.5 $\mathbf{RA3} \circ \mathbf{RA3}(P) = \mathbf{RA3}(P)$

Theorem T.G.3.6 $P \sqsubseteq Q \Rightarrow \mathbf{RA3}(P) \sqsubseteq \mathbf{RA3}(Q)$

Properties with respect to PBMH

Theorem T.G.3.7 $\mathbf{PBMH} \circ \mathbf{RA3}(P) = \mathbf{RA3} \circ \mathbf{PBMH}(P)$

Properties with respect to A2

Theorem T.G.3.8 $\mathbf{A2} \circ \mathbf{RA3}(P) = \mathbf{RA3} \circ \mathbf{A2}(P)$

Theorem T.G.3.9 $\mathbf{A2} \circ \mathbf{RA3} \circ \mathbf{A2}(P) = \mathbf{RA3} \circ \mathbf{A2}(P)$

Lemma L.G.3.1 $\mathbf{A2}(\mathit{II}_{\mathbf{RAD}}) = \mathit{II}_{\mathbf{RAD}}$

G.3.3 Substitution Lemmas

Lemma L.5.2.1 $\mathbf{RA3}(P) = \mathbf{RA3}(P_f)$

Lemma L.G.3.2 $\mathbf{RA3}(P)^o_f = P^o_f$

Lemma L.G.3.3 $\mathbf{RA3}(P)^o_w = (\mathit{II}_{\mathbf{RAD}})^o_w \lhd w \rhd P^o_w$

G.4 RA

G.4.1 Definition

Definition 113 $\mathbf{RA}(P) \mathrel{\widehat{=}} \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{RA3}(P)$

Theorem T.5.2.20 $\mathbf{RAD}(P) = \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f)$

Theorem T.5.2.21 *Provided P is* **RAD***-healthy,* $\mathbf{PBMH}(P) = P$.

Lemma L.5.2.2 $\mathbf{RAD}(P) = \mathbf{RA}(\neg\, \mathbf{PBMH}(P)^f_f \vdash \mathbf{PBMH}(P)^t_f)$

Theorem T.G.4.1 $\mathbf{RA}(P \land Q) = \mathbf{RA}(P) \land \mathbf{RA}(Q)$

Theorem T.G.4.2 $\mathbf{RA}(P \lor Q) = \mathbf{RA}(P) \lor \mathbf{RA}(Q)$

Theorem T.G.4.3 $\mathbf{RA} \circ \mathbf{RA}(P) = \mathbf{RA}(P)$

Theorem T.G.4.4 *Provided P is* **PBMH***-healthy,*

$$\mathbf{PBMH} \circ \mathbf{RA}(P) = \mathbf{RA}(P)$$

Theorem T.G.4.5

$$\mathbf{RA} \circ \mathbf{A}(\neg\, (\mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f))^f_f \vdash (\mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f))^t_f)$$
$$=$$
$$\mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f)$$

Lemma L.G.4.1

$$\mathbf{RA1} \circ \mathbf{RA3}(P \vdash Q)$$
$$=$$
$$\mathbf{RA1}((\mathit{true} \lhd s.\mathit{wait} \rhd P) \vdash (s \in ac' \lhd s.\mathit{wait} \rhd Q))$$

Lemma L.G.4.2

$$\mathbf{RA1} \circ \mathbf{RA3}(\neg\, ok) = \mathbf{RA1}(\neg\, ok) \lor (s.\mathit{wait} \land I\!I_{\mathbf{RAD}})$$

Lemma L.G.4.3 $I\!I_{\mathbf{RAD}} = \mathbf{RA1}(\neg\, ok) \lor (ok' \land s \in ac')$

Lemma L.G.4.4

$$\mathbf{RA1} \circ \mathbf{RA3}(P) = (s.\mathit{wait} \land I\!I_{\mathbf{RAD}}) \lor \mathbf{RA1} \circ \mathbf{RA3}(P)$$

Lemma L.G.4.5 $\mathbf{RA1} \circ \mathbf{RA3}(P) = I\!I_{\mathbf{RAD}} \lhd s.\mathit{wait} \rhd \mathbf{RA1}(P)$

Lemma L.G.4.6 $\mathbf{RA}(P)^o_f = \mathbf{RA2} \circ \mathbf{RA1}(P^o_f)$

Lemma L.G.4.7

$$(\mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f))^o_w$$
$$=$$
$$\mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(\neg\, ok \lor P^f_f \lor (P^t_f \land o))$$

Lemma L.G.4.8

$$(\mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f))^f_f$$
$$=$$
$$\mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(\neg\, ok \lor P^f_f)$$

Lemma L.G.4.9

$$(\mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f))^t_f$$
$$=$$
$$\mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(\neg\, ok \lor P^f_f \lor P^t_f)$$

Lemma L.G.4.10

$$\exists\, ac' \bullet \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P) = \exists\, ac' \bullet \mathbf{RA2} \circ \mathbf{PBMH}(P)$$

Lemma L.G.4.11

$$\mathbf{RA} \circ \mathbf{A}(\neg\, \mathbf{RA2} \circ \mathbf{PBMH}(P) \vdash \mathbf{RA2} \circ \mathbf{PBMH}(Q))$$
$$=$$
$$\mathbf{RA} \circ \mathbf{A}(\neg\, P \vdash Q)$$

Lemma L.G.4.12

$$\mathbf{RA} \circ \mathbf{A}(\neg\, \mathbf{RA2} \circ \mathbf{PBMH}(P) \vdash Q)$$
$$=$$
$$\mathbf{RA} \circ \mathbf{A}(\neg\, P \vdash Q)$$

Lemma L.G.4.13

$$\mathbf{RA} \circ \mathbf{A}(P \vdash \mathbf{RA2} \circ \mathbf{PBMH}(Q))$$
$$=$$
$$\mathbf{RA} \circ \mathbf{A}(P \vdash Q)$$

Lemma L.G.4.14

$$\mathbf{RA} \circ \mathbf{A}(P \vdash \mathbf{RA1} \circ \mathbf{PBMH}(Q))$$
$$=$$
$$\mathbf{RA} \circ \mathbf{A}(P \vdash Q)$$

Lemma L.G.4.15

$$\mathbf{RA} \circ \mathbf{A}(\neg\, \mathbf{RA1} \circ \mathbf{PBMH}(P) \vdash Q)$$
$$=$$

$$\mathbf{RA} \circ \mathbf{A}(\neg\, P \vdash Q)$$

G.4.2 Properties with respect to A2

Theorem T.G.4.6 $\mathbf{RA} \circ \mathbf{A} \circ \mathbf{A2}(P) = \mathbf{A2} \circ \mathbf{RA} \circ \mathbf{A} \circ \mathbf{A2}(P)$

Theorem T.G.4.7 *Provided P is* **A2***-healthy,*

$$\mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) = \mathbf{A2} \circ \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f)$$

Lemma L.G.4.16

$$\begin{array}{l} \mathbf{RA} \circ \mathbf{A}(\neg\, \mathbf{A2}(P) \vdash \mathbf{A2}(Q)) \\ = \\ \mathbf{A2} \circ \mathbf{RA} \circ \mathbf{A}(\neg\, \mathbf{A2}(P) \vdash \mathbf{A2}(Q)) \end{array}$$

G.5 CSPA1

Lemma L.G.5.1 $\mathbf{CSPA1}(P) = P \lor (\neg\, ok \land \exists\, z \bullet s.tr \leq z.tr \land z \in ac')$

G.5.1 Properties

Theorem T.5.2.18 $\mathbf{CSPA1} \circ \mathbf{RA1}(P) = \mathbf{RA1} \circ \mathbf{H1}(P)$

Theorem T.5.2.19 *Provided P is* **PBMH***-healthy,*

$$\mathbf{PBMH} \circ \mathbf{CSPA1}(P) = \mathbf{CSPA1}(P)$$

Theorem T.G.5.1 $\mathbf{CSPA1} \circ \mathbf{CSPA1}(P) = \mathbf{CSPA1}(P)$

Theorem T.G.5.2 $P \sqsubseteq Q \Rightarrow \mathbf{CSPA1}(P) \sqsubseteq \mathbf{CSPA1}(Q)$

Properties with respect to RA1 and H1

Theorem T.G.5.3 $\mathbf{RA1} \circ \mathbf{CSPA1}(P) = \mathbf{RA1} \circ \mathbf{H1}(P)$

Theorem T.G.5.4 $\mathbf{RA1} \circ \mathbf{CSPA}(P) = \mathbf{CSPA1} \circ \mathbf{RA1}(P)$

G.6 $\mathbf{ND_{RAD}}$

Theorem T.5.5.2 *Provided P is* **RAD**-*healthy,*

$$\mathbf{ND_{RAD}}(P) = \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash P^t_f)$$

Theorem T.5.5.3 *Provided P is* **RAD**-*healthy,*

$$\mathbf{ND_{RAD}}(P) = P \Leftrightarrow \forall\, s, ac' \bullet \neg\, P^f_f$$

Theorem T.5.5.1 $\mathbf{ND_{RAD}} \circ \mathbf{ND_{RAD}}(P) = \mathbf{ND_{RAD}}(P)$

Theorem T.G.6.1 *Provided P and Q are reactive angelic designs and* $\mathbf{ND_{RAD}}$-*healthy,*

$$\begin{array}{l} P \mathbin{;_{\mathcal{D}ac}} Q \\ = \\ \mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} \mathit{true} \\ \vdash \\ \mathbf{RA1}(P^t_f) \mathbin{;_{\mathcal{A}}} (s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA2} \circ \mathbf{RA1}(Q^t_f)) \end{array} \right) \end{array}$$

Lemma L.G.6.1 $\mathbf{ND_{RAD}}(\mathit{Chaos}_{\mathbf{RAD}}) = \mathit{Choice}_{\mathbf{RAD}}$

Lemma L.G.6.2 $\mathbf{ND_{RAD}}(a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) = a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}$

G.7 Relationship with CSP

G.7.1 Results with respect to R

Theorem T.5.3.1 *Provided P is* **PBMH**-*healthy,* $\mathit{ac2p} \circ \mathbf{RA}(P) = \mathbf{R} \circ \mathit{ac2p}(P)$

Theorem T.5.3.2 $\mathit{ac2p} \circ \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) = \mathbf{R}(\neg\, \mathit{ac2p}(P^f_f) \vdash \mathit{ac2p}(P^t_f))$

Theorem T.G.7.1 *Provided P is* **PBMH**-*healthy,*

$$\mathit{ac2p} \circ \mathbf{RA1}(P) = \mathbf{R1} \circ \mathit{ac2p}(P)$$

Theorem T.G.7.2 *Provided P is* **PBMH**-*healthy,*

$$\mathit{ac2p} \circ \mathbf{RA1} \circ \mathbf{RA2}(P) = \mathbf{R1} \circ \mathbf{R2} \circ \mathit{ac2p}(P)$$

Theorem T.G.7.3 $ac2p \circ \mathbf{RA3}(P) = \mathbf{R3} \circ ac2p(P)$

Theorem T.G.7.4 *Provided* $out\alpha = \{tr', ref', wait'\}$,

$$ac2p(I\!I_{\mathbf{RAD}}) = I\!I_{rea}$$

Theorem T.5.3.3 $p2ac \circ \mathbf{R}(P) = \mathbf{RA} \circ p2ac(P)$

Theorem T.5.3.4 $p2ac \circ \mathbf{R}(\neg\, P^f_f \vdash P^t_f) = \mathbf{RA} \circ \mathbf{A}(\neg\, p2ac(P^f_f) \vdash p2ac(P^t_f))$

Theorem T.G.7.5 $p2ac \circ \mathbf{R}(\neg\, P^f \vdash P^t) = \mathbf{RA}(\neg\, p2ac(P^f) \vdash p2ac(P^t))$

Theorem T.G.7.6

$$p2ac \circ \mathbf{R}(\neg\, P^f \vdash P^t)$$
$$=$$
$$\mathbf{RA} \circ \mathbf{A}(\neg\, p2ac(P^f) \wedge (\neg\, P^f[\mathbf{s}/in\alpha] \;;\; true) \vdash p2ac(P^t))$$

Theorem T.G.7.7

$$p2ac \circ \mathbf{R}(\neg\, P^f \vdash P^t)$$
$$=$$
$$\mathbf{RA}(\neg\, p2ac(P^f) \wedge (\neg\, P^f[\mathbf{s}/in\alpha] \;;\; true) \vdash p2ac(P^t))$$

Theorem T.G.7.8 $\mathbf{RA1} \circ p2ac(P) = p2ac \circ \mathbf{R1}(P)$

Theorem T.G.7.9 $p2ac \circ \mathbf{R1} \circ \mathbf{R2}(P) = \mathbf{RA2} \circ p2ac(P)$

Theorem T.G.7.10 $p2ac \circ \mathbf{R3}(P) = \mathbf{RA3} \circ p2ac(P)$

Lemma L.G.7.1

$$I\!I_{\mathbf{RAD}} = (\neg\, ok \wedge \exists\, z \bullet s.tr \leq z.tr \wedge z \in ac') \vee (ok' \wedge s \in ac')$$

Lemma L.G.7.2 $p2ac(I\!I_{rea}) = I\!I_{\mathbf{RAD}}$

G.7.2 *ac2p*

Lemma L.G.7.3 *Provided ac' is not free in P,*

$$ac2p(\circledin^{y}_{ac'}(P)) = P[State_{I\!I}(in\alpha)/s][undash(State_{I\!I}(out\alpha_{-ok'}))/y]$$

Lemma L.G.7.4

$$\begin{array}{l} ac2p(P) \mathbin{;} ac2p(Q) \\ = \\ \exists\, ok_0, y \bullet \left(\begin{array}{l} (\exists\, ac' \bullet P[State_{\mathit{II}}(in\alpha_{-ok})/s][ok_0/ok'] \land ac' \subseteq \{y\}) \\ \land \\ (\exists\, ac' \bullet Q[y/s][ok_0/ok] \land ac' \subseteq \{z \mid \bigwedge x : out\alpha_{-ok'} \bullet dash(z).x = x\}) \end{array}\right) \end{array}$$

Lemma L.G.7.5

$$\begin{array}{l} ac2p(P) \mathbin{;} ac2p(Q) \\ = \\ \exists\, ok_0, y \bullet \left(\begin{array}{l} (P[\emptyset/ac'] \lor P[\{y\}/ac'])[State_{\mathit{II}}(in\alpha_{-ok})/s][ok_0/ok'] \\ \land \\ (\exists\, ac' \bullet Q[y/s][ok_0/ok] \land ac' \subseteq \{z \mid \bigwedge x : out\alpha_{-ok'} \bullet dash(z).x = x\}) \end{array}\right) \end{array}$$

Lemma L.G.7.6

$$\begin{array}{l} ac2p(P) \mathbin{;} ac2p(Q) \\ = \\ \left(\begin{array}{l} (\exists\, ac' \bullet P[State_{\mathit{II}}(in\alpha_{-ok})/s] \land ac' \subseteq \{s'\}) \\ \; ; \\ (\exists\, ac' \bullet Q \land ac' \subseteq \{z \mid \bigwedge x : out\alpha_{-ok'} \bullet dash(z).x = x\}) \end{array}\right) \end{array}$$

Lemma L.G.7.7

$$\begin{array}{l} ac2p(P) \mathbin{;} ac2p(Q) \\ = \\ \left(\begin{array}{l} \exists\, ok_0 \bullet \left(\begin{array}{l} P[\emptyset/ac'][State_{\mathit{II}}(in\alpha_{-ok})/s][ok_0/ok'] \\ \land \\ (\exists\, ac', s \bullet Q[ok_0/ok] \land ac' \subseteq \{z \mid \bigwedge x : out\alpha_{-ok'} \bullet dash(z).x = x\}) \end{array}\right) \\ \lor \\ \left(\begin{array}{l} P[\{s'\}/ac'][State_{\mathit{II}}(in\alpha_{-ok})/s] \\ \; ; \\ (\exists\, ac' \bullet Q \land ac' \subseteq \{z \mid \bigwedge x : out\alpha_{-ok'} \bullet dash(z).x = x\}) \end{array}\right) \end{array}\right) \end{array}$$

G.7.3 $p2ac$

Theorem T.G.7.11 $p2ac(P \; ; \; Q) = p2ac(P) \; ;_{\mathcal{D}ac} \; p2ac(Q)$

Theorem T.G.7.12 *Provided ok' is not free in P and ok is not free in Q,*

$$p2ac(P) \; ;_{\mathcal{A}} \; p2ac(Q) = p2ac(P \; ; \; Q)$$

Lemma L.G.7.8 $p2ac(P)[\{z\}/ac'] \land z \in ac' = p2ac(P)[\{z\} \cap ac'/ac'] \land z \in ac'$

G.7.4 $p2ac$ and $ac2p$

Theorem T.5.3.5 $ac2p \circ p2ac(P) = P$

Theorem T.5.3.6 *Provided P is* **PBMH***-healthy, $p2ac \circ ac2p(P) \sqsupseteq P$.*

Theorem T.G.7.13 $p2ac \circ ac2p(P) \sqsupseteq \mathbf{PBMH}(P)$

Theorem T.G.7.14

$$p2ac(ac2p(P) \; ; \; ac2p(Q)) = (\exists \, ac' \bullet P \land ac' \subseteq \{s'\}) \; ; \; p2ac \circ ac2p(Q)$$

Theorem T.G.7.15 *Provided Q is* **PBMH***-healthy and s' is not free in P,*

$$p2ac(ac2p(P) \; ; \; ac2p(Q)) \Rightarrow \mathbf{PBMH}(P) \; ;_{\mathcal{D}ac} \; Q$$

Lemma L.G.7.9 *Provided P and Q are* **PBMH***-healthy, s' is not free in P, ok' is not free in P and ok is not free in Q,*

$$p2ac(ac2p(P) \; ; \; ac2p(Q)) \Rightarrow P \; ;_{\mathcal{A}} \; Q$$

Lemma L.G.7.10 *Provided P and Q are* **PBMH***-healthy, s' is not free in P, ok' is not free in P.*

$$p2ac(ac2p(P) \; ; \; ac2p(Q)) \Rightarrow P \; ;_{\mathcal{A}} \; (\exists \, ok \bullet Q)$$

Results with respect to A2

Theorem T.5.3.7 *Provided P^f_f and P^t_f are* **A2***-healthy,*

$$p2ac \circ ac2p \circ \mathbf{RA} \circ \mathbf{A}(\neg \, P^f_f \vdash P^t_f) = \mathbf{RA} \circ \mathbf{A}(\neg \, P^f_f \vdash P^t_f)$$

Lemma L.G.7.11 *Provided P is* **A2***-healthy,* $p2ac \circ ac2p(P) = P \land ac' \neq \emptyset$

Lemma L.G.7.12 *Provided P is* **A2***-healthy,*

$$p2ac \circ ac2p(P)[\{x\}/ac'] = P[\{x\}/ac']$$

G.7.5 Lifting

Definition 122 $\circledin_{ac'}^{y}(P) \mathrel{\widehat=} \exists\, y \bullet y \in ac' \land P[\{y\}/ac']$

Lemma L.G.7.13 *Provided ac' is not free in P,*

$$\mathbf{PBMH}(\circledin_{ac'}^{y}(P)) = \circledin_{ac'}^{y}(P)$$

Lemma L.G.7.14

$$\mathbf{RA1}(\circledin_{ac'}^{y}(P)) = \circledin_{ac'}^{y}(\mathbf{RA1}(P[\{y\} \cap ac'/ac']) \land s.tr \leq y.tr)$$

Lemma L.G.7.15 *Provided ac' is not free in P,*

$$\mathbf{RA1}(\circledin_{ac'}^{y}(P)) = \circledin_{ac'}^{y}(P \land s.tr \leq y.tr)$$

Lemma L.G.7.16 *Provided ac' is not free in P,*

$$\begin{aligned}&\mathbf{RA2}(\circledin_{ac'}^{y}(P))\\ &=\\ &\exists\, y \bullet \mathbf{RA2}(P) \land \circledin_{ac'}^{z}(s.tr \leq z.tr \land y = z \oplus \{tr \mapsto z.tr - s.tr\})\end{aligned}$$

Lemma L.G.7.17 *Provided x is not s,*

$$\mathbf{RA2}(x \in ac') = \circledin_{ac'}^{z}(s.tr \leq z.tr \land x = z \oplus \{tr \mapsto z.tr - s.tr\})$$

Lemma L.G.7.18 $\mathbf{RA2}(x \in ac') = x \oplus \{tr \mapsto s.tr \frown x.tr\} \in ac'$

Lemma L.G.7.19 *Provided x is not in the set $\{s, ac'\}$,*

$$\mathbf{RA2}(\exists\, x \bullet P) = \exists\, x \bullet \mathbf{RA2}(P)$$

Lemma L.G.7.20 *Provided ac' is not free in P,*

$$\mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(\circledin_{ac'}^{y}(P))$$

$$=$$
$$\circledin^{z}_{ac'}(P[s \oplus \{tr \mapsto \langle\rangle\}/s][z \oplus \{tr \mapsto z.tr - s.tr\}/y] \wedge s.tr \leq z.tr)$$

Lemma L.G.7.21 *Provided ac' is not free in P,*

$$\mathbf{RA2}(P) = P[s \oplus \{tr \mapsto \langle\rangle\}/s]$$

Lemma L.G.7.22

$$\mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(\circledin^{y}_{ac'}(y.tr = s.tr \wedge a \notin y.ref \wedge y.wait))$$
$$=$$
$$\circledin^{y}_{ac'}(y.tr = s.tr \wedge a \notin y.ref \wedge y.wait)$$

Lemma L.G.7.23

$$\mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(\circledin^{y}_{ac'}(y.tr = s.tr \frown \langle a \rangle \wedge \neg\, y.wait))$$
$$=$$
$$\circledin^{y}_{ac'}(y.tr = s.tr \frown \langle a \rangle \wedge \neg\, y.wait)$$

Lemma L.G.7.24

$$\mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(\circledin^{y}_{ac'}(s.tr \frown \langle a \rangle \leq y.tr))$$
$$=$$
$$\circledin^{y}_{ac'}(s.tr \frown \langle a \rangle \leq y.tr)$$

Lemma L.G.7.25 $\circledin^{y}_{ac'}(P \vee Q) = \circledin^{y}_{ac'}(P) \vee \circledin^{y}_{ac'}(Q)$

Lemma L.G.7.26

$$\mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}\left(\circledin^{y}_{ac'}\left(\begin{array}{l}(y.tr = s.tr \wedge a \notin y.ref)\\ \lhd y.wait \rhd\\ (y.tr = s.tr \frown \langle a \rangle)\end{array}\right)\right)$$
$$=$$
$$\left(\circledin^{y}_{ac'}\left(\begin{array}{l}(y.tr = s.tr \wedge a \notin y.ref)\\ \lhd y.wait \rhd\\ (y.tr = s.tr \frown \langle a \rangle)\end{array}\right)\right)$$

Lemma L.G.7.27 *Provided ac' is not free in P,*

$$\circledin_{ac'}^{y}(P) \;;_{\mathcal{A}}\; Q = \exists\, y \bullet P \land Q[y/s]$$

Lemma L.G.7.28 *Provided P is* **PBMH**-*healthy,*

$$\circledin_{ac'}^{y}(P) = \exists\, y \bullet P[\{y\}/ac'] \land y \in ac'$$

Lemma L.G.7.29 *Provided ac' is not free in P,* $\circledin_{ac'}^{y}(P) = \exists\, y \bullet P \land y \in ac'$

Lemma L.G.7.30 $\circledin_{ac'}^{y}(P \lor Q) = \circledin_{ac'}^{y}(P) \lor \circledin_{ac'}^{y}(Q)$

Lemma L.G.7.31

$$\begin{array}{l} \circledin_{ac'}^{y}(P \lhd c_0 \land \ldots \land c_n \rhd Q) \\ = \\ \circledin_{ac'}^{y}(c_0 \land \ldots \land c_n \land P) \lor \circledin_{ac'}^{y}(\neg\, c_0 \land Q) \lor \ldots \lor \circledin_{ac'}^{y}(\neg\, c_n \land Q) \end{array}$$

Lemma L.G.7.32 *Provided $s.tr \le z.tr$,*

$$\begin{array}{l} (s.tr = y.tr \land y.wait)[s \oplus \{tr \mapsto \langle\rangle\}/s][y \oplus \{tr \mapsto y.tr - s.tr\}/y] \\ = \\ (s.tr = y.tr \land y.wait) \end{array}$$

Lemma L.G.7.33 *Provided $s.tr \le y.tr$,*

$$\begin{array}{l} (s.tr \neq y.tr)[s \oplus \{tr \mapsto \langle\rangle\}/s][y \oplus \{tr \mapsto y.tr - s.tr\}/y] \\ = \\ (s.tr \neq y.tr) \end{array}$$

Lemma L.G.7.34 *Provided x is not tr,*

$$\begin{array}{l} (y.x)[s \oplus \{tr \mapsto \langle\rangle\}/s][y \oplus \{tr \mapsto y.tr - s.tr\}/y] \\ = \\ (y.x) \end{array}$$

Lemma L.G.7.35 *Provided:*

- *P and Q are* **PBMH**-*healthy*

- *For* $0 \leq i \leq n$*:* ac' *is not free in* c_i
- $(c_0 \wedge \ldots \wedge c_n)[s \oplus \{tr \mapsto \langle\rangle\}/s][y \oplus \{tr \mapsto y.tr - s.tr\}/y] = (c_0 \wedge \ldots \wedge c_n)$, *assuming* $s.tr \leq y.tr$
- *For* $0 \leq i \leq n$*:* $(\neg\ c_i)[s \oplus \{tr \mapsto \langle\rangle\}/s][y \oplus \{tr \mapsto y.tr - s.tr\}/y] = \neg\ c_i$, *assuming* $s.tr \leq y.tr$

$$\begin{array}{l}\textcircled{$\in$}_{ac'}^{y}(\mathbf{RA2} \circ \mathbf{RA1}(P) \lhd c_0 \wedge \ldots \wedge c_n \rhd \mathbf{RA2} \circ \mathbf{RA1}(Q)) \\ = \\ \mathbf{RA2}(\textcircled{$\in$}_{ac'}^{y}(P \lhd (c_0 \wedge \ldots \wedge c_n) \rhd Q))\end{array}$$

Lemma L.G.7.36 *Provided that* P *and* Q *are* **PBMH**-*healthy,*

$$\begin{array}{l}\textcircled{$\in$}_{ac'}^{y}(\mathbf{RA2} \circ \mathbf{RA1}(P) \lhd ytr = s.tr \wedge y.wait \rhd \mathbf{RA2} \circ \mathbf{RA1}(Q)) \\ = \\ \mathbf{RA2}(\textcircled{$\in$}_{ac'}^{y}(P \lhd ytr = s.tr \wedge y.wait \rhd Q))\end{array}$$

Lemma L.G.7.37 $\textcircled{$\in$}_{ac'}^{y}(P \wedge \textcircled{$\in$}_{ac'}^{z}(Q)) = \textcircled{$\in$}_{ac'}^{y}(P \wedge Q[y/z])$

Lemma L.G.7.38 $\textcircled{$\in$}_{ac'}^{z}(Q)[\{y\} \cap ac'/ac'] = Q[y/z][\{y\} \cap ac'/ac'] \wedge y \in ac'$

Properties with respect to PBMH

Lemma L.G.7.39 $\textcircled{$\in$}_{ac'}^{y}(\mathbf{PBMH}(P) \wedge Q) \Rightarrow \mathbf{PBMH}(P)$

Lemma L.G.7.40

$$\begin{array}{l}\neg\ \mathbf{PBMH}(P) \wedge \textcircled{$\in$}_{ac'}^{y}(((\mathbf{PBMH}(P) \wedge Q) \vee R) \lhd c \rhd T) \\ = \\ \neg\ \mathbf{PBMH}(P) \wedge \textcircled{$\in$}_{ac'}^{y}(R \lhd c \rhd T)\end{array}$$

Lemma L.G.7.41

$$\begin{array}{l}\neg\ \mathbf{PBMH}(P) \wedge \textcircled{$\in$}_{ac'}^{y}(Q \lhd c \rhd (\mathbf{PBMH}(P) \vee R)) \\ = \\ \neg\ \mathbf{PBMH}(P) \wedge \textcircled{$\in$}_{ac'}^{y}(Q \lhd c \rhd R)\end{array}$$

Properties with respect to *ac2p*

Theorem T.G.7.16 *Provided ac' is not free in P, Q and R, and y is not free in P nor Q,*

$$\begin{array}{l} ac2p(\circledin_{ac'}^{y}(p2ac(P) \land p2ac(Q) \land R)) \\ = \\ P \land Q \land R[undash(State_{II}(out\alpha_{-ok'}))/y][State_{II}(in\alpha_{-ok})/s] \end{array}$$

Lemma L.G.7.42 *Provided ac' is not free in P, Q and R, and y is not free in P nor Q,*

$$\begin{array}{l} ac2p(\circledin_{ac'}^{y}(p2ac(P) \land R)) \\ = \\ P \land R[undash(State_{II}(out\alpha_{-ok'}))/y][State_{II}(in\alpha_{-ok})/s] \end{array}$$

Lemma L.G.7.43 *Provided P is* **PBMH**-*healthy,* $\mathbf{PBMH}(\circledin_{ac'}^{y}(P)) = \circledin_{ac'}^{y}(P)$

Lemma L.G.7.44

$$\begin{array}{l} \mathbf{RA2}(\circledin_{ac'}^{y}(y.wait \land y.tr = s.tr)) \\ = \\ \circledin_{ac'}^{y}(y.wait \land y.tr = s.tr) \end{array}$$

Properties with respect to A2

Lemma L.G.7.45 $\mathbf{A2}(\circledin_{ac'}^{y}(P)) = \exists\, y \bullet P[\{y\}/ac'] \land y \in ac'$

Theorem T.G.7.17 *Provided P is* **PBMH**-*healthy,* $\mathbf{A2}(\circledin_{ac'}^{y}(P)) = \circledin_{ac'}^{y}(P)$.

G.8 Operators

G.8.1 Angelic Choice

Theorem T.5.4.1 *Provided P and Q are reactive angelic designs,*

$$P \sqcup Q = \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \lor \neg\, Q^f_f \vdash (\neg\, P^f_f \Rightarrow P^t_f) \land (\neg\, Q^f_f \Rightarrow Q^t_f))$$

Theorem T.5.4.2 $ac2p(p2ac(P) \sqcup_{\mathbf{RAD}} p2ac(Q)) = P \sqcup_{\mathbf{R}} Q$

Theorem T.5.4.3 *Provided that P and Q are reactive angelic designs,*

$$p2ac(ac2p(P) \sqcup_{\mathbf{R}} ac2p(Q)) \sqsupseteq P \sqcup_{\mathbf{RAD}} Q$$

Theorem T.G.8.1

$$\begin{array}{l}
\mathbf{RA} \circ \mathbf{A}(P \vdash Q) \sqcup \mathbf{RA} \circ \mathbf{A}(R \vdash S) \\
= \\
\mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l}
(\neg\, \mathbf{PBMH}(\neg\, P) \vee \neg\, \mathbf{PBMH}(\neg\, R)) \\
\vdash \\
\left(\begin{array}{l}
(\neg\, \mathbf{PBMH}(\neg\, P) \Rightarrow \mathbf{PBMH}(Q)) \\
\wedge \\
(\neg\, \mathbf{PBMH}(\neg\, R) \Rightarrow \mathbf{PBMH}(S)
\end{array} \right)
\end{array} \right)
\end{array}$$

Theorem T.G.8.2 *Provided $\neg\, P$, $\neg\, Q$, R and S are* **PBMH**-*healthy.*

$$\begin{array}{l}
\mathbf{RA} \circ \mathbf{A}(P \vdash Q) \sqcup \mathbf{RA} \circ \mathbf{A}(R \vdash S) \\
= \\
\mathbf{RA} \circ \mathbf{A}(P \vee R \vdash (P \Rightarrow Q) \wedge (R \Rightarrow S))
\end{array}$$

Theorem T.G.8.3 *Provided P is a reactive angelic design,* $Chaos_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) = \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f)$

G.8.2 Demonic Choice

Theorem T.5.4.4 *Provided P and Q are reactive angelic processes,*

$$P \sqcap_{\mathbf{RAD}} Q = \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \wedge \neg\, Q^f_f \vdash P^t_f \vee Q^t_f)$$

Theorem T.5.4.5

$$p2ac(ac2p(P) \sqcap_{\mathbf{R}} ac2p(Q)) = p2ac \circ ac2p(P) \sqcap_{\mathbf{RAD}} p2ac \circ ac2p(Q)$$

Theorem T.5.4.6 $ac2p(p2ac(P) \sqcap_{\mathbf{RAD}} p2ac(Q)) = P \sqcap_{\mathbf{R}} Q$

Theorem T.5.4.7 *Provided P is a reactive angelic design,*

$$Chaos_{\mathbf{RAD}} \sqcap_{\mathbf{RAD}} P = Chaos_{\mathbf{RAD}}$$

Theorem T.G.8.4

$$\mathbf{RA} \circ \mathbf{A}(P \vdash Q) \sqcap \mathbf{RA} \circ \mathbf{A}(R \vdash S) = \mathbf{RA} \circ \mathbf{A}(P \land R \vdash Q \lor S)$$

Lemma L.5.4.1 *Provided P and Q are reactive angelic designs and* **A2***-healthy,*

$$p2ac(ac2p(P) \sqcap_{\mathbf{R}} ac2p(Q)) = P \sqcap_{\mathbf{RAD}} Q$$

G.8.3 Chaos

Theorem T.5.4.8 *Provided P is a reactive angelic design,*

$$Chaos_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P = P$$

Theorem T.5.4.9 $ac2p(Chaos_{\mathbf{RAD}}) = Chaos_{\mathbf{R}}$

Theorem T.5.4.10 $p2ac(Chaos_{\mathbf{R}}) = Chaos_{\mathbf{RAD}}$

G.8.4 Choice

Theorem T.5.4.11 $p2ac(Choice_{\mathbf{R}}) = Choice_{\mathbf{RAD}}$

Theorem T.5.4.12 $ac2p(Choice_{\mathbf{RAD}}) = Choice_{\mathbf{R}}$

Theorem T.5.4.13 *Provided P is* **RAD***-healthy,*

$$Choice_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P = \mathbf{RA} \circ \mathbf{A}(true \vdash P^t_f)$$

Theorem T.5.4.14 *Provided P is* **RAD***-healthy,*

$$Choice_{\mathbf{RAD}} \sqcap_{\mathbf{RAD}} P = \mathbf{RA} \circ \mathbf{A}(\neg P^f_f \vdash ac' \neq \emptyset)$$

G.8.5 Stop

Theorem T.5.4.15 *Provided P is* **RAD***-healthy,*

$$\begin{aligned}
&Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P \\
&= \\
&\mathbf{RA} \circ \mathbf{A}(true \vdash (\neg P^f_f \Rightarrow P^t_f) \land \circledin^{y}_{ac'}(y.tr = s.tr \land y.wait))
\end{aligned}$$

Theorem T.5.4.16 $p2ac(Stop_{\mathbf{R}}) = Stop_{\mathbf{RAD}}$

Theorem T.5.4.17 $ac2p(Stop_{\mathbf{RAD}}) = Stop_{\mathbf{R}}$

G.8.6 Skip

Theorem T.5.4.18 *Provided P is* **RAD**-*healthy,*

$$\begin{array}{l} Skip_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P \\ = \\ \mathbf{RA} \circ \mathbf{A}(true \vdash \circledcirc\!\!\in_{ac'}^{y}(\neg\, y.wait \land y.tr = s.tr)) \land (\neg\, P^f_f \Rightarrow P^t_t)) \end{array}$$

Theorem T.5.4.19 $p2ac(Skip_{\mathbf{R}}) = Skip_{\mathbf{RAD}}$

Theorem T.5.4.20 $ac2p(Skip_{\mathbf{RAD}}) = Skip_{\mathbf{R}}$

Lemma L.5.4.2 $ac2p(Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} Skip_{\mathbf{RAD}}) = \top_{\mathbf{R}}$

G.8.7 Sequential Composition

Theorem T.5.4.21 *Provided P and Q are reactive angelic designs,*

$$\begin{array}{l} P \mathbin{;_{\mathcal{D}ac}} Q \\ = \\ \mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} \left(\begin{array}{l} \neg\, (\mathbf{RA1}(P^f_f) \mathbin{;_{\mathcal{A}}} \mathbf{RA1}(true)) \\ \land \\ \neg\, (\mathbf{RA1}(P^t_f) \mathbin{;_{\mathcal{A}}} (\neg\, s.wait \land \mathbf{RA2} \circ \mathbf{RA1}(Q^f_f))) \end{array} \right) \\ \vdash \\ \mathbf{RA1}(P^t_f) \mathbin{;_{\mathcal{A}}} (s \in ac' \lhd s.wait \rhd (\mathbf{RA2} \circ \mathbf{RA1}(\neg\, Q^f_f \Rightarrow Q^t_f))) \end{array} \right) \end{array}$$

Theorem T.5.4.22 *Provided P and Q are reactive angelic designs,*

$$p2ac(ac2p(P) \mathbin{;} ac2p(Q)) \sqsupseteq P \mathbin{;_{\mathcal{D}ac}} Q$$

Theorem T.5.4.23 *Provided P and Q are* **RAD**-*healthy and* **A2**-*healthy,*

$$p2ac(ac2p(P) \mathbin{;} ac2p(Q)) = P \mathbin{;_{\mathcal{D}ac}} Q$$

Theorem T.5.4.24 $ac2p(p2ac(P) \mathbin{;_{\mathcal{D}ac}} p2ac(Q)) = P \mathbin{;} Q$

Theorem T.5.4.25 *Provided P and Q are reactive angelic designs and* **A2**-*healthy,* $\mathbf{A2}(P \mathbin{;_{\mathcal{D}ac}} Q) = P \mathbin{;_{\mathcal{D}ac}} Q$

Theorem T.G.8.5 *Provided* $\neg P$, $\neg R$, Q *and* S *are* **PBMH**-*healthy and* ok, ok' *are not free in* P, Q, R *and* S,

$$\begin{array}{l}\mathbf{RA}(P \vdash Q) \mathbin{;_{\mathcal{D}ac}} \mathbf{RA}(R \vdash S)\\ =\\ \mathbf{RA}\left(\begin{array}{l}\left(\begin{array}{l}\neg (\mathbf{RA1}(\neg P) \mathbin{;_{\mathcal{A}}} \mathbf{RA1}(true))\\ \land\\ \neg (\mathbf{RA1}(Q) \mathbin{;_{\mathcal{A}}} (\neg s.wait \land \mathbf{RA2} \circ \mathbf{RA1}(\neg R)))\end{array}\right)\\ \vdash\\ \mathbf{RA1}(Q) \mathbin{;_{\mathcal{A}}} (s \in ac' \lhd s.wait \rhd \mathbf{RA2} \circ \mathbf{RA1}(R \Rightarrow S))\end{array}\right)\end{array}$$

Theorem T.G.8.6 *Provided* $\neg P$,Q,$\neg R$ *and* S *are* **PBMH**-*healthy, and* ok *and* ok' *are not free in* P,Q,R *and* S,

$$\begin{array}{l}\mathbf{RA1}(P \vdash Q) \mathbin{;_{\mathcal{D}ac}} \mathbf{RA1}(R \vdash S)\\ =\\ \mathbf{RA1}\left(\begin{array}{l}\neg (\mathbf{RA1}(\neg P) \mathbin{;_{\mathcal{A}}} \mathbf{RA1}(true)) \land \neg (\mathbf{RA1}(Q) \mathbin{;_{\mathcal{A}}} \mathbf{RA1}(\neg R))\\ \vdash\\ \mathbf{RA1}(Q) \mathbin{;_{\mathcal{A}}} \mathbf{RA1}(R \Rightarrow S)\end{array}\right)\end{array}$$

Lemma L.5.4.3 $(Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} Skip_{\mathbf{RAD}}) \mathbin{;_{\mathcal{D}ac}} Chaos_{\mathbf{RAD}} = Stop_{\mathbf{RAD}}$

G.8.8 Event Prefixing

Theorem T.5.4.26 *Provided* P *is a reactive angelic design,*

$$\begin{array}{l}a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P\\ =\\ \mathbf{RA} \circ \mathbf{A}\left(true \vdash \textcircled{\in}_{ac'}^{y}\left(\begin{array}{l}(y.tr = s.tr \land a \notin y.ref)\\ \lhd y.wait \rhd\\ (y.tr = s.tr \frown \langle a \rangle)\end{array}\right) \land (\neg P_f^f \Rightarrow P_f^t)\right)\end{array}$$

Relationship with CSP

Theorem T.5.4.27 $ac2p(a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}) = a \rightarrow_{\mathbf{R}} Skip_{\mathbf{R}}$

Theorem T.5.4.28 $p2ac(a \rightarrow_{\mathbf{R}} Skip_{\mathbf{R}}) = a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$

Theorem T.5.4.29 *Provided P is* **RAD**-*healthy,*

$$a \rightarrow_{\mathbf{RAD}} P$$
$$=$$
$$\mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} \neg\ \exists\, y \bullet y.tr = s.tr \frown \langle a \rangle \wedge \neg\ y.wait \wedge (\mathbf{RA2} \circ \mathbf{RA1}(P^f_f))[y/s] \\ \vdash \\ \exists\, y \bullet \left(\begin{array}{l} (y \in ac' \wedge y.tr = s.tr \wedge a \notin y.ref) \\ \lhd y.wait \rhd \\ (y.tr = s.tr \frown \langle a \rangle \wedge (\mathbf{RA2} \circ \mathbf{RA1}(P^t_f))[y/s]) \end{array} \right) \end{array} \right)$$

Lemma L.G.8.1

$$ac2p(a \rightarrow_{\mathbf{RAD}} Chaos_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} Chaos_{\mathbf{RAD}})$$
$$=$$
$$\mathbf{R}(true \vdash tr' = tr \wedge wait' \wedge a \notin ref' \wedge b \notin ref')$$

Lemma L.G.8.2

$$ac2p(a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}})$$
$$=$$
$$a \rightarrow_{\mathbf{R}} Stop_{\mathbf{R}} \sqcup_{\mathbf{R}} b \rightarrow_{\mathbf{R}} Stop_{\mathbf{R}}$$

Lemma L.G.8.3

$$p2ac(a \rightarrow_{\mathbf{R}} Stop_{\mathbf{R}} \sqcup_{\mathbf{R}} b \rightarrow_{\mathbf{R}} Stop_{\mathbf{R}})$$
$$= \mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} true \\ \vdash \\ \exists\, y \bullet y.wait \wedge y.tr = s.tr \wedge a \notin y.ref \wedge b \notin y.ref \wedge y \in ac' \end{array} \right)$$

Lemma L.G.8.4 $ac2p(a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}) = a \rightarrow_{\mathbf{R}} Stop_{\mathbf{R}}$

Properties and Examples

Theorem T.G.8.7 *Provided P is* **RAD**-*healthy,*

$$P \ ;_{\mathcal{R}\mathbf{ac}} Chaos_{\mathbf{RAD}}$$
$$=$$

$$\mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} \left(\begin{array}{l} \neg\, (\mathbf{RA1}(P^f_f) \;;_{\mathcal{A}}\; \mathbf{RA1}(\mathit{true})) \\ \wedge \\ \neg\, (\mathbf{RA1}(P^t_f) \;;_{\mathcal{A}}\; (\neg\, s.\mathit{wait} \wedge \mathbf{RA2} \circ \mathbf{RA1}(\mathit{true}))) \end{array} \right) \\ \vdash \\ \mathbf{RA1}(P^t_f) \;;_{\mathcal{A}}\; (s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA2} \circ \mathbf{RA1}(\mathit{true})) \end{array} \right)$$

Theorem T.G.8.8

$$a \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}} = \mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} \neg\, \textcircled{\in}^{z}_{ac'}(s.tr \frown \langle a \rangle \leq z.tr) \\ \vdash \\ \textcircled{\in}^{y}_{ac'}(y.\mathit{wait} \wedge y.tr = s.tr \wedge a \notin y.\mathit{ref}) \end{array} \right)$$

Lemma L.G.8.5

$$\begin{array}{l} a \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}} \\ = \\ \mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} \neg\, \textcircled{\in}^{y}_{ac'}(s.tr \frown \langle a \rangle \leq y.tr) \\ \vdash \\ \textcircled{\in}^{y}_{ac'}(y.\mathit{wait} \wedge y.tr = s.tr \wedge a \notin y.\mathit{ref}) \end{array} \right) \end{array}$$

Lemma L.G.8.6

$$\begin{array}{l} a \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}} \\ = \\ \mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} \neg\, (\textcircled{\in}^{y}_{ac'}(s.tr \frown \langle a \rangle \leq y.tr) \wedge \textcircled{\in}^{y}_{ac'}(s.tr \frown \langle b \rangle \leq y.tr)) \\ \vdash \\ \left(\begin{array}{l} \textcircled{\in}^{y}_{ac'}((y.\mathit{wait} \wedge a \notin y.\mathit{ref}) \lhd y.tr = s.tr \rhd (s.tr \frown \langle a \rangle \leq y.tr)) \\ \wedge \\ \textcircled{\in}^{y}_{ac'}((y.\mathit{wait} \wedge b \notin y.\mathit{ref}) \lhd y.tr = s.tr \rhd (s.tr \frown \langle b \rangle \leq y.tr)) \end{array} \right) \end{array} \right) \end{array}$$

Lemma L.G.8.7

$$\begin{array}{l} (a \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}}) \;\square_{\mathbf{RAD}}\; \mathit{Stop}_{\mathbf{RAD}} \\ = \\ \mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} \neg\, (\textcircled{\in}^{y}_{ac'}(s.tr \frown \langle a \rangle \leq y.tr) \wedge \textcircled{\in}^{y}_{ac'}(s.tr \frown \langle b \rangle \leq y.tr)) \\ \vdash \\ \textcircled{\in}^{y}_{ac'}(y.\mathit{wait} \wedge y.tr = s.tr \wedge a \notin y.\mathit{ref} \wedge b \notin y.\mathit{ref}) \end{array} \right) \end{array}$$

Lemma L.G.8.8

$$a \rightarrow_{\mathbf{RAD}} \mathit{Stop}_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} \mathit{Stop}_{\mathbf{RAD}}$$

$$=$$

$$\mathbf{RA} \circ \mathbf{A}\left(\begin{array}{l} \mathit{true} \\ \vdash \\ \left(\begin{array}{l} \circledin^{y}_{ac'}(y.\mathit{wait} \wedge ((y.tr = s.tr \wedge a \notin y.\mathit{ref}) \vee y.tr = s.tr \frown \langle a \rangle)) \\ \wedge \\ \circledin^{y}_{ac'}(y.\mathit{wait} \wedge ((y.tr = s.tr \wedge b \notin y.\mathit{ref}) \vee y.tr = s.tr \frown \langle b \rangle)) \end{array}\right) \end{array}\right)$$

Lemma L.G.8.9

$$(a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} (b \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}})$$

$$=$$

$$(a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} (b \rightarrow_{\mathbf{RAD}} \mathit{Choice}_{\mathbf{RAD}})$$

Lemma L.G.8.10 *Provided P is* **RAD***-healthy,*

$$a \rightarrow_{\mathbf{RAD}} P$$

$$=$$

$$\mathbf{RA} \circ \mathbf{A}\left(\begin{array}{l} \left(\begin{array}{l} \circledin^{z}_{ac'}(s.tr \frown \langle a \rangle \leq z.tr) \\ \Rightarrow \\ \neg \left(\exists\, \mathit{ref} \bullet (\mathbf{RA2}(P^f_f)) \left[\left\{\begin{array}{l} tr \mapsto s.tr \frown \langle a \rangle, \\ \mathit{wait} \mapsto \mathit{false}, \\ \mathit{ref} \mapsto \mathit{ref} \end{array}\right\} / s\right]\right) \end{array}\right) \\ \vdash \\ \left(\begin{array}{l} \circledin^{y}_{ac'}(y.\mathit{wait} \wedge y.tr = s.tr \wedge a \notin y.\mathit{ref}) \\ \vee \\ \left(\begin{array}{l} \left(\exists\, \mathit{ref} \bullet \mathbf{RA2}(P^t_f) \left[\left\{\begin{array}{l} tr \mapsto s.tr \frown \langle a \rangle, \\ \mathit{wait} \mapsto \mathit{false}, \\ \mathit{ref} \mapsto \mathit{ref} \end{array}\right\} / s\right]\right) \\ \wedge \\ \circledin^{z}_{ac'}(s.tr \frown \langle a \rangle \leq z.tr) \end{array}\right) \end{array}\right) \end{array}\right)$$

Lemma L.G.8.11

$$a \rightarrow_{\mathbf{RAD}} \mathit{Stop}_{\mathbf{RAD}}$$

$$=$$

$$\mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} true \\ \vdash \\ ⓔ^{y}_{ac'}(y.wait \land ((y.tr = s.tr \land a \notin y.ref) \lor y.tr = s.tr \frown \langle a \rangle)) \end{array} \right)$$

Lemma L.G.8.12

$$(a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$$
$$=$$
$$\mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} true \\ \vdash \\ \left(\begin{array}{l} ⓔ^{y}_{ac'}(y.wait \land ((y.tr = s.tr \land a \notin y.ref) \lor y.tr = s.tr \frown \langle a \rangle)) \\ \land \\ ⓔ^{y}_{ac'}(\neg\, y.wait \land y.tr = s.tr) \end{array} \right) \end{array} \right)$$

Lemma L.G.8.13

$$((a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} Skip_{\mathbf{RAD}}) \;;_{\mathcal{D}ac}\; Chaos_{\mathbf{RAD}}$$
$$=$$
$$a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}$$

Lemma L.G.8.14

$$a \rightarrow_{\mathbf{RAD}} Choice_{\mathbf{RAD}}$$
$$=$$
$$\mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} true \\ \vdash \\ ⓔ^{y}_{ac'}((y.wait \land y.tr = s.tr \land a \notin y.ref) \lor (s.tr \frown \langle a \rangle \leq y.tr)) \end{array} \right)$$

G.8.9 External Choice

Theorem T.5.4.30 *Provided P is a reactive angelic design,*

$$P \,\Box_{\mathbf{RAD}}\, Stop_{\mathbf{RAD}} = \mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash \exists\, y \bullet (P^t_f)[\{y\}/ac'] \land y \in ac')$$

Theorem T.5.4.31 *Provided P is a reactive angelic design and* **A2***-healthy,*

$$P \,\Box_{\mathbf{RAD}}\, Stop_{\mathbf{RAD}} = P$$

Theorem T.G.8.9

$$\begin{aligned}&\mathbf{RA} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f) \mathbin{\Box_{\mathbf{RAD}}} \mathbf{RA} \circ \mathbf{A}(\neg\, Q^f_f \vdash Q^t_f)\\ &=\\ &\mathbf{RA} \circ \mathbf{A}\left(\begin{array}{l}\neg\, P^f_f \land \neg\, Q^f_f\\ \vdash\\ \circledin^y_{ac'}\left(\begin{array}{l}(\mathbf{PBMH}(P^t_f) \land \mathbf{PBMH}(Q^t_f))\\ \lhd y.wait \land y.tr = s.tr \rhd\\ (\mathbf{PBMH}(P^t_f) \lor \mathbf{PBMH}(Q^t_f))\end{array}\right)\end{array}\right)\end{aligned}$$

Theorem T.G.8.10 *Provided P and Q are reactive angelic designs,*

$$\begin{aligned}&P \mathbin{\Box_{\mathbf{RAD}}} Q\\ &=\\ &\mathbf{RA} \circ \mathbf{A}\left(\begin{array}{l}\neg\, P^f_f \land \neg\, Q^f_f\\ \vdash\\ \circledin^y_{ac'}\left(\begin{array}{l}(P^t_f \land Q^t_f)\\ \lhd y.wait \land y.tr = s.tr \rhd\\ (P^t_f \lor Q^t_f)\end{array}\right)\end{array}\right)\end{aligned}$$

Relationship with CSP

Theorem T.5.4.32 *Provided that P and Q are CSP processes,*

$$ac2p(p2ac(P) \mathbin{\Box_{\mathbf{RAD}}} p2ac(Q)) = P \mathbin{\Box_{\mathbf{R}}} Q$$

Theorem T.5.4.33 *Provided P and Q are reactive angelic designs,*

$$p2ac(ac2p(P) \mathbin{\Box_{\mathbf{R}}} ac2p(Q)) \sqsupseteq P \mathbin{\Box_{\mathbf{RAD}}} Q$$

Theorem T.5.4.34 *Provided P and Q are* **RAD**-*healthy and* **A2**-*healthy,*

$$p2ac(ac2p(P) \mathbin{\Box_{\mathbf{R}}} ac2p(Q)) = P \mathbin{\Box_{\mathbf{RAD}}} Q$$

Closure

Theorem T.5.4.35 *Provided P and Q are reactive angelic designs and* **A2***-healthy,*

$$\mathbf{A2}(P \Box_{\mathbf{RAD}} Q) = P \Box_{\mathbf{RAD}} Q$$

Properties and Examples

Lemma L.G.8.15 $(\mathit{Skip}_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} \mathit{Stop}_{\mathbf{RAD}}) \Box_{\mathbf{RAD}} \mathit{Stop}_{\mathbf{RAD}} = \top_{\mathbf{RAD}}$

Lemma L.G.8.16 $(\mathit{Skip}_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} \mathit{Stop}_{\mathbf{RAD}}) \Box_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}} = \mathit{Skip}_{\mathbf{RAD}}$

Appendix H

Angelic Processes

H.1 Healthiness Conditions

H.1.1 $\mathit{II}_{\mathbf{AP}}$

Lemma L.H.1.1 $\mathbf{RA2}(\mathit{II}_{\mathbf{AP}}) = \mathit{II}_{\mathbf{AP}}$

Lemma L.H.1.2 $\mathbf{RA1}(\mathit{II}_{\mathbf{AP}}) = \mathit{II}_{\mathbf{RAD}}$

H.1.2 $\mathbf{RA3_{AP}}$

Theorem T.6.2.1 $\mathbf{RA3_{AP}} \circ \mathbf{RA3_{AP}}(P) = \mathbf{RA3_{AP}}(P)$

Theorem T.6.2.2 $P \sqsubseteq Q \Rightarrow \mathbf{RA3_{AP}}(P) \sqsubseteq \mathbf{RA3_{AP}}(Q)$

Theorem T.6.2.3 $\mathbf{RA3_{AP}}(P \land Q) = \mathbf{RA3_{AP}}(P) \land \mathbf{RA3_{AP}}(Q)$

Theorem T.6.2.4 $\mathbf{RA3_{AP}}(P \lor Q) = \mathbf{RA3_{AP}}(P) \lor \mathbf{RA3_{AP}}(Q)$

Theorem T.6.2.5 *Provided P and Q are $\mathbf{RA3_{AP}}$-healthy,*

$$\mathbf{RA3_{AP}}(P \,;_{\mathcal{A}}\, Q) = P \,;_{\mathcal{A}}\, Q$$

Theorem T.6.2.6 $\mathbf{RA3_{AP}} \circ \mathbf{PBMH}(P) = \mathbf{PBMH} \circ \mathbf{RA3_{AP}}(P)$

Theorem T.6.2.7 $\mathbf{RA2} \circ \mathbf{RA3_{AP}}(P) = \mathbf{RA3_{AP}} \circ \mathbf{RA2}(P)$

Lemma L.6.2.1 $\mathbf{PBMH} \circ \mathbf{RA3_{AP}} \circ \mathbf{PBMH}(P) = \mathbf{RA3_{AP}} \circ \mathbf{PBMH}(P)$

Theorem T.H.1.1 $\mathbf{RA1} \circ \mathbf{RA3_{AP}}(P) = \mathbf{RA3} \circ \mathbf{RA1}(P)$

Properties

Lemma L.H.1.3

$$\mathbf{RA3_{AP}} \circ \mathbf{H1}(P) = \mathbf{H1}((ok' \land s \in ac') \lhd s.wait \rhd P)$$

Lemma L.H.1.4

$$\mathbf{RA3_{AP}}(P \vdash Q) = (true \lhd s.wait \rhd P \vdash s \in ac' \lhd s.wait \rhd Q)$$

H.1.3 AP

Main Results

Theorem T.6.2.8 $\mathbf{AP}(P) = \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f)$

Theorem T.6.2.9

$$\mathbf{AP}(P) = \left(\begin{array}{l} true \lhd s.wait \rhd \neg\, \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f) \\ \vdash \\ s \in ac' \lhd s.wait \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f) \end{array}\right)$$

Theorem T.H.1.2 $\mathbf{AP} \circ \mathbf{AP}(P) = P$

Theorem T.H.1.3 $\mathbf{PBMH} \circ \mathbf{AP}(P) = \mathbf{AP}(P)$

Theorem T.H.1.4

$$\begin{array}{l} \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A}(P \vdash Q) \\ = \\ \left(\begin{array}{l} true \lhd s.wait \rhd \neg\, \mathbf{RA2} \circ \mathbf{PBMH}(\neg\, P) \\ \vdash \\ s \in ac' \lhd s.wait \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(Q) \end{array}\right) \end{array}$$

Lemma L.H.1.5

$$\mathbf{AP}(P)^o_f = \left(\begin{array}{l} (ok \land \neg\, \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f)) \\ \Rightarrow \\ (\mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f) \land o) \end{array}\right)$$

Lemma L.H.1.6 $\mathbf{AP}(P)^f_f = ok \Rightarrow \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f)$

Lemma L.H.1.7

$$\begin{array}{l}\mathbf{AP}(P)^t_f\\ =\\ (ok \land \neg\, \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f)) \Rightarrow \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f)\end{array}$$

Lemma L.H.1.8

$$\mathbf{RA2} \circ \mathbf{PBMH}(\mathbf{AP}(P)^t_f) = \mathbf{AP}(P)^t_f$$

Lemma L.H.1.9

$$\mathbf{AP}(\mathit{true} \vdash P^t_f) = (\mathit{true} \vdash s \in ac' \lhd s.wait \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f))$$

Lemma L.H.1.10

$$\begin{array}{l}\mathbf{AP}(\neg\, P^f_f \vdash P^t_f)\\ =\\ \left(\begin{array}{l}\mathit{true} \lhd s.wait \rhd \neg\, \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f)\\ \vdash\\ s \in ac' \lhd s.wait \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f)\end{array}\right)\end{array}$$

Lemma L.H.1.11 $\mathbf{AP}(\neg\, P^f_f \vdash P^t_f) = \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A}(\neg\, P^f_f \vdash P^t_f)$

H.1.4 $\mathbf{ND_{AP_N}}$

Theorem T.6.2.10 *Provided P is* **AP***-healthy.*

$$\mathit{Choice}_{\mathbf{AP}} \sqcup P = (\mathit{true} \vdash s \in ac' \lhd s.wait \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f))$$

H.2 Relationship with Reactive Angelic Designs

H.2.1 From RAD to AP

Theorem T.6.3.1

$$\mathbf{H1} \circ \mathbf{RAD}(P) = \left(\begin{array}{l}\mathit{true} \lhd s.wait \rhd \neg\, \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f)\\ \vdash\\ s \in ac' \lhd s.wait \rhd \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P^t_f)\end{array}\right)$$

Theorem T.H.2.1 $\mathbf{A} \circ \mathbf{H1} \circ \mathbf{RAD}(P) = \mathbf{H1} \circ \mathbf{RAD}(P)$

Lemma L.6.3.1

$$\begin{aligned}&\mathbf{H1} \circ \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash P^t_f)\\ &=\\ &(\mathit{true} \vdash s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f))\end{aligned}$$

Lemma L.H.2.1

$$\begin{aligned}&\mathbf{H3} \circ \mathbf{H1} \circ \mathbf{RAD}(P)\\ &=\\ &\left(\begin{array}{l} \mathit{true} \lhd s.\mathit{wait} \rhd \neg\ \exists\, ac' \bullet \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P^f_f)\\ \vdash\\ s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P^t_f) \end{array}\right)\end{aligned}$$

Lemma L.H.2.2 $\mathbf{H1} \circ \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash P^t_f) = \mathbf{AP}(\mathit{true} \vdash P^t_f)$

Lemma L.H.2.3

$$\begin{aligned}&\mathbf{H3} \circ \mathbf{H1} \circ \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash P^t_f)\\ &=\\ &(\mathit{true} \vdash s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P^t_f))\end{aligned}$$

Lemma L.H.2.4 $\mathbf{H1} \circ \mathbf{RAD}(P) = \mathbf{AP}(\neg\ \mathbf{RA1} \circ \mathbf{PBMH}(P^f_f) \vdash P^t_f)$

Lemma L.H.2.5 *Provided P is a reactive angelic process,*

$$\mathbf{H1}(P) = \mathbf{AP}(\neg\ \mathbf{RA1}(P^f_f) \vdash P^t_f)$$

H.2.2 From AP to RAD

Theorem T.6.3.2 $\mathbf{RA1} \circ \mathbf{AP}(P) = \mathbf{RA} \circ \mathbf{A}(\neg\ P^f_f \vdash P^t_f)$

H.2.3 Galois Connection and Isomorphism

Theorem T.6.3.3 $\mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{RAD}(P) = \mathbf{RAD}(P)$

Theorem T.6.3.4 $\mathbf{H1} \circ \mathbf{RA1} \circ \mathbf{AP}(P) \sqsupseteq \mathbf{AP}(P)$

Theorem T.6.3.5 $\mathbf{H1} \circ \mathbf{RA1} \circ \mathbf{ND_{AP}} \circ \mathbf{AP}(P) = \mathbf{ND_{AP}} \circ \mathbf{AP}(P)$

Theorem T.H.2.2 $\mathbf{RA1} \circ \mathbf{H3} \circ \mathbf{H1} \circ \mathbf{RAD}(P) \sqsubseteq \mathbf{RAD}(P)$

Theorem T.H.2.3 $\mathbf{H3} \circ \mathbf{H1} \circ \mathbf{RA1} \circ \mathbf{AP}(P) \sqsubseteq \mathbf{AP}(P)$

Theorem T.H.2.4

$$\mathbf{RA1} \circ \mathbf{H3} \circ \mathbf{H1} \circ \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash P^t_f) = \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash P^t_f)$$

Theorem T.H.2.5 *Provided P is* **AP***-healthy,*

$$\mathbf{H3} \circ \mathbf{H1} \circ \mathbf{RA1} \circ \mathbf{ND_{AP}}(P) = \mathbf{ND_{AP}}(P)$$

Lemma L.H.2.6 $\mathbf{H1} \circ \mathbf{RA1}(P \vdash Q) = (\neg\, \mathbf{RA1}(\neg\, P) \vdash \mathbf{RA1}(Q))$

Lemma L.H.2.7

$$\begin{aligned}&\mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(Q))\\ &=\\ &\mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash Q)\end{aligned}$$

H.3 Operators

H.3.1 Angelic Choice

Closure

Theorem T.6.4.1 *Provided P and Q are* **AP***-healthy,*

$$\mathbf{AP}(P \sqcup_{\mathbf{AP}} Q) = P \sqcup_{\mathbf{AP}} Q$$

Theorem T.6.4.2 *Provided P and Q are* $\mathbf{ND_{AP}}$*-healthy,*

$$\mathbf{ND_{AP}}(P \sqcup_{\mathbf{AP}} Q) = P \sqcup_{\mathbf{AP}} Q$$

Lemma L.H.3.1

$$\begin{aligned}&\mathbf{AP}(P) \sqcup \mathbf{AP}(Q)\\ &=\end{aligned}$$

$$\left(\begin{array}{l}\left(\begin{array}{l}true\\ \lhd s.wait \rhd\\ \neg\,\mathbf{RA2}\circ\mathbf{PBMH}(\mathbf{RA2}\circ\mathbf{PBMH}(P^f_f)\land\mathbf{RA2}\circ\mathbf{PBMH}(Q^f_f))\end{array}\right)\\ \vdash\\ \left(\begin{array}{l}s\in ac'\\ \lhd s.wait \rhd\\ \mathbf{RA2}\circ\mathbf{RA1}\circ\mathbf{PBMH}\left(\begin{array}{l}\left(\begin{array}{l}\mathbf{RA2}\circ\mathbf{PBMH}(P^f_f)\\ \land\\ \mathbf{RA2}\circ\mathbf{RA1}\circ\mathbf{PBMH}(Q^t_f)\end{array}\right)\\ \lor\\ \left(\begin{array}{l}\mathbf{RA2}\circ\mathbf{RA1}\circ\mathbf{PBMH}(P^t_f)\\ \land\\ \mathbf{RA2}\circ\mathbf{PBMH}(Q^f_f)\end{array}\right)\\ \lor\\ \left(\begin{array}{l}\mathbf{RA2}\circ\mathbf{RA1}\circ\mathbf{PBMH}(P^t_f)\\ \land\\ \mathbf{RA2}\circ\mathbf{RA1}\circ\mathbf{PBMH}(Q^t_f)\end{array}\right)\end{array}\right)\end{array}\right)\end{array}\right)$$

Linking

Theorem T.6.4.3 *Provided P and Q are* **RAD***-healthy,*

$$\mathbf{RA1}(\mathbf{H1}(P)\sqcup_{\mathbf{AP}}\mathbf{H1}(Q)) = P\sqcup_{\mathbf{RAD}} Q$$

Theorem T.6.4.4 *Provided P and Q are* **AP***-healthy,*

$$\mathbf{H1}(\mathbf{RA1}(P)\sqcup_{\mathbf{RAD}}\mathbf{RA1}(Q)) \sqsupseteq P\sqcup_{\mathbf{AP}} Q$$

H.3.2 Demonic Choice

Closure

Theorem T.6.4.5 *Provided P and Q are* **AP***-healthy,* $\mathbf{AP}(P\sqcap Q) = P\sqcap Q$.

Theorem T.6.4.6 *Provided P and Q are* $\mathbf{ND_{AP}}$*-healthy,*

$$\mathbf{ND_{AP}}(P\sqcap_{\mathbf{AP}} Q) = P\sqcap_{\mathbf{AP}} Q$$

Lemma L.H.3.2

$$\mathbf{AP}(P)\sqcap\mathbf{AP}(Q) = \mathbf{AP}(\neg\, P^f_f\land\neg\, Q^f_f\vdash P^t_f\lor Q^t_f)$$

Linking

Theorem T.6.4.7 *Provided P and Q* **RAD**-*healthy,*

$$\mathbf{RA1}(\mathbf{H1}(P) \sqcap_{\mathbf{AP}} \mathbf{H1}(Q)) = P \sqcap_{\mathbf{RAD}} Q$$

Theorem T.6.4.8 *Provided P and Q are* **AP**-*healthy,*

$$\mathbf{H1}(\mathbf{RA1}(P) \sqcap_{\mathbf{RAD}} \mathbf{RA1}(Q)) \sqsupseteq P \sqcap_{\mathbf{AP}} Q$$

H.3.3 Divergence: Chaos and Chaos of CSP

Theorem T.6.4.9 *Provided P is* **AP**-*healthy,* $P \sqcup_{\mathbf{AP}} \mathit{Chaos}_{\mathbf{AP}} = P$

Theorem T.6.4.10 $\mathbf{H1}(\mathit{Chaos}_{\mathbf{RAD}}) = \mathit{ChaosCSP}_{\mathbf{AP}}$

Theorem T.6.4.11 $\mathbf{RA1}(\mathit{ChaosCSP}_{\mathbf{AP}}) = \mathit{Chaos}_{\mathbf{RAD}}$

Theorem T.H.3.1 $\mathbf{H3} \circ \mathbf{H1}(\mathit{Chaos}_{\mathbf{RAD}}) = \mathit{Chaos}_{\mathbf{AP}}$

Lemma L.6.4.1 $\mathit{Chaos}_{\mathbf{AP}} = (s.\mathit{wait} \vdash s \in ac')$

Lemma L.6.4.2 $\mathit{ChaosCSP}_{\mathbf{AP}} = (s.\mathit{wait} \lor \neg\, \mathbf{RA1}(\mathit{true}) \vdash s.\mathit{wait} \land s \in ac')$

H.3.4 Choice

Properties

Lemma L.6.4.3 $\mathbf{AP}(\mathit{true} \vdash ac' \neq \emptyset) = (\mathit{true} \vdash s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA1}(\mathit{true}))$

Lemma L.H.3.3

$$\begin{array}{l}
\mathbf{AP}(\mathit{true} \vdash ac' \neq \emptyset) \\
= \\
\left(\begin{array}{l}
\mathit{true} \lhd s.\mathit{wait} \rhd \neg\, \mathbf{RA2} \circ \mathbf{PBMH}(\mathit{false}) \\
\vdash \\
s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(\mathit{true})
\end{array}\right)
\end{array}$$

Linking

Theorem T.6.4.12 $\mathbf{H1}(\mathit{Choice}_{\mathbf{RAD}}) = \mathit{Choice}_{\mathbf{AP}}$

Theorem T.6.4.13 $\mathbf{RA1}(\mathit{Choice}_{\mathbf{AP}}) = \mathit{Choice}_{\mathbf{RAD}}$

H.3.5 Stop

Theorem T.6.4.14 $\mathbf{H1}(\mathit{Stop}_{\mathbf{RAD}}) = \mathit{Stop}_{\mathbf{AP}}$

Theorem T.6.4.15 $\mathbf{RA1}(\mathit{Stop}_{\mathbf{AP}}) = \mathit{Stop}_{\mathbf{RAD}}$

H.3.6 Skip

Theorem T.6.4.16 $\mathbf{H1}(\mathit{Skip}_{\mathbf{RAD}}) = \mathit{Skip}_{\mathbf{AP}}$

Theorem T.6.4.17 $\mathbf{RA1}(\mathit{Skip}_{\mathbf{AP}}) = \mathit{Skip}_{\mathbf{RAD}}$

H.3.7 Sequential Composition

Theorem T.6.4.18 *Provided P and Q are* **AP***-healthy,*

$$P \mathbin{;_{\mathcal{D}ac}} Q$$
$$=$$
$$\mathbf{AP}\left(\begin{array}{l} \neg\,(P^f_f \mathbin{;_{\mathcal{A}}} \mathit{true}) \land \neg\,(\mathbf{RA1}(P^t_f) \mathbin{;_{\mathcal{A}}} (\neg\, s.\mathit{wait} \land \mathbf{RA2}(Q^f_f))) \\ \vdash \\ \mathbf{RA1}(P^t_f) \mathbin{;_{\mathcal{A}}} (s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA2}(\neg\, Q^f_f \Rightarrow \mathbf{RA1}(Q^t_f))) \end{array}\right)$$

Theorem T.H.3.2

$$\left(\begin{array}{l} (\mathit{true} \lhd s.\mathit{wait} \rhd P \vdash s \in ac' \lhd s.\mathit{wait} \rhd Q) \\ ;_{\mathcal{D}ac} \\ (\mathit{true} \lhd s.\mathit{wait} \rhd R \vdash s \in ac' \lhd s.\mathit{wait} \rhd S) \end{array}\right)$$
$$=$$
$$\left(\begin{array}{l} \mathit{true} \lhd s.\mathit{wait} \rhd \neg\,((\neg\, P \mathbin{;_{\mathcal{A}}} \mathit{true}) \lor (Q \mathbin{;_{\mathcal{A}}} (\neg\, s.\mathit{wait} \land \neg\, R))) \\ \vdash \\ s \in ac' \lhd s.\mathit{wait} \rhd (Q \mathbin{;_{\mathcal{A}}} (s \in ac' \lhd s.\mathit{wait} \rhd (R \Rightarrow S))) \end{array}\right)$$

Theorem T.H.3.3

$$\mathbf{AP}(P) \mathbin{;_{\mathcal{D}ac}} \mathbf{AP}(Q)$$
$$=$$

$$\mathbf{AP}\left(\begin{array}{l}\left(\begin{array}{l}\neg\,(\mathbf{PBMH}(P^f_f)\ ;_{\mathcal{A}}\ true)\\ \land\\ \neg\,(\mathbf{RA1}\circ\mathbf{PBMH}(P^t_f)\ ;_{\mathcal{A}}\ (\neg\, s.wait \land \mathbf{RA2}\circ\mathbf{PBMH}(Q^f_f)))\end{array}\right)\\ \vdash\\ \left(\begin{array}{l}\mathbf{RA1}\circ\mathbf{PBMH}(P^t_f)\\ ;_{\mathcal{A}}\\ s\in ac' \lhd s.wait \rhd \mathbf{RA2}\left(\begin{array}{l}\neg\,\mathbf{PBMH}(Q^f_f)\\ \Rightarrow\\ (\mathbf{RA1}\circ\mathbf{PBMH}(Q^t_f))\end{array}\right)\end{array}\right)\end{array}\right)$$

Lemma L.H.3.4

$$\begin{array}{l}P\ ;_{\mathcal{D}ac}\ Chaos_{\mathbf{AP}}\\ =\\ \mathbf{AP}\left(\begin{array}{l}\neg\,(P^f_f\ ;_{\mathcal{A}}\ true)\land\neg\,(\mathbf{RA1}(P^t_f)\ ;_{\mathcal{A}}\ \neg\, s.wait)\\ \vdash\\ \mathbf{RA1}(P^t_f)\ ;_{\mathcal{A}}\ (s\in ac' \lor \neg\, s.wait)\end{array}\right)\end{array}$$

Lemma L.H.3.5

$$\begin{array}{l}Skip_{\mathbf{AP}} \sqcup_{\mathbf{AP}} Stop_{\mathbf{AP}}\\ =\\ \left(\begin{array}{l}true\\ \vdash\\ \left(s\in ac' \lhd s.wait \rhd \left(\begin{array}{l}ⓔ^{y}_{ac'}(y.tr = s.tr \land \neg\, y.wait)\\ \land\\ ⓔ^{y}_{ac'}(y.tr = s.tr \land y.wait)\end{array}\right)\right)\end{array}\right)\end{array}$$

Lemma L.H.3.6 $(Skip_{\mathbf{AP}} \sqcup_{\mathbf{AP}} Stop_{\mathbf{AP}})\ ;_{\mathcal{D}ac}\ Chaos_{\mathbf{AP}} = Stop_{\mathbf{AP}}$

Linking

Theorem T.6.4.19 *Provided P and Q are reactive angelic designs,*

$$\mathbf{RA1}(\mathbf{H1}(P)\ ;_{\mathcal{D}ac}\ \mathbf{H1}(Q)) \sqsubseteq P\ ;_{\mathcal{D}ac}\ Q$$

Theorem T.6.4.20 *Provided P and Q are* **AP***-healthy,*

$$\mathbf{H1}(\mathbf{RA1}(P)\ ;_{\mathcal{D}ac}\ \mathbf{RA1}(Q)) \sqsupseteq P\ ;_{\mathcal{D}ac}\ Q$$

Theorem T.6.4.21 *Provided P and Q are reactive angelic designs and $\mathbf{ND_{RAD}}$-healthy,*

$$\mathbf{RA1}(\mathbf{H1}(P) ;_{\mathcal{D}ac} \mathbf{H1}(Q)) = P ;_{\mathcal{D}ac} Q$$

Closure

Theorem T.6.4.22 *Provided P and Q are angelic processes and $\mathbf{ND_{AP}}$-healthy,*

$$\mathbf{ND_{AP}}(P ;_{\mathcal{D}ac} Q) = P ;_{\mathcal{D}ac} Q$$

Theorem T.H.3.4 *Provided P and Q are angelic processes,*

$$\begin{array}{l} \mathbf{ND_{AP}}(P) ;_{\mathcal{D}ac} \mathbf{ND_{AP}}(Q) \\ = \\ \left(\begin{array}{l} true \\ \vdash \\ s \in ac' \lhd s.wait \rhd \left(\begin{array}{l} \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f) \\ ;_{\mathcal{A}} \\ (s \in ac' \lhd s.wait \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(Q^t_f)) \end{array} \right) \end{array} \right) \end{array}$$

Lemma L.H.3.7

$$\left(\begin{array}{l} \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P) \\ ;_{\mathcal{A}} \\ (s \in ac \lhd c \rhd \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(Q)) \end{array} \right) \Rightarrow \mathbf{RA1}(true)$$

H.3.8 Prefixing

Theorem T.6.4.23 *Provided P is $\mathbf{AP}$-healthy,*

$$\begin{array}{l} a \rightarrow P \\ = \\ \mathbf{AP} \left(\begin{array}{l} \neg (\exists y \bullet \neg y.wait \wedge y.tr = s.tr \frown \langle a \rangle \wedge (\mathbf{RA2} \circ \mathbf{PBMH}(P^f_f))[y/s]) \\ \vdash \\ \exists y \bullet \left(\begin{array}{l} (y.tr = s.tr \wedge a \notin y.ref \wedge y \in ac') \\ \lhd y.wait \rhd \\ (y.tr = s.tr \frown \langle a \rangle \wedge \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P^t_f)[y/s]) \end{array} \right) \end{array} \right) \end{array}$$

Lemma L.6.4.4 $\mathbf{H1}(a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}) = a \rightarrow_{\mathbf{AP}} Skip_{\mathbf{AP}}$

Lemma L.6.4.5 $\mathbf{RA1}(a \rightarrow_{\mathbf{AP}} \mathit{Skip}_{\mathbf{AP}}) = a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}$

Lemma L.6.4.6

$$\begin{aligned}&a \rightarrow_{\mathbf{AP}} \mathit{ChaosCSP}_{\mathbf{AP}}\\ &=\\ &\mathbf{AP}(\neg \circledin_{ac'}^{y}(s.tr ^\frown \langle a\rangle \leq y.tr) \vdash \circledin_{ac'}^{y}(y.wait \land y.tr = s.tr \land a \notin y.ref))\end{aligned}$$

Lemma L.H.3.8 $a \rightarrow_{\mathbf{AP}} \mathit{Chaos}_{\mathbf{AP}} = \mathit{Chaos}_{\mathbf{AP}}$

Lemma L.H.3.9 $\mathbf{PBMH} \circ \mathbf{RA1}(\mathit{true}) = \mathbf{RA1}(\mathit{true})$

Linking

Theorem T.H.3.5

$$\begin{aligned}&\mathbf{H1}(a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}})\\ &=\\ &\mathbf{AP}\left(\mathit{true} \vdash \circledin_{ac'}^{y}\begin{pmatrix}(y.tr = s.tr \land a \notin y.ref)\\ \lhd y.wait \rhd\\ (y.tr = s.tr ^\frown \langle a\rangle)\end{pmatrix}\right)\end{aligned}$$

Theorem T.H.3.6

$$\begin{aligned}&\mathbf{H3} \circ \mathbf{H1}(a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}})\\ &=\\ &\left(\mathit{true} \vdash s \in ac' \lhd s.wait \rhd \circledin_{ac'}^{y}\begin{pmatrix}(y.tr = s.tr \land a \notin y.ref)\\ \lhd y.wait \rhd\\ (y.tr = s.tr ^\frown \langle a\rangle)\end{pmatrix}\right)\end{aligned}$$

Appendix I

Set Theory

Lemmas

Lemma L.I.0.10

$$ac_0 \subseteq \{s \mid \{s\} = ac_1\} = ac_0 \subseteq ac_1 \land ac_0 \subseteq \{s \mid ac_1 \subseteq \{s\}\}$$

Proof.

$ac_0 \subseteq \{s \mid \{s\} = ac_1\}$ {Definition of subset inclusion}

$= \forall x \bullet x \in ac_0 \Rightarrow x \in \{s \mid \{s\} = ac_1\}$ {Property of sets}

$= \forall x \bullet x \in ac_0 \Rightarrow \{x\} = ac_1$ {Property of sets}

$= \forall x \bullet x \in ac_0 \Rightarrow (\{x\} \subseteq ac_1 \land ac_1 \subseteq \{x\})$ {Property of sets}

$= \forall x \bullet x \in ac_0 \Rightarrow (x \in ac_1 \land ac_1 \subseteq \{x\})$ {Predicate calculus}

$= (\forall x \bullet x \in ac_0 \Rightarrow x \in ac_1) \land (\forall x \bullet x \in ac_0 \Rightarrow ac_1 \subseteq \{x\})$ {Property of sets}

$= (\forall x \bullet x \in ac_0 \Rightarrow x \in ac_1) \land (\forall x \bullet x \in ac_0 \Rightarrow x \in \{s \mid ac_1 \subseteq \{s\}\}$

{Definition of subset inclusion}

$= ac_0 \subseteq ac_1 \land ac_0 \subseteq \{s \mid ac_1 \subseteq \{s\}\}$

□

Lemma L.I.0.11

$$ac_0 \subseteq \{s \mid ac_1 \subseteq \{s\}\} = ac_1 \subseteq \{s \mid ac_0 \subseteq \{s\}\}$$

Proof.

$ac_0 \subseteq \{s \mid ac_1 \subseteq \{s\}\}$ {Definition of subset inclusion}

$= \forall x \bullet x \in ac_0 \Rightarrow x \in \{s \mid ac_1 \subseteq \{s\}\}$ {Property of sets}

$= \forall x \bullet x \in ac_0 \Rightarrow ac_1 \subseteq \{x\}$ {Definition of subset inclusion}

$= \forall x \bullet x \in ac_0 \Rightarrow (\forall y \bullet y \in ac_1 \Rightarrow y \in \{x\})$ {Property of sets}

$= \forall x \bullet x \in ac_0 \Rightarrow (\forall y \bullet y \in ac_1 \Rightarrow y = x)$ {Predicate calculus}

$= \forall x, y \bullet x \in ac_0 \Rightarrow (y \in ac_1 \Rightarrow y \in y = x)$ {Predicate calculus}

$= \forall x, y \bullet x \in ac_0 \wedge y \in ac_1 \Rightarrow y = x$ {Predicate calculus}

$= \forall x, y \bullet y \in ac_1 \Rightarrow (x \in ac_0 \Rightarrow y = x)$ {Predicate calculus}

$= \forall y \bullet y \in ac_1 \Rightarrow (\forall x \bullet x \in ac_0 \Rightarrow y = x)$ {Property of sets}

$= \forall y \bullet y \in ac_1 \Rightarrow (\forall x \bullet x \in ac_0 \Rightarrow x \in \{y\})$ {Definition of subset inclusion}

$= \forall y \bullet y \in ac_1 \Rightarrow ac_0 \subseteq \{y\}$ {Property of sets}

$= \forall y \bullet y \in ac_1 \Rightarrow y \in \{s \mid ac_0 \subseteq \{s\}\}$ {Definition of subset inclusion}

$= ac_1 \subseteq \{s \mid ac_0 \subseteq \{s\}\}$

□

Lemma L.I.0.12

$$ac_0 \subseteq \{s \mid ac_0 \subseteq ac'\} = ac_0 = \emptyset \vee ac_0 \subseteq ac'$$

Proof.

$ac_0 \subseteq \{s \mid ac_0 \subseteq ac'\}$ {Definition of subset inclusion}

$= \forall x \bullet x \in ac_0 \Rightarrow x \in \{s \mid ac_0 \subseteq ac'\}$ {Property of sets}

$= \forall x \bullet x \in ac_0 \Rightarrow ac_0 \subseteq ac'$ {Predicate calculus}

$= \forall x \bullet (x \notin ac_0 \vee ac_0 \subseteq ac')$ {Predicate calculus}

$= (\forall x \bullet x \notin ac_0) \vee ac_0 \subseteq ac'$ {Property of sets}

$= ac_0 = \emptyset \vee ac_0 \subseteq ac'$

□

Lemma L.I.0.13 *Provided v is not s,*

$$\exists v \bullet t \subseteq \{s \mid Q\} \Rightarrow t \subseteq \{s \mid \exists v \bullet Q\}$$

Proof.

$$
\begin{array}{ll}
\exists\, v \bullet t \subseteq \{s \mid Q\} & \{\text{Property of sets, } x \text{ is fresh}\}\\
= \exists\, v \bullet (\forall\, x \bullet x \in t \Rightarrow (\exists\, s \bullet Q \land x = s)) & \{\text{Predicate calculus}\}\\
\Rightarrow \forall\, x \bullet (\exists\, v \bullet x \in t \Rightarrow (\exists\, s \bullet Q \land x = s)) & \{\text{Predicate calculus}\}\\
= \forall\, x \bullet x \in t \Rightarrow (\exists\, v \bullet (\exists\, s \bullet Q \land x = s)) & \{\text{Predicate calculus: } v \text{ is not } s\}\\
= \forall\, x \bullet x \in t \Rightarrow (\exists\, s \bullet (\exists\, v \bullet Q) \land x = s)) & \{\text{Property of sets}\}\\
= \forall\, x \bullet x \in t \Rightarrow x \in \{s \mid \exists\, v \bullet Q\} & \{\text{Property of sets}\}\\
= t \subseteq \{s \mid \exists\, v \bullet Q\} &
\end{array}
$$

□

Lemma L.I.0.14 *Provided $\preceq$ is transitive,*

$$x \preceq y \land A \subseteq \{z \mid y \preceq z \land x \preceq z \land e\} = x \preceq y \land A \subseteq \{z \mid y \preceq z \land e\}$$

Proof.

$$
\begin{array}{ll}
x \preceq y \land A \subseteq \{z \mid x \preceq z \land e\} & \{\text{Property of sets}\}\\
= x \preceq y \land \forall\, z \bullet z \in A \Rightarrow (y \preceq z \land x \preceq z \land e) & \{\text{Predicate calculus}\}\\
= \forall\, z \bullet x \preceq y \land (z \in A \Rightarrow (y \preceq z \land x \preceq z \land e)) & \\
 & \{\text{Predicate calculus: } \preceq \text{ is transitivite}\}\\
= \forall\, z \bullet x \preceq y \land (z \in A \Rightarrow (y \preceq z \land e)) & \{\text{Predicate calculus}\}\\
= x \preceq y \land \forall\, z \bullet z \in A \Rightarrow (y \preceq z \land e) & \{\text{Property of sets}\}\\
= x \preceq y \land A \subseteq \{z \mid y \preceq z \land e\} &
\end{array}
$$

□

Lemma L.I.0.15

$$\exists\, B \bullet B \neq \emptyset \land B \subseteq C \Leftrightarrow C \neq \emptyset$$

Proof. (Implication) By contradiction: Suppose the consequent is false yet the antecedent is true. Then $C = \emptyset$.

$$
\begin{array}{ll}
\exists\, B \bullet B \neq \emptyset \land B \subseteq C & \{\text{Assumption: } C = \emptyset\}\\
= \exists\, B \bullet B \neq \emptyset \land B \subseteq \emptyset & \{\text{Property of subset inclusion}\}
\end{array}
$$

$= \exists B \bullet B \neq \emptyset \wedge B = \emptyset$ {Propositional calculus}

$= \mathit{false}$

$\square$

Proof. (Reverse implication)

$C \neq \emptyset \Rightarrow \exists B \bullet B \neq \emptyset \wedge B \subseteq C$ {Choose $B = C$}

$= C \neq \emptyset \Rightarrow C \neq \emptyset \wedge C \subset C$ {Reflexivity of subset inclusion}

$= C \neq \emptyset \Rightarrow C \neq \emptyset$ {Propositional calculus}

$= \mathit{true}$

$\square$

Lemma L.I.0.16

$\exists ac_0 \bullet s \in ac_0 \wedge ac_0 \subseteq ac' \Leftrightarrow s \in ac'$

Proof. (Implication)

$\exists ac_0 \bullet s \in ac_0 \wedge ac_0 \subseteq ac'$ {Definition of subset inclusion}

$= \exists ac_0 \bullet s \in ac_0 \wedge (\forall z \bullet z \in ac_0 \Rightarrow z \in ac')$

{Assume $s \in ac_0$ then there is a case when $z = s$}

$= \exists ac_0 \bullet s \in ac_0 \wedge (\forall z \bullet z \in ac_0 \Rightarrow z \in ac') \wedge (s \in ac_0 \Rightarrow s \in ac')$

{Assume $s \in ac_0$ and propositional calculus}

$\Rightarrow s \in ac'$

$\square$

Proof. (Reverse implication)

$s \in ac' \Rightarrow (\exists ac_0 \bullet s \in ac_0 \wedge ac_0 \subseteq ac')$ {Choose $ac_0 = ac'$}

$= (s \in ac') \Rightarrow (s \in ac' \wedge ac' \subseteq ac')$

{Reflexivity of subset inclusion and propositional calculus}

$= \mathit{true}$

$\square$

Lemma L.I.0.17 *Provided that $P[y/z]$ holds,*

$$\{z \mid P \wedge z = y \bullet Q\} = \{Q[y/z]\}$$

Appendix J

Definitions: Alphabets and Healthiness Conditions

J.1 Binary Multirelations

Definition 11 $BM \mathrel{\widehat=} State \leftrightarrow \mathbb{P}\, State$

J.1.1 Healthiness Conditions

Definition 12 $\mathbf{BMH} \mathrel{\widehat=} \forall\, s, ss_0, ss_1 \bullet ((s, ss_0) \in B \land ss_0 \subseteq ss_1) \Rightarrow (s, ss_1) \in B$

J.2 Designs

J.2.1 Alphabet

$ok, ok' : \{true, false\}$

J.2.2 Healthiness Conditions

Definition 26 (H1) $\mathbf{H1}(P) \mathrel{\widehat=} ok \Rightarrow P$

Definition 27 (H2) $\mathbf{H2}(P) \mathrel{\widehat=} [P[false/ok'] \Rightarrow P[true/ok']]$

Definition 30 (H3) $\mathbf{H3}(P) \mathrel{\widehat=} P \mathbin{;} I\!I_{\mathcal{D}}$

Definition 31 (H4) $P \mathbin{;} true = true$

J.3 Reactive Processes and CSP

J.3.1 Alphabet

$$\begin{aligned}
&ok, ok', wait, wait' : \{true, false\}\\
&tr, tr' : \mathrm{seq}\, Event\\
&ref, ref' : \mathbb{P}\, Event
\end{aligned}$$

J.3.2 Healthiness Conditions

Definition 57

$$\begin{aligned}
&\mathbf{R1}(P) \mathrel{\widehat=} P \wedge tr \le tr'\\
&\mathbf{R2}(P) \mathrel{\widehat=} P[\langle\rangle, tr' - tr/tr, tr']\\
&\mathbf{R3}(P) \mathrel{\widehat=} I\!I_{rea} \lhd wait \rhd P\\
&\mathbf{R}(P) \mathrel{\widehat=} \mathbf{R3} \circ \mathbf{R1} \circ \mathbf{R2}(P)
\end{aligned}$$

Definition 58

$$\begin{aligned}
&\mathbf{CSP1}(P) \mathrel{\widehat=} P \vee \mathbf{R1}(\neg\, ok)\\
&\mathbf{CSP2}(P) \mathrel{\widehat=} P \mathbin{;} ((ok \Rightarrow ok') \wedge tr' = tr \wedge ref' = ref \wedge wait' = wait)
\end{aligned}$$

J.4 Extended Binary Multirelations

Definition 66

$$\begin{aligned}
&State_\perp == State \cup \{\perp\}\\
&BM_\perp == State \leftrightarrow \mathbb{P}\, State_\perp
\end{aligned}$$

J.4.1 Healthiness Conditions

Definition 67 (BMH0)

$$\forall\, s, ss_0, ss_1 \bullet ((s, ss_0) \in B \wedge ss_0 \subseteq ss_1 \wedge (\perp \in ss_0 \Leftrightarrow \perp \in ss_1)) \Rightarrow (s, ss_1) \in B$$

Definition 68 (BMH1) $\forall\, s : State, ss : \mathbb{P}\, State_\perp \bullet (s, ss \cup \{\perp\}) \in B \Rightarrow (s, ss) \in B$

Definition 69 (BMH2) $\forall\, s : State \bullet (s, \emptyset) \in B \Leftrightarrow (s, \{\perp\}) \in B$

Definition 70 (BMH3)

$$\forall\, s : State \bullet (s, \emptyset) \notin B \Rightarrow (\forall\, ss : \mathbb{P}\, State_{\perp} \bullet (s, ss) \in B \Rightarrow \perp \notin ss)$$

J.5 Angelic Designs

J.5.1 Alphabet

Definition 85

$$s : State(S\alpha)$$
$$ac' : \mathbb{P}\, State(S\alpha)$$
$$ok, ok' : \{true, false\}$$
$$State(S\alpha) = \{x, e \mid x \in S\alpha\}$$

J.5.2 Healthiness Conditions

Definition 87 $\mathbf{A0}(P) \mathrel{\widehat=} P \wedge ((ok \wedge \neg\, P^f) \Rightarrow (ok' \Rightarrow ac' \neq \emptyset))$

Definition 88 $\mathbf{PBMH}(P) \mathrel{\widehat=} P \;;\; ac \subseteq ac' \wedge ok' = ok$

Definition 89 $\mathbf{A1}(P \vdash Q) \mathrel{\widehat=} (\neg\, \mathbf{PBMH}(\neg\, P) \vdash \mathbf{PBMH}(Q))$

Definition 90 $\mathbf{A}(P) \mathrel{\widehat=} \mathbf{A0} \circ \mathbf{A1}(P)$

Definition 91 $\mathbf{A2}(P) \mathrel{\widehat=} \mathbf{PBMH}(P \;;_{\mathcal{A}}\; \{s\} = ac')$

J.6 Reactive Angelic Designs

J.6.1 Alphabet

Definition 107

$$ok, ok' : \{true, false\}, s : State(\{tr, ref, wait\}), ac' : \mathbb{P}\, State(\{tr, ref, wait\})$$

J.6.2 Healthiness Conditions

Definition 109 $\mathbf{RA1}(P) \mathrel{\widehat=} (P \wedge ac' \neq \emptyset)[States_{tr \leq tr'}(s) \cap ac'/ac']$

Definition 110

$$\mathbf{RA2}(P) \mathrel{\widehat=} P\left[s \oplus \{tr \mapsto \langle\rangle\}, \left\{z \;\middle|\; \begin{array}{l} z \in ac' \land s.tr \leq z.tr \\ \bullet\ z \oplus \{tr \mapsto z.tr - s.tr\} \end{array}\right\} \middle/ s, ac'\right]$$

Definition 112 $\mathbf{RA3}(P) \mathrel{\widehat=} I\!I_{\mathbf{RAD}} \lhd s.wait \rhd P$

Definition 113 $\mathbf{RA}(P) \mathrel{\widehat=} \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{RA3}(P)$

Definition 114 $\mathbf{CSPA1}(P) \mathrel{\widehat=} P \lor \mathbf{RA1}(\neg\, ok)$

Definition 115 $\mathbf{CSPA2}(P) \mathrel{\widehat=} \mathbf{H2}(P)$

Definition 116 $\mathbf{RAD}(P) \mathrel{\widehat=} \mathbf{RA} \circ \mathbf{CSPA1} \circ \mathbf{CSPA2} \circ \mathbf{PBMH}(P)$

Definition 126 $\mathbf{ND_{RAD}}(P) = P \sqcup_{\mathbf{RAD}} Choice_{\mathbf{RAD}}$

J.7 Angelic Processes

J.7.1 Healthiness Conditions

Definition 127 $\mathbf{AP}(P) \mathrel{\widehat=} \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A} \circ \mathbf{H1} \circ \mathbf{CSPA2}(P)$

Definition 129 $\mathbf{RA3_{AP}}(P) \mathrel{\widehat=} I\!I_{\mathbf{AP}} \lhd s.wait \rhd P$

Definition 130 $\mathbf{ND_{AP}}(P) \mathrel{\widehat=} Choice_{\mathbf{AP}} \sqcup_{\mathbf{AP}} P$

Glossary

ACP	Algebra of Communicating Processes
ASM	Abstract State Machine
BNF	Backus-Naur Normal Form
CCS	Calculus of Concurrent Systems
CSP	Communicating Sequential Processes
FCD	Free Completely Distributive
FDR	Failures-Divergence Refinement
FSM	Finite State Machines
JCSP	Java Communicating Sequential Processes
LTS	Labelled Transition System
SOS	Structured Operational Semantics
UTP	Unifying Theories of Programming
VDM	Vienna Development Method
ZRC	Z Refinement Calculus

Bibliography

[1] P. Ribeiro and A. Cavalcanti, “Designs with Angelic Nondeterminism,” in *Theoretical Aspects of Software Engineering (TASE), 2013 International Symposium on.* IEEE, 2013, pp. 71–78.

[2] ——, “Angelicism in the Theory of Reactive Processes,” in *Unifying Theories of Programming*, ser. Lecture Notes in Computer Science, D. Naumann, Ed. Springer International Publishing, 2015, vol. 8963, pp. 42–61.

[3] ——, “UTP Designs for Binary Multirelations,” in *Theoretical Aspects of Computing ICTAC 2014*, ser. Lecture Notes in Computer Science, G. Ciobanu and D. Méry, Eds. Springer International Publishing, 2014, vol. 8687, pp. 388–405.

[4] E. W. Dijkstra, “The humble programmer,” *Commun. ACM*, vol. 15, pp. 859–866, October 1972. [Online]. Available: http://doi.acm.org/10.1145/355604.361591

[5] R. W. Floyd, “Assigning meanings to programs,” in *Proceedings of Symposia in Applied Mathematics*, vol. 19, 1967, pp. 19–32. [Online]. Available: http://www.cs.virginia.edu/~weimer/2007-615/reading/FloydMeaning.pdf

[6] C. A. R. Hoare, “An axiomatic basis for computer programming,” *Commun. ACM*, vol. 12, pp. 576–580, October 1969. [Online]. Available: http://doi.acm.org/10.1145/363235.363259

[7] E. W. Dijkstra, “Guarded commands, nondeterminacy and formal derivation of programs,” *Commun. ACM*, vol. 18, pp. 453–457, August 1975. [Online]. Available: http://doi.acm.org/10.1145/360933.360975

[8] J. Woodcock and J. Davies, *Using Z: Specification, Refinement, and Proof.* Prentice Hall, 1996.

[9] J. M. Spivey, *The Z notation: A Reference Manual.* Prentice Hall, 1989. [Online]. Available: http://spivey.oriel.ox.ac.uk/~mike/zrm/zrm.pdf

[10] C. Fischer, “How to Combine Z with a Process Algebra,” in *Proceedings of the 11th International Conference of Z Users on The Z Formal Specification Notation.* London, UK: Springer-Verlag, 1998, pp. 5–23. [Online]. Available: http://portal.acm.org/citation.cfm?id=647283.722938

[11] C. B. Jones, *Systematic software development using VDM.* Prentice Hall International, 1986.

[12] E. Börger, “The ASM Refinement Method,” *Formal Aspects of Computing*, vol. V15, no. 2, pp. 237–257, Nov. 2003. [Online]. Available: http://dx.doi.org/10.1007/s00165-003-0012-7

[13] ——, “The ASM Method for System Design and Analysis. A Tutorial Introduction,” in *Frontiers of Combining Systems, 5th International Workshop, FroCoS 2005, Vienna, Austria, September 19-21, 2005, Proceedings*, ser. Lecture Notes in Computer Science, B. Gramlich, Ed., vol. 3717. Springer, 2005, pp. 264–283.

[14] J.-R. Abrial, *The B-Book.* Cambridge University Press, 1996.

[15] ——, “Formal Methods: Theory Becoming Practice,” *Journal of Universal Computer Science*, vol. 13, no. 5, pp. 619–628, May 2007. [Online]. Available: http://www.jucs.org/jucs_13_5/formal_methods_theory_becoming

[16] C. A. R. Hoare, *Communicating Sequential Processes.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1985.

[17] A. W. Roscoe, *The Theory and Practice of Concurrency.* Prentice Hall, 1998.

[18] ——, *Understanding concurrent systems.* Springer, 2010.

[19] R. Milner, *Communication and Concurrency.* Prentice Hall, 1989.

[20] J. Bergstra and J. Klop, “Algebra of communicating processes with abstraction,” *Theoretical Computer Science*, vol. 37, no. 0, pp. 77 – 121, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/030439758590088X

[21] A. Cavalcanti, A. Sampaio, and J. Woodcock, “A Refinement Strategy for *Circus*,” *Formal Aspects of Computing*, vol. 15, pp. 146–181, 2003. [Online]. Available: http://dx.doi.org/10.1007/s00165-003-0006-5

[22] M. Oliveira, “Formal Derivation of State-Rich Reactive Programs using *Circus*,” Ph.D. dissertation, University of York, 2005. [Online]. Available: https://www.cs.york.ac.uk/circus/publications/papers/06-oliveira.pdf

[23] G. Smith and J. Derrick, “Specification, Refinement and Verification of Concurrent Systems - An Integration of Object-Z and CSP,” *Formal Methods in System Design*, vol. 18, pp. 249–284, 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1011269103179

[24] ——, “Abstract Specification in Object-Z and CSP,” in *Formal Methods and Software Engineering*, ser. Lecture Notes in Computer Science, C. George and H. Miao, Eds. Springer Berlin / Heidelberg, 2002, vol. 2495, pp. 108–119. [Online]. Available: http://dx.doi.org/10.1007/3-540-36103-0_14

[25] S. Schneider and H. Treharne, “Communicating B Machines,” in *ZB 2002:Formal Specification and Development in Z and B*, ser. Lecture Notes in Computer Science, D. Bert, J. Bowen, M. Henson, and K. Robinson, Eds. Springer Berlin / Heidelberg, 2002, vol. 2272, pp. 251–258. [Online]. Available: http://dx.doi.org/10.1007/3-540-45648-1_22

[26] M. Butler and M. Leuschel, “Combining CSP and B for Specification and Property Verification,” in *FM 2005: Formal Methods*, ser. Lecture Notes in Computer Science, J. Fitzgerald, I. J. Hayes, and A. Tarlecki, Eds. Springer Berlin / Heidelberg, 2005, vol. 3582, pp. 221–236. [Online]. Available: http://dx.doi.org/10.1007/11526841_16

[27] S. Schneider, H. Treharne, and H. Wehrheim, “A CSP Approach to Control in Event-B,” in *Integrated Formal Methods*, ser. Lecture Notes in Computer Science, D. Mary and S. Merz, Eds. Springer Berlin / Heidelberg, 2010, vol. 6396, pp. 260–274. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16265-7_19

[28] R.-J. Back, “On the correctness of refinement in program development,” Ph.D. dissertation, Department of Computer Science, University of Helsinki, 1978.

[29] J. M. Morris, “A theoretical basis for stepwise refinement and the programming calculus,” *Sci. Comput. Program.*, vol. 9, pp. 287–306, December 1987. [Online]. Available: http://dl.acm.org/citation.cfm?id=34898.34903

[30] C. Morgan, “The specification statement,” *ACM Trans. Program. Lang. Syst.*, vol. 10, pp. 403–419, July 1988. [Online]. Available: http://doi.acm.org/10.1145/44501.44503

[31] ——, *Programming from specifications.* Prentice Hall, 1994.

[32] R. Back and J. Wright, *Refinement calculus: a systematic introduction*, ser. Graduate texts in computer science. Springer, 1998.

[33] P. Gardiner and C. Morgan, "Data refinement of predicate transformers," *Theoretical Computer Science*, vol. 87, no. 1, pp. 143 – 162, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0304397591900292

[34] C. Morgan and P. Gardiner, "Data refinement by calculation," *Acta Informatica*, vol. 27, no. 6, pp. 481–503, 1990. [Online]. Available: http://dx.doi.org/10.1007/BF00277386

[35] I. Rewitzky, "Binary Multirelations," in *Theory and Applications of Relational Structures as Knowledge Instruments*, ser. Lecture Notes in Computer Science, H. de Swart, E. Orlowska, G. Schmidt, and M. Roubens, Eds. Springer Berlin / Heidelberg, 2003, vol. 2929, pp. 1964–1964. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24615-2_12

[36] C. E. Martin, S. A. Curtis, and I. Rewitzky, "Modelling Nondeterminism," in *MPC, volume 3125 of LNCS.* Springer, 2004, pp. 228–251.

[37] W. Guttmann, "Algebras for correctness of sequential computations," *Science of Computer Programming*, vol. 85, Part B, no. 0, pp. 224 – 240, 2014, special Issue on Mathematics of Program Construction 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642313002013

[38] A. Cavalcanti, J. Woodcock, and S. Dunne, "Angelic nondeterminism in the unifying theories of programming," *Formal Aspects of Computing*, vol. 18, pp. 288–307, 2006. [Online]. Available: http://dx.doi.org/10.1007/s00165-006-0001-8

[39] C. A. R. Hoare and H. Jifeng, *Unifying Theories of Programming.* Prentice Hall International Series in Computer Science, 1998.

[40] J. M. Morris and M. Tyrrell, "Terms with unbounded demonic and angelic nondeterminacy," *Science of Computer Programming*, vol. 65, no. 2, pp. 159 – 172, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642306002127

[41] J. Morris, "Augmenting Types with Unbounded Demonic and Angelic Nondeterminacy," in *Mathematics of Program Construction*, ser. Lecture Notes in Computer Science, D. Kozen, Ed. Springer Berlin / Heidelberg,

2004, vol. 3125, pp. 274–288. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-27764-4_15

[42] W. H. Hesselink, "Alternating states for dual nondeterminism in imperative programming," *Theoretical Computer Science*, vol. 411, no. 22-24, pp. 2317 – 2330, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S030439751000143X

[43] M. Tyrrell, J. Morris, A. Butterfield, and A. Hughes, "A Lattice-Theoretic Model for an Algebra of Communicating Sequential Processes," in *Theoretical Aspects of Computing - ICTAC 2006*, ser. Lecture Notes in Computer Science, K. Barkaoui, A. Cavalcanti, and A. Cerone, Eds. Springer Berlin / Heidelberg, 2006, vol. 4281, pp. 123–137. [Online]. Available: http://dx.doi.org/10.1007/11921240_9

[44] A. Cavalcanti and J. Woodcock, "A Tutorial Introduction to CSP in *Unifying Theories of Programming*," in *Refinement Techniques in Software Engineering*, ser. Lecture Notes in Computer Science, A. Cavalcanti, A. Sampaio, and J. Woodcock, Eds. Springer Berlin / Heidelberg, 2006, vol. 3167, pp. 220–268. [Online]. Available: http://dx.doi.org/10.1007/11889229_6

[45] T. Santos, A. Cavalcanti, and A. Sampaio, "Object-Orientation in the UTP," in *Unifying Theories of Programming*, ser. Lecture Notes in Computer Science, S. Dunne and B. Stoddart, Eds. Springer Berlin / Heidelberg, 2006, vol. 4010, pp. 18–37. [Online]. Available: http://dx.doi.org/10.1007/11768173_2

[46] F. Zeyda, T. Santos, A. Cavalcanti, and A. Sampaio, "A Modular Theory of Object Orientation in Higher-Order UTP," in *FM 2014: Formal Methods*, ser. Lecture Notes in Computer Science, C. Jones, P. Pihlajasaari, and J. Sun, Eds. Springer International Publishing, 2014, vol. 8442, pp. 627–642. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-06410-9_42

[47] W. Harwood, A. Cavalcanti, and J. Woodcock, "A Theory of Pointers for the UTP," in *Theoretical Aspects of Computing - ICTAC 2008*, ser. Lecture Notes in Computer Science, J. Fitzgerald, A. Haxthausen, and H. Yenigun, Eds. Springer Berlin / Heidelberg, 2008, vol. 5160, pp. 141–155. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85762-4_10

[48] A. Sherif and J. He, "Towards a Time Model for *Circus*," in *Proceedings of the 4th International Conference on Formal Engineering Methods: Formal Methods and Software Engineering*, ser. ICFEM '02. London,

UK, UK: Springer-Verlag, 2002, pp. 613–624. [Online]. Available: http://portal.acm.org/citation.cfm?id=646272.685816

[49] A. Sherif, "A Framework for Specification and Validation of Real-Time Systems using *Circus Actions*," Ph.D. dissertation, Center of Informatics - Federal University of Pernambuco, Brazil, 2006. [Online]. Available: http://www.cs.york.ac.uk/circus/publications/papers/06-sherif.pdf

[50] K. Wei, J. Woodcock, and A. Cavalcanti, "New *Circus Time*," University of York, Tech. Rep., February 2012. [Online]. Available: http://www.cs.york.ac.uk/circus/publications/techreports/reports/Circus%20Time.pdf

[51] J. Woodcock and A. Cavalcanti, "A Tutorial Introduction to Designs in Unifying Theories of Programming," in *Integrated Formal Methods*, ser. Lecture Notes in Computer Science, E. Boiten, J. Derrick, and G. Smith, Eds. Springer Berlin / Heidelberg, 2004, vol. 2999, pp. 40–66. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24756-2_4

[52] M. O. Rabin and D. Scott, "Finite Automata and Their Decision Problems," *IBM J. Res. Dev.*, vol. 3, no. 2, pp. 114–125, Apr. 1959. [Online]. Available: http://dx.doi.org/10.1147/rd.32.0114

[53] S. A. Cook, "The Complexity of Theorem-proving Procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, ser. STOC '71. New York, NY, USA: ACM, 1971, pp. 151–158. [Online]. Available: http://doi.acm.org/10.1145/800157.805047

[54] M. Schützenberger, "On context-free languages and push-down automata," *Information and Control*, vol. 6, no. 3, pp. 246 – 264, 1963. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0019995863903061

[55] W. H. Hesselink, "LR-parsing derived," *Science of Computer Programming*, vol. 19, no. 2, pp. 171 – 196, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/016764239290007X

[56] A. P. Martin, P. H. B. Gardiner, and J. C. P. Woodcock, "A tactic calculus - abridged version," *Formal Aspects of Computing*, vol. 8, no. 4, pp. 479–489, 1996, 10.1007/BF01213535. [Online]. Available: http://dx.doi.org/10.1007/BF01213535

[57] M. Oliveira, A. Cavalcanti, and J. Woodcock, “ArcAngel: a Tactic Language for Refinement,” *Formal Aspects of Computing*, vol. 15, no. 1, pp. 28–47, 2003. [Online]. Available: http://dx.doi.org/10.1007/s00165-003-0003-8

[58] R. Jagadeesan, V. A. Saraswat, and V. Shanbhogue, “Angelic non-determinism in concurrent constraint programming,” Xerox Park, Tech. Rep., January 1991.

[59] J. N. Kok, “On Logic Programming and the Refinement Calculus: Semantics Based Program Transformations,” Utrecht University, Technical Report RUU-CS-90-39, December 1990.

[60] R. W. Floyd, “Nondeterministic Algorithms,” *J. ACM*, vol. 14, no. 4, pp. 636–644, Oct. 1967. [Online]. Available: http://doi.acm.org/10.1145/321420.321422

[61] N. Ward and I. Hayes, “Applications of Angelic Nondeterminism,” in *Australian Software Engineering Conference 1991: Engineering Safe Software; Proceedings*, P. A. Bailes, Ed. Sydney: N.S.W.: Australian Computer Society, 1991, pp. 391–404.

[62] E. W. Dijkstra, *A Discipline of Programming*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall, 1976.

[63] R. Back and J. von Wright, “Combining angels, demons and miracles in program specifications,” *Theoretical Computer Science*, vol. 100, no. 2, pp. 365 – 383, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0304397592903094

[64] A. Cavalcanti and J. Woodcock, “Angelic Nondeterminism and Unifying Theories of Programming,” University of Kent, Tech. Rep., 2004. [Online]. Available: http://kar.kent.ac.uk/14151/

[65] W. Guttmann, “Multirelations with infinite computations,” *Journal of Logical and Algebraic Methods in Programming*, vol. 83, no. 2, pp. 194 – 211, 2014, festschrift in Honour of Gunther Schmidt on the Occasion of his 75th Birthday. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1567832614000095

[66] R. J. R. Back, “Changing data representation in the refinement calculus,” in *System Sciences, 1989. Vol.II: Software Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on*, vol. 2, Jan 1989, pp. 231–242 vol.2.

[67] B. Davey and H. Priestley, *Introduction to Lattices and Order*, ser. Cambridge mathematical textbooks. Cambridge University Press, 2002. [Online]. Available: http://books.google.co.uk/books?id=vVVTxeuiyvQC

[68] C. A. R. Hoare, "Communicating Sequential Processes," *Commun. ACM*, vol. 21, no. 8, pp. 666–677, Aug. 1978. [Online]. Available: http://doi.acm.org/10.1145/359576.359585

[69] ——, "A model for communicating sequential processes," Department of Computing Science, University of Wollongong, Tech. Rep. Working Paper 80-1, 1980.

[70] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe, "A Theory of Communicating Sequential Processes," *J. ACM*, vol. 31, no. 3, pp. 560–599, Jun. 1984. [Online]. Available: http://doi.acm.org/10.1145/828.833

[71] FDR. [Online]. Available: http://www.fsel.com/

[72] T. Gibson-Robinson, P. Armstrong, A. Boulgakov, and A. Roscoe, "FDR3 — A Modern Refinement Checker for CSP," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, E. ÃĄbrahÃąm and K. Havelund, Eds., vol. 8413, 2014, pp. 187–201.

[73] G. D. Plotkin, "A structual approach to operational semantics," *Journal of Logic and Algebraic Programming*, vol. 60, pp. 17–140, 2004.

[74] P. Ribeiro, "Angelic Processes," Ph.D. dissertation (extended version), University of York, December 2014. [Online]. Available: http://arxiv.org/abs/1505.04726

[75] J. Woodcock, "The Miracle of Reactive Programming," in *Unifying Theories of Programming*, ser. Lecture Notes in Computer Science, A. Butterfield, Ed. Springer Berlin Heidelberg, 2010, vol. 5713, pp. 202–217. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14521-6_12

[76] K. Wei, J. Woodcock, and A. Burns, "A Timed Model of Circus with the Reactive Design Miracle," in *Software Engineering and Formal Methods (SEFM), 2010 8th IEEE International Conference on*, Sept 2010, pp. 315–319.

[77] ——, "Timed Circus: Timed CSP with the Miracle," in *Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference on*, April 2011, pp. 55–64.

[78] K. Wei, J. Woodcock, and A. Cavalcanti, "*Circus Time* with Reactive Designs," in *Unifying Theories of Programming*, ser. Lecture Notes in Computer Science, B. Wolff, M.-C. Gaudel, and A. Feliachi, Eds. Springer Berlin Heidelberg, 2013, vol. 7681, pp. 68–87. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35705-3_3

[79] C. E. Martin and S. A. Curtis, "The algebra of multirelations," *Mathematical Structures in Computer Science*, vol. 23, pp. 635–674, 6 2013. [Online]. Available: http://journals.cambridge.org/article_S0960129512000965

[80] A. Cavalcanti, A. Mota, and J. Woodcock, "Simulink Timed Models for Program Verification," in *Theories of Programming and Formal Methods*, ser. Lecture Notes in Computer Science, Z. Liu, J. Woodcock, and H. Zhu, Eds. Springer Berlin Heidelberg, 2013, vol. 8051, pp. 82–99. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-39698-4_6

[81] S. Schneider, *Concurrent and real-time systems: the CSP approach*, ser. Worldwide series in computer science. John Wiley, 2000.

[82] S. Foster, F. Zeyda, and J. Woodcock, "Isabelle/UTP: A Mechanised Theory Engineering Framework," in *Unifying Theories of Programming*, ser. Lecture Notes in Computer Science, D. Naumann, Ed. Springer International Publishing, 2015, vol. 8963, pp. 21–41.

[83] A. Feliachi, M.-C. Gaudel, and B. Wolff, "Unifying Theories in Isabelle/HOL," in *Unifying Theories of Programming*, ser. Lecture Notes in Computer Science, S. Qin, Ed. Springer Berlin Heidelberg, 2010, vol. 6445, pp. 188–206. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16690-7_9

[84] F. Zeyda and A. Cavalcanti, "Encoding Circus Programs in ProofPowerZ," in *Unifying Theories of Programming*, ser. Lecture Notes in Computer Science, A. Butterfield, Ed. Springer Berlin Heidelberg, 2010, vol. 5713, pp. 218–237. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14521-6_13

[85] A. Butterfield, "Saoithín: A Theorem Prover for UTP," in *Unifying Theories of Programming*, ser. Lecture Notes in Computer Science, S. Qin, Ed. Springer Berlin Heidelberg, 2010, vol. 6445, pp. 137–156. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16690-7_6

[86] ——, "The Logic of U(TP)2," in *Unifying Theories of Programming*, ser. Lecture Notes in Computer Science, B. Wolff, M.-C. Gaudel, and A. Feliachi,

Eds. Springer Berlin Heidelberg, 2013, vol. 7681, pp. 124–143. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35705-3_6