Angelic Processes for CSP via the UTP

Pedro Ribeiro, Ana Cavalcanti

Department of Computer Science, Univesity of York, UK

Abstract

Demonic and angelic nondeterminism play fundamental roles as abstraction mechanisms for formal modelling. In contrast with its demonic counterpart, in an angelic choice failure is avoided whenever possible. Although it has been extensively studied in refinement calculi, in the context of process algebras, and of the Communicating Sequential Processes (CSP) algebra for refinement, in particular, it has been elusive. We show here that a semantics for an extended version of CSP that includes both demonic and angelic choice can be provided using Hoare and He's Unifying Theories of Programming (UTP). Since CSP is given semantics in the UTP via reactive designs (pre and postcondition pairs) we have developed a theory of angelic designs and a conservative extension of the CSP theory using reactive angelic designs. To characterise angelic nondeterminism appropriately in an algebra of processes, however, a notion of divergence that can undo the history of events needs to be considered. Taking this view, we present a model for CSP where angelic choice completely avoids divergence just like in the refinement calculi for sequential programs.

Keywords: Semantics, formal specification, process algebras, CSP, UTP
2010 MSC: 68Q55, 68Q60, 68Q85

1. Introduction

An essential abstraction mechanism that is pervasive across modelling approaches is nondeterministic choice. It can be used to specify purely nondeterministic behaviour, in which no particular bias in choices is guaranteed, but also to describe concisely a form of choice in which, if there are options that lead to success, they are guaranteed to be selected. The former is normally named demonic, while the latter is named angelic. Operationally, both 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 a refinement, the behaviour can be more deterministic than that specified, while adhering to the possible externally observable behaviours. In other words, the user is unable to force any particular choice and must accept any outcome, including failure, if this is a possibility. This corresponds to the semantics of nondeterminism in Dijkstra [1]'s guarded commands, and internal choice in CSP [2], 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 that leads to success, then the program is guaranteed to achieve a satisfying outcome. Metaphorically, the choice is said to be made by an angel, while the arbitrary form of choice is made by a demon. Operationally, angelic nondeterminism can be interpreted as a backtracking mechanism. This is similar to the underlying concept involved in searching for solutions in a given space.

Angelic nondeterminism has traditionally been studied in the refinement calculi [3, 4, 5], 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. Morgan and Gardiner [6] have

Email addresses: pedro.ribeiro@york.ac.uk (Pedro Ribeiro), ana.cavalcanti@york.ac.uk (Ana Cavalcanti)

Accepted for publication in Theoretical Computer Science *8 October 2018*

used the least upper bound to define logical variables, which, for example, enable the postcondition of a specification statement to refer to the initial value of a program variable. Logical variables are also central to their calculational data-refinement approach.

In the context of reactive and concurrent systems, however, the notions of angelic nondeterminism considered so far have been notably different. Tyrrell et al. [7] have proposed an axiomatized algebra of processes resembling CSP. Their external choice is referred to as angelic choice, but, in their model deadlock is not distinguishable from divergence. Therefore, that model and its associated notion of refinement are different from the standard (failures-divergences) semantics of CSP [8].

More recently, Roscoe [2] has proposed an angelic choice operator $P \boxdot Q$ defined using an operational combinator semantics for CSP. It is an alternative to the external choice operator 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.

A suitable notion of angelic nondeterminism for CSP, and indeed any process algebra, needs to avoid divergence. For example, the angelic choice $a \to \mathit{Chaos} \sqcup b \to \mathit{Skip}$, between a process that performs an event a followed by divergence[2], and a process that performs an event b followed by termination, should always be resolved in favour of $b \to \mathit{Skip}$. This is required to characterise angelic and demonic nondeterminism in reactive system models for both data and behavioural refinement. We show here that the UTP of Hoare and He [10] is a suitable framework to characterise angelic nondeterminism in CSP.

Since in the UTP CSP processes are defined by reactive designs, that is, via pre and postcondition pairs, we develop first a UTP theory of angelic designs based on a relational encoding of upward-closed binary multirelations [11, 12]. Using this theory, we develop a conservative extension of CSP with angelic nondeterminism. In this theory, the choice $a \to \mathit{Skip} \sqcup b \to \mathit{Chaos}$ is resolved in favour of $a \to \mathit{Skip}$ provided a and b are the same event. Otherwise, the behaviour can be described by $a \to \mathit{Skip} \sqcup b \to \mathit{Choice}$, where *Choice* is the most nondeterministic process that does not diverge. This is a consequence of the fact that, in a theory of reactive processes, the history of events observed cannot be undone. So the angel cannot change history, despite its capability to avoid divergence.

Therefore, we finally propose a new theory of angelic processes where we give up the healthiness conditions of CSP that ensure that the history of events must be preserved. In this theory angelic choice completely avoids potentially divergent processes as required, so that $a \to \mathit{Skip} \sqcup b \to \mathit{Chaos} = a \to \mathit{Skip}$.

We present the definition of basic CSP operators in this theory. Moreover, we study the relationship between our new theory and reactive angelic designs to ensure that the semantics of CSP operators is preserved. In particular, CSP programs that do not use angelic nondeterminism have the usual semantics.

We show that all theories discussed in this paper can be related using Galois connections.

The remainder of this paper is structured as follows. In Section 2 we discuss the applications of angelic nondeterminism and CSP. In Section 3 we introduce the UTP and its theories of designs and reactive designs. In Section 4 we present our framework for angelic nondeterminism. Section 5 discusses the theory of angelic designs. The theory of reactive angelic designs is introduced in Section 6 along with the characterisation of new operators. Section 7 contains our final theory for a version of CSP that can describe truly angelic processes. Finally, we summarize our results in Section 8, where we also discuss related and future work.

2. Angelic Nondeterminism and CSP

In this section we first discuss the notion of angelic determinism and its applications in Section 2.1, followed by the standard theory of CSP in Section 2.2.

2.1. Angelic Nondeterminism and its Applications

The earliest known use of angelic nondeterminism can be found in automata theory [13] and Turing machines [14], for example, to characterise context-free languages [15], and the class of NP-problems [14],

[2]Here we follow the notation of *Circus* [9], and use *Chaos* to represent divergence, which corresponds to **div** in Roscoe [2]'s presentation, and similarly we use *Choice* to represent the most nondeterministic non-divergent process, which corresponds to Roscoe [2]'s *Chaos*.

whose solutions can be found efficiently given an angelic nondeterministic machine. Since then, angelic nondeterminism has been used as a specification and programming construct in several applications, including parsing [16], modelling of game-like scenarios [4] and user interactions, theorem proving tactics [17, 18], constraint programming [19] and logic programming [20]. These are applications where finding a solution often involves a combination of search and backtracking, and thus can be specified rather abstractly using angelic choices.

Floyd [21] envisioned angelic choice as a mechanism for the abstract specification of algorithms. In his flowcharts, Floyd introduced explicit nondeterministic choice points, and appropriate notions of success and failure, to avoid implementation details of particular execution strategies. When Dijkstra [1, 22] introduced the language of guarded commands, however, the nondeterministic choice considered was no longer angelic. The semantic model is that of conjunctive, but not disjunctive, weakest preconditions that excludes angelic (as well as infeasible programs) by observing the so called "*Law of the Excluded Miracle*".

When Back [23], Morris [5] and Morgan [3] introduced the refinement calculus, miracles were allowed back into their models to yield complete lattices. Back and Wright [4] extensively studied sublattices, where choice can be either angelic or demonic. The most important model in all these works is that of monotonic predicate transformers, where angelic and demonic choice are modelled as the least upper bound and the greatest lower bound of the lattice, both extensively used in modelling.

Angelic choice plays a significant role amongst data refinement techniques, such as that of Gardiner and Morgan [24], where the least upper bound is used to formalize logical variables. This allowed for a simpler stepwise approach to data refinement via calculation.

Ward and Hayes [25], in their work 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 was no longer restricted to choice points, but applicable to any angelic construct, including a new form of angelic assignment of values to program variables, exploited in the refinement of programs from high-level specifications.

A practical application of angelic choice can be found, for example, in the approach of Cavalcanti et al. [26], where control systems specified using Simulink are modelled using *Circus* [9], a combination of the Z [27] specification language and CSP. This work demonstrates that angelic nondeterminism is useful to reason about simulations of reactive systems. Simulations have an idealised view of computation, which can take place infinitely fast. To relate the view of a system defined by the results of the idealised simulation to that of a program, we can interpret the computations in simulations as angelic choices.

In the context of theories of total correctness, computations can be specified through relations between initial and final states, such as the view adopted in Z and VDM [28]. However, as Back and Wright [4] have noted, relations can only capture one type of nondeterminism, either angelic or demonic, but not both. Furthermore, Cavalcanti and Woodcock [29] have established that, in general, UTP relations are isomorphic to conjunctive predicate transformers. To define a theory of relations with angelic nondeterminism they considered a predicative encoding of Rewitzky [11]'s upward-closed binary multirelations, which is at the core of our work on catering for angelic and demonic nondeterminism in CSP.

2.2. CSP

The central notion of CSP is that of communicating processes. These include basic processes, such as *Skip*, which terminates successfully, *Stop*, which behaves as deadlock and hence refuses to do anything, and **div**, which behaves unpredictably. In CSP communication is modelled by defining events, which the system can perform only with the cooperation of its environment. Once agreement is reached, events happen instantaneously and atomically. The prefixing $a \rightarrow P$ first offers the environment the possibility to perform event a, while allowing any other events to be refused, after which it behaves like P. An external choice $P \,\Box\, Q$ offers the environment the possibility to behave as P or Q, with the choice being resolved either through a communication or via termination. On the other hand, an internal choice $P \sqcap Q$, which is demonic, is under the control of the system, and so the environment cannot possibly force the system into behaving as either P or Q. Finally two processes P and Q can be sequentially composed $P \,;\, Q$, so that the termination of P leads to the behaviour of Q. The treatment of other important CSP operators like parallel composition and hiding can be found in [2].

The simplest semantic model for CSP considers the observable sequences of events that a process may produce as a trace, a sequence of type Σ ($Trace : \operatorname{seq}\Sigma$, following the Z notation), where Σ is the set of all possible events. Refinement in this model allows reasoning about safety, that is, a valid refinement can never perform traces not in the original specification, but may refuse to perform any event at all, like *Stop*. To capture liveness the failures model records the events refused after having performed a trace. In this case we record for each process a pair of type $Trace \times \mathbb{P}\, Failure$, where $Failure : (Trace \times \mathbb{P}\Sigma)$ is a pair, whose first component is a trace and whose second component is a set of events, containing those events refused after the trace. A refinement cannot refuse more events, and so in this model *Stop* is not a valid refinement of every process, unlike in the traces model. Finally, to capture divergent behaviour, such as that exhibited by **div**, the traces leading to a divergence are recorded, with the view that two processes that can diverge immediately are equivalent and useless, and once divergent can perform any trace and refuse any event (also known as divergence strictness). In this case we record for each process a pair of type $\mathbb{P}\, Failure \times Divergences$, where $Divergences$ is of type $\mathbb{P}(\operatorname{seq}\Sigma)$.

The results by Back and Wright [4], Cavalcanti and Woodcock [29] indicate that both angelic and demonic nondeterminism cannot be characterised in the traditional failures-divergences model. To illustrate this result, we construct a simple relational model isomorphic to failures-divergences (without considering the treatment of termination) by defining the following pair of mapping functions, $fd2r$ that takes a pair of failures-divergences (F, D) and yields a relation on a type $State = Failure \cup \bot$, with $\bot$ used to record divergent behaviour, and $r2fd$ that maps in the opposite direction.

Definition 1.

$$fd2r : (\mathbb{P}\, Failure \times Divergences) \to (State \leftrightarrow State)$$

$$fd2r(F, D) = \left(\begin{array}{l} \{(s, s') \mid \{s, s'\} \subseteq F \land \exists\, a : \Sigma \bullet trace(s') = trace(s) \frown \langle a \rangle\} \cup \\ \{(s, s') \mid trace(s) \in D \land (s' = \bot \lor \exists\, a : \Sigma \bullet trace(s') = trace(s) \frown \langle a \rangle)\} \cup \{(\bot, \bot)\} \end{array} \right)$$

The definition of $fd2r$ is the union of three sets. The first set captures the valid traces of a process by relating every initial state s to a final state s', such that s and s' are in the set of failures F and the $trace(s)$ of s is a proper prefix of $trace(s')$ that is one event shorter, where *trace* is a function that yields the traces in a state s. The second set considers divergences by relating every state s whose trace $trace(s)$ leads to a divergence in D with $\bot$, to record that there is a divergence, and with every failure whose trace is one event shorter. Finally, for consistency we also relate $\bot$ with $\bot$.

Similarly, the function $r2fd$ can be defined by considering the pair whose components are given by $r2f$, which yields a set of failures, and $r2d$, which yields a set of diverging traces.

Definition 2.

$$r2f(R) = \{s' \mid \exists\, s \bullet (s, s') \in R \land s \neq \bot \land s' \neq \bot\}$$

$$r2d(R) = \{t \mid \exists\, s \bullet (s, \bot) \in R \land t = trace(s) \land s \neq \bot\} \qquad r2fd(R) = (r2f(R), r2d(R))$$

The function $r2f$ is defined by considering every proper failure, that is, different form $\bot$, s' that is related to a proper failure. The definition of $r2d$ considers the traces t of every state s related to $\bot$ in R. Under the healthiness conditions of the model of failures-divergences, we can establish the following Theorem 1, proof of which is available in [30]. Prefix-closure requires that for every valid trace its prefix can also be observed. Failures-strictness requires that upon divergence, every event can be refused, and divergence-strictness requires that every extension of a divergent trace is also divergent.

Theorem 1. *Provided F and R are prefix-closed, failures-strict, and D and R are divergence-strict, then $r2fd \circ fd2r(F, D) = (F, D)$ and $fd2r \circ r2fd(R) = R$.*

Furthermore, we can also show that under the same assumptions the mappings are closed under the healthiness conditions. Therefore, we can conclude the model is isomorphic to a relational model, which we know cannot model both demonic and angelic nondeterminism. To model both forms of nondeterminism, we need a complete lattice. We use the framework of the UTP, as described in the next few sections.

3. Unifying Theories of Programming

The UTP of Hoare and He [10] is an alphabetized, predicative theory of relations suitable for characterising different programming paradigms. It promotes unification of semantic models, while enabling different aspects of paradigms (data, reaction, time, and so on) to be considered in isolation. Several theories have been proposed that consider concurrency, logic and higher-order programming, object-orientation, pointers, probability and others. A survey is available in [31].

A UTP theory is characterised by three components: an alphabet, a set of healthiness conditions, and a set of operators. The UTP is based on the principle of observation, and so the discourse 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 theory is formed by predicates described using its operators and whose free variables are in its alphabet. Healthiness conditions define the valid predicates of the theory.

In what follows we introduce the UTP theoretical framework for CSP. Section 3.1 discusses the core UTP theory of relations, Section 3.2, the theory of designs, Section 3.3, theory of reactive processes, and, finally, Section 3.4 discusses the theory of CSP (without angelic nondeterminism).

3.1. Relations

In the UTP relations are alphabetised, that is, a set of named and typed variables given by $\alpha(P)$, is used to refer to the components of the source S and target T of a relation $P : S \leftrightarrow T$. For example, in a theory of discrete time we may have variables t and t' of type $\mathbb{N}$ to record the time, with t used to record the initial value, and t' to record the final or after value. A program, which is a relation, that increments the initial value of t can be specified by the predicate $t' = t + 1$.

The alphabet $\alpha(P)$ of a relation P is split into two disjoint subsets: $in\alpha(P)$, which contains undashed variables recording initial observations of P, and $out\alpha(P)$, containing dashed counterparts for after or final observations. 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, it is homogeneous.

Definition 3. *A relation P is homogeneous if, and only if, $(in\alpha(P))' = out\alpha(P)$.*

Here $(in\alpha(P))'$ is the set of variables obtained by dashing every variable in the set $in\alpha(P)$.

3.1.1. Healthiness Conditions

The healthiness conditions of a theory are usually specified as idempotent and monotonic functions from predicates to predicates. The healthy (valid) predicates are the fixed points of these functions.

Example 1. *For example, considering our previous example of a theory of discrete-time using a variable t to record time, a plausible expectation of such a model is that a system must guarantee that time is increasingly monotonic. This can be described by the following conjunctive healthiness condition* **HC**. $\mathbf{HC}(P) \mathrel{\widehat{=}} P \wedge t \leq t'$. *It requires that the initial value of t is less than or equal to the final or after value t'.* □

Conjunctive healthiness conditions are known to be idempotent and monotonic [32].

3.1.2. Refinement

The theory of relations forms a complete lattice, with the order given by reverse universal implication, corresponding to the notion of refinement defined below.

Definition 4 (Refinement). $P \sqsubseteq Q \mathrel{\widehat{=}} [Q \Rightarrow P]$

We use square brackets in $[P]$ to stand for universal quantification over all the variables in the alphabet of P. The top of the lattice is *false* and the bottom is *true*.

Refinement can be understood as capturing the notion of correctness in the sense that, if a relation 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.

3.1.3. Operators

A UTP theory comprises a number of operators that allow more complex behaviours to be specified. In the theory of relations there are core operators that correspond to typical constructs found in programming languages, such as conditional, sequential composition and recursion. We present their definition next.

Conditional. The following operator is similar to a conditional statement in an imperative language.

Definition 5 (Conditional). $P \lhd Q \rhd R \mathrel{\widehat{=}} (Q \land P) \lor (\neg\, Q \land R)$

If the relation Q holds, then the program behaves as P, otherwise it behaves as R. If Q has no free dashed variables, then Q is a condition, and the conditional is a standard if-then-else.

Sequential Composition. In theories whose relations are homogeneous, sequential composition is defined as relational composition. The definition is reproduced below.

Definition 6 (Sequential Composition). $P\,;\,Q \mathrel{\widehat{=}} \exists\, v_0 \bullet P[v_0/v'] \land Q[v_0/v]$

The intuition is that the sequential composition of two relations P and Q involves some intermediate, unobservable state, characterised by values for the vector of (input) alphabet variables 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.

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 Tarski [10, 33]. In the following definition, F is a monotonic function and $\bigsqcap$ is the greatest lower bound.

Definition 7 (Recursion). $\mu\, X \bullet F(X) \mathrel{\widehat{=}} \bigsqcap\{X \mid [F(X) \sqsubseteq X]\}$

A nonterminating recursion, such as $(\mu\, Y \bullet Y)$, is equated with the bottom of the lattice, *true* [10]. Intuitively, this means that it does not terminate, but if we sequentially compose this recursion with another program, for example $(\mu\, Y \bullet Y)\,;\; x := 2$, then it becomes possible to recover from the nontermination, as the calculation yields $x := 2$. To appropriately capture total correctness, this has motivated Hoare and He [10] to propose the theory of designs that we discuss next.

3.2. Designs

In the UTP, total correctness is characterised through the theory of designs [10, 34], whose alphabet includes besides the program variables and their dashed counterparts, two auxiliary Boolean variables: ok and ok'. They 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*.

3.2.1. Healthiness Conditions

Valid predicates of the theory of designs must obey two principles: that no guarantees can be made by a program before it has started, and, that no program may require nontermination. These are characterised by the healthiness conditions **H1** and **H2**, respectively, which we reproduce below.

Definition 8 (Healthiness Conditions of Designs).

$$\mathbf{H1}(P) \mathrel{\widehat{=}} ok \Rightarrow P \qquad\qquad \mathbf{H2}(P) \mathrel{\widehat{=}} P\,;((ok \Rightarrow ok') \land v' = v)$$

The definition of **H1** states that any guarantees made by P can only be established once it has started. The healthiness condition **H2**, which is defined using sequential composition, allows the value of ok to increase monotonically, while every other variable v is unchanged. In other words, a design that is unstable $(\neg\, ok)$, may or may not terminate, but cannot enforce non-termination.

A fixed point of both **H1** and **H2** satisfies the equality below [10].

Lemma 1 (Design). $\mathbf{H1} \circ \mathbf{H2}(P) = (ok \wedge \neg\, P[false/ok']) \Rightarrow (P[true/ok'] \wedge ok')$

Here the design is split into two parts: a precondition and a postcondition. It is defined using the notation of Hoare and He [10] as shown in the following definition, where $\vdash$ binds weaker than the logic operators. We observe that a precondition can refer to the after or final value of an observation variable, which, as we discuss in the next paragraph, is required to accommodate the CSP theory.

Definition 9 (Design). $(P \vdash Q) \mathrel{\widehat{=}} (ok \wedge P) \Rightarrow (ok' \wedge Q)$

A design can also be written using the following notation, where we use the shorthand $P^a = P[a/ok']$, where a can be either $t = true$ or $f = false$, as introduced by Woodcock and Cavalcanti [34], 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 2 (Design). *A predicate P is a design if, and only if, it can be written as: $(\neg\, P^f \vdash P^t)$.*

We observe that the functions **H1** and **H2** are idempotent and monotonic with respect to refinement [10]. 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.

In the theory of designs, an additional healthiness condition **H3** requires $\mathbb{I}_{\mathcal{D}}$, the identity of the theory, to be a right-unit for sequential composition.

Definition 10. $\mathbf{H3}(P) \mathrel{\widehat{=}} P \,;\, \mathbb{I}_{\mathcal{D}}$ *where* $\mathbb{I}_{\mathcal{D}} \mathrel{\widehat{=}} (true \vdash v' = v)$

H3 requires the precondition not to have any dashed variables. In order to understand the intuition behind it we consider an example of a design that is not **H3**-healthy.

Example 2.

$$
\begin{array}{ll}
(x' \neq 2 \vdash true) & \{\text{Definition of designs}\} \\
= (ok \wedge x' \neq 2) \Rightarrow ok' & \{\text{Propositional calculus}\} \\
= ok \Rightarrow (x' = 2 \vee ok') &
\end{array}
$$

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 x and termination is then not required. In a theory of total correctness for sequential programs, this is a behaviour that would not normally be expected. □

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 [12].

The theory of designs is a complete lattice [10] with the everywhere miraculous program as the top $\top_D \mathrel{\widehat{=}} (true \vdash false)$ and abort as the bottom $\bot_D \mathrel{\widehat{=}} (false \vdash true)$. Miracle cannot be started while abort has arbitrary behaviour. Both programs play an important role in refinement calculi.

3.3. Reactive Processes

CSP can be characterised in the UTP through the theory of reactive processes. In addition to ok and ok', this theory includes the variables $wait$, tr, ref and their dashed counterparts, that record information about termination, possible interactions with the environment, and possibility of refusing interaction. Similarly, $wait'$ records this information 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 occurs in states where ok' and $\neg\, wait'$ are $true$.

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.

The theory of reactive processes contains the fixed points of the functional composition of the following three healthiness conditions **R1**, **R2** and **R3**.

Definition 11 (Healthiness Conditions of Reactive Processes).

$$\mathbf{R1}(P) \mathrel{\widehat{=}} P \wedge tr \leq tr' \qquad \mathbf{R2}(P) \mathrel{\widehat{=}} P[\langle\rangle, tr' - tr/tr, tr']$$
$$\mathbf{R3}(P) \mathrel{\widehat{=}} \mathbb{I}_{rea} \lhd wait \rhd P, \text{ where } \mathbb{I}_{rea} \mathrel{\widehat{=}} (\mathbf{R1}(\neg\, ok) \vee (v' = v))$$

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 tr. As we discuss later, this condition prevents a proper characterisation of angelic nondeterminism for reactive processes. We let go of it in our final theory of angelic processes.

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, that is, *wait* is *true*, then the current process P behaves as the identity of the theory $\mathbb{I}_{rea}$, otherwise it behaves as P itself. Finally, the healthiness condition of the theory of reactive processes is **R**, the functional composition of **R1**, **R2** and **R3**.

Definition 12 (Reactive Process). $\mathbf{R}(P) \mathrel{\widehat{=}} \mathbf{R1} \circ \mathbf{R2} \circ \mathbf{R3}(P)$

Like the theory of designs, this theory also forms a complete lattice under refinement.

3.4. CSP Processes as Reactive Designs

The theory of CSP can be described by reactive processes that, in addition, satisfy two healthiness conditions, **CSP1** and **CSP2**, whose definitions we reproduce below.

Definition 13 (Healthiness Conditions of CSP).

$$\mathbf{CSP1}(P) \mathrel{\widehat{=}} P \vee \mathbf{R1}(\neg\, ok)$$
$$\mathbf{CSP2}(P) \mathrel{\widehat{=}} P\,;((ok \Rightarrow ok') \wedge tr' = tr \wedge ref' = ref \wedge wait' = wait)$$

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** 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. We reproduce this result below, where we use the notation $P_w = P[w/wait]$.

Theorem 2. *For every CSP process* P, $\mathbf{R}(\neg\, P_f^f \vdash P_f^t) = P$.

This result is important as it establishes that CSP processes can be specified by pre and postconditions, like sequential programs, with **R** enforcing the required reactive behaviour. For example, $a \rightarrow Chaos$, which diverges after performing the event a, is specified by a non-**H3** reactive design as follows.

Example 3. $a \rightarrow Chaos \mathrel{\widehat{=}} \mathbf{R}(\neg\, tr \frown \langle a\rangle \leq tr' \vdash tr' = tr \wedge a \notin ref' \wedge wait')$

The precondition requires the concatenation of tr with a not to be a prefix of tr': when a happens, the process diverges. The postcondition describes the behaviour before a is performed: while the process is waiting, when $wait'$ is *true*, the trace stays unchanged and a is not refused. Once a occurs, the postcondition makes no guarantees. The application of **R1**, however, still guarantees extension of the trace.

Despite their name, reactive designs are not designs since **R1** requires the extension of the trace to be observed in every circumstance, whereas **H1** states that there are no guarantees in an unstable state. The application of **R1** after **H1** turns a design into a reactive process. Conversely, a reactive process can be turned into a design, by applying **H1**. These two functions form a Galois connection, which relates non-isomorphic theories with different expressivity. Here we reproduce the general definition of Hoare and He [10] and provide a pictorial illustration in Fig. 1.

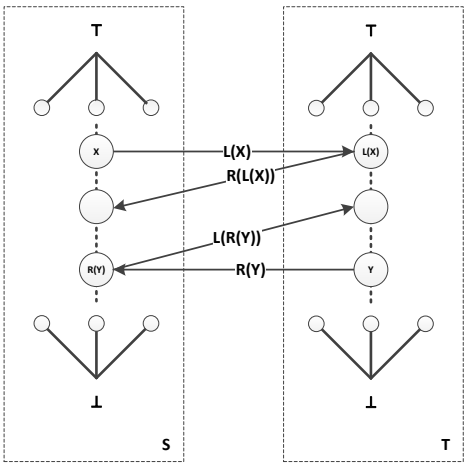


Figure 1: Galois connection between two lattices, S and T

Definition 14 (Galois Connection). *For lattices S and T, a pair (L, R) of functions $L : S \to T$ and $R : T \to S$ is defined to be a Galois connection if, and only if, for all $X \in S$ and $Y \in T$:*

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

A link stronger than a Galois connection is a bijection, where each function completely undoes the effect of the other. Not every bijection is a Galois connection. Hoare and He [10] give the example of negation whose inverse is itself, but is not monotonic. It is a known property of Galois connections that the functions are monotonic. In addition, the composition of Galois connections is also a Galois connection.

Our theories, which we describe in the following sections, extend all the theories presented above to cater for a theory that can be used to define angelic nondeterminism in CSP.

4. UTP Framework for Angelic Nondeterminism

As already discussed, when relational models are considered, only one type of nondeterminism can be modelled, either angelic or demonic, but not both [4, 29]. Our semantic framework is based on an encoding of multirelations.

In Fig. 2 we illustrate the complete UTP framework for angelic nondeterminism. Each theory of interest is depicted by an ellipse. Labels correspond to the name of the characterising healthiness condition of each theory. 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). The following sections contain a brief introduction to each theory of interest, followed by a complete formal characterisation in the sequel. In Section 4.1 we discuss the multirelational encoding of upward-closed binary multirelations. Section 4.2 presents the theory of angelic designs. The theory of reactive angelic designs is discussed in Section 4.3 followed by the theory of angelic processes in Section 4.4.

4.1. Binary Multirelations and their UTP encoding

To model both forms of nondeterminism in the UTP, a multirelational approach has been considered by Cavalcanti et al. [12]. That work has proposed a predicative encoding of upward-closed binary multirelations, that is, relations between initial states and sets of final states [11]. In that theory, the alphabet consists of input program variables and a sole output variable ac' that records a set of final states available for angelic choice. Intuitively, the states available for angelic choice are those in ac', while demonic choice is captured by choice over the value of ac' itself.

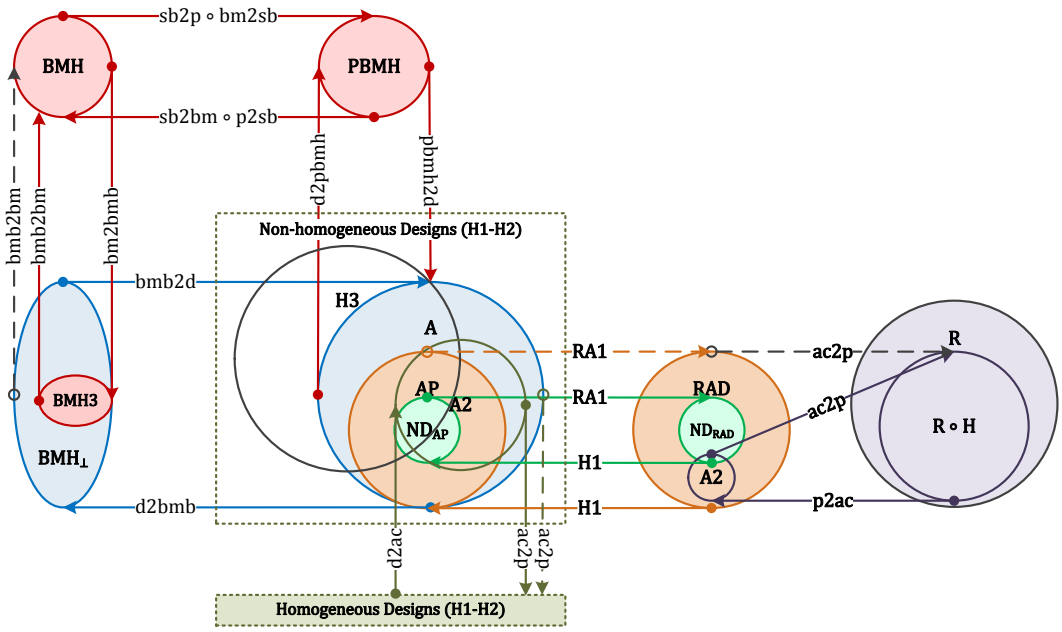


Figure 2: Theories and their relationship through linking functions

Example 4. *We consider the following example, where we define an angelic choice between the assignment of* 1 *and* 2 *to the only program variable* x*:* $x := 1 \sqcup x := 2 = \{x' \mapsto 1\} \in ac' \wedge \{x' \mapsto 2\} \in ac'$. □

Angelic choice is the least upper bound operator ($\sqcup$). As shown above, in every possible set of angelic choices ac', both assignments are defined to be available to the angel. It is the states that are in the intersection of all possible values of ac' that are effectively guaranteed choices for the angel, whatever choice the demon makes. This example illustrates a fundamental property of multirelations: upward closure. This property is enforced by the following healthiness condition, where v and v' refer to every variable other than ac and ac', respectively, and PBMH stands for Predicative Binary Multirelation Healthiness Condition.

Definition 15. $\mathbf{PBMH}(P) \mathrel{\widehat{=}} P \,;(ac \subseteq ac' \wedge v' = v)$

A fixed point of **PBMH** requires that, if some set of final states ac' is available for angelic choice, then all of its supersets are also available. The use of ac yields a homogeneous relation suitable for the composition, while $v' = v$ generalises the definition of Cavalcanti et al. [12] by keeping other variables unchanged.

Since the refinement order is defined as universal reverse implication, like in other UTP theories, the number of angelic choices can be augmented in a refinement step (by further constraining the possible values of ac' and, therefore, enlarging their distributed intersection). Conversely, demonic choice is the greatest lower bound, disjunction, and so demonic choices can be refined away as usual.

An immediate consequence of **PBMH** is that no well-behaved program can require the set of final states ac' to be empty, as established by the following lemma.

Lemma 3. $\mathbf{PBMH}(ac' = \emptyset) = \mathit{true}$

Since there is a requirement that ac' is upward closed, this theory also satisfies the constraint enforced by **H3**: arbitrary behaviour when there is nontermination. The proof of Lemma 3, and all other proofs omitted in the sequel, are available in [35].

In Fig. 2 we depict the **PBMH** theory. It is isomorphic [12] to that of binary multirelations of Rewitzky [11] with the label **BMH**. The adjoints, themselves compositions of linking functions, $sb2p \circ bm2sb$ and $sb2bm \circ p2sb$, are defined in [12] and included for completeness.

	Description
A0	Whenever the precondition of a design is satisfied, then the set of angelic choices is not 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: Healthiness Conditions of Angelic Designs

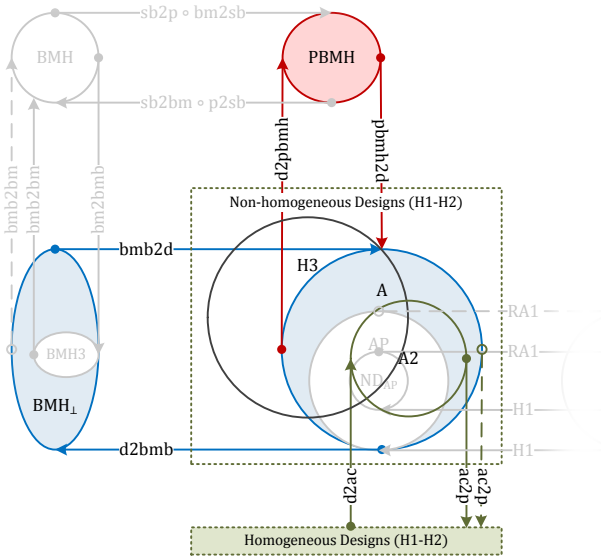


Figure 3: Theory of angelic designs and links

4.2. Angelic Designs

Based on the encoding of the **PBMH** theory, we develop a theory of angelic designs where we reintroduce the auxiliary Boolean variables ok and ok' 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), and **H1** and **H2** of the original theory of designs. In Fig. 3, we highlight the theory of angelic designs in the context of Fig. 2. We formalise this theory in Section 5.

The additional healthiness condition **A2** characterises the subset of **A** designs that do not exhibit angelic nondeterminism. It is useful for the validation: we establish that the subset of **A2** angelic designs is isomorphic to the original theory of homogeneous designs, via the linking functions $d2ac$ and $ac2p$. Moreover, we also establish that the subset of angelic designs that is **H3**-healthy is isomorphic to the **PBMH** theory. This uses two linking functions, $d2pbmh$ and $pbmh2d$, which map predicates in that theory to angelic designs, and vice versa. The formal definition of these healthiness conditions is discussed in Section 5: monotonicity, idempotency, commutativity and the isomorphisms are proved in [35].

For validation, we have also developed an extended set-based model of binary multirelations (labelled as $\mathbf{BMH}_\perp$) that is isomorphic to **A**-healthy designs [36, 35]. This model is more expressive than the original model of binary multirelations in that it can capture non-**H3** designs. That model is not only useful to study certain aspects of angelic designs, but also allows related theories to be studied in the set-based model of binary multirelations. In Fig. 2 we illustrate the relevant isomorphism via the pair of linking functions $bmb2d$ and $d2bmb$.

	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 terminated.
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 2: Healthiness Conditions of Reactive Angelic Designs

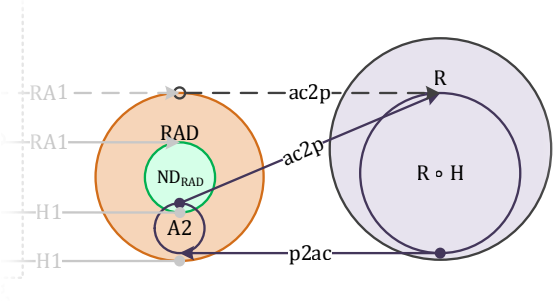


Figure 4: Theory of reactive angelic designs and links with CSP

4.3. Reactive Angelic Designs

Based on the theory of angelic designs, we then define a conservative extension of CSP with angelic nondeterminism. It encodes the observational variables of reactive processes as record components of states, and adopts every healthiness condition of CSP in the context of this encoding. For each healthiness condition **R1**, **R2**, **R3**, **CSP1** and **CSP2**, we introduce a counterpart in this model, as summarized in Table 2.

The theory is characterised by **RAD**, which is defined by the composition of all healthiness conditions of interest. The healthiness condition $\mathbf{ND_{RAD}}$ is useful for studying the subset of non-divergent processes.

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 $ac2p$ and $p2ac$. In general, if we consider the superset **RAD**, a Galois connection exists between the theories. This relationship is illustrated in Fig. 4. The formalisation of reactive angelic designs is the subject of Section 6.

4.4. Angelic Processes

As already said, to allow angelic choice to exclude potentially divergent processes, we enrich 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. To this end, we redefine **RA3** as $\mathbf{RA3_{AP}}$, and define the healthiness condition **AP** via the composition described in Table 3. When compared to **RAD**, we let go of **RA1** and, therefore, **CSP1**, and adopt $\mathbf{RA3_{AP}}$ instead of **RA3**. As a consequence this model is effectively a theory of angelic designs, where **RA1** is required in the postcondition.

	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 3: Healthiness Conditions of Angelic Processes

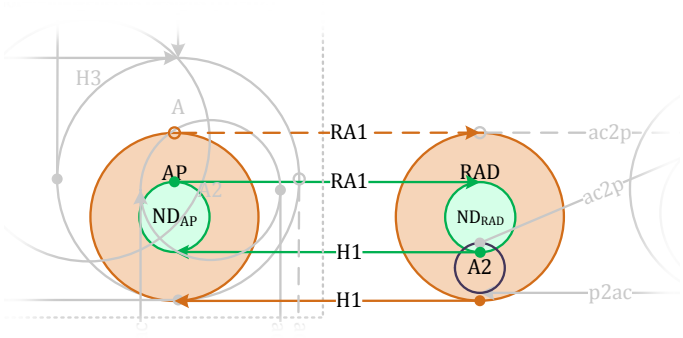


Figure 5: Theory of angelic processes and link with reactive angelic designs

We establish a Galois connection between the theory of angelic processes and the theory of reactive angelic designs, and also prove that an isomorphism exists for 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 Fig. 5 where we highlight both theories in the context of Fig. 2.

A detailed account of the theory of angelic processes is presented in Section 7.

Having now given an overview of our theories for angelic nondeterminism, we present in the next sections a more detailed account of their alphabet, healthiness conditions, and Galois connections. For the final theory of angelic processes, we also present the definition of some operators.

5. Angelic Designs

Our theory of angelic designs is defined by considering the observational variables ok and ok' and two additional variables $s : State(S\alpha)$ and $ac' : \mathbb{P}\, State(S\alpha)$, where $State(S\alpha)$ is a record type parametrised by a set of variables $S\alpha$ specifying the names of the record components. We represent a record as a set of ordered pairs, where the first component is a variable name, from the set of all possible variables $S\alpha$, and the second component corresponds to the associated value.

Definition 16 (Alphabet).

$$s : State(S\alpha);\ ac' : \mathbb{P}\, State(S\alpha);\ ok, ok' : \{true, false\},\ where\ State(S\alpha) = \{x \mapsto e \mid x \in S\alpha\}$$

The variable s encapsulates the initial values of program variables as record components of s. The set of final states ac' is similar to that of the **PBMH** theory. (A notable difference is that we do not dash the variable names in the record components. This deliberate choice bears no consequences, other than simplifying notation.) The set of program variables $S\alpha$ recorded in both s and final states of ac' is the same.

Example 5. *We consider the following angelic design, where the value of the expression e is assigned to the program variable x: $x := e \mathrel{\widehat{=}} (true \vdash s \oplus \{x \mapsto e\} \in ac')$. The precondition is true, while the postcondition states that there is a final state in ac' where the initial state s is overridden so that the component x takes the value of e.* □

5.1. Healthiness Conditions

The healthiness conditions for angelic designs are **H1** and **H2** from the theory of designs, and **A**, defined as the functional composition of **A0** and **A1** presented below.

Definition 17.

$$\mathbf{A0}(P) \mathrel{\widehat{=}} P \wedge ((ok \wedge \neg P^f) \Rightarrow (ok' \Rightarrow ac' \neq \emptyset))$$
$$\mathbf{A1}(P) \mathrel{\widehat{=}} (\neg\, \mathbf{PBMH}(P^f) \vdash \mathbf{PBMH}(P^t))$$
$$\mathbf{A}(P) \mathrel{\widehat{=}} \mathbf{A0} \circ \mathbf{A1}(P)$$

The healthiness condition **A0** requires that when a design terminates successfully, then there must be some final state in ac' available for angelic choice. The application of **A0** to a design strengthens its postcondition to require the set of states ac' not to be empty; this is established by the following theorem. Detailed proof of this result and all others omitted in the sequel can be found in [35].

Theorem 3. $\mathbf{A0}(\neg P^f \vdash P^t) = (\neg P^f \vdash P^t \wedge ac' \neq \emptyset)$

A1 requires that the final set of states in both the postcondition and the negation of the precondition are upward closed. We observe that **A1** can also be expressed as the application of **PBMH** to the whole of the design P. We consider the following example, where **A** is applied to the assignment operator.

Example 6.

$$\begin{array}{ll}
\mathbf{A}(x := e) & \{\text{Definition of assignment}\} \\
= \mathbf{A}(true \vdash s \oplus \{x \mapsto e\} \in ac') & \{\text{Definition of } \mathbf{A}\} \\
= \mathbf{A0} \circ \mathbf{A1}(true \vdash s \oplus \{x \mapsto e\} \in ac') & \{\text{Definition of } \mathbf{A0} \text{ and } \mathbf{A1}\} \\
= (\neg\, \mathbf{PBMH}(\neg\, true) \vdash \mathbf{PBMH}(s \oplus \{x \mapsto e\} \in ac') \wedge ac' \neq \emptyset) & \{\text{Predicate calculus}\} \\
= (\neg\, \mathbf{PBMH}(false) \vdash \mathbf{PBMH}(s \oplus \{x \mapsto e\} \in ac') \wedge ac' \neq \emptyset) & \{\text{Definition of } \mathbf{PBMH}\} \\
= (\neg\, false \vdash s \oplus \{x \mapsto e\} \in ac' \wedge ac' \neq \emptyset) & \{\text{Predicate calculus}\} \\
= (true \vdash s \oplus \{x \mapsto e\} \in ac') &
\end{array}$$

The application of **A1** *to the design results in the application of* **PBMH** *to both the negation of the precondition and the postcondition. When* **A0** *is applied to the design, following from Theorem 3, the conjunct $ac' \neq \emptyset$ appears in the postcondition. The resulting design is a fixed point of* **A**. □

Since **H1**, **H2** and **A** commute, and these functions are all idempotent and monotonic, so is the functional composition of **H1**, **H2** and **A**. Furthermore, because **A** is idempotent and monotonic, and the theory of designs is a complete lattice, so is our theory of **A**-healthy designs [10, Sec. 4.1, Ch. 4].

5.2. Sequential Composition

Amongst the operators in the theory of angelic designs we single out sequential composition as the most interesting due to the use of non-homogeneous relations. Its definition is given by considering the auxiliary variables ok and ok' separately, as follows.

Definition 18. $P \;;_{\mathcal{D}}\; Q \mathrel{\widehat{=}} \exists\, ok_0 \bullet P[ok_0/ok'] \;;_{\mathcal{A}}\; Q[ok_0/ok]$

This definition resembles relational composition with the difference that, instead of relational composition, we use the operator $;_{\mathcal{A}}$ that handles the non-homogeneous alphabet. Below, we reproduce the definition of the sequential composition operator $;_{\mathcal{A}}$ of the **PBMH** theory, in the context of this theory.

Definition 19. $P \mathbin{;_{\mathcal{A}}} Q \mathrel{\widehat{=}} P[\{s \mid Q\}/ac']$

The resulting sets of angelic choices are those of Q, that can be reached from initial states of Q that are available for P as a set ac' of angelic choices. This use of substitution can be interpreted as back propagating the necessary information concerning the final states. By way of illustration, we consider the following example. The choice is between the assignment of *true* or *false* to the program variable b, as denoted by t and f, respectively. This is sequentially composed with the program that maintains the value of b provided that the initial value of b is *true*, and otherwise aborts.

Example 7.

$$
\begin{array}{ll}
(\{b \mapsto t\} \in ac' \sqcup \{b \mapsto f\} \in ac') \mathbin{;_{\mathcal{A}}} (s.b \Rightarrow s \in ac') & \{\text{Definition of } \sqcup\} \\
= (\{b \mapsto t\} \in ac' \wedge \{b \mapsto f\} \in ac') \mathbin{;_{\mathcal{A}}} (s.b \Rightarrow s \in ac') & \{\text{Definition of } ;_{\mathcal{A}}\} \\
= (\{b \mapsto t\} \in ac' \wedge \{b \mapsto f\} \in ac')[\{s \mid s.b \Rightarrow s \in ac'\}/ac'] & \{\text{Substitution}\} \\
= \{b \mapsto t\} \in \{s \mid s.b \Rightarrow s \in ac'\} \wedge \{b \mapsto f\} \in \{s \mid s.b \Rightarrow s \in ac'\} & \{\text{Property of sets}\} \\
= (\{b \mapsto t\}.b \Rightarrow \{b \mapsto t\} \in ac') \wedge (\{b \mapsto f\}.b \Rightarrow \{b \mapsto f\} \in ac') & \{\text{Record component } b\} \\
= (\mathit{true} \Rightarrow \{b \mapsto t\} \in ac') \wedge (\mathit{false} \Rightarrow \{b \mapsto f\} \in ac') & \{\text{Predicate calculus}\} \\
= \{b \mapsto t\} \in ac' &
\end{array}
$$

The only possible result is the assignment of true to b, since this avoids aborting. □

Proof for closure of sequential composition and other operators under **A** is available in [35].

5.3. Characterising Designs without Angelic Nondeterminism

The sequential composition operator allows us to characterise the subset of angelic designs that do not exhibit angelic nondeterminism. Such a design always provides at most one angelic choice. In other words, for every initial state, there is at most one final state available in the distributed intersection over all possible values of ac'. This leads to the following healthiness condition **A2**.

Definition 20. $\mathbf{A2}(P) \mathrel{\widehat{=}} \mathbf{PBMH}(P \mathbin{;_{\mathcal{A}}} \{s\} = ac')$

Informally, **A2** requires the set of final states in P to be either empty or a singleton, otherwise it becomes *false*. Since this purposely 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 4.

Theorem 4. $\mathbf{A2}(P) = P[\emptyset/ac'] \vee (\exists\, y \bullet P[\{y\}/ac'] \wedge y \in ac')$

It confirms our intuition that P must hold when ac' is the empty set, and thus the resulting ac' is unconstrained by upward-closure, and when ac' is a singleton $\{y\}$, in which case y is in every ac'.

We consider the following example, where **A2** is applied to the angelic choice in Example 4.

Example 8.

$$
\begin{array}{ll}
\mathbf{A2}(x := 1 \sqcup x := 2) & \{\text{Definition of assignment}\} \\
= \mathbf{A2}((\mathit{true} \vdash s \oplus \{x \mapsto 1\} \in ac') \sqcup (\mathit{true} \vdash s \oplus \{x \mapsto 2\} \in ac')) & \{\text{Definition of } \sqcup\} \\
= \mathbf{A2}(\mathit{true} \vdash s \oplus \{x \mapsto 1\} \in ac' \wedge s \oplus \{x \mapsto 2\} \in ac') & \{\text{Lemma 17}\} \\
= (\neg\, \mathbf{A2}(\mathit{false}) \vdash \mathbf{A2}(s \oplus \{x \mapsto 1\} \in ac' \wedge s \oplus \{x \mapsto 2\} \in ac')) & \{\text{Definition of } \mathbf{A2}\} \\
= (\mathit{true} \vdash \mathit{false}) &
\end{array}
$$

The result is not a fixed point of **A2***, rather it is an angelic design with postcondition false. This design is the top of the lattice and captures the notion of miracle. Like in the theory of designs, this is an infeasible program that can never be started. Lemma 17 and all other results referenced in examples and proofs here are in the appendix.* □

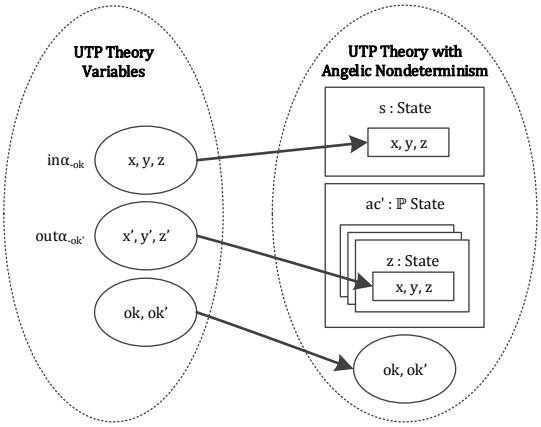


Figure 6: Encoding variables in a theory of angelic designs using $p2ac$

5.4. From Designs to Angelic Designs

To study the correspondence of results across the models depicted in Fig. 3, we consider the mapping from designs to angelic designs. The main concern is encoding both the pre and postcondition in terms of a single initial state s and a set of final states ac'. Since angelic designs are designs, ok and ok' retain the same meaning. The function $d2ac$, that maps from designs to angelic designs, is defined as follows.

Definition 21. $d2ac(P) \mathrel{\widehat{=}} (\neg\, p2ac(P^f) \land (\neg\, P^f[\mathbf{s}/in\alpha_{-ok}]\,;true) \vdash p2ac(P^t))$

The negation of the precondition P^f and the postcondition P^t are mapped using the function $p2ac$, defined below, while the second conjunct in the precondition of the angelic design requires that whenever the precondition $\neg\, P^f$ holds, then there is some final observation of the values of the variables in $out\alpha$. Essentially this allows the value of ac' to be unspecified when the precondition $\neg\, P^f$ is not satisfied.

The predicate $\neg\, P^f[\mathbf{s}/in\alpha_{-ok}]\,;true$ can be restated as $\exists\, out\alpha \bullet \neg\, P^f[\mathbf{s}/in\alpha_{-ok}]$, where the sets $in\alpha$ and $out\alpha$ include the program variables of the target theory of designs, and $in\alpha_{-ok}$ excludes the variable ok from the set. In a substitution $P[\mathbf{s}/S\alpha]$, 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 $S\alpha$, this substitution is defined as follows.

Definition 22. $P[\mathbf{z}/S\alpha] \mathrel{\widehat{=}} P[z.s_0, \ldots, z.s_n/s_0, \ldots, s_n]$

Each variable s_i in $S\alpha$ is replaced with $z.s_i$. As an example, we consider the following 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.

The main purpose of $p2ac$, which we define below, 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 Fig. 6.

Definition 23. $p2ac(P) \mathrel{\widehat{=}} \exists\, z \bullet P[\mathbf{s}, \mathbf{z}/in\alpha_{-ok}, out\alpha_{-ok'}] \land undash(z) \in ac'$

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 ac'. Since in our encoding states have undashed components, we use the function $undash$ to rename every record component in z, and thus require $undash(z)$ to be in ac'. The variables ok and ok' are unchanged.

Example 9. *We consider the example where we apply d2ac to the non-***H3** *design* $(y = 1 \land x' \neq 2 \vdash x' = 1)$*, equal to* $(ok \land y = 1) \Rightarrow (x' = 2 \lor (x' = 1 \land ok'))$*, which, provided the initial value of y is 1, either terminates with a value 1 for x, or gives x the value* 2*, irrespective of termination. We assume that* $in\alpha = \{ok, x, y\}$ *and* $out\alpha = \{ok', x', y'\}$*. (Non-***H3** *designs are of interest in a theory of CSP.)*

$$
\begin{array}{ll}
& d2ac(y = 1 \land x' \neq 2 \vdash x' = 1) \qquad \{\text{Definition of } d2ac\} \\
= & (\neg\, p2ac(\neg\,(y = 1 \land x' \neq 2)) \land ((y = 1 \land x' \neq 2)[\mathbf{s}/in\alpha_{-ok}]\,;true) \vdash p2ac(x' = 1)) \qquad \{\text{Substitution}\} \\
= & (\neg\, p2ac(\neg\,(y = 1 \land x' \neq 2)) \land ((s.y = 1 \land x' \neq 2)\,;true) \vdash p2ac(x' = 1)) \qquad \{\text{Sequential composition}\} \\
= & (\neg\, p2ac(\neg\,(y = 1 \land x' \neq 2)) \land s.y = 1 \vdash p2ac(x' = 1)) \qquad \{\text{Predicate calculus}\} \\
= & (\neg\, p2ac(y = 1 \land x' = 2) \land s.y = 1 \vdash p2ac(x' = 1)) \qquad \{\text{Definition of } p2ac\} \\
= & \left(\begin{array}{l} \neg\,(s.y = 1 \land \exists\, z \bullet z.x' = 2 \land undash(z) \in ac') \land s.y = 1 \\ \vdash \\ \exists\, z \bullet z.x' = 1 \land undash(z) \in ac' \end{array}\right) \qquad \{\text{Property of } undash \text{ and calculus}\} \\
= & ((\neg\, \exists\, z \bullet z.x = 2 \land z \in ac') \land s.y = 1 \vdash \exists\, z \bullet z.x = 1 \land z \in ac')
\end{array}
$$

The result is a design whose precondition states that there is no final state z in ac' *whose component x has value* 2*, and the value of component y in s is* 1*. Similarly, the postcondition states that there is a state z in the set of angelic choices* ac'*, where the value of x is* 1*. Because the precondition of* $(y = 1 \land x' \neq 2 \vdash x' = 1)$ *refers to* x'*, in the angelic design, we have a reference to* ac' *in the precondition. In the postcondition, the choices for the angel are those that guarantee termination, and so record a value of 1 for x.* □

5.5. From Angelic Designs to Designs

We now consider the mapping from angelic designs to designs. It is defined by the function $ac2p$, whose goal is to collapse the set of final states ac' into a single state, and, introduce the input and output variables as variables in their own right, rather than as components of records. Its definition is presented below.

Definition 24. $ac2p(P) \mathrel{\widehat{=}} \mathbf{PBMH}(P)[State_{\Pi}(in\alpha_{-ok})/s]\ ;_{\mathcal{A}} \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x$

We define $ac2p(P)$ in terms of the predicate obtained by applying **PBMH** to P; this ensures upward closure of ac'. In this way, instead of requiring P to be **PBMH**-healthy, we make $ac2p$ more widely applicable. This is useful in the definition of a link between angelic designs and reactive processes (see Section 6.3).

Above, in $\mathbf{PBMH}(P)$, we replace references to components of s to introduce the corresponding input variables of the set $in\alpha_{-ok}$, which excludes ok. For a set of variables $S\alpha$, $State_{\Pi}(S\alpha)$ is an identity record, whose components s_i are mapped to the respective variables s_i. As already mentioned, ok and ok' retain the same meaning. Finally, the resulting predicate is sequentially composed, using $;_{\mathcal{A}}$, with a predicate that introduces the output variables of the final state, except for ok'. Since in our encoding the components of a state are always undashed, we apply the function $dash$, the inverse of $undash$, to s.

Definition 25. $State_{\Pi}(S\alpha) \mathrel{\widehat{=}} \{s_0 \mapsto s_0, \ldots, s_n \mapsto s_n\}$

As an example, we consider the substitution $(s.x = 1 \land ok)[State_{\Pi_{-ok}}(in\alpha)/s]$, 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 Lemma 4.

Lemma 4. $ac2p(P) = \exists\, ac' \bullet P[State_{\Pi}(in\alpha_{-ok})/s] \land \forall\, z \bullet z \in ac' \Rightarrow \bigwedge x : out\alpha_{-ok'} \bullet dash(z).x = x$

The variable ac' is quantified away, and for each state z in ac', the output variables in $out\alpha$, except for ok', are introduced and set to their respective values in z. If there is more than one state in ac', then $ac2p$ yields *false* since x cannot take more than one value.

Example 10. *We consider below the application of ac2p to the angelic choice in Example 4.*

$$
\begin{aligned}
& ac2p(x := 1 \sqcup x := 2) && \{\text{Definition of assignment}\} \\
&= ac2p((true \vdash \{x \mapsto 1\} \in ac') \sqcup (true \vdash \{x \mapsto 2\} \in ac')) && \{\text{Conjunction of designs}\} \\
&= ac2p(true \vdash \{x \mapsto 1\} \in ac' \wedge \{x \mapsto 2\} \in ac') && \{\text{Lemma 24}\} \\
&= (\neg\, ac2p(\neg\, true) \vdash ac2p(\{x \mapsto 1\} \in ac' \wedge \{x \mapsto 2\} \in ac')) && \{\text{Definition of } ac2p\} \\
&= (\neg\, false \vdash x' = 1 \wedge x' = 2) && \{\text{Predicate calculus}\} \\
&= (true \vdash false)
\end{aligned}
$$

The angelic nondeterminism is collapsed and the result is the top of the lattice of designs, a miraculous design, since angelic choice cannot be represented in a design. □

5.6. Isomorphism and Galois Connection

Having defined a pair of linking functions between the theories of angelic designs and designs, we show that there is a Galois connection between the two theories. In addition, when we consider the subset of **A2**-healthy designs, we have an isomorphism.

The mapping of a design P through $d2ac$ and then $ac2p$ yields the same design P as established below.

Theorem 5. *Provided that P is a design, $ac2p \circ d2ac(P) = P$.*

In other words, in our theory of angelic designs we can model the original designs of Hoare and He [10] without angelic nondeterminism. This is a reassuring result regarding the suitability of our model.

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 6.

Theorem 6. *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, as illustrated in Example 10. This is because the angelic nondeterminism is removed. The results of Theorems 5 and 6 establish that we have a Galois connection between the two theories.

If we consider the subset of **A**-healthy designs that are in addition **A2**-healthy, then we can prove the reverse implication as established below.

Theorem 7. *Provided P is an* **A0**-**A2***-healthy design, $d2ac \circ ac2p(P) \sqsubseteq P$.*

These results establish that there is a bijection for the subset of **A2**-healthy designs.

Theorem 8. *Provided P is a design that is* **A0**-**A2***-healthy, $d2ac \circ ac2p(P) = P$*

This result confirms that these models are isomorphic as depicted in Figs. 2 and 3.

As previously discussed, CSP processes can be specified as the image of non-**H3** designs through the healthiness condition **R** of reactive processes. Angelic designs, which encompass non-**H3** designs, are a basis for the specification of CSP processes with angelic nondeterminism, which we pursue next.

6. Reactive Angelic Designs

We consider a theory where the observational variables of the theory of reactive processes are encoded as components of a *State*. We define its alphabet as follows.

Definition 26 (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 ok and ok', we require that every *State* has components named tr, $wait$ and ref. This enables the angelic choice over the final or intermediate observations of the observational variables tr, ref and $wait$ for reactive processes.

Next, in Section 6.1 we show how we can express in this encoding every healthiness condition of the original theory of reactive processes, followed by the healthiness conditions of CSP in Section 6.2. We then study the link between CSP and reactive angelic designs in Section 6.3. This is followed by a discussion of important operators and interesting algebraic properties in Section 6.4. Finally, in Section 6.5 we discuss the link with the subset of non-divergent reactive angelic designs.

6.1. Healthiness Conditions

The first property of interest that underpins the theory of reactive processes is that the history of events observed cannot be undone. We recall that this is established by the healthiness condition **R1**. In general, for any initial state x, the set of all final states that satisfy this property is given by $States_{tr \leq tr'}(x)$ below.

Definition 27. $States_{tr \leq tr'}(x) \mathrel{\widehat{=}} \{z : State(\{tr, ref, wait\}) \mid x.tr \leq z.tr\}$

This definition is used for characterising the first healthiness condition, **RA1**.

Definition 28. $\mathbf{RA1}(P) \mathrel{\widehat{=}} (P \wedge ac' \neq \emptyset)[States_{tr \leq tr'}(s) \cap ac'/ac']$

RA1 not only enforces trace extension for final states in ac', but also that there is some final state satisfying this property available for angelic choice. This is achieved by substituting ac' with the intersection of ac' and the set $State_{tr \leq tr'}(s)$ of final states whose trace is a suffix of $s.tr$ in $P \wedge ac' \neq \emptyset$. We require $ac' \neq \emptyset$ because, if the angel is to ensure trace extension, it cannot be given an empty set of choices. We note that, even if P is **PBMH** healthy, it does not ensure $ac' \neq \emptyset$, although it may not require $ac' = \emptyset$.

A consequence of the definition of **RA1** is that it also enforces **A0**.

Theorem 9. $\mathbf{RA1} \circ \mathbf{A0}(P) = \mathbf{RA1}(P)$

Although **A0** only requires ac' not to be empty in the postcondition of an angelic design, **RA1** requires this property under all circumstances, that is, even if the precondition is not satisfied.

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 29. $\mathbf{RA2}(P) \mathrel{\widehat{=}} P\left[s \oplus \{tr \mapsto \langle\rangle\}, \left\{ z \;\middle|\; \begin{array}{l} z \in ac' \wedge s.tr \leq z.tr \\ \bullet\ z \oplus \{tr \mapsto z.tr - s.tr\} \end{array} \right\} \middle/ s, ac' \right]$

It defines the component tr in the initial state s to be the empty sequence. For the set ac', it considers the states z in there whose traces $z.tr$ are an extension of $s.tr$. The substitution replaces ac' with the set of states obtained by changing the trace of each such z with the difference with respect to $s.tr$.

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 our new theory is **RA3**. Before exploring its definition, we introduce the identity $\mathbb{I}_{\mathcal{R}ac}$ of our theory, used to define **RA3**.

Definition 30. $\mathbb{I}_{\mathcal{R}ac} \mathrel{\widehat{=}} (\mathbf{RA1}(\neg\ ok) \vee (ok' \wedge s \in ac'))$

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'. Using $\mathbb{I}_{\mathcal{R}ac}$, we define **RA3** below.

Definition 31. $\mathbf{RA3}(P) \mathrel{\widehat{=}} \mathbb{I}_{\mathcal{R}ac} \lhd s.wait \rhd P$

This definition is similar to that of **R3**, except that we use $\mathbb{I}_{\mathcal{R}ac}$ as the identity, and $s.wait$ instead of $wait$ as a condition, since $wait$ is a component of the initial state s.

Our theory of reactive angelic designs is characterised by the functional composition of **RA1**, **RA2**, and **RA3**, besides **PBMH**. To provide a parallel with the original theory of reactive processes, we define part of this composition as **RA**. The order of the functional composition in the definition of **RA** is not important since **RA1**, **RA2** and **RA3** commute [35].

Definition 32. $\mathbf{RA}(P) \mathrel{\widehat{=}} \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{RA3}(P)$

PBMH and **RA1**, however, do not commute. We consider the following Example 11, where the healthiness conditions **R1A** and **PBMH** are applied to the relation $ac' = \emptyset$, which is not **PBMH**-healthy.

Example 11.

$$
\begin{array}{ll}
\mathbf{RA1} \circ \mathbf{PBMH}(ac' = \emptyset) & \{\text{Definition of } \mathbf{PBMH}\} \\
= \mathbf{RA1}(\exists\, ac_0 \bullet ac_0 = \emptyset \land ac_0 \subseteq ac') & \{\text{One-point rule and property of sets}\} \\
= \mathbf{RA1}(\mathit{true}) &
\end{array}
$$

$$
\begin{array}{ll}
\mathbf{PBMH} \circ \mathbf{RA1}(ac' = \emptyset) & \{\text{Definition of } \mathbf{RA1}\} \\
= \mathbf{PBMH}((ac' = \emptyset \land ac' \neq \emptyset)[\mathit{States}_{tr \leq tr'}(s) \cap ac'/ac']) & \{\text{Predicate calculus}\} \\
= \mathbf{PBMH}(\mathit{false}) & \{\text{Definition of } \mathbf{PBMH}\} \\
= \mathit{false} &
\end{array}
$$

In the first case, the application of **PBMH** *yields true. The result of the functional composition is then* $\mathbf{RA1}(\mathit{true})$*. 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.* □

In the definition of the healthiness condition of the theory of reactive angelic designs, presented in the next section, **PBMH** is applied before **RA**. Since all conditions are monotonic and idempotent [35], our theory is a complete lattice [10, Sec. 4.1, Ch. 4].

6.2. Reactive Angelic Designs

As mentioned in Section 3, in the original theory of CSP, another two healthiness conditions, **CSP1** and **CSP2**, are required, in addition to **R**. To consider a theory of CSP with angelic nondeterminism we define a counterpart to these functions below. The first healthiness condition of interest is **CSPA1**, which is the counterpart to **CSP1** in the new theory. Its definition is presented below.

Definition 33. $\mathbf{CSPA1}(P) \mathrel{\widehat{=}} P \lor \mathbf{RA1}(\neg\, ok)$

A CSP process with angelic nondeterminism P is required to observe **RA1** when in an unstable state. For an **RA**-healthy process, this property is already enforced by **RA1** under all circumstances. The following theorem establishes that this behaviour can also be described as the composition of **RA1** after **H1**.

Theorem 10. $\mathbf{RA1} \circ \mathbf{CSPA1}(P) = \mathbf{RA1} \circ \mathbf{H1}(P)$

This is because **CSPA1** requires that in an unstable state there are no meaningful observations other than those where the trace is extended, a requirement equally enforced by **H1** when composed with **RA1**.

The counterpart to **CSP2** is defined as **H2** with the extended alphabet that includes s and ac'.

Definition 34. $\mathbf{CSPA2}(P) \mathrel{\widehat{=}} \mathbf{H2}(P)$

That is, $\mathbf{CSPA2}(P) = P\,;((ok \Rightarrow ok') \land s' = s \land ac' = ac)$.

Finally, as previously indicated, our conservative extension of the theory of CSP processes is defined by **RAD**, the composition of all the healthiness conditions of interest.

Definition 35. $\mathbf{RAD}(P) \mathrel{\widehat{=}} \mathbf{RA} \circ \mathbf{CSPA1} \circ \mathbf{CSPA2} \circ \mathbf{PBMH}(P)$

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 the image of an angelic design through the function **RA** as established by the following theorem, where $P^o_w = P[o, s \oplus \{\mathit{wait} \mapsto w\}/ok', s]$.

Theorem 11. $\mathbf{RAD}(P) = \mathbf{RA} \circ \mathbf{A}(\neg\ P_f^f \vdash P_f^t)$

PROOF.

$$
\begin{array}{ll}
\mathbf{RAD}(P) & \{\text{Definition of } \mathbf{RAD}\} \\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{CSPA1} \circ \mathbf{CSPA2} \circ \mathbf{PBMH}(P) & \{\text{Theorem 10}\} \\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{CSPA2} \circ \mathbf{PBMH}(P) & \{\mathbf{CSPA2} \text{ is } \mathbf{H2}\} \\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{H2} \circ \mathbf{PBMH}(P) & \{\text{Theorem 9}\} \\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{A0} \circ \mathbf{H1} \circ \mathbf{H2} \circ \mathbf{PBMH}(P) & \{\text{Theorems 64 and 65}\} \\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{A0} \circ \mathbf{PBMH} \circ \mathbf{H1} \circ \mathbf{H2}(P) & \{\text{Definition of design}\} \\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{A0} \circ \mathbf{PBMH}(\neg\ P^f \vdash P^t) & \{\text{Definition of } \mathbf{A}\} \\
= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{A}(\neg\ P^f \vdash P^t) & \{\text{Theorems 68, 69 and 71}\} \\
= \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{RA3} \circ \mathbf{A}(\neg\ P^f \vdash P^t) & \{\text{Lemmas 18 and 19}\} \\
= \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{RA3} \circ \mathbf{A}((\neg\ P^f \vdash P^t)_f) & \{\text{Substitution}\} \\
= \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{RA3} \circ \mathbf{A}(\neg\ P_f^f \vdash P_f^t) & \{\text{Definition of } \mathbf{RA}\} \\
= \mathbf{RA} \circ \mathbf{A}(\neg\ P_f^f \vdash P_f^t) & \Box
\end{array}
$$

This is a result similar to that for CSP processes as the image of designs through **R**. Since both **RA** and **A** are monotonic and idempotent, and the theory of designs is a complete lattice [10], so is the theory of reactive angelic designs just presented.

6.3. Relationship with CSP

The theory of reactive angelic designs can be related to the original theory of CSP through the pair of linking functions $ac2p$ and $p2ac$ previously introduced in Section 5. We employ them by considering the set of variables $in\alpha$ to be $\{tr, ref, wait, ok\}$, and a corresponding set of variables $out\alpha$ with dashed counterparts.

The relationship between the models has previously been illustrated in Fig. 2. Here we focus our attention on the relationship with CSP. In Fig. 7a 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 discussed in Section 5 that characterises predicates with no angelic nondeterminism.

Fig. 7b illustrates the relationship between the predicates of each theory. For a reactive angelic design P, the composition $p2ac \circ ac2p(P)$ yields a stronger predicate, since any angelic nondeterminism in P is collapsed into a single final state. On the other hand, for a predicate Q of the CSP theory, $ac2p \circ p2ac(Q)$ yields exactly Q. Thus a Galois connection exists between the theories.

6.3.1. From Reactive Angelic Designs to CSP

Application of the function $ac2p$ 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.

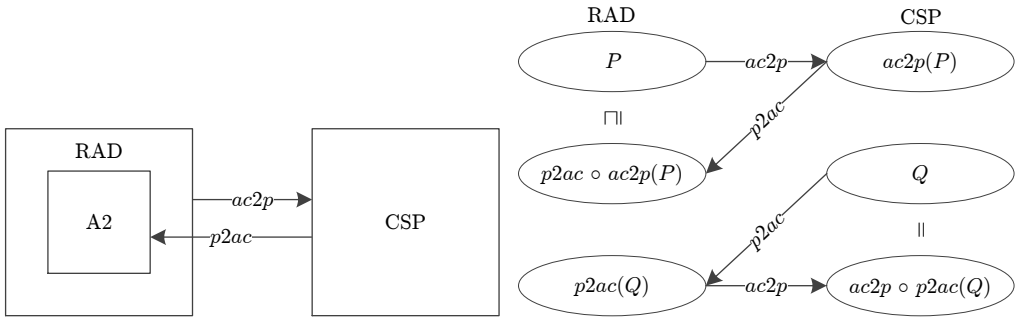
Theorem 12. *Provided P is* **PBMH***-healthy, $ac2p \circ \mathbf{RA}(P) = \mathbf{R} \circ ac2p(P)$.*

If P is a reactive angelic design, then the application of $ac2p$ yields a reactive design.

Theorem 13. $ac2p \circ \mathbf{RA} \circ \mathbf{A}(\neg\ P_f^f \vdash P_f^t) = \mathbf{R}(\neg\ ac2p(P_f^f) \vdash ac2p(P_f^t))$

PROOF.

$$
\begin{array}{ll}
ac2p \circ \mathbf{RA} \circ \mathbf{A}(\neg\ P_f^f \vdash P_f^t) & \{\text{Theorem 67}\} \\
= ac2p \circ \mathbf{RA} \circ \mathbf{PBMH}(\neg\ P_f^f \vdash P_f^t) & \{\text{Theorem 12}\} \\
= \mathbf{R} \circ ac2p \circ \mathbf{PBMH}(\neg\ P_f^f \vdash P_f^t) & \{\text{Lemma 25}\} \\
= \mathbf{R} \circ ac2p(\neg\ P_f^f \vdash P_f^t) & \{\text{Lemma 24}\} \\
= \mathbf{R}(\neg\ ac2p(P_f^f) \vdash ac2p(P_f^t)) & \Box
\end{array}
$$



(a) Theories and links (b) Predicates and links

Figure 7: Relationship between theories

The function $ac2p$ can be applied directly to the pre and postconditions of the reactive angelic design. This is a pleasing result that supports the reuse of results across the theories by defining an approach for calculating corresponding CSP operators from reactive angelic designs.

In the next example, $ac2p$ is applied to $a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}$, the angelic choice between a prefixing on a followed by deadlock, and on b followed by deadlock. The operators of the theory of reactive angelic designs have subscript $_{\mathbf{RAD}}$ to distinguish them from those of the original theory of CSP, which have subscript $_{\mathbf{R}}$. To describe this process as a reactive angelic design, we use the following operator.

Definition 36. $\circledin^{y}_{ac'}(P) \mathrel{\widehat{=}} \exists\, y \bullet y \in ac' \wedge P[\{y\}/ac']$

The predicate $\circledin^{y}_{ac'}(P)$ requires that P admits a state y as a single option for angelic choice, effectively defining y as the only (angelic) choice available. In general, this predicate allows the definition of CSP operators to be lifted into the theory of reactive angelic designs as further illustrated in the next section.

Example 12. *By the definitions of prefixing, $Stop_{\mathbf{RAD}}$ and $\sqcup_{\mathbf{RAD}}$, we can describe the reactive angelic design $a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}$ as shown below.*

$$\mathbf{RA} \circ \mathbf{A} \left(\begin{array}{l} true \vdash \left(\begin{array}{l} \circledin^{y}_{ac'} \left(y.wait \wedge \left(\begin{array}{l} (y.tr = s.tr \wedge a \notin y.ref) \vee y.tr = s.tr \frown \langle a \rangle \end{array} \right) \right) \\ \wedge \\ \circledin^{y}_{ac'} \left(y.wait \wedge \left(\begin{array}{l} (y.tr = s.tr \wedge b \notin y.ref) \vee y.tr = s.tr \frown \langle b \rangle \end{array} \right) \right) \end{array} \right) \end{array} \right)$$

This process never diverges, does not terminate ($y.wait$), and either the trace has not changed ($y.tr = s.tr$) and we have an angelic choice of not refusing a ($a \notin y.ref$) or b ($b \notin y.ref$), or the trace is extended with a or b. With this, we can calculate the result of applying $ac2p$ as follows.

$ac2p(a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}})$ {Theorem 13}

$$= \mathbf{R} \left(\begin{array}{l} \neg\, ac2p(false) \\ \vdash \\ ac2p \left(\begin{array}{l} \circledin^{y}_{ac'} \left(y.wait \wedge \left(\begin{array}{l} (y.tr = s.tr \wedge a \notin y.ref) \vee y.tr = s.tr \frown \langle a \rangle \end{array} \right) \right) \\ \wedge \\ \circledin^{y}_{ac'} \left(y.wait \wedge \left(\begin{array}{l} (y.tr = s.tr \wedge b \notin y.ref) \vee y.tr = s.tr \frown \langle b \rangle \end{array} \right) \right) \end{array} \right) \end{array} \right)$$

{Definition of $\circledin^{y}_{ac'}$ and $ac2p$}

$$= \mathbf{R} \left(\begin{array}{l} true \\ \vdash \\ \left(\begin{array}{l} wait' \wedge ((tr' = tr \wedge a \notin ref') \vee (tr' = tr \frown \langle a \rangle)) \wedge \\ wait' \wedge ((tr' = tr \wedge b \notin ref') \vee (tr' = tr \frown \langle b \rangle)) \end{array} \right) \end{array} \right)$$

{Predicate calculus}

$= \mathbf{R}(true \vdash wait' \wedge tr' = tr \wedge a \notin ref' \wedge b \notin ref')$ {Lemma 5}

$= a \rightarrow_{\mathbf{R}} Stop_{\mathbf{R}} \sqcup_{\mathbf{R}} b \rightarrow_{\mathbf{R}} Stop_{\mathbf{R}}$

The result is the least upper bound ($\sqcup_{\mathbf{R}}$) of the corresponding CSP processes, defined as 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. The conjunction of the CSP processes in the example is established by the following lemma.

Lemma 5. $a \rightarrow_{\mathbf{R}} Stop_{\mathbf{R}} \sqcup_{\mathbf{R}} b \rightarrow_{\mathbf{R}} Stop_{\mathbf{R}} = \mathbf{R}(true \vdash tr' = tr \wedge a \notin ref' \wedge b \notin ref' \wedge wait')$

In this case, both processes can only agree on the trace of events remaining unchanged, and not refusing events a and b, while waiting. This is effectively $Stop_{\mathbf{R}}$ but with a stronger postcondition. □

6.3.2. From CSP to Reactive Angelic Designs

The mapping in the opposite direction, from CSP processes to reactive angelic designs, is $p2ac$. The application of $p2ac$ to a process P that is **R**-healthy can be described by the application of the functional composition of **RA** with $p2ac$ to the original process P.

Theorem 14. $p2ac \circ \mathbf{R}(P) = \mathbf{RA} \circ p2ac(P)$

The result of applying $p2ac$ to a reactive design is established below: $p2ac$ can be directly applied to the pre and postconditions separately, followed by **A** and **RA**.

Theorem 15. $p2ac \circ \mathbf{R}(\neg\, P_f^f \vdash P_f^t) = \mathbf{RA} \circ \mathbf{A}(\neg\, p2ac(P_f^f) \vdash p2ac(P_f^t))$

Proof.

$$
\begin{aligned}
& p2ac \circ \mathbf{R}(\neg\, P_f^f \vdash P_f^t) && \{\text{Theorem 14 and definition of } \mathbf{RA}\} \\
&= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ p2ac(\neg\, P_f^f \vdash P_f^t) && \{\text{Definition of } \mathbf{RA1} \text{ and predicate calculus}\} \\
&= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1}(p2ac(\neg\, P_f^f \vdash P_f^t) \wedge ac' \neq \emptyset) && \{\text{Theorem 72}\} \\
&= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1}((\neg\, p2ac(P_f^f) \vdash p2ac(P_f^t)) \wedge ac' \neq \emptyset) && \{\mathbf{RA1} \text{ and } \mathbf{RA}\} \\
&= \mathbf{RA}(\neg\, p2ac(P_f^f) \vdash p2ac(P_f^t)) && \{\text{Lemma 26}\} \\
&= \mathbf{RA}(\neg\, \mathbf{PBMH} \circ p2ac(P_f^f) \vdash \mathbf{PBMH} \circ p2ac(P_f^t)) && \{\text{Definition of } \mathbf{A1}\} \\
&= \mathbf{RA} \circ \mathbf{A1}(\neg\, p2ac(P_f^f) \vdash p2ac(P^t)) && \{\text{Definition of } \mathbf{RA} \text{ and Theorem 9}\} \\
&= \mathbf{RA} \circ \mathbf{A0} \circ \mathbf{A1}(\neg\, p2ac(P_f^f) \vdash p2ac(P^t)) && \{\text{Definition of } \mathbf{A}\} \\
&= \mathbf{RA} \circ \mathbf{A}(\neg\, p2ac(P_f^f) \vdash p2ac(P^t)) &&
\end{aligned}
$$

□

Example 13. *As an example, we consider the CSP terminating process $Skip_{\mathbf{R}}$, defined by the following reactive design $\mathbf{R}(true \vdash tr' = tr \wedge \neg\, wait')$. It specifies that $Skip_{\mathbf{R}}$ does not diverge, since its precondition is true, and then terminates ($\neg\, wait'$) without engaging in any event (that is, $tr' = tr$). Below, it is mapped through $p2ac$ to a reactive angelic design.*

$$p2ac(Skip_{\mathbf{R}}) = \mathbf{RA} \circ \mathbf{A}(true \vdash \exists\, y \bullet \neg\, y.wait \wedge y.tr = s.tr \wedge y \in ac')$$

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. □

As already said, the pair of linking functions we have considered form a Galois connection. When considering the mapping from the original theory of reactive processes, followed by the mapping in the opposite direction, we obtain an exact correspondence.

Theorem 16. $ac2p \circ p2ac(P) = P$

Proof.

$$
\begin{array}{ll}
ac2p \circ p2ac(P) & \{\text{Definition of } ac2p\} \\
= (\mathbf{PBMH} \circ p2ac(P))[State_{\Pi}(in\alpha_{-ok})/s] \mathbin{;_{\mathcal{A}}} \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x & \{\text{Lemma 26}\} \\
= p2ac(P)[State_{\Pi}(in\alpha_{-ok})/s] \mathbin{;_{\mathcal{A}}} \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x & \{\text{Definition of } p2ac\} \\
= \left(\begin{array}{l} (\exists\, z \bullet P[\mathbf{s}, \mathbf{z}/in\alpha_{-ok}, out\alpha_{-ok'}] \wedge undash(z) \in ac')[State_{\Pi}(in\alpha_{-ok})/s] \mathbin{;_{\mathcal{A}}} \\ \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x \end{array} \right) & \{\text{Substitution}\} \\
= \left(\begin{array}{l} (\exists\, z \bullet P[\mathbf{s}, \mathbf{z}/in\alpha_{-ok}, out\alpha_{-ok'}][State_{\Pi}(in\alpha_{-ok})/s] \wedge undash(z) \in ac') \mathbin{;_{\mathcal{A}}} \\ \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x \end{array} \right) & \{\text{Lemma 27}\} \\
= (\exists\, z \bullet P[\mathbf{z}/out\alpha_{-ok'}] \wedge undash(z) \in ac') \mathbin{;_{\mathcal{A}}} \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x & \{\text{Def. of } ;_{\mathcal{A}} \text{ substitution}\} \\
= \exists\, z \bullet P[\mathbf{z}/out\alpha_{-ok'}] \wedge undash(z) \in \{s \mid \bigwedge x : out\alpha_{-ok'} \bullet dash(s).x = x\} & \{\text{Property of sets}\} \\
= \exists\, z \bullet P[\mathbf{z}/out\alpha_{-ok'}] \wedge \bigwedge x : out\alpha_{-ok'} \bullet dash(undash(z)).x = x & \{\text{Property of } dash \text{ and } undash\} \\
= \exists\, z \bullet P[\mathbf{z}/out\alpha_{-ok'}] \wedge \bigwedge x : out\alpha_{-ok'} \bullet z.x = x & \{\text{Lemma 28}\} \\
= P[\mathbf{z}/out\alpha_{-ok'}][State_{\Pi}(out\alpha_{-ok'})/z] & \{\text{Lemma 27}\} \\
= P & \square
\end{array}
$$

This reassuring result establishes that our theory can accommodate the existing CSP processes, that is, those without angelic nondeterminism, appropriately. Their semantics is preserved.

When considering the mapping in the opposite direction we obtain the result below.

Lemma 6. $p2ac \circ ac2p(P) = \exists\, ac_0, y \bullet P[ac_0/ac'] \wedge ac_0 \subseteq \{y\} \wedge y \in ac'$

If the set of final states ac_0 in P contains 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 the element of ac_0, or empty, in which case ac' is arbitrary. In other words, the functional composition only preserves predicates whose set of angelic choices is either empty or a singleton, otherwise the result is *false*.

In the following example, Lemma 6 is applied to the process in Example 12.

Example 14.

$$
\begin{array}{l}
p2ac \circ ac2p(a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}) \\
= \mathbf{RA} \circ \mathbf{A}\left(true \vdash \circledin{}^{y}_{ac'}(y.wait \wedge y.tr = s.tr \wedge a \notin y.ref \wedge b \notin y.ref)\right)
\end{array}
$$

This process results from the application of $p2ac$ to the process in Example 12. In this case, the process is always waiting, keeps the trace of events unchanged, but does not refuse either a or b. This is different from the reactive angelic design $a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}$ defined in Example 12. $\square$

If we consider the result of Lemma 6 in the context of the predicates of our theory, that is, those which are **PBMH**-healthy, then we obtain an inequality as established below.

Theorem 17. *Provided P is* **PBMH***-healthy, $p2ac \circ ac2p(P) \sqsupseteq P$.*

Proof.

$$
\begin{array}{ll}
p2ac \circ ac2p(P) & \{\text{Lemma 6}\} \\
= \exists\, ac_0, y \bullet P[ac_0/ac'] \wedge ac_0 \subseteq \{y\} \wedge y \in ac' & \{\text{Property of sets}\} \\
= \exists\, ac_0, y \bullet P[ac_0/ac'] \wedge ac_0 \subseteq \{y\} \wedge \{y\} \subseteq ac' & \{\text{Predicate calculus}\} \\
\Rightarrow \exists\, ac_0 \bullet P[ac_0/ac'] \wedge ac_0 \subseteq ac' & \{\text{Definition of } \mathbf{PBMH}\} \\
= \mathbf{PBMH}(P) & \{\text{Assumption: } P \text{ is } \mathbf{PBMH}\text{-healthy}\} \\
= P & \square
\end{array}
$$

This theorem, together with Theorem 16, and the closure results of Theorems 13 and 15, establishes the existence of a Galois connection between the theory of CSP and reactive angelic designs. We observe, that in general, this result also establishes a Galois connection between relations and **PBMH**-healthy relations.

The result of Theorem 17 can be strengthened into an equality by considering the subset of reactive angelic designs that are **A2**-healthy. In other words, for reactive angelic designs, **A2** characterises the same fixed points as $p2ac \circ ac2p(P)$. For example, if we consider the application of **A2** to the following 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 14.

(We observe, however, that, in general, **A2** permits $ac' = \emptyset$, whereas in the theory of reactive angelic designs, both **RA1** and the mapping $p2ac$ require ac' not to be empty. For example, in the theory of angelic designs, the bottom of the lattice, which is *true*, is a fixed point of **A2**. On the other hand, the bottom of the theory of reactive angelic designs is $\mathbf{RA} \circ \mathbf{A}(false \vdash true)$, which, on account of **RA1** ensures $ac' \neq \emptyset$.)

Theorem 18 establishes that the result of $p2ac \circ ac2p(P)$, for a reactive angelic design P that is **A2**-healthy, is exactly the same reactive angelic design P.

Theorem 18. *Provided P_f^f and P_f^t are* **A2***-healthy,*

$$p2ac \circ ac2p \circ \mathbf{RA} \circ \mathbf{A}(\neg\, P_f^f \vdash P_f^t) = \mathbf{RA} \circ \mathbf{A}(\neg\, P_f^f \vdash P_f^t)$$

In summary, when we consider the theory of **A2**-healthy reactive angelic designs, 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.

6.4. Operators

Having introduced the alphabet and the healthiness conditions of the theory of reactive angelic designs, in this section we introduce the definition of important operators of CSP in the new model. We illustrate how operators relate to their original CSP counterparts.

6.4.1. Angelic Choice

The first operator of interest is angelic choice. Like in the theory of angelic designs, it is defined as the least upper bound operator, which is conjunction.

Definition 37. $P \sqcup_{\mathbf{RAD}} Q \mathrel{\widehat{=}} P \wedge Q$

For reactive angelic designs P and Q, angelic choice can be described in terms of a pre and a postcondition as established by the following theorem. 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.

Theorem 19. *Provided P and Q are* **RAD***-healthy,*

$$P \sqcup_{\mathbf{RAD}} Q = \mathbf{RA} \circ \mathbf{A}(\neg\, P_f^f \vee \neg\, Q_f^f \vdash (\neg\, P_f^f \Rightarrow P_f^t) \wedge (\neg\, Q_f^f \Rightarrow Q_f^t))$$

If either the precondition of P or Q holds, then the corresponding postcondition is established.

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 established by Theorem 20.

Theorem 20. $ac2p(p2ac(P) \sqcup_{\mathbf{RAD}} p2ac(Q)) = P \sqcup_{\mathbf{R}} Q$

This is expected, since we can express every standard CSP process in the new theory of reactive angelic designs. The result in the opposite direction, however, is an inequality.

Theorem 21. *Provided that P and Q are* **RAD***-healthy, $p2ac(ac2p(P) \sqcup_{\mathbf{R}} ac2p(Q)) \sqsupseteq P \sqcup_{\mathbf{RAD}} Q$.*

This is also expected, as $ac2p$ collapses angelic nondeterminism, and $p2ac$ cannot undo such effect.

6.4.2. Demonic Choice

Similarly to internal choice in CSP, in our theory, this operator is defined using the greatest lower bound.

Definition 38. $P \sqcap_{\mathbf{RAD}} Q \mathrel{\widehat{=}} P \vee Q$

For any P and Q, their demonic choice can be described as a reactive angelic design as stated below.

Theorem 22. *Provided P and Q are* **RAD***-healthy,* $P \sqcap_{\mathbf{RAD}} Q = \mathbf{RA} \circ \mathbf{A}(\neg P_f^f \wedge \neg Q_f^f \vdash P_f^t \vee Q_f^t)$.

In words, the resulting precondition is the conjunction of the respective preconditions of P and Q, while the postcondition is the disjunction of the postconditions. Intuitively, in a demonic choice, divergence is avoided if both preconditions are satisfied, while either the postcondition of P or Q may be observed.

The greatest lower bound of the theory of reactive angelic designs can be related with that of CSP through the pair of linking functions $p2ac$ and $ac2p$. Since $p2ac$ distributes through disjunction we can establish the following general result in Theorem 23.

Theorem 23. $p2ac(ac2p(P) \sqcap_{\mathbf{R}} ac2p(Q)) = p2ac \circ ac2p(P) \sqcap_{\mathbf{RAD}} p2ac \circ ac2p(Q)$

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 18) we obtain the result shown in Lemma 7.

Lemma 7. *Provided P and Q are* **RAD** *and* **A2***-healthy,* $p2ac(ac2p(P) \sqcap_{\mathbf{R}} ac2p(Q)) = P \sqcap_{\mathbf{RAD}} Q$.

In words, for reactive angelic designs with no angelic nondeterminism, the demonic choice of our theory and that of the CSP theory are in correspondence. Since $ac2p$ also distributes through disjunction, we can establish the following result. This means that the greatest lower bounds are in correspondence.

Theorem 24. $ac2p(p2ac(P) \sqcap_{\mathbf{RAD}} p2ac(Q)) = P \sqcap_{\mathbf{R}} Q$

Finally, since the least upper bound is conjunction, and the greatest lower bound is disjunction, angelic and demonic choice distribute over each other, as expected.

6.4.3. Chaos

The next operator of interest is $Chaos_{\mathbf{RAD}}$, which is the bottom of the lattice of reactive angelic designs.

Definition 39. $Chaos_{\mathbf{RAD}} \mathrel{\widehat{=}} \mathbf{RA} \circ \mathbf{A}(\mathit{false} \vdash ac' \neq \emptyset)$

Its precondition is *false* while the postcondition requires that ac' is not empty. As indicated previously, the postcondition can alternatively be specified as *true*, since both **A** and **RA1** ensure that the design is **A0**-healthy. Like for every reactive angelic design, the complete behaviour is constrained by **RA** and thus the final states in ac' observe the properties it enforces.

Like in the original theory, if a process in a demonic choice may diverge immediately, then this is the only choice ($Chaos_{\mathbf{RAD}} \sqcap_{\mathbf{RAD}} P = Chaos_{\mathbf{RAD}}$). The dual of this property is the unit law for angelic choice.

Theorem 25. *Provided P is* **RAD***-healthy,* $Chaos_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P = P$.

When there is an angelic choice between diverging immediately or behaving as P, then the choice is resolved in favour of P. This is a fundamental property of an angelic choice: if possible, it avoids divergence.

6.4.4. Choice

The next operator we introduce in this section is $Choice_{\mathbf{RAD}}$, the most nondeterministic process that does not diverge. (This is the counterpart to *Chaos* in Roscoe [2]'s presentation of CSP)

Definition 40. $Choice_{\mathbf{RAD}} \mathrel{\widehat{=}} \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash ac' \neq \emptyset)$

The precondition is *true* while the postcondition allows any non-empty set ac'.

As is discussed in Section 6.5, $Choice_{\mathbf{RAD}}$ plays an important role in the characterisation of the non-divergent reactive angelic designs. This uses the general result below regarding the least upper bound between an arbitrary process P and $Choice_{\mathbf{RAD}}$.

Theorem 26. *Provided P is* **RAD***-healthy, $Choice_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P = \mathbf{RA} \circ \mathbf{A}(true \vdash P_f^t)$.*

Proof.

$$
\begin{array}{ll}
Choice_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P & \{\text{Definition of } Choice_{\mathbf{RAD}}\} \\
= \mathbf{RA} \circ \mathbf{A}(true \vdash ac' \neq \emptyset) \sqcup_{\mathbf{RAD}} P & \{\text{Definition of } \mathbf{A}\text{, Lemma 15 and Theorem 3}\} \\
= \mathbf{RA} \circ \mathbf{A}(true \vdash true) \sqcup_{\mathbf{RAD}} P & \{\text{Assumption: } P \text{ is } \mathbf{RAD}\text{-healthy}\} \\
= \mathbf{RA} \circ \mathbf{A}(true \vdash true) \sqcup_{\mathbf{RAD}} \mathbf{RA} \circ \mathbf{A}(\neg\, P_f^f \vdash P_f^t) & \{\text{Theorem 19}\} \\
= \mathbf{RA} \circ \mathbf{A}(true \vdash true \wedge (\neg\, P_f^f \Rightarrow P_f^t)) & \{\text{Definition of design and predicate calculus}\} \\
= \mathbf{RA} \circ \mathbf{A}(true \vdash (ok \wedge \neg\, P_f^f) \Rightarrow P_f^t) & \{\text{Theorem 67}\} \\
= \mathbf{RA} \circ \mathbf{PBMH}(true \vdash (ok \wedge \neg\, P_f^f) \Rightarrow P_f^t) & \{\text{Lemma 16}\} \\
= \mathbf{RA}(\neg\, \mathbf{PBMH}(false) \vdash \mathbf{PBMH}((ok \wedge \neg\, P_f^f) \Rightarrow P_f^t)) & \{\text{Lemma 22}\} \\
= \mathbf{RA}(\neg\, \mathbf{PBMH}(false) \vdash \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}((ok \wedge \neg\, P_f^f) \Rightarrow P_f^t)) & \{\text{Theorems 66 and 70}\} \\
= \mathbf{RA}(\neg\, \mathbf{PBMH}(false) \vdash \mathbf{PBMH} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}((ok \wedge \neg\, P_f^f) \Rightarrow P_f^t)) & \{\text{Lemma 21}\} \\
= \mathbf{RA}(\neg\, \mathbf{PBMH}(false) \vdash \mathbf{PBMH}((\mathbf{RA} \circ \mathbf{A}(\neg\, P_f^f \vdash P_f^t))_f^t)) & \{\text{Assumption: } P \text{ is } \mathbf{RAD}\text{-healthy}\} \\
= \mathbf{RA}(\neg\, \mathbf{PBMH}(false) \vdash \mathbf{PBMH}(P_f^t)) & \{\text{Lemma 16}\} \\
= \mathbf{RA} \circ \mathbf{PBMH}(true \vdash P_f^t) & \{\text{Theorem 67}\} \\
= \mathbf{RA} \circ \mathbf{A}(true \vdash P_f^t) & \square
\end{array}
$$

This result establishes that $Choice_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P$ can be characterised as a reactive angelic design in which the precondition is *true*, while the postcondition P_f^t 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, the greatest lower bound ($\sqcap_{\mathbf{RAD}}$) between P and $Choice_{\mathbf{RAD}}$ is as follows.

Theorem 27. *Provided P is* **RAD***-healthy, $Choice_{\mathbf{RAD}} \sqcap_{\mathbf{RAD}} P = \mathbf{RA} \circ \mathbf{A}(\neg\, P_f^f \vdash ac' \neq \emptyset)$.*

The precondition of P is maintained, while the postcondition requires ac' not to be empty. In other words, if P had the possibility to diverge, this is still the case, but if the process does not diverge, then it behaves nondeterministically like $Choice_{\mathbf{RAD}}$, but also does not diverge.

6.4.5. Stop

Like in CSP, the notion of deadlock is captured by $Stop_{\mathbf{RAD}}$.

Definition 41. $Stop_{\mathbf{RAD}} \mathrel{\widehat{=}} \mathbf{RA} \circ \mathbf{A}(true \vdash \circledin{}^{y}_{ac'}(y.tr = s.tr \wedge y.wait))$

The precondition is *true* while the postcondition requires the process to be always waiting and keep the trace unchanged. Use of the $\circledin{}^{y}_{ac'}$ operator gives definitions similar to those of the original theory.

An angelic choice between a process P and $Stop_{\mathbf{RAD}}$ is, in general, not resolved in favour of either.

Theorem 28. *Provided P is* **RAD***-healthy,*

$$Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P = \mathbf{RA} \circ \mathbf{A}(true \vdash (\neg\, P_f^f \Rightarrow P_f^t) \wedge \circledin{}^{y}_{ac'}(y.tr = s.tr \wedge y.wait))$$

Any divergence in P_f^f is avoided, and so the precondition is *true*. If P diverges, then the process behaves as $Stop_{\mathbf{RAD}}$, otherwise there is an angelic choice between P and $Stop_{\mathbf{RAD}}$ as characterised by the conjunction of their postconditions. This process does not force the deadlock, but insists that it is possible.

6.4.6. Skip

The process that always terminates successfully is defined as $Skip_{\mathbf{RAD}}$.

Definition 42. $Skip_{\mathbf{RAD}} \mathrel{\widehat{=}} \mathbf{RA} \circ \mathbf{A}(true \vdash \circledin{}^{y}_{ac'}(\neg\, y.wait \wedge y.tr = s.tr))$

Its precondition is *true* while the postcondition requires that there is a final state y in ac' that records termination ($\neg\, y.wait$) and no change in the trace of events $s.tr$. 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.

Theorem 29. *Provided P is* **RAD***-healthy,*

$$Skip_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P = \mathbf{RA} \circ \mathbf{A}(true \vdash \circledin{}^{y}_{ac'}(\neg\, y.wait \wedge y.tr = s.tr)) \wedge (\neg\, P^{f}_{f} \Rightarrow P^{t}_{t}))$$

Again, divergence in P is avoided. If P diverges, because its precondition $\neg\, P^{f}_{f}$ does not hold, the angelic choice avoids that divergence behaving as $Skip_{\mathbf{RAD}}$. Otherwise, the behaviour is given by the conjunction of the postconditions of P, that is, P^{t}_{t}, and $Skip_{\mathbf{RAD}}$.

Example 15. *We consider an angelic choice between termination and deadlock.*

$$\begin{array}{ll}
& Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} Skip_{\mathbf{RAD}} & \{\text{Definition of } Stop_{\mathbf{RAD}} \text{ and } Skip_{\mathbf{RAD}}\} \\
= & \left(\begin{array}{l} \mathbf{RA} \circ \mathbf{A}(true \vdash \circledin{}^{y}_{ac'}(y.tr = s.tr \wedge y.wait)) \\ \sqcup_{\mathbf{RAD}} \\ \mathbf{RA} \circ \mathbf{A}(true \vdash \circledin{}^{y}_{ac'}(\neg\, y.wait \wedge y.tr = s.tr)) \end{array}\right) & \{\text{Theorem 19}\} \\
= & \mathbf{RA} \circ \mathbf{A}\left(\begin{array}{l} true \vee true \\ \vdash \\ \left(\begin{array}{l} (true \Rightarrow \circledin{}^{y}_{ac'}(y.tr = s.tr \wedge y.wait)) \\ \wedge \\ (true \Rightarrow \circledin{}^{y}_{ac'}(\neg\, y.wait \wedge y.tr = s.tr)) \end{array}\right) \end{array}\right) & \{\text{Predicate calculus}\} \\
= & \mathbf{RA} \circ \mathbf{A}(true \vdash \circledin{}^{y}_{ac'}(y.tr = s.tr \wedge y.wait) \wedge \circledin{}^{y}_{ac'}(\neg\, y.wait \wedge y.tr = s.tr)) &
\end{array}$$

In this case, the choice is not resolved by either process. □

If we map Example 15 into the original theory of CSP, then we obtain the top $\top_{\mathbf{R}}$ of that lattice, defined as a reactive design by $\top_{\mathbf{R}} = \mathbf{R}(true \vdash false)$, as Lemma 8 establishes.

Lemma 8. $ac2p(Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} Skip_{\mathbf{RAD}}) = \top_{\mathbf{R}}$

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 example of the general strengthening indicated by Theorem 21. Although the miraculous process $\top_{\mathbf{R}}$ is not part of the standard CSP semantics [2] it plays an important role, for example, in the characterisation of deadline operators in the context of timed versions of process calculi [37, 38, 39].

6.4.7. Sequential Composition

Since we have a theory of non-homogeneous relations, sequential composition is given by $;_{\mathcal{D}\mathbf{ac}}$, previously defined in the theory of angelic designs, which is itself layered upon $;_{\mathcal{A}}$. When considering reactive angelic designs P and Q, we obtain the following result for $P\ ;_{\mathcal{D}\mathbf{ac}}\ Q$ as a reactive angelic design.

Theorem 30.

$$P\ ;_{\mathcal{D}\mathbf{ac}}\ Q = \mathbf{RA} \circ \mathbf{A}\left(\begin{array}{l} \left(\begin{array}{l} \neg\, (\mathbf{RA1}(P^{f}_{f})\ ;_{\mathcal{A}}\ \mathbf{RA1}(true)) \\ \wedge \\ \neg\, (\mathbf{RA1}(P^{t}_{f})\ ;_{\mathcal{A}}\ (\neg\, s.wait \wedge \mathbf{RA2} \circ \mathbf{RA1}(Q^{f}_{f}))) \end{array}\right) \\ \vdash \\ \mathbf{RA1}(P^{t}_{f})\ ;_{\mathcal{A}}\ (s \in ac' \lhd s.wait \rhd (\mathbf{RA2} \circ \mathbf{RA1}(\neg\, Q^{f}_{f} \Rightarrow Q^{t}_{f}))) \end{array}\right)$$

The precondition is the conjunction of two sequential compositions defined using $;_{\mathcal{A}}$. The first requires that the precondition of P, that is $\neg\ P_f^f$, is satisfied, by stating that it is not the case that the negation (P_f^f) leads to an extension of the trace via $\mathbf{RA1}(true)$. The second composition requires that P_f^t, the postcondition of P, satisfies the precondition of Q when $s.wait$ is $false$, that is, when Q starts. **RA1** and **RA2** must hold for the negation of the precondition of Q. Finally, the postcondition is given by the sequential composition of the postcondition of P with a condition, where: if P is still waiting for the environment, that is, $s.wait$ is $true$, then it behaves as the identity $s \in ac'$, otherwise as the implication between the pre and postcondition of Q, where **RA1** and **RA2** are required to hold. This implication acts as a filter that eliminates final states of P that fail to satisfy the precondition of Q and yet satisfy **RA1** and **RA2**.

As an example, we consider the following result established in Lemma 9.

Lemma 9. $(Stop_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} Skip_{\mathbf{RAD}})\ ;_{\mathcal{D}\mathbf{ac}}\ Chaos_{\mathbf{RAD}} = Stop_{\mathbf{RAD}}$

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 $Chaos_{\mathbf{RAD}}$ is unsatisfiable: $false$. Once the preceding process of the sequential composition terminates, that is, $wait$ is $false$, the composition diverges. Because there is an angelic choice with the non-terminating $Stop_{\mathbf{RAD}}$, the divergence is avoided.

6.4.8. 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. The prefixing of a followed by an arbitrary process P ($a \rightarrow_{\mathbf{RAD}} P$) can be obtained by considering the sequential composition $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}\,;P$.

Definition 43.

$$a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}} \mathrel{\widehat{=}} \mathbf{RA} \circ \mathbf{A}\left(true \vdash \circledin{}^{y}_{ac'} \left(\begin{array}{l}(y.tr = s.tr \wedge a \notin y.ref)\\ \quad \lhd\, y.wait \,\rhd\\ (y.tr = s.tr \frown \langle a \rangle)\end{array}\right)\right)$$

The precondition is $true$ because a simple prefixing $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$ cannot diverge. 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 unchanged. In the second case, the process has interacted with the environment, and so the only guarantee is that 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 the following prefixing $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$ avoids divergence as established below.

Theorem 31. *Provided P is* **RAD***-healthy,*

$$a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} P = \mathbf{RA} \circ \mathbf{A}\left(true \vdash \circledin{}^{y}_{ac'} \left(\begin{array}{l}(y.tr = s.tr \wedge a \notin y.ref)\\ \quad \lhd\, y.wait \,\rhd\\ (y.tr = s.tr \frown \langle a \rangle)\end{array}\right) \wedge (\neg\ P_f^f \Rightarrow P_f^t)\right)$$

The behaviour of this process depends on that of P. If P diverges, then it behaves as $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$, otherwise there is an angelic choice between $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$ and P. We consider three examples.

Example 16. *Here we have a choice between terminating and deadlocking following the event a, sequentially composed with $Chaos_{\mathbf{RAD}}$:* $((a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} Skip_{\mathbf{RAD}})\ ;_{\mathcal{D}\mathbf{ac}}\ Chaos_{\mathbf{RAD}} = a \rightarrow_{\mathbf{RAD}} Stop_{\mathbf{RAD}}$ *The angelic choice avoids divergence by choosing non-termination: allowing the environment to perform the event a and then deadlocking.* □

Example 17. *Now we have a choice between terminating or diverging upon performing the event a.*

$$(a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} (a \rightarrow_{\mathbf{RAD}} Chaos_{\mathbf{RAD}}) = a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$$

The result is a process that following event a can only terminate, and thus avoids divergence. This illustrates a situation where our angelic choice operator is a counterpart to that of the refinement calculus: it resolves choices to avoid divergence. □

Example 18. *If we consider an angelic choice over prefixes on different events, the result is rather different.*

$$(a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} (b \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}}) = (a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) \sqcup_{\mathbf{RAD}} (b \rightarrow_{\mathbf{RAD}} \mathit{Choice}_{\mathbf{RAD}})$$

In this case, the possibility of diverging after the event b is avoided by turning $\mathit{Chaos}_{\mathbf{RAD}}$ into $\mathit{Choice}_{\mathbf{RAD}}$. The possibility for engaging in the event b 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 behind the theory of angelic processes that we discuss in Section 7.* □

6.4.9. External Choice

An external choice $P \,\Box_{\mathbf{RAD}}\, Q$, which offers the environment the choice over the events initially offered by processes P and Q, has a definition in our theory that is very similar to that in the original CSP theory.

Definition 44.

$$P \,\Box_{\mathbf{RAD}}\, Q \mathrel{\widehat{=}} \mathbf{RA} \circ \mathbf{A}\left((\neg\, P_f^f \wedge \neg\, Q_f^f) \vdash \circledin{}^{y}_{ac'}((P_f^t \wedge Q_f^t) \lhd y.tr = s.tr \wedge y.wait \rhd (P_f^t \vee Q_f^t))\right)$$

The precondition is the conjunction of the preconditions of P and Q, while the postcondition is split into two cases. When the trace $s.tr$ is unchanged and the process is waiting, the choice has not been resolved and the behaviour is given by the conjunction of the postconditions of P and Q. In this scenario, P and Q must be in agreement, and so, if there is angelic nondeterminism in P or Q, there must be an agreement on a single common state in ac'. If the choice has been made, the behaviour is given by the disjunction of the postconditions: the behaviour is that of P or Q, whichever has been chosen. Even if there is angelic nondeterminism in P or Q, there is also a requirement for an agreement on a final state, as enforced by $\circledin{}^{y}_{ac'}$.

We consider, for example, an external choice involving $\mathit{Stop}_{\mathbf{RAD}}$ and a reactive angelic design P.

Theorem 32. *Provided P is* **RAD***-healthy,*

$$P \,\Box_{\mathbf{RAD}}\, \mathit{Stop}_{\mathbf{RAD}} = \mathbf{RA} \circ \mathbf{A}(\neg\, P_f^f \vdash \exists\, y \bullet (P_f^t)[\{y\}/ac'] \wedge y \in ac')$$

In this case, the angelic nondeterminism in the postcondition P_f^t of P is collapsed, by requiring that there exists a state y such that P_f^t holds when ac' is a singleton, and that y is in every resulting ac'. Unlike in the original theory of CSP, $\mathit{Stop}_{\mathbf{RAD}}$ is not 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.

Theorem 33. *Provided P is* **RAD** *and* **A2***-healthy, $P \,\Box_{\mathbf{RAD}}\, \mathit{Stop}_{\mathbf{RAD}} = P$.*

Example 19. *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.*

$$\begin{array}{l}(a \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}} \sqcup_{\mathbf{RAD}} b \rightarrow_{\mathbf{RAD}} \mathit{Chaos}_{\mathbf{RAD}}) \,\Box_{\mathbf{RAD}}\, \mathit{Stop}_{\mathbf{RAD}} \\ = \\ \mathbf{RA} \circ \mathbf{A}\left(\begin{array}{l} \neg\, (\circledin{}^{y}_{ac'}(s.tr \frown \langle a \rangle \leq y.tr) \wedge \circledin{}^{y}_{ac'}(s.tr \frown \langle b \rangle \leq y.tr)) \\ \vdash \\ \circledin{}^{y}_{ac'}(y.wait \wedge y.tr = s.tr \wedge a \notin y.ref \wedge b \notin y.ref) \end{array}\right)\end{array}$$

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. So the external choice cannot be made, as it leads to divergence. □

6.5. Non-divergent Reactive Angelic Designs

As discussed in Section 4, 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 $\mathbf{ND_{RAD}}$ defined below.

Definition 45. $\mathbf{ND_{RAD}}(P) = P \sqcup_{\mathbf{RAD}} \mathit{Choice}_{\mathbf{RAD}}$

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.

Example 20. *Here we consider the case where* $\mathbf{ND_{RAD}}$ *is applied to the bottom of the lattice* $\mathit{Chaos}_{\mathbf{RAD}}$*:* $\mathbf{ND_{RAD}}(\mathit{Chaos}_{\mathbf{RAD}}) = \mathit{Choice}_{\mathbf{RAD}}$*. The divergence is avoided and the result is the process* $\mathit{Choice}_{\mathbf{RAD}}$*.* □

Example 21. *If instead we consider a process that is not divergent, such as* $\mathit{Skip}_{\mathbf{RAD}}$*, then* $\mathbf{ND_{RAD}}$ *is innocuous:* $\mathbf{ND_{RAD}}(a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}) = a \rightarrow_{\mathbf{RAD}} \mathit{Skip}_{\mathbf{RAD}}$*. The non-divergent process is a fixed point.* □

As stated below, the application of $\mathbf{ND_{RAD}}$ to a reactive angelic design P gives a process with the same postcondition, but precondition *true*. So, the resulting process is non-divergent.

Theorem 34. *Provided* P *is* **RAD***-healthy,* $\mathbf{ND_{RAD}}(P) = \mathbf{RA} \circ \mathbf{A}(\mathit{true} \vdash P_f^t)$.

Furthermore, Theorem 35 states that the fixed points P of $\mathbf{ND_{RAD}}$ have a precondition $\neg\ P_f^f$ that is satisfied for every possible initial state s and set of final states ac'.

Theorem 35. *Provided* P *is* **RAD***-healthy,* $\mathbf{ND_{RAD}}(P) = P \Leftrightarrow \forall\, s, ac' \bullet \neg\ P_f^f$.

These complementary results confirm our intuition about the definition of $\mathbf{ND_{RAD}}$: it characterises non-divergent processes. It is used in the next section to establish the correspondence between the subset of non-divergent reactive angelic designs and non-divergent angelic processes (see Fig. 2).

7. Angelic Processes

As discussed previously, in the theory of reactive angelic designs, as required by **RA1**, processes never undo the history of events. For example, $\mathit{Chaos}_{\mathbf{RAD}}$, 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 $s.tr$.

Since angelic choice is the least upper bound and $\mathit{Chaos}_{\mathbf{RAD}}$ the bottom of the lattice of reactive angelic designs, immediate divergence is avoided, if possible, in an angelic choice. However, once there is the possibility for interacting with the environment, like in the case of the process in Example 16, 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 section we propose a theory of angelic processes, like that of reactive angelic designs but which does not enforce **RA1** when a process diverges. This is a departure from the norm for a theory of CSP. The main consequence is that divergent processes have a different semantics, where any divergence is equated with immediate divergence. For example, in this theory, the angelic choice $a \rightarrow_{\mathbf{AP}} \mathit{Chaos}_{\mathbf{AP}} \sqcup b \rightarrow_{\mathbf{AP}} \mathit{Skip}_{\mathbf{AP}}$ is resolved in favour of $b \rightarrow_{\mathbf{AP}} \mathit{Skip}_{\mathbf{AP}}$ as $a \rightarrow_{\mathbf{AP}} \mathit{Chaos}_{\mathbf{AP}}$ is equated with $\mathit{Chaos}_{\mathbf{AP}}$, where the processes of the theory of angelic processes, which we define in the following sections, are subscripted with $_{\mathbf{AP}}$. This result, as well as other algebraic laws of interest are established in Theorem 50 and Examples 26 to 28. However, the subset of non-divergent processes preserves the existing semantics defined by **RAD**, and by extension, the semantics of non-divergent CSP processes.

In Section 7.1 we discuss the healthiness conditions of the theory of angelic processes. In Section 7.2 we study the relationship between angelic processes and reactive angelic designs. Finally in Section 7.3 we discuss the definition of the operators and important algebraic properties.

7.1. Healthiness Conditions

The alphabet of angelic processes is the same as that of reactive angelic designs: we have ok, ok', s and ac', where a *State* is defined with components tr, ref and $wait$.

As with every UTP theory, we define healthiness conditions. Since we aim to define a theory like **RAD**, but without enforcing **RA1**, we revisit **RAD** reproduced 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 $\mathbb{I}_{\mathcal{R}ac}$ 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 46. $\mathbf{AP}(P) \mathrel{\widehat{=}} \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A} \circ \mathbf{H1} \circ \mathbf{CSPA2}(P)$

The healthiness condition **RA3** used in **RAD** is replaced with $\mathbf{RA3_{AP}}$, which does not require **RA1**. The condition **A** is included in the functional composition, since it enforces both **A0** and **A1** (itself **PBMH** as previously discussed in Section 5) as required. **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 order of the composition of **A** with **H1** and **CSPA2** in the definition of **AP** is not relevant, as **A** commutes with **H**. This order captures the definition of an angelic design.

$\mathbf{RA3_{AP}}$ is introduced in the following Section 7.1.1. In Section 7.1.2 **AP** is explored. Finally, in Section 7.1.3 the subset of non-divergent angelic processes is characterised by another condition $\mathbf{ND_{AP}}$.

7.1.1. $\mathbf{RA3_{AP}}$

We define a new identity $\mathbb{I}_{\mathbf{AP}}$ as follows.

Definition 47. $\mathbb{I}_{\mathbf{AP}} \mathrel{\widehat{=}} \mathbf{H1}(ok' \wedge s \in ac')$

In contrast with the definition for $\mathbb{I}_{\mathcal{R}ac}$, there is no longer a requirement for **RA1** to be enforced when the process is unstable. Instead, the only guarantee is that, if the process is stable, that is, ok is *true*, then stability is maintained and the state is unchanged: the initial state s is in the set of final states ac'.

We can now define the healthiness condition $\mathbf{RA3_{AP}}$.

Definition 48. $\mathbf{RA3_{AP}}(P) \mathrel{\widehat{=}} \mathbb{I}_{\mathbf{AP}} \lhd s.wait \rhd P$

The definition of $\mathbf{RA3_{AP}}$ is similar to that of **RA3**, but we use $\mathbb{I}_{\mathbf{AP}}$, instead of $\mathbb{I}_{\mathcal{R}ac}$.

7.1.2. Angelic Processes

As already mentioned, the theory of angelic processes is characterised by the composition of $\mathbf{RA3_{AP}}$, **RA2**, **A**, **H1** and **CSPA2**. A result similar to Theorem 11 from the theory of reactive angelic designs can be obtained: **AP** processes can also be expressed in terms of a design.

Theorem 36. $\mathbf{AP}(P) = \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{A}(\neg\ P_f^f \vdash P_f^t)$

This establishes that an angelic process can also be specified 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 [10], 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*. In contrast, since we have dropped **RA1**, the theory of angelic processes is a theory of angelic designs as established by the following Theorem 37.

Theorem 37. $\mathbf{AP}(P) = \left(\begin{array}{l} \mathit{true} \lhd s.\mathit{wait} \rhd \neg\, \mathbf{RA2} \circ \mathbf{PBMH}(P_f^f) \\ \vdash \\ s \in ac' \lhd s.\mathit{wait} \rhd \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P_f^t) \end{array} \right)$

The precondition of the design has a conditional on $s.\mathit{wait}$. 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 $s.\mathit{wait}$. When the previous process has not terminated, 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 ac' is not empty and we enforce **RA2**, this means that **RA1** is enforced in the postcondition. 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**. It is, therefore, only in a situation of divergence, that **RA1** is no longer required to hold.

7.1.3. Non-divergent Angelic Processes

Like in the theory of reactive angelic designs, it is possible to identify the subset of non-divergent angelic processes. These are processes that satisfy the following healthiness condition $\mathbf{ND_{AP}}$. As depicted in Figs. 2 and 5 we show that the subsets of non-divergent processes and reactive angelic designs are isomorphic. This is a key result that supports the preservation of the semantics of non-divergent CSP processes.

Definition 49. $\mathbf{ND_{AP}}(P) \mathrel{\widehat{=}} \mathit{Choice}_{\mathbf{AP}} \sqcup_{\mathbf{AP}} P$

The definition of $\mathbf{ND_{AP}}$ is similar to that of $\mathbf{ND_{RAD}}$, but here we use the 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 shown next in Theorem 38.

Theorem 38. *Provided P is* **AP***-healthy,* $\mathit{Choice}_{\mathbf{AP}} \sqcup P = (\mathit{true} \vdash \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P_f^t))$.

The precondition is *true*, while the postcondition corresponds to that of P.

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 non-divergent. For example, $(s.\mathit{wait} \vdash s \in ac')$ is **H3**-healthy, however, it diverges when $s.\mathit{wait}$ is *false*.

7.2. Relationship with Reactive Angelic Designs

As part of our approach for validating the theories we propose, we study the relationship between the theories of angelic processes and of reactive angelic designs. Through the links in Section 4 between reactive angelic designs and CSP, the results here also link the theories of angelic processes and of CSP.

In Section 7.2.1 we discuss how reactive angelic designs can be mapped into the theory of angelic processes. In Section 7.2.3 we present the reverse mapping. Finally in Section 7.2.4 we show that the subsets of non-divergent processes of these theories are isomorphic.

7.2.1. From Reactive Angelic Designs to Angelic Processes

As already said, angelic processes are designs, and so are **H1** and **H2**-healthy. A reactive angelic design fails to be a design because of **RA1**, which prescribes a particular behaviour when *ok* is *false*. Accordingly, a reactive angelic design can be turned into an angelic process by applying **H1** and removing any guarantees when *ok* is *false*. **CSPA2** is basically **H2** and so enforced in both theories.

Below, we characterise designs obtained by applying **H1** to a reactive angelic design.

Theorem 39.

$$\mathbf{H1} \circ \mathbf{RAD}(P) = \left(\begin{array}{l} \mathit{true} \lhd s.\mathit{wait} \rhd \neg\, \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P_f^f) \\ \vdash \\ \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P_f^t) \end{array} \right)$$

In words, 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 23).

Example 22. *We consider the following example where* **H1** *is applied to* $Chaos_{\mathbf{RAD}}$.

$$\mathbf{H1}(Chaos_{\mathbf{RAD}}) = (s.wait \vee \neg\, \mathbf{RA1}(true) \vdash s.wait \wedge s \in ac')$$

In this case, if the previous process has not terminated, that is, $s.wait$ *is true, then the state is unchanged (*s *is required to be in the set of final states* ac'*). Once the process starts, that is,* $s.wait$ *is false, the design can be restated as* $ok \Rightarrow \mathbf{RA1}(true)$. □

7.2.2. Non-divergent Processes

The application of **H1** to a reactive angelic design that is non-divergent, that is $\mathbf{ND_{RAD}}$-healthy, yields a precondition that is still *true*, while the postcondition is that corresponding to the mapping through **H1**.

Lemma 10. $\mathbf{H1} \circ \mathbf{RA} \circ \mathbf{A}(true \vdash P_f^t) = (true \vdash \mathbf{RA3_{AP}} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P_f^t))$

Example 23. *We consider, for example, the mapping of the process* $Skip_{\mathbf{RAD}}$ *through* **H1**.

$$\mathbf{H1}(Skip_{\mathbf{RAD}}) = (true \vdash \mathbf{RA3_{AP}} \circ \textcircled{\in}^{y}_{ac'}(\neg\, y.wait \wedge y.tr = s.tr))$$

The original postcondition of $Skip_{\mathbf{RAD}}$ *is kept intact, in the context of* $\mathbf{RA3_{AP}}$. □

7.2.3. 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. In fact, the mapping we need is **RA1** itself. The result of applying **RA1** to an angelic process is established by Theorem 40.

Theorem 40. $\mathbf{RA1} \circ \mathbf{AP}(P) = \mathbf{RA} \circ \mathbf{A}(\neg\, P_f^f \vdash P_f^t)$

In the reactive angelic design $\mathbf{RA} \circ \mathbf{A}(\neg\, P_f^f \vdash P_f^t)$, **RA1** applies to the whole angelic design.

Example 24. *Here we apply* **RA1** *to the design of the previous Example 22.*

$$\mathbf{RA1}(s.wait \vee \neg\, \mathbf{RA1}(true) \vdash s.wait \wedge s \in ac') = Chaos_{\mathbf{RAD}}$$

This result shows that we obtain the original $Chaos_{\mathbf{RAD}}$ *of reactive angelic designs.* □

As we discuss in the next Section 7.2.4, every reactive angelic design mapped into this theory can be mapped back through **RA1** to obtain a reactive angelic design.

7.2.4. Isomorphism and Galois Connection

The discussion above suggests 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 41.

Theorem 41. $\mathbf{RA1} \circ \mathbf{H1} \circ \mathbf{RAD}(P) = \mathbf{RAD}(P)$

This is a fundamental result, which, together with the links between the theories of reactive angelic designs and CSP, establishes that every CSP process can be modelled as an angelic process. This follows from the fact the composition of Galois connections is also a Galois connection (Theorem 4.2.5 in [10]). When we consider the mapping in the opposite direction, however, an inequality is obtained.

Theorem 42. $\mathbf{H1} \circ \mathbf{RA1} \circ \mathbf{AP}(P) \sqsupseteq \mathbf{AP}(P)$

This is expected, since reactive angelic designs require **RA1** in all circumstances, whereas angelic processes are designs. Thus there is a Galois connection between reactive angelic designs and angelic processes.

Example 25. *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.*

$$\mathbf{H1} \circ \mathbf{RA1}(s.wait \vdash s \in ac') = (s.wait \vee \neg\, \mathbf{RA1}(true) \vdash s.wait \wedge s \in ac')$$

The result is exactly the same as that of applying **H1** *to* $Chaos_{\mathbf{RAD}}$*. This angelic process has a weaker precondition than that of the bottom* $\bot_{\mathbf{AP}}$*, and so is a refinement of* $\bot_{\mathbf{AP}}$*.* □

If we restrict our attention to the subset of angelic processes that are non-divergent, then Theorem 42 can be strengthened into an equality. Therefore, the subsets of non-divergent processes of the theories of angelic processes and of reactive angelic designs are isomorphic.

Theorem 43. $\mathbf{H1} \circ \mathbf{RA1} \circ \mathbf{ND_{AP}} \circ \mathbf{AP}(P) = \mathbf{ND_{AP}} \circ \mathbf{AP}(P)$

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 see that there is a subset corresponding exactly to non-divergent CSP processes in the theory of angelic processes.

7.3. Operators

In this section we present some operators of the theory of angelic processes. We also study the relationship between these operators and their counterparts as reactive angelic designs presented in Section 6.4.

7.3.1. Angelic Choice

The angelic choice operator of this theory is also defined through the least upper bound.

Definition 50. $P \sqcup_{\mathbf{AP}} Q \mathrel{\widehat{=}} P \wedge Q$

The subset of non-divergent angelic processes, characterised by $\mathbf{ND_{AP}}$, is closed under angelic choice.

Theorem 44. *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.

Theorem 45. *Provided* P *and* Q *are* $\mathbf{RAD}$*-healthy,* $\mathbf{RA1}(\mathbf{H1}(P) \sqcup_{\mathbf{AP}} \mathbf{H1}(Q)) = P \sqcup_{\mathbf{RAD}} Q$*.*

In words, if we consider two reactive angelic designs P and Q, and after mapping them through **H1** we take the least upper bound $\sqcup_{\mathbf{AP}}$ and apply **RA1**, then we obtain the same result as the least upper bound $\sqcup_{\mathbf{RAD}}$ of P and Q. Together with Theorem 44, 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 application of **H1** to the least upper bound of two angelic processes P and Q mapped through **RA1**, we do not obtain an equality.

Theorem 46. *Provided* P *and* Q *are* $\mathbf{AP}$*-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 **RA1**.

7.3.2. Demonic Choice

Demonic choice is defined using the greatest lower bound, which is disjunction.

Definition 51. $P \sqcap_{\mathbf{AP}} Q \mathrel{\widehat{=}} P \vee Q$

The subset of non-divergent processes is closed under demonic choice.

Theorem 47. *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 can be equally obtained through the greatest lower bound.

Theorem 48. *Provided P and Q are* **RAD***-healthy,* $\mathbf{RA1}(\mathbf{H1}(P) \sqcap_{\mathbf{AP}} \mathbf{H1}(Q)) = P \sqcap_{\mathbf{RAD}} Q$.

If we map P and Q through **H1**, take the greatest lower bound $\sqcap_{\mathbf{AP}}$, and then apply **RA1**, then we obtain the greatest lower bound of P and Q. With this result, together with the closure of $\sqcap_{\mathbf{AP}}$ under $\mathbf{ND}_{\mathbf{AP}}$ (Theorem 47), it is possible to ascertain that the demonic choice for non-divergent processes is in correspondence in both theories. On the other hand, in general, the greatest lower bound of the theory of angelic processes cannot be replicated in the theory of reactive angelic designs.

Theorem 49. *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 angelic processes do not enforce **RA1**.

7.3.3. Divergence

The bottom of the lattice is $Chaos_{\mathbf{AP}}$, defined in terms of the bottom of the lattice of designs, as follows.

Definition 52. $Chaos_{\mathbf{AP}} \mathrel{\widehat{=}} \mathbf{AP}(false \vdash true)$

$Chaos_{\mathbf{AP}}$ can be explicitly described as a design as established by Lemma 11.

Lemma 11. $Chaos_{\mathbf{AP}} = (s.wait \vdash s \in ac')$

The precondition requires the component *wait* of the initial state s to be *true*, while the postcondition defines that the state is kept unchanged (by requiring s to be in ac'). As long as the previous process does not terminate, the state is kept unchanged. However, once it does terminate, then $Chaos_{\mathbf{AP}}$ diverges.

$Chaos_{\mathbf{AP}}$ is a unit for angelic choice as established by Theorem 50.

Theorem 50. *Provided P is* **AP***-healthy,* $P \sqcup_{\mathbf{AP}} Chaos_{\mathbf{AP}} = P$.

In other words, if possible, in an angelic choice, divergence is avoided.

The process that corresponds to $Chaos_{\mathbf{RAD}}$ is $ChaosCSP_{\mathbf{AP}}$, defined through a design as shown below.

Definition 53. $ChaosCSP_{\mathbf{AP}} \mathrel{\widehat{=}} \mathbf{AP}(\neg\, \mathbf{RA1}(true) \vdash true)$

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 corresponds exactly to the mapping of $Chaos_{\mathbf{RAD}}$ through the linking function **H1**.

Theorem 51. $\mathbf{H1}(Chaos_{\mathbf{RAD}}) = ChaosCSP_{\mathbf{AP}}$

Additionally, if we map $ChaosCSP_{\mathbf{AP}}$ through **RA1**, we obtain the bottom of the lattice $Chaos_{\mathbf{RAD}}$.

Theorem 52. $\mathbf{RA1}(ChaosCSP_{\mathbf{AP}}) = Chaos_{\mathbf{RAD}}$

This follows from the general result of Theorem 41.

7.3.4. Choice

The most nondeterministic process that does not diverge is defined as $Choice_{\mathbf{AP}}$.

Definition 54. $Choice_{\mathbf{AP}} \mathrel{\widehat{=}} \mathbf{AP}(true \vdash ac' \neq \emptyset)$

The precondition is *true*, while any non-empty set of final states ac' is acceptable.

As previously indicated, $Choice_{\mathbf{AP}}$ is used to characterise algebraically the subset of angelic processes that are non-divergent. It is a fixed point of $\mathbf{ND}_{\mathbf{AP}}$ and, by definition, also of **AP**. It is the counterpart to $Choice_{\mathbf{RAD}}$ of the theory of reactive angelic designs as established by the following Theorems 53 and 54.

Theorem 53. $\mathbf{H1}(Choice_{\mathbf{RAD}}) = Choice_{\mathbf{AP}}$

Theorem 54. $\mathbf{RA1}(Choice_{\mathbf{AP}}) = Choice_{\mathbf{RAD}}$

Theorem 54 follows from Theorem 53 and the general result of Theorem 41.

7.3.5. Stop

Deadlock is modelled by $Stop_{\mathbf{AP}}$, whose definition is similar to that of $Stop_{\mathbf{RAD}}$.

Definition 55. $Stop_{\mathbf{AP}} \mathrel{\widehat{=}} \mathbf{AP}(true \vdash \circledS^{y}_{ac'}(y.tr = s.tr \wedge y.wait))$

The postcondition states that there is a final state y in ac' where the trace is kept unchanged and the process is always waiting. This definition can be obtained by applying **H1** to $Stop_{\mathbf{RAD}}$.

Theorem 55. $\mathbf{H1}(Stop_{\mathbf{RAD}}) = Stop_{\mathbf{AP}}$

Similarly, $Stop_{\mathbf{RAD}}$ can be obtained by applying **RA1** to $Stop_{\mathbf{AP}}$.

Theorem 56. $\mathbf{RA1}(Stop_{\mathbf{AP}}) = Stop_{\mathbf{RAD}}$

This is expected, since $Stop_{\mathbf{AP}}$ is a non-divergent angelic process.

7.3.6. Skip

The process that always terminates successfully is characterised by $Skip_{\mathbf{AP}}$.

Definition 56. $Skip_{\mathbf{AP}} \mathrel{\widehat{=}} \mathbf{AP}(true \vdash \circledS^{y}_{ac'}(y.tr = s.tr \wedge \neg\, y.wait))$

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*. $Skip_{\mathbf{AP}}$ is in correspondence with $Skip_{\mathbf{RAD}}$ of the theory of reactive angelic designs as established by the following Theorems 57 and 58.

Theorem 57. $\mathbf{H1}(Skip_{\mathbf{RAD}}) = Skip_{\mathbf{AP}}$

Theorem 58. $\mathbf{RA1}(Skip_{\mathbf{AP}}) = Skip_{\mathbf{RAD}}$

These results are expected since $Skip_{\mathbf{AP}}$ and $Skip_{\mathbf{RAD}}$ are both non-divergent processes.

7.3.7. Sequential composition

For angelic processes, sequential composition is also $;_{\mathcal{D}\mathbf{ac}}$. When we consider angelic processes P and Q, we have the result below, which is similar to that obtained for reactive angelic designs (Theorem 30).

Theorem 59. *Provided P and Q are* **AP***-healthy,*

$$P \;;_{\mathcal{D}\mathbf{ac}}\; Q = \mathbf{AP}\left(\begin{array}{l} \neg\,(P_f^f \;;_{\mathcal{A}}\; true) \wedge \neg\,(\mathbf{RA1}(P_f^t) \;;_{\mathcal{A}}\; (\neg\, s.wait \wedge \mathbf{RA2}(Q_f^f))) \\ \vdash \\ \mathbf{RA1}(P_f^t) \;;_{\mathcal{A}}\; (s \in ac' \lhd s.wait \rhd \mathbf{RA2}(\neg\, Q_f^f \Rightarrow \mathbf{RA1}(Q_f^t))) \end{array}\right)$$

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, $;_{\mathcal{D}\mathbf{ac}}$ is a sequential composition operator that behaves differently to that of CSP. It can back propagate the divergence of Q through P, irrespective of other interactions that happen in P, if eventually P terminates and Q takes over.

Example 26. *We consider the following example: $(Stop_{\mathbf{AP}} \sqcup_{\mathbf{AP}} Skip_{\mathbf{AP}}) \;;_{\mathcal{D}\mathbf{ac}} Chaos_{\mathbf{AP}} = Stop_{\mathbf{AP}}$. 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.

Theorem 60. *Provided P and Q are* **RAD***-healthy,* $\mathbf{RA1}(\mathbf{H1}(P) \ ;_{\mathcal{D}\mathbf{ac}} \mathbf{H1}(Q)) \sqsubseteq P \ ;_{\mathcal{D}\mathbf{ac}} Q$.

This is because the possibility to diverge in Q, 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 **RA1**, there is a weakening. Similarly, the reverse mapping through **H1** of the sequential composition of two angelic processes P and Q mapped through **RA1** is a refinement of the sequential composition of P and Q, as established by Theorem 61.

Theorem 61. *Provided P and Q are* **AP***-healthy,* $\mathbf{H1}(\mathbf{RA1}(P) \ ;_{\mathcal{D}\mathbf{ac}} \mathbf{RA1}(Q)) \sqsupseteq P \ ;_{\mathcal{D}\mathbf{ac}} Q$.

This is due to the fact that the notions of divergence in the two theories are different. In a sequential composition of P with the bottom of the lattice $Chaos_{\mathbf{AP}}$, the result is also $Chaos_{\mathbf{AP}}$. If we map $Chaos_{\mathbf{AP}}$ through **RA1**, the result is $Chaos_{\mathbf{RAD}}$ (Theorem 52), which when sequentially composed after the process $\mathbf{RA1}(P)$, still preserves the history of events in P. On the other hand, 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 60 can be strengthened into an equality.

Theorem 62. *Provided P and Q are* **RAD** *and* $\mathbf{ND_{RAD}}$*-healthy,* $\mathbf{RA1}(\mathbf{H1}(P) \ ;_{\mathcal{D}\mathbf{ac}} \mathbf{H1}(Q)) = P \ ;_{\mathcal{D}\mathbf{ac}} Q$.

In addition, the set of non-divergent angelic processes characterised by $\mathbf{ND_{AP}}$ is closed under the $;_{\mathcal{D}\mathbf{ac}}$ operator, as established by the following Theorem 63.

Theorem 63. *Provided P and Q are* **AP** *and* $\mathbf{ND_{AP}}$*-healthy,* $\mathbf{ND_{AP}}(P \ ;_{\mathcal{D}\mathbf{ac}} Q) = P \ ;_{\mathcal{D}\mathbf{ac}} Q$.

Thus, as long as P and Q are non-divergent, $;_{\mathcal{D}\mathbf{ac}}$ 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. Proof of closure for angelic processes that are not necessarily non-divergent, including proofs for closure of other operators, is available in [35].

7.3.8. Prefixing

Similarly to the previous non-divergent processes, event prefixing has a definition similar to that of $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$ in the theory of reactive angelic designs.

Definition 57. $a \rightarrow_{\mathbf{AP}} Skip_{\mathbf{AP}} \mathrel{\widehat{=}} \mathbf{AP}(true \vdash \circledin{}^{y}_{ac'}((y.tr = s.tr \wedge a \notin y.ref) \lhd y.wait \rhd (y.tr = s.tr \frown \langle a \rangle)))$

The postcondition is like that of the following reactive angelic design $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$ (Definition 43). The correspondence between $a \rightarrow_{\mathbf{AP}} Skip_{\mathbf{AP}}$ and $a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$ is established by Lemmas 12 and 13.

Lemma 12. $\mathbf{H1}(a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}) = a \rightarrow_{\mathbf{AP}} Skip_{\mathbf{AP}}$

Lemma 13. $\mathbf{RA1}(a \rightarrow_{\mathbf{AP}} Skip_{\mathbf{AP}}) = a \rightarrow_{\mathbf{RAD}} Skip_{\mathbf{RAD}}$

Example 27. *In order to illustrate the behaviour of prefixing in the presence of divergence in the theory of angelic processes, we consider the following example: $a \rightarrow_{\mathbf{AP}} Chaos_{\mathbf{AP}} = Chaos_{\mathbf{AP}}$.*

Proof.

$$
\begin{array}{ll}
a \rightarrow_{\mathbf{AP}} Chaos_{\mathbf{AP}} & \{\text{Definition of } Chaos_{\mathbf{AP}} \text{ and prefixing (Theorem T.6.4.23 in [35])}\} \\
= \mathbf{AP}\left(\begin{array}{l} \neg\,(\exists\, y \bullet \neg\, y.wait \wedge y.tr = s.tr \frown \langle a \rangle \wedge (\mathbf{RA2} \circ \mathbf{PBMH}(true))[y/s]) \\ \vdash \\ \exists\, y \bullet \left(\begin{array}{l} (y.tr = s.tr \wedge a \notin y.ref \wedge y \in ac') \\ \lhd y.wait \rhd \\ (y.tr = s.tr \frown \langle a \rangle \wedge \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(true)[y/s]) \end{array}\right) \end{array}\right) & \{\text{Theorem 71}\}
\end{array}
$$

$$= \mathbf{AP}\left(\begin{array}{l} \neg\,(\exists\, y \bullet \neg\, y.wait \wedge y.tr = s.tr \frown \langle a\rangle \wedge (\mathbf{RA2} \circ \mathbf{PBMH}(true))[y/s]) \\ \vdash \\ \exists\, y \bullet \left(\begin{array}{l} (y.tr = s.tr \wedge a \notin y.ref \wedge y \in ac') \\ \lhd y.wait \rhd \\ (y.tr = s.tr \frown \langle a\rangle \wedge \mathbf{RA1} \circ \mathbf{RA2} \circ \mathbf{PBMH}(true)[y/s]) \end{array}\right) \end{array}\right)$$

{Def. of **PBMH** and **RA2**}

$$= \mathbf{AP}\left(\begin{array}{l} \neg\,(\exists\, y \bullet \neg\, y.wait \wedge y.tr = s.tr \frown \langle a\rangle \wedge (true)[y/s]) \\ \vdash \\ \exists\, y \bullet \left(\begin{array}{l} (y.tr = s.tr \wedge a \notin y.ref \wedge y \in ac') \\ \lhd y.wait \rhd \\ (y.tr = s.tr \frown \langle a\rangle \wedge \mathbf{RA1}(true)[y/s]) \end{array}\right) \end{array}\right)$$

{Substitution and predicate calculus}

$$= \mathbf{AP}\left(\begin{array}{l} false \\ \vdash \\ \exists\, y \bullet \left(\begin{array}{l} (y.tr = s.tr \wedge a \notin y.ref \wedge y \in ac') \\ \lhd y.wait \rhd \\ (y.tr = s.tr \frown \langle a\rangle \wedge \mathbf{RA1}(true)[y/s]) \end{array}\right) \end{array}\right)$$

{Def. of design and predicate calculus}

$$= \mathbf{AP}(false \vdash true)$$

{Definition of $Chaos_{\mathbf{AP}}$}

$$= Chaos_{\mathbf{AP}}$$

In this case, the potential for divergence after performing the event a leads to immediate divergence. If, instead, we sequentially compose prefixing on the event a with $ChaosCSP_{\mathbf{AP}}$, the behaviour is different.

Lemma 14.

$$a \rightarrow_{\mathbf{AP}} ChaosCSP_{\mathbf{AP}} = \mathbf{AP}(\neg \circledin{}^{y}_{ac'}(s.tr \frown \langle a\rangle \leq y.tr) \vdash \circledin{}^{y}_{ac'}(y.wait \wedge y.tr = s.tr \wedge a \notin y.ref))$$

This mirrors the behaviour of $a \rightarrow_{\mathbf{RAD}} Chaos_{\mathbf{RAD}}$ of the theory of reactive angelic designs, since the process $ChaosCSP_{\mathbf{AP}}$ corresponds to $Chaos_{\mathbf{RAD}}$ and does not allow the occurrence of a to be backtracked to avoid the divergence that follows. □

Example 28. *Finally, we revisit the choice in Example 18 in the theory of angelic processes.*

$$a \rightarrow_{\mathbf{AP}} Chaos_{\mathbf{AP}} \sqcup_{\mathbf{AP}} b \rightarrow_{\mathbf{AP}} Skip_{\mathbf{AP}} = b \rightarrow_{\mathbf{AP}} Skip_{\mathbf{AP}}$$

Now, the possibility for divergence is avoided altogether, and the result is the prefixing on b. As required, the angelic choice can avoid processes that may lead to divergence, a property that is not observed in the theory of reactive angelic designs. □

In summary, because we do not adopt **RA1** as a healthiness condition, an angelic choice can discard traces of events leading to divergence. The result 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 behaves arbitrarily, and may even undo the history of interactions. More importantly, in an angelic choice involving other interactions, it becomes possible for an angelic choice to undo the history of events, if necessary, and avoid divergence. This is much closer in spirit to the angelic choice of the refinement calculus.

8. Conclusions

In general, angelic nondeterminism enables a high degree of abstraction in the context of formal models. Its characterisation in the context of process algebras, such as CSP has, however, to the best of our knowledge, been elusive. The existing approaches have either considered notions of angelic nondeterminism different from that of refinement calculi [2], or different CSP semantics [7].

Angelic nondeterminism has traditionally been studied in the context of theories of correctness for sequential computations, such as the refinement calculus [4, 5, 3], where it is characterised as the least upper

bound of the lattice of monotonic predicate transformers. Isomorphic models include Rewitzky [11]'s binary multirelations, which is the foundation of our approach.

The work presented in this paper is the foundation for the development of process algebras with angelic nondeterminism in the wider context of state-rich reactive systems. Our approach has focused mainly on CSP, but because we use the UTP, our results are applicable to other process calculi. This may include, for example, process algebras like rTiMo [40] and CSP# [41], which have also been given UTP semantics [42, 43]. Event-based modelling languages may also benefit from the inclusion of angelic choice as a specification construct. SystemC, for example, has a UTP semantics [44] that could make our results also applicable.

While we have studied a number of CSP operators, an interesting avenue for future work is the treatment of other operators, such as hiding and parallel composition. (Recursion can be treated in a similar way to other UTP theories as the weakest fixed point.) We expect that, for many operators, 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. Input synchronisations can typically be treated through replicated external choices, although in the case of infinite sets this may require an adequate treatment of variables, following, for example, the definition for input prefixing in *Circus* [9, p. 37]. A definition for parallel composition, using the parallel by merge approach [10] will require support for renaming state 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 [10, 45] of CSP and whether the addition of angelic choice may be useful to enable or simplify the algebraic specification of these.

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. This 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. [46] and Feliachi et al. [47] using Isabelle/HOL, Zeyda and Cavalcanti [48] and Oliveira [9] using ProofPower/Z, and Butterfield [49].

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 other process algebra for refinement, and for the additional exploration of algebra, techniques, and applications of angelic nondeterminism in the context of concurrent and reactive models.

Appendix A. Proofs

Here we include results supporting proofs shown above. In particular, Appendix A.1 contains results about the healthiness conditions of angelic designs, while Appendix A.2 contains results about the healthiness conditions of reactive angelic designs. Appendix A.3 contains results pertaining to the linking functions, while Appendix A.4 contains lemmas about substitutions with record states. As mentioned before, proofs of results not explicitly included here can be found in [35].

Appendix A.1. Angelic Designs

Lemma 15. $\mathbf{PBMH}(ac' \neq \emptyset) = ac' \neq \emptyset$

Proof.

$$
\begin{array}{ll}
\mathbf{PBMH}(ac' \neq \emptyset) & \{\text{Definition of } \mathbf{PBMH}\} \\
= \exists\, ac_0 \bullet ac_0 \neq \emptyset \land ac_0 \subseteq ac' & \{\text{Property of sets}\} \\
= ac' \neq \emptyset & \square
\end{array}
$$

Theorem 64. $\mathbf{H1} \circ \mathbf{PBMH}(P) = \mathbf{PBMH} \circ \mathbf{H1}(P)$

Proof.

$$
\begin{array}{ll}
\mathbf{PBMH} \circ \mathbf{H1}(P) & \{\text{Definition of } \mathbf{H1} \text{ and predicate calculus}\} \\
= \mathbf{PBMH}(\neg\, ok \vee P) & \{\text{Distributivity of } \mathbf{PBMH}\} \\
= \mathbf{PBMH}(\neg\, ok) \vee \mathbf{PBMH}(P) & \{\text{Definition of } \mathbf{PBMH}\} \\
= \neg\, ok \vee \mathbf{PBMH}(P) & \{\text{Predicate calculus}\} \\
= ok \Rightarrow \mathbf{PBMH}(P) & \{\text{Definition of } \mathbf{H1}\} \\
= \mathbf{H1} \circ \mathbf{PBMH}(P) & \square
\end{array}
$$

Theorem 65. $\mathbf{H2} \circ \mathbf{PBMH}(P) = \mathbf{PBMH} \circ \mathbf{H2}(P)$

Proof.

$$
\begin{array}{ll}
\mathbf{H2} \circ \mathbf{PBMH}(P) & \{\text{Definition of } \mathbf{H2} \text{ (J-split)}\} \\
= \mathbf{PBMH}(P)\,;((ok \Rightarrow ok') \wedge v' = v) & \{\text{Definition of } \mathbf{PBMH}\} \\
= (P\,; ac \subseteq ac' \wedge v' = v)\,;((ok \Rightarrow ok') \wedge v' = v) & \{\text{Composition is associative}\} \\
= P\,;((ac \subseteq ac' \wedge v' = v)\,;((ok \Rightarrow ok') \wedge v' = v)) & \{\text{Lemma L.E.6.1 in [35]}\} \\
= P\,;(((ok \Rightarrow ok') \wedge v' = v)\,;(ac \subseteq ac' \wedge v' = v)) & \{\text{Composition is associative}\} \\
= (P\,;((ok \Rightarrow ok') \wedge v' = v))\,;(ac \subseteq ac' \wedge v' = v) & \{\text{Definition of } \mathbf{PBMH}\} \\
= \mathbf{PBMH}(P\,;((ok \Rightarrow ok') \wedge v' = v)) & \{\text{Definition of } \mathbf{H2}\} \\
= \mathbf{PBMH} \circ \mathbf{H2}(P) & \square
\end{array}
$$

Lemma 16. $\mathbf{PBMH}(P \vdash Q) = (\neg\, \mathbf{PBMH}(\neg\, P) \vdash \mathbf{PBMH}(Q))$

Proof.

$$
\begin{array}{ll}
\mathbf{PBMH}(P \vdash Q) & \{\text{Definition of design}\} \\
= \mathbf{PBMH}((ok \wedge P) \Rightarrow (Q \wedge ok')) & \{\text{Predicate calculus}\} \\
= \mathbf{PBMH}(\neg\, ok \vee \neg\, P \vee (Q \wedge ok')) & \{\text{Distributivity of } \mathbf{PBMH}\} \\
= \mathbf{PBMH}(\neg\, ok) \vee \mathbf{PBMH}(\neg\, P) \vee \mathbf{PBMH}(Q \wedge ok') & \{\text{Definition of } \mathbf{PBMH}\} \\
= \neg\, ok \vee \mathbf{PBMH}(\neg\, P) \vee \mathbf{PBMH}(Q \wedge ok') & \{\text{Definition of } \mathbf{PBMH}\} \\
= \neg\, ok \vee \mathbf{PBMH}(\neg\, P) \vee (\mathbf{PBMH}(Q) \wedge ok') & \{\text{Predicate calculus}\} \\
= (ok \wedge \neg\, \mathbf{PBMH}(\neg\, P)) \Rightarrow (\mathbf{PBMH}(Q) \wedge ok') & \{\text{Definition of design}\} \\
= (\neg\, \mathbf{PBMH}(\neg\, P) \vdash \mathbf{PBMH}(Q)) & \square
\end{array}
$$

Lemma 17. $\mathbf{A2}(P \vdash Q) = (\neg\, \mathbf{A2}(\neg\, P) \vdash \mathbf{A2}(Q))$

Proof.

$$
\begin{array}{ll}
\mathbf{A2} \circ \mathbf{A}(P \vdash Q) & \{\text{Definition of design}\} \\
= \mathbf{A2}((ok \wedge P) \Rightarrow (Q \wedge ok')) & \{\text{Predicate calculus}\} \\
= \mathbf{A2}(\neg\, ok \vee \neg\, P \vee (Q \wedge ok')) & \{\text{Distributivity of } \mathbf{A2}\} \\
= \mathbf{A2}(\neg\, ok) \vee \mathbf{A2}(\neg\, P) \vee (\mathbf{A2}(Q) \wedge ok') & \{\text{Definition of } \mathbf{A2}\} \\
= \neg\, ok \vee \mathbf{A2}(\neg\, P) \vee (\mathbf{A2}(Q) \wedge ok') & \{\text{Predicate calculus}\} \\
= (ok \wedge \neg\, \mathbf{A2}(\neg\, P)) \Rightarrow (\mathbf{A2}(Q) \wedge ok') & \{\text{Definition of design}\} \\
= (\neg\, \mathbf{A2}(\neg\, P) \vdash \mathbf{A2}(Q)) & \square
\end{array}
$$

Lemma 18. *Provided ok' and ac' are not free in e, $\mathbf{A}(P)[e/s] = \mathbf{A}(P[e/s])$.*

Proof.

$$
\begin{array}{ll}
\mathbf{A}(P)[e/s] & \{\text{Definition of } \mathbf{A}\} \\
= (\mathbf{A0} \circ \mathbf{PBMH}(P))[e/s] & \{\text{Lemma L.C.1.1 in [35]}\} \\
= \mathbf{A0} \circ (\mathbf{PBMH}(P))[e/s] & \{\text{Lemma L.E.5.2 in [35]}\} \\
= \mathbf{A0} \circ \mathbf{PBMH}(P[e/s]) & \{\text{Definition of } \mathbf{A}\} \\
= \mathbf{A}(P[e/s]) & \Box
\end{array}
$$

Appendix A.2. Reactive Angelic Designs

Theorem 66. $\mathbf{PBMH} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P) = \mathbf{RA2} \circ \mathbf{PBMH}(P)$

Proof.

$$
\begin{array}{ll}
\mathbf{PBMH} \circ \mathbf{RA2} \circ \mathbf{PBMH}(P) & \{\text{Definition of } \mathbf{PBMH}\} \\
= \mathbf{PBMH} \circ \mathbf{RA2}(\exists\, ac_0 \bullet P[ac_0/ac'] \wedge ac_0 \subseteq ac') & \{\text{Definition of } \mathbf{RA2}\} \\
= \mathbf{PBMH} \left(\begin{array}{l} (\exists\, ac_0 \bullet P[ac_0/ac'] \wedge ac_0 \subseteq ac') \\ \left[\begin{array}{l} s \oplus \{tr \mapsto \langle\rangle\} \\ \{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\} \end{array} \middle/ \begin{array}{r} s \\ ac' \end{array} \right] \end{array} \right) & \{\text{Substitution}\} \\
= \mathbf{PBMH} \left(\begin{array}{l} \exists\, ac_0 \bullet P[ac_0/ac'][s \oplus \{tr \mapsto \langle\rangle\}/s] \\ \wedge\ ac_0 \subseteq \{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\} \end{array} \right) & \{\text{Definition of } \mathbf{PBMH}\} \\
= \left(\begin{array}{l} \exists\, ac_1, ac_0 \bullet P[ac_0/ac'][s \oplus \{tr \mapsto \langle\rangle\}/s] \\ \wedge\ ac_0 \subseteq \{z \mid z \in ac_1 \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\} \\ \wedge\ ac_1 \subseteq ac' \end{array} \right) & \{\text{Definition of subset inclusion}\} \\
= \left(\begin{array}{l} \exists\, ac_1, ac_0 \bullet P[ac_0/ac'][s \oplus \{tr \mapsto \langle\rangle\}/s] \\ \wedge\ \forall\, x \bullet x \in ac_0 \Rightarrow x \in \{z \mid z \in ac_1 \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\} \\ \wedge\ ac_1 \subseteq ac' \end{array} \right) & \{\text{Property of sets}\} \\
= \left(\begin{array}{l} \exists\, ac_1, ac_0 \bullet P[ac_0/ac'][s \oplus \{tr \mapsto \langle\rangle\}/s] \\ \wedge\ \forall\, x \bullet x \in ac_0 \Rightarrow \exists\, z \bullet z \in ac_1 \wedge s.tr \leq z.tr \wedge x = z \oplus \{tr \mapsto z.tr - s.tr\} \\ \wedge\ ac_1 \subseteq ac' \end{array} \right) & \{\text{Lemma L.G.1.8 in [35]}\} \\
= \left(\begin{array}{l} \exists\, ac_1, ac_0 \bullet P[ac_0/ac'][s \oplus \{tr \mapsto \langle\rangle\}/s] \\ \wedge\ \forall\, x \bullet x \in ac_0 \Rightarrow x \oplus \{tr \mapsto s.tr \frown x.tr\} \in ac_1 \\ \wedge\ ac_1 \subseteq ac' \end{array} \right) & \{\text{Lemma L.E.4.13 in [35]}\} \\
= \left(\begin{array}{l} \exists\, ac_0 \bullet P[ac_0/ac'][s \oplus \{tr \mapsto \langle\rangle\}/s] \\ \wedge \\ \forall\, x \bullet x \in ac_0 \Rightarrow (x \oplus \{tr \mapsto s.tr \frown x.tr\}) \in ac' \end{array} \right) & \{\text{Lemma L.G.1.8 in [35]}\} \\
= \left(\begin{array}{l} \exists\, ac_0 \bullet P[ac_0/ac'][s \oplus \{tr \mapsto \langle\rangle\}/s] \\ \wedge\ \forall\, x \bullet x \in ac_0 \Rightarrow \exists\, z \bullet z \in ac' \wedge s.tr \leq z.tr \wedge x = z \oplus \{tr \mapsto z.tr - s.tr\} \end{array} \right) & \{\text{Property of sets}\} \\
= \left(\begin{array}{l} \exists\, ac_0 \bullet P[ac_0/ac'][s \oplus \{tr \mapsto \langle\rangle\}/s] \\ \wedge \\ ac_0 \subseteq \{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\} \end{array} \right) & \{\text{Substitution}\} \\
= \left(\begin{array}{l} (\exists\, ac_0 \bullet P[ac_0/ac'] \wedge ac_0 \subseteq ac') \\ \left[\begin{array}{l} s \oplus \{tr \mapsto \langle\rangle\} \\ \{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\} \end{array} \middle/ \begin{array}{r} s \\ ac' \end{array} \right] \end{array} \right) & \{\text{Definition of } \mathbf{RA2}\} \\
= \mathbf{RA2}(\exists\, ac_0 \bullet P[ac_0/ac'] \wedge ac_0 \subseteq ac') & \{\text{Definition of } \mathbf{PBMH}\} \\
= \mathbf{RA2} \circ \mathbf{PBMH}(P) & \Box
\end{array}
$$

Theorem 67. $\mathbf{RA} \circ \mathbf{A}(P) = \mathbf{RA} \circ \mathbf{PBMH}(P)$

Proof.

$\mathbf{RA} \circ \mathbf{A}(P)$ {Definition of **RA** and **A**}
$= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{A0} \circ \mathbf{A1}(P)$ {**A1** is **PBMH**}
$= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{A0} \circ \mathbf{PBMH}(P)$ {Theorem 9}
$= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P)$ {Definition of **RA**}
$= \mathbf{RA} \circ \mathbf{PBMH}(P)$ □

Theorem 68. $\mathbf{RA3} \circ \mathbf{RA1}(P) = \mathbf{RA3} \circ \mathbf{RA1}(P)$

Proof.

$\mathbf{RA1} \circ \mathbf{RA3}(P)$ {Definition of **RA3**}
$= \mathbf{RA1}(\mathbb{I}_{\mathcal{R}ac} \lhd s.wait \rhd P)$ {Lemma L.G.1.15 in [35]}
$= \mathbf{RA1}(\mathbb{I}_{\mathcal{R}ac}) \lhd s.wait \rhd \mathbf{RA1}(P)$ {Theorem T.G.3.1 in [35]}
$= \mathbb{I}_{\mathcal{R}ac} \lhd s.wait \rhd \mathbf{RA1}(P)$ {Definition of **RA3**}
$= \mathbf{RA3} \circ \mathbf{RA1}(P)$ □

Theorem 69. $\mathbf{RA2} \circ \mathbf{RA3}(P) = \mathbf{RA3} \circ \mathbf{RA2}(P)$

Proof.

$\mathbf{RA2} \circ \mathbf{RA3}(P)$ {Definition of **RA3**}
$= \mathbf{RA2}(\mathbb{I}_{\mathcal{R}ac} \lhd s.wait \rhd P)$ {Lemma L.G.2.6 in [35]}
$= \mathbf{RA2}(\mathbb{I}_{\mathcal{R}ac}) \lhd s.wait \rhd \mathbf{RA2}(P)$ {Theorem T.G.3.2 in [35]}
$= \mathbb{I}_{\mathcal{R}ac} \lhd s.wait \rhd \mathbf{RA2}(P)$ {Definition of **RA3**}
$= \mathbf{RA3} \circ \mathbf{RA2}(P)$ □

Lemma 19. $\mathbf{RA3}(P) = \mathbf{RA3}(P_f)$

Proof.

$\mathbf{RA3}(P)$ {Definition of **RA3**}
$= (\mathbb{I}_{\mathcal{R}ac} \lhd s.wait \rhd P)$ {Definition of conditional and predicate calculus}
$= (\mathbb{I}_{\mathcal{R}ac} \lhd s.wait \rhd (\neg\, s.wait \wedge P))$ {Predicate calculus}
$= (\mathbb{I}_{\mathcal{R}ac} \lhd s.wait \rhd (s.wait = false \wedge P))$ {Leibniz substitution}
$= (\mathbb{I}_{\mathcal{R}ac} \lhd s.wait \rhd (s.wait = false \wedge P[s \oplus \{wait \mapsto false\}/s]))$ {Predicate calculus}
$= (\mathbb{I}_{\mathcal{R}ac} \lhd s.wait \rhd P[s \oplus \{wait \mapsto false\}/s])$ {Definition of **RA3**}
$= \mathbf{RA3}(P[s \oplus \{wait \mapsto false\}/s])$ {Substitution abbreviation}
$= \mathbf{RA3}(P_f)$ □

Lemma 20. $(\mathbf{RA} \circ \mathbf{A}(\neg\, P_f^f \vdash P_f^t))_f^f = \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(\neg\, ok \vee P_f^f)$

Proof.

$(\mathbf{RA} \circ \mathbf{A}(\neg\, P_f^f \vdash P_f^t))_f^f$ {Lemma L.G.4.7 in [35]}
$= \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(\neg\, ok \vee P_f^f \vee (P_f^t \wedge false))$ {Predicate calculus}
$= \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(\neg\, ok \vee P_f^f)$ □

Theorem 70. $\mathbf{PBMH} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P) = \mathbf{RA1} \circ \mathbf{PBMH}(P)$

Proof.

$$
\begin{array}{ll}
\mathbf{PBMH} \circ \mathbf{RA1} \circ \mathbf{PBMH}(P) & \{\text{Defnition of } \mathbf{RA1}\} \\
= \mathbf{PBMH} \left(\begin{array}{l} \mathbf{PBMH}(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' \end{array} \right) & \{\text{Definition of } \mathbf{PBMH}\} \\
= \mathbf{PBMH} \left(\begin{array}{l} (\exists\, ac_0 \bullet P[ac_0/ac'] \land ac_0 \subseteq ac')[\{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' \end{array} \right) & \{\text{Substitution}\} \\
= \mathbf{PBMH} \left(\begin{array}{l} (\exists\, ac_0 \bullet P[ac_0/ac'] \land ac_0 \subseteq \{z \mid z \in ac' \land s.tr \leq z.tr\}) \\ \land \\ \exists\, z \bullet s.tr \leq z.tr \land z \in ac' \end{array} \right) & \{\text{Property of sets}\} \\
= \mathbf{PBMH} \left(\begin{array}{l} (\exists\, ac_0 \bullet P[ac_0/ac'] \land \forall\, x \bullet x \in ac_0 \Rightarrow (x \in ac' \land s.tr \leq x.tr)) \\ \land \\ \exists\, z \bullet s.tr \leq z.tr \land z \in ac' \end{array} \right) & \{\text{Predicate calculus}\} \\
= \mathbf{PBMH} \left(\begin{array}{l} \exists\, ac_0 \bullet \left(\begin{array}{l} P[ac_0/ac'] \land (\forall\, x \bullet x \in ac_0 \Rightarrow x \in ac') \\ \land \\ (\forall\, x \bullet x \in ac_0 \Rightarrow s.tr \leq x.tr) \end{array} \right) \land \\ \exists\, z \bullet s.tr \leq z.tr \land z \in ac' \end{array} \right) & \{\text{Definition of } \mathbf{PBMH}\} \\
= \exists\, ac_1 \bullet \left(\begin{array}{l} \exists\, ac_0 \bullet \left(\begin{array}{l} P[ac_0/ac'] \land (\forall\, x \bullet x \in ac_0 \Rightarrow x \in ac') \\ \land \\ (\forall\, x \bullet x \in ac_0 \Rightarrow s.tr \leq x.tr) \end{array} \right) \land \\ \exists\, z \bullet s.tr \leq z.tr \land z \in ac' \end{array} \right) [ac_1/ac'] \land ac_1 \subseteq ac' & \\
& \{\text{Substitution}\} \\
= \exists\, ac_1 \bullet \left(\begin{array}{l} \exists\, ac_0 \bullet \left(\begin{array}{l} P[ac_0/ac'] \land (\forall\, x \bullet x \in ac_0 \Rightarrow x \in ac_1) \\ \land \\ (\forall\, x \bullet x \in ac_0 \Rightarrow s.tr \leq x.tr) \end{array} \right) \\ \land \\ \exists\, z \bullet s.tr \leq z.tr \land z \in ac_1 \end{array} \right) \land ac_1 \subseteq ac' & \{\text{Predicate calculus}\} \\
= \left(\begin{array}{l} \exists\, ac_0 \bullet \left(\begin{array}{l} P[ac_0/ac'] \land (\forall\, x \bullet x \in ac_0 \Rightarrow x \in ac') \\ \land \\ (\forall\, x \bullet x \in ac_0 \Rightarrow s.tr \leq x.tr) \end{array} \right) \\ \land \\ \exists\, z \bullet s.tr \leq z.tr \land z \in ac' \end{array} \right) & \{\text{Predicate calculus}\} \\
= \left(\begin{array}{l} \exists\, ac_0 \bullet P[ac_0/ac'] \land (\forall\, x \bullet x \in ac_0 \Rightarrow (x \in ac' \land s.tr \leq x.tr)) \\ \land \\ \exists\, z \bullet s.tr \leq z.tr \land z \in ac' \end{array} \right) & \{\text{Property of sets}\} \\
= \left(\begin{array}{l} \exists\, ac_0 \bullet P[ac_0/ac'] \land ac_0 \subseteq \{z \mid z \in ac' \land s.tr \leq z.tr\} \\ \land \\ \exists\, z \bullet s.tr \leq z.tr \land z \in ac' \end{array} \right) & \{\text{Substitution}\} \\
= \left(\begin{array}{l} (\exists\, ac_0 \bullet P[ac_0/ac'] \land ac_0 \subseteq ac')[\{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' \end{array} \right) & \{\text{Definition of } \mathbf{PBMH}\} \\
= \mathbf{PBMH}(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' & \{\text{Definition of } \mathbf{RA1}\} \\
= \mathbf{RA1} \circ \mathbf{PBMH}(P) & \square
\end{array}
$$

Theorem 71. $\mathbf{RA2} \circ \mathbf{RA1}(P) = \mathbf{RA1} \circ \mathbf{RA2}(P)$

Proof.

$$
\begin{array}{ll}
& \mathbf{RA2} \circ \mathbf{RA1}(P) \qquad \{\text{Definition of } \mathbf{RA2}\} \\
= & \mathbf{RA1}(P)[s \oplus \{tr \mapsto \langle\rangle\}, \{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\}/s, ac'] \\
& \qquad \{\text{Definition of } \mathbf{RA1}\} \\
= & \left(P \wedge ac' \neq \emptyset\right)[\{z \mid z \in ac' \wedge s.tr \leq z.tr\}/ac'] \\
& [s \oplus \{tr \mapsto \langle\rangle\}, \{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\}/s, ac'] \\
& \qquad \{\text{Substitution of } s\} \\
= & \left(\begin{array}{l} P[s \oplus \{tr \mapsto \langle\rangle\}/s] \wedge \\ ac' \neq \emptyset \end{array}\right) \begin{array}{l}[\{z \mid z \in ac' \wedge (s \oplus \{tr \mapsto \langle\rangle\}).tr \leq z.tr\}/ac'] \\ [\{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\}/ac']\end{array} \\
& \qquad \{\text{Value of state component } tr\} \\
= & \left(\begin{array}{l} P[s \oplus \{tr \mapsto \langle\rangle\}/s] \wedge \\ ac' \neq \emptyset \end{array}\right) \begin{array}{l}[\{z \mid z \in ac' \wedge \langle\rangle \leq z.tr\}/ac'] \\ [\{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\}/ac']\end{array} \\
& \qquad \{\text{Property of sequence prefixing}\} \\
= & \left(\begin{array}{l} P[s \oplus \{tr \mapsto \langle\rangle\}/s] \wedge \\ ac' \neq \emptyset \end{array}\right) \begin{array}{l}[\{z \mid z \in ac'\}/ac'] \\ [\{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\}/ac']\end{array} \\
& \qquad \{\text{Property of sets}\} \\
= & \left(\begin{array}{l} P[s \oplus \{tr \mapsto \langle\rangle\}/s] \wedge \\ ac' \neq \emptyset \end{array}\right) \begin{array}{l}[ac'/ac'] \\ [\{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\}/ac']\end{array} \\
& \qquad \{\text{Property of substitution}\} \\
= & \left(\ P[s \oplus \{tr \mapsto \langle\rangle\}/s] \wedge ac' \neq \emptyset\ \right)[\{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\}/ac'] \\
& \qquad \{\text{Substitution}\} \\
= & \left(\begin{array}{l} P[s \oplus \{tr \mapsto \langle\rangle\}/s][\{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\}/ac'] \\ \wedge \\ \{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\} \neq \emptyset \end{array}\right) \qquad \{\text{Property of sets}\} \\
= & \left(\begin{array}{l} P[s \oplus \{tr \mapsto \langle\rangle\}/s][\{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\}/ac'] \\ \wedge \\ \exists\, y, z \bullet z \in ac' \wedge s.tr \leq z.tr \wedge y = z \oplus \{tr \mapsto z.tr - s.tr\} \end{array}\right) \qquad \{\text{One-point rule}\} \\
= & \left(\begin{array}{l} P[s \oplus \{tr \mapsto \langle\rangle\}/s][\{z \mid z \in ac' \wedge s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\}\}/ac'] \\ \wedge \\ \exists\, z \bullet z \in ac' \wedge s.tr \leq z.tr \end{array}\right) \qquad \{\text{Property of sets}\} \\
= & \left(\begin{array}{l} P[s \oplus \{tr \mapsto \langle\rangle\}/s]\left[\left\{\ z \;\middle|\; \begin{array}{l} z \in \{z \mid z \in ac' \wedge s.tr \leq z.tr\} \\ \wedge\ s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\} \end{array}\ \right\}/ac'\right] \\ \wedge \\ \exists\, z \bullet z \in ac' \wedge s.tr \leq z.tr \end{array}\right) \\
& \qquad \{\text{Property of substitution}\} \\
= & \left(\begin{array}{l} P[s \oplus \{tr \mapsto \langle\rangle\}/s]\left[\left\{\ z \;\middle|\; \begin{array}{l} z \in ac' \\ \wedge\ s.tr \leq z.tr \bullet z \oplus \{tr \mapsto z.tr - s.tr\} \end{array}\ \right\}/ac'\right] \\ \qquad\qquad [\{z \mid z \in ac' \wedge s.tr \leq z.tr\}/ac'] \\ \wedge \\ \exists\, z \bullet z \in ac' \wedge s.tr \leq z.tr \end{array}\right) \\
& \qquad \{\text{Definition of } \mathbf{RA2}\} \\
= & (\mathbf{RA2}(P)[\{z \mid z \in ac' \wedge s.tr \leq z.tr\}/ac'] \wedge \exists\, z \bullet z \in ac' \wedge s.tr \leq z.tr) \qquad \{\text{Definition of } \mathbf{RA1}\} \\
= & \mathbf{RA1} \circ \mathbf{RA2}(P)
\end{array}
$$

□

Lemma 21. $(\mathbf{RA} \circ \mathbf{A}(\neg\ P_f^f \vdash P_f^t))_f^t = \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(\neg\ ok \vee P_f^f \vee P_f^t)$

Proof.

$(\mathbf{RA} \circ \mathbf{A}(\neg\ P_f^f \vdash P_f^t))_f^t$ {Lemma L.G.4.7 in [35]}
$= \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(\neg\ ok \vee P_f^f \vee (P_f^t \wedge true))$ {Predicate calculus}
$= \mathbf{RA2} \circ \mathbf{RA1} \circ \mathbf{PBMH}(\neg\ ok \vee P_f^f \vee P_f^t)$ □

Lemma 22. $\mathbf{RA}(P \vdash Q) = \mathbf{RA}(P \vdash \mathbf{RA2} \circ \mathbf{RA1}(Q))$

Proof.

$\mathbf{RA}(P \vdash Q)$ {Definition of **RA**}
$= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1}(P \vdash Q)$ {Lemma L.G.1.20 in [35]}
$= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1}(P \vdash \mathbf{RA1}(Q))$ {Theorem 71}
$= \mathbf{RA3} \circ \mathbf{RA1} \circ \mathbf{RA2}(P \vdash \mathbf{RA1}(Q))$ {Lemma L.G.2.16 in [35]}
$= \mathbf{RA3} \circ \mathbf{RA1} \circ \mathbf{RA2}(P \vdash \mathbf{RA2} \circ \mathbf{RA1}(Q))$ {Theorem 71}
$= \mathbf{RA3} \circ \mathbf{RA2} \circ \mathbf{RA1}(P \vdash \mathbf{RA2} \circ \mathbf{RA1}(Q))$ {Definition of **RA**}
$= \mathbf{RA}(P \vdash \mathbf{RA2} \circ \mathbf{RA1}(Q))$ □

Lemma 23. $\mathbf{RA1}(P \vdash Q) = \mathbf{RA1}(\neg\ \mathbf{RA1}(\neg\ P) \vdash Q)$

Proof.

$\mathbf{RA1}(P \vdash Q)$ {Definition of design}
$= \mathbf{RA1}((ok \wedge P) \Rightarrow (Q \wedge ok'))$ {Predicate calculus}
$= \mathbf{RA1}(\neg\ ok \vee \neg\ P \vee (Q \wedge ok'))$ {Distributivity of **RA1**}
$= \mathbf{RA1}(\neg\ ok) \vee \mathbf{RA1}(\neg\ P) \vee \mathbf{RA1}(Q \wedge ok')$ {**RA1** is idempotent}
$= \mathbf{RA1}(\neg\ ok) \vee \mathbf{RA1} \circ \mathbf{RA1}(\neg\ P) \vee \mathbf{RA1}(Q \wedge ok')$ {Distributivity of **RA1**}
$= \mathbf{RA1}(\neg\ ok \vee \mathbf{RA1}(\neg\ P) \vee (Q \wedge ok'))$ {Predicate calculus}
$= \mathbf{RA1}((ok \wedge \neg\ \mathbf{RA1}(\neg\ P)) \Rightarrow (Q \wedge ok'))$ {Definition of design}
$= \mathbf{RA1}(\neg\ \mathbf{RA1}(\neg\ P) \vdash Q)$ □

Appendix A.3. Linking

Theorem 72. $ac' \neq \emptyset \wedge p2ac(\neg\ P^f \vdash P^t) = ac' \neq \emptyset \wedge (\neg\ p2ac(P^f) \vdash p2ac(P^t))$

Proof.

$ac' \neq \emptyset \wedge p2ac(\neg\ P^f \vdash P^t)$ {Definition of design}
$= ac' \neq \emptyset \wedge p2ac((ok \wedge \neg\ P^f) \Rightarrow (P^t \wedge ok'))$ {Predicate calculus}
$= ac' \neq \emptyset \wedge p2ac(\neg\ ok \vee P^f \vee (P^t \wedge ok'))$ {Distributivity of $p2ac$ (Theorem T.4.6.1 in [35])}
$= ac' \neq \emptyset \wedge (p2ac(\neg\ ok) \vee p2ac(P^f) \vee p2ac(P^t \wedge ok'))$ {Lemmas L.C.5.5 and L.C.5.6 in [35]}
$= ac' \neq \emptyset \wedge ((\neg\ ok \wedge ac' \neq \emptyset) \vee p2ac(P^f) \vee (p2ac(P^t) \wedge ok'))$ {Predicate calculus}
$= ac' \neq \emptyset \wedge (\neg\ ok \vee p2ac(P^f) \vee (p2ac(P^t) \wedge ok'))$ {Predicate calculus}
$= ac' \neq \emptyset \wedge ((ok \wedge \neg\ p2ac(P^f)) \Rightarrow (p2ac(P^t) \wedge ok'))$ {Definition of design}
$= ac' \neq \emptyset \wedge (\neg\ p2ac(P^f) \vdash p2ac(P^t))$

□

Lemma 24. *Provided P is a design, $ac2p(P) = (\neg\ ac2p(P^f) \vdash ac2p(P^t))$.*

Proof.

$$
\begin{array}{ll}
ac2p(P) & \{\text{Assumption: } P \text{ is a design}\} \\
= ac2p(\neg\ P^f \vdash P^t) & \{\text{Definition of design}\} \\
= ac2p((ok \wedge \neg\ P^f) \Rightarrow (P^t \wedge ok')) & \{\text{Predicate calculus and distributivity of } ac2p\} \\
= ac2p(\neg\ ok) \vee ac2p(P^f) \vee ac2p(P^t \wedge ok') & \{ac' \text{ is not free}\} \\
= \neg\ ok \vee ac2p(P^f) \vee (ac2p(P^t) \wedge ok') & \{\text{Predicate calculus}\} \\
= (ok \wedge \neg\ ac2p(P^f)) \Rightarrow (ac2p(P^t) \wedge ok') & \{\text{Definition of design}\} \\
= (\neg\ ac2p(P^f) \vdash ac2p(P^t)) & \square
\end{array}
$$

Lemma 25. $ac2p \circ \mathbf{PBMH}(P) = ac2p(P)$

Proof.

$$
\begin{array}{ll}
ac2p \circ \mathbf{PBMH}(P) & \{\text{Definition of } ac2p\} \\
= \mathbf{PBMH}(\mathbf{PBMH}(P))[State_{\amalg}(in\alpha)/s]\ ;_{\mathcal{A}}\ \bigwedge x' : out\alpha \bullet s.x = x' & \{\mathbf{PBMH}\text{-idempotent}\} \\
= \mathbf{PBMH}(P)[State_{\amalg}(in\alpha)/s]\ ;_{\mathcal{A}}\ \bigwedge x' : out\alpha \bullet s.x = x' & \{\text{Definition of } ac2p\} \\
= ac2p(P) & \square
\end{array}
$$

Lemma 26. $\mathbf{PBMH} \circ p2ac(P) = p2ac(P)$

Proof.

$$
\begin{array}{ll}
\mathbf{PBMH} \circ p2ac(P) & \{\text{Definition of } \mathbf{PBMH}\} \\
= \exists\, ac_0 \bullet p2ac(P)[ac_0/ac'] \wedge ac_0 \subseteq ac' & \{\text{Definition of } p2ac\} \\
= \exists\, ac_0 \bullet (\exists\, z \bullet P[\mathbf{s}, \mathbf{z}'/in\alpha_{-ok}, out\alpha_{-ok'}] \wedge z \in ac')[ac_0/ac'] \wedge ac_0 \subseteq ac' & \{\text{Substitution}\} \\
= \exists\, ac_0 \bullet (\exists\, z \bullet P[\mathbf{s}, \mathbf{z}'/in\alpha_{-ok}, out\alpha_{-ok'}] \wedge z \in ac_0) \wedge ac_0 \subseteq ac' & \{\text{Property of sets}\} \\
= \exists\, z \bullet P[\mathbf{s}, \mathbf{z}'/in\alpha_{-ok}, out\alpha_{-ok'}] \wedge z \in ac' & \{\text{Definition of } p2ac\} \\
= p2ac(P) & \square
\end{array}
$$

Appendix A.4. State Substitution Lemmas

Lemma 27. *Provided z is not free in P, $P[\mathbf{z}/S\alpha][State_{\amalg}(S\alpha)/z] = P$.*

Proof.

$$
\begin{array}{ll}
P[\mathbf{z}/S\alpha][State_{\amalg}(S\alpha)/z] & \{\text{Definition of state substitution}\} \\
= P[z.x_0, \ldots, z.x_n/x_0, \ldots, x_n][State_{\amalg}(S\alpha)/z] & \{\text{Definition of } State_{\amalg}(S\alpha)\} \\
= P[z.x_0, \ldots, z.x_n/x_0, \ldots, x_n][\{x_0 \mapsto x_0, \ldots, x_n \mapsto x_n\}/z] & \{z \text{ is not free in } P\} \\
= P[\{x_0 \mapsto x_0, \ldots, x_n \mapsto x_n\}.x_0, \ldots, \{x_0 \mapsto x_0, \ldots, x_n \mapsto x_n\}.x_n/x_0, \ldots, x_n] & \{\text{Value of state component}\} \\
= P[x_0, \ldots, x_n/x_0, \ldots, x_n] & \{\text{Property of substitution}\} \\
= P & \square
\end{array}
$$

Lemma 28. $\exists\, z : State(S\alpha) \bullet P \wedge (\bigwedge x : S\alpha \bullet z.x = x) = P[State_{\Pi}(S\alpha)/z]$

PROOF.

$$
\begin{aligned}
&\exists\, z : State(S\alpha) \bullet P \wedge (\textstyle\bigwedge x : S\alpha \bullet z.x = x) && \{\text{Equality of records}\}\\
&= \exists\, z : State(S\alpha) \bullet P \wedge z = \{x_0 \mapsto x_0, \ldots, x_n \mapsto x_n\} && \{\text{Definition of } State_{\Pi}\}\\
&= \exists\, z : State(S\alpha) \bullet P \wedge State_{\Pi}(S\alpha) = z && \{\text{One-point rule}\}\\
&= P[State_{\Pi}(S\alpha)/z] && \square
\end{aligned}
$$

Acknowledgments

The authors are grateful to the UK EPSRC for funding this work, and to Andrew Butterfield, Detlef Plump, and Steve Schneider for their comments on Pedro Ribeiro's PhD thesis whose results are described here.

References

[1] E. W. Dijkstra, Guarded commands, nondeterminacy and formal derivation of programs, Commun. ACM 18 (1975) 453–457.
[2] A. W. Roscoe, Understanding concurrent systems, Springer, 2010.
[3] C. Morgan, Programming from specifications, Prentice Hall, 1994.
[4] R. Back, J. Wright, Refinement calculus: a systematic introduction, Graduate texts in computer science, Springer, 1998.
[5] J. M. Morris, A theoretical basis for stepwise refinement and the programming calculus, Sci. Comput. Program. 9 (1987) 287–306.
[6] C. Morgan, P. Gardiner, Data refinement by calculation, Acta Informatica 27 (1990) 481–503.
[7] M. Tyrrell, J. Morris, A. Butterfield, A. Hughes, A Lattice-Theoretic Model for an Algebra of Communicating Sequential Processes, in: K. Barkaoui, A. Cavalcanti, A. Cerone (Eds.), Theoretical Aspects of Computing - ICTAC 2006, volume 4281 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2006, pp. 123–137. doi:10.1007/11921240_9.
[8] A. W. Roscoe, The Theory and Practice of Concurrency, Prentice Hall, 1998.
[9] M. Oliveira, Formal Derivation of State-Rich Reactive Programs using *Circus*, Ph.D. thesis, University of York, 2005. URL: https://www.cs.york.ac.uk/circus/publications/papers/06-oliveira.pdf.
[10] C. A. R. Hoare, J. He, Unifying Theories of Programming, Prentice Hall International Series in Computer Science, 1998.
[11] I. Rewitzky, Binary Multirelations, in: H. de Swart, E. Orlowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments, volume 2929 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2003, pp. 1964–1964. doi:10.1007/978-3-540-24615-2_12.
[12] A. Cavalcanti, J. Woodcock, S. Dunne, Angelic nondeterminism in the unifying theories of programming, Formal Aspects of Computing 18 (2006) 288–307.
[13] M. O. Rabin, D. Scott, Finite Automata and Their Decision Problems, IBM J. Res. Dev. 3 (1959) 114–125.
[14] S. A. Cook, The Complexity of Theorem-proving Procedures, in: Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC '71, ACM, New York, NY, USA, 1971, pp. 151–158. doi:10.1145/800157.805047.
[15] M. Schützenberger, On context-free languages and push-down automata, Information and Control 6 (1963) 246 – 264.
[16] W. H. Hesselink, LR-parsing derived, Science of Computer Programming 19 (1992) 171 – 196.
[17] A. P. Martin, P. H. B. Gardiner, J. C. P. Woodcock, A tactic calculus - abridged version, Formal Aspects of Computing 8 (1996) 479–489. 10.1007/BF01213535.
[18] M. Oliveira, A. Cavalcanti, J. Woodcock, ArcAngel: a Tactic Language for Refinement, Formal Aspects of Computing 15 (2003) 28–47.
[19] R. Jagadeesan, V. A. Saraswat, V. Shanbhogue, Angelic non-determinism in concurrent constraint programming, Technical Report, Xerox Park, 1991.
[20] J. N. Kok, On Logic Programming and the Refinement Calculus: Semantics Based Program Transformations, Technical Report RUU-CS-90-39, Utrecht University, 1990.
[21] R. W. Floyd, Nondeterministic Algorithms, J. ACM 14 (1967) 636–644.
[22] E. W. Dijkstra, A Discipline of Programming, 1st ed., Prentice Hall, Upper Saddle River, NJ, USA, 1976.
[23] R.-J. Back, On the correctness of refinement in program development, Ph.D. thesis, Department of Computer Science, University of Helsinki, 1978.
[24] P. Gardiner, C. Morgan, Data refinement of predicate transformers, Theoretical Computer Science 87 (1991) 143 – 162.
[25] N. Ward, I. Hayes, Applications of Angelic Nondeterminism, in: P. A. Bailes (Ed.), Australian Software Engineering Conference 1991: Engineering Safe Software; Proceedings, N.S.W.: Australian Computer Society, Sydney, 1991, pp. 391–404.

[26] A. Cavalcanti, A. Mota, J. Woodcock, Simulink Timed Models for Program Verification, in: Z. Liu, J. Woodcock, H. Zhu (Eds.), Theories of Programming and Formal Methods, volume 8051 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2013, pp. 82–99. doi:10.1007/978-3-642-39698-4_6.
[27] J. Woodcock, J. Davies, Using Z: Specification, Refinement, and Proof, Prentice Hall, 1996.
[28] C. B. Jones, Systematic software development using VDM, Prentice Hall International, 1986.
[29] A. Cavalcanti, J. Woodcock, Angelic Nondeterminism and Unifying Theories of Programming, Technical Report, University of Kent, 2004. URL: http://kar.kent.ac.uk/14151/.
[30] P. Ribeiro, Relational CSP, 2017. URL: https://perma.cc/DR5D-A6TE.
[31] J. Woodcock, J. Bryans, S. Canham, S. Foster, The COMPASS Modelling Language: Timed Semantics in UTP, Communicating Process Architectures (2014).
[32] W. Harwood, A. Cavalcanti, J. Woodcock, A Theory of Pointers for the UTP, in: J. Fitzgerald, A. Haxthausen, H. Yenigun (Eds.), Theoretical Aspects of Computing - ICTAC 2008, volume 5160 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2008, pp. 141–155. doi:10.1007/978-3-540-85762-4_10.
[33] B. Davey, H. Priestley, Introduction to Lattices and Order, Cambridge mathematical textbooks, Cambridge University Press, 2002.
[34] J. Woodcock, A. Cavalcanti, A Tutorial Introduction to Designs in Unifying Theories of Programming, in: E. Boiten, J. Derrick, G. Smith (Eds.), Integrated Formal Methods, volume 2999 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2004, pp. 40–66. doi:10.1007/978-3-540-24756-2_4.
[35] P. Ribeiro, Angelic Processes, Ph.D. dissertation (extended version), University of York, 2014. URL: http://arxiv.org/abs/1505.04726.
[36] P. Ribeiro, A. Cavalcanti, UTP Designs for Binary Multirelations, in: G. Ciobanu, D. Méry (Eds.), Theoretical Aspects of Computing ICTAC 2014, volume 8687 of *Lecture Notes in Computer Science*, Springer International Publishing, 2014, pp. 388–405. doi:10.1007/978-3-319-10882-7_23.
[37] J. Woodcock, The Miracle of Reactive Programming, in: A. Butterfield (Ed.), Unifying Theories of Programming, volume 5713 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2010, pp. 202–217. doi:10.1007/978-3-642-14521-6_12.
[38] K. Wei, J. Woodcock, 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, 2010, pp. 315–319. doi:10.1109/SEFM.2010.40.
[39] K. Wei, J. Woodcock, A. Burns, Timed *Circus*: Timed CSP with the Miracle, in: Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference on, 2011, pp. 55–64. doi:10.1109/ICECCS.2011.13.
[40] B. Aman, G. Ciobanu, Real-time migration properties of rTiMo verified in UPPAAL, in: Proceedings of the 11th International Conference on Software Engineering and Formal Methods - Volume 8137, SEFM 2013, Springer-Verlag, Berlin, Heidelberg, 2013, pp. 31–45. doi:10.1007/978-3-642-40561-7_3.
[41] J. Sun, Y. Liu, J. S. Dong, C. Chen, Integrating specification and programs for system modeling and verification, in: 2009 Third IEEE International Symposium on Theoretical Aspects of Software Engineering, 2009, pp. 127–135. doi:10.1109/TASE.2009.32.
[42] W. Xie, S. Xiang, H. Zhu, A UTP approach for rTiMo, Formal Aspects of Computing (2018).
[43] L. Shi, Y. Zhao, Y. Liu, J. Sun, J. S. Dong, S. Qin, A UTP semantics for communicating processes with shared variables and its formal encoding in PVS, Formal Aspects of Computing 30 (2018) 351–380.
[44] H. Zhu, J. He, S. Qin, P. J. Brooke, Denotational semantics and its algebraic derivation for an event-driven system-level language, Formal Aspects of Computing 27 (2015) 133–166.
[45] A. Cavalcanti, J. Woodcock, A Tutorial Introduction to CSP in *Unifying Theories of Programming*, in: A. Cavalcanti, A. Sampaio, J. Woodcock (Eds.), Refinement Techniques in Software Engineering, volume 3167 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2006, pp. 220–268. doi:10.1007/11889229_6.
[46] S. Foster, F. Zeyda, J. Woodcock, Isabelle/UTP: A Mechanised Theory Engineering Framework, in: D. Naumann (Ed.), Unifying Theories of Programming, volume 8963 of *Lecture Notes in Computer Science*, Springer International Publishing, 2015, pp. 21–41. doi:10.1007/978-3-319-14806-9_2.
[47] A. Feliachi, M.-C. Gaudel, B. Wolff, Unifying Theories in Isabelle/HOL, in: S. Qin (Ed.), Unifying Theories of Programming, volume 6445 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2010, pp. 188–206. doi:10.1007/978-3-642-16690-7_9.
[48] F. Zeyda, A. Cavalcanti, Encoding *Circus* Programs in ProofPowerZ, in: A. Butterfield (Ed.), Unifying Theories of Programming, volume 5713 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2010, pp. 218–237. doi:10.1007/978-3-642-14521-6_13.
[49] A. Butterfield, The Logic of $U{\cdot}(TP)^2$, in: B. Wolff, M.-C. Gaudel, A. Feliachi (Eds.), Unifying Theories of Programming, volume 7681 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2013, pp. 124–143. doi:10.1007/978-3-642-35705-3_6.