Algebras for correctness of sequential computations

Walter Guttmann

Department of Computer Science and Software Engineering, University of Canterbury, New Zealand
walter.guttmann@canterbury.ac.nz

Abstract

Previous work gives algebras for uniformly describing correctness statements and calculi in various relational and matrix-based computation models. These models support a single kind of non-determinism, which is either angelic, demonic or erratic with respect to the infinite executions of a computation. Other models, notably isotone predicate transformers or up-closed multirelations, offer both angelic and demonic choice with respect to finite executions. We propose algebras for a theory of correctness which covers these multirelational models in addition to relational and matrix-based models. Existing algebraic descriptions, in particular general refinement algebras and monotonic Boolean transformers, are instances of our theory. Our new description includes a precondition operation that instantiates to both modal diamond and modal box operators. We verify all results in Isabelle, heavily using its automated theorem provers. We integrate our theories with the Isabelle theory of monotonic Boolean transformers making our results applicable to that setting.

Keywords: algebraic structures, axiomatic semantics, Conway semirings, Hoare calculus, modal operators, multirelations, preconditions, program semantics, sequential computations

1. Introduction

Computation models establish the setting for reasoning about programs by providing the semantics of programs, specifications, correctness statements, correctness calculi and related ingredients. The models differ in the kinds of computations they can represent and the precision they achieve, for example, covering deterministic/non-deterministic, sequential/concurrent, strict/lazy computations, angelic/demonic choice and finite/infinite/aborting executions. Reasoning in a given model is limited to the computations and precision that can be represented, yet less detailed models might be better in terms of comprehension or automation.

Algebras structure this diversity of models to avoid the repeated development of similar theories. Elements of the carriers of the algebras are computations representing programs and specifications. Operations of the algebras are program constructs. Axioms of the algebras are the laws of programming. Derived theorems state program transformations, refinements and correctness statements. Computation models are characterised by the operations they support and the axioms these operations satisfy.

In this algebraic setting, a correctness statement is an equation in the underlying algebra. Our previous work [21] shows that this is unifying in two dimensions: a given equation may express different correctness statements both in the same computation model and across different computation models, depending on how the operations are instantiated. That work covers non-deterministic, sequential, strict computations with relational or matrix-based models that distinguish finite, infinite and aborting executions with varying precision. These models can represent a single kind of non-determinism, which is either angelic, demonic or erratic with respect to the infinite executions of a computation.

The purpose of the present paper is to extend the algebraic approach to correctness so that it covers models that support both angelic and demonic choice with respect to finite executions [3, 4]. Such models are useful for modelling contracts between agents, interaction, games and protocols [2, 35] besides being of theoretical interest [10]. Due to the increased expressivity, computations are no longer represented by

relations but by multirelations or isotone predicate transformers. At the algebraic level, these structures fail to satisfy left distributivity of sequential composition over non-deterministic choice, but sequential composition is still isotone. This weakening of distributive operations to isotone operations has been studied for idempotent semirings, Kleene algebras, omega algebras and their extensions by a domain operation [39] as well as demonic refinement algebras [48]. We therefore investigate whether the algebraic structures used to describe correctness in [21] can be generalised accordingly in order to support correctness reasoning in multirelational models. Unlike the matrix-based models, the multirelational models covered in this paper do not distinguish infinite and aborting executions.

The contributions of the present paper are:

* Generalised algebras for correctness reasoning – in particular, relative domain semirings, Conway semirings and precondition algebras – that capture multirelational models in addition to relational and matrix-based models.
* Axioms for preconditions which instantiate to both modal diamond and modal box operators.
* A sound and relatively complete, propositional correctness calculus that uniformly covers multirelational computation models in addition to relational and matrix-based models, and different kinds of correctness statements in each model.
* Instances of the above theories for general refinement algebras [48] and monotonic Boolean transformers [45]. This makes all our results available in these algebras and involves a substantial extension of the Isabelle theory of monotonic Boolean transformers by assumptions and continuity.

Together we obtain a framework that unifies correctness reasoning for various multirelational, relational and matrix-based computation models and various correctness claims.

All results are verified in Isabelle making heavy use of its integrated automated theorem provers. The proofs can be found in the theory files, which are available at http://www.csse.canterbury.ac.nz/walter.guttmann/algebra/.

This paper is structured by the abstraction level at which computation models are discussed. Section 2 is the most concrete and motivates the subsequent development by a selection of relational, matrix-based and multirelational models of sequential computations. Our contributions follow in Sections 3–5, starting at the most abstract level and getting more and more concrete. Section 3 describes the bare minimum required for correctness reasoning: algebras for tests, preconditions, their effect on while-programs and correctness statements. We give a propositional correctness calculus and prove its soundness and relative completeness. Section 4 instantiates the algebras of Section 3 by modal semirings and a generalisation of Conway semirings that omits left distributivity of sequential composition. This intermediate level provides additional expressiveness but is still general enough to capture all intended models. Section 5 instantiates the algebras of Section 4 by a wide variety of structures proposed in the literature for describing computations. We focus on monotonic Boolean transformers which capture up-closed multirelations and satisfy the axioms of our algebras in several ways. In this paper we use the name 'monotonic Boolean transformers' coined by [45], and take 'isotone' in other contexts; both adjectives mean order-preserving.

2. Relational, matrix-based and multirelational computation models

This section recalls a selection of relational, matrix-based and multirelational models of sequential computations. We focus on the multirelational model to point out the changes necessary in generalising our previous treatment of correctness. In particular, we leave out the detailed description of several matrix-based computation models and their various correctness statements given in [21].

Let A be the state space of a computation, which is given by the values the program variables can take. Our examples in this section have $A \subseteq \mathbb{N}$ which amounts to a single variable x with natural numbers as values.

2.1. A relational computation model

In our first model, a computation is a binary relation R over the state space A. A pair (x, x') in R specifies that there is an execution of R which starts in state x and terminates in x'. Hence x is the value of the variable before execution of R and x' is the value afterwards. More than one x' may be related to a given x, which amounts to non-determinism. For example, using $A = \mathbb{N}$, the relation R_1 given by

$$\begin{array}{lcl} 0 & \mapsto & \{0,3\} \\ 1 & \mapsto & \{2,4\} \\ 2 & \mapsto & \{4,5\} \\ 3 & \mapsto & \{6\} \\ 4 & \mapsto & \{7,8\} \\ & \ldots & \end{array}$$

models a computation that either adds 3 to its input x or multiplies it by 2. Thus non-deterministic choice is modelled by set union. Infinite executions are not represented in this simple model, but the absence of finite executions can be interpreted as non-termination, in which case the endless loop is the empty relation $\emptyset$. Because $\emptyset$ is a neutral element of set union, choice is angelic with respect to non-termination: the non-deterministic choice between the endless loop and any relation R is just R. Another consequence is that the typical correctness claim in this model is a partial-correctness claim. For sets of states p and q, the Hoare triple $p\{R\}q$ expresses that every execution of R which starts in a state in p and terminates, does so in a state in q [27]. Because the statement has to be proved for *every* execution of R, choice is demonic with respect to finite executions.

The Hoare triple $p\{R\}q$ is algebraically formalised by $p \cdot R \cdot q' \leq 0$ using tests p and q [34]. Tests represent subsets of the state space A and act as filters in a sequential composition. In the relational model, tests are subsets of the identity relation 1. The operation $'$ complements tests relative to 1, the operation $\cdot$ is relational composition, $\leq$ is subset and 0 is the empty relation. According to the inequality there is no execution in R which starts in p and ends in a state not in q.

The Hoare triple can be expressed as $p \leq |R]q$ using a modal box operator [40]. This expresses that if a state is in the subset represented by p, all executions of R from that state lead to a state in the subset represented by q. Thus box represents the weakest liberal precondition in this computation model; it appropriately signifies the universal quantification that takes into account every execution of R. The box operator is defined as $|R]q = d(R \cdot q')'$ in terms of the operation d that describes the domain of a computation [14]. In the relational model, domain is given by $d(R) = (R \cdot \top) \cap 1$ using the universal relation $\top = A \times A$. A relation of the form $R \cdot \top$ is a *vector*, that is, it relates every state either to all states or to none, and therefore corresponds to a set of states. Intersection with 1 represents this set as a test. More details about domain and box follow in Section 4.

Iteration is described using the Kleene star * in this computation model, which amounts to the reflexive transitive closure of a relation. Kleene algebras capture * as a least fixpoint by induction axioms and provide further properties useful for reasoning about programs. More details about operations for iteration follow in Section 4.

2.2. A matrix-based computation model

Our second model augments the first model in order to represent infinite executions. It is an abstraction of the 'designs' of the Unifying Theories of Programming [30]. As the relational model, it disregards the intermediate states in a computation. For infinite executions this means that only their presence or absence has to be recorded for each starting state. To this end, the model adds a component that represents the set of states from which infinite executions exist. Together with the finite executions this is conveniently described by a matrix. Thus, a computation is a 2×2 matrix whose entries are relations over A. The matrix has the form

$$R = \begin{pmatrix} \top & \top \\ W & X \end{pmatrix}$$

where $W \subseteq X$ and both entries in the top row are the universal relation. The relation W represents the states from which infinite executions exist; it is a vector. The entry X represents the finite executions of the computation in states where infinite executions do not exist; additionally, states where infinite executions do exist are related by X to all states because $W \subseteq X$. This means that the presence or absence of finite executions cannot be distinguished in the presence of infinite ones. Non-deterministic choice is modelled by componentwise union. Sequential composition is given by the matrix product, where relational composition and union play the roles of multiplication and addition of the components.

For example, consider the relation R_2 for $A = \mathbb{N}$ given by

$$\begin{array}{lcl} 0 & \mapsto & \mathbb{N} \\ 1 & \mapsto & \emptyset \\ 2 & \mapsto & \mathbb{N} \\ 3 & \mapsto & \emptyset \\ 4 & \mapsto & \mathbb{N} \\ & \dots & \end{array}$$

representing the set of states $\{0, 2, 4, \dots\}$ as a vector. Then the matrix

$$\begin{pmatrix} \top & \top \\ R_2 & R_1 \cup R_2 \end{pmatrix}$$

models a computation that either adds 3 to its input x or multiplies it by 2, when x is odd, but need not terminate when x is even. Thus R_2 describes the states from which termination is not guaranteed. This model does not distinguish between infinite and aborting executions.

The endless loop is the matrix with four $\top$ entries; infinite executions exist from each state. Because this matrix is an annihilator of componentwise union, choice is demonic with respect to non-termination. Another consequence is that this model supports total-correctness claims. The Hoare triple $p\{R\}q$ now expresses that every execution of R which starts in p terminates in q. Again this is formalised by $p \cdot R \cdot q' \leq 0$ [48]. The order $\leq$ is the subset relation lifted componentwise to matrices. The test p is represented just as a program by a 2×2 matrix of the above form, using $W = \emptyset$ and a subset of the identity relation as X; a similar representation is used for the test q'. The computation 0 with no executions is represented by the matrix with $W = X = \emptyset$.

Observe that the entries in the first row of the matrices in this computation model are fixed to $\top$. This propagates the information in the entry W appropriately through sequential composition. Other choices in these entries and in matrices of size 3×3 yield different computation models, in which combinations of finite, infinite and aborting executions can be represented with varying precision [21]. Several kinds of correctness claims can be stated in each of these models. Most claims take the form $p \cdot R \cdot q' \leq \mathsf{Z}$ for a constant Z depending on the claim and the model. This constant captures executions that are ignored by the correctness claim; for example, Z could contain all infinite executions or all aborting executions. Also this more general claim can be expressed as $p \leq |R]q$ using a relativised modal box operator [21]. Depending on the computation model, the box operator represents the weakest liberal precondition, the weakest precondition or other kinds of preconditions, which avoid aborting executions in addition to finite or infinite ones. Also depending on the model, various fixpoints are needed to describe iteration. These fixpoints are uniformly captured by simulation axioms instead of the induction axioms of Kleene algebras.

In the models we have considered so far, non-deterministic choices are made by a single agent. These choices are demonic in the sense that all executions must be taken into account to guarantee correctness.

2.3. A multirelational computation model

Our third model can represent computations which involve both angelic and demonic choices as regards finite executions. To achieve this, relations and their matrix-based extensions are replaced with up-closed multirelations or isotone predicate transformers [2, 35]. The latter two formalisms are isomorphic [2, 46, 26]; we use up-closed multirelations in the following.

A relation is a subset of the Cartesian product $A \times B$ for sets A and B. In contrast, a *multirelation* [46] is a subset of the Cartesian product $A \times \mathsf{P}B$ where $\mathsf{P}B$ is the powerset of B. Thus it maps an element of A to a set of subsets of B. A multirelation R is *up-closed* if $(x, X) \in R \wedge X \subseteq Y \Rightarrow (x, Y) \in R$ for each $x \in A$ and $X, Y \subseteq B$. Thus, if an element of A is related to a set X, it must be related to all supersets of X. The *dual* $R^\smile$ of a multirelation R is given by $(x, X) \in R^\smile \Leftrightarrow (x, B \setminus X) \notin R$.

A multirelation models a computation as follows; see also [47]. Consider a state $x \in A$ and the set of subsets Xs it is mapped to. The outer set structure of Xs represents an angelic choice: the 'angel' chooses a set $X \in Xs$. The inner set structure of Xs represents a demonic choice: the 'demon' subsequently chooses an element $x' \in X$ which is the next state.

For example, consider the multirelation R_3 for $A = B = \{0, 1, 2, 3, 4\}$ given by

$$\begin{array}{lcl} 0 & \mapsto & \{\{1,2\},\{1,3,4\}\} \\ 1 & \mapsto & \{\{1\},\{2\}\} \\ 2 & \mapsto & \{\{1,2\}\} \\ 3 & \mapsto & \{\emptyset\} \\ 4 & \mapsto & \emptyset \end{array}$$

It describes the following computation. In state 0 an angelic choice between two sets $\{1, 2\}$ and $\{1, 3, 4\}$ is made. If the angel chooses $\{1, 3, 4\}$ the demon chooses which of 1, 3 and 4 is the next state. If the angel chooses $\{1, 2\}$ the demon chooses one of 1 and 2 as the next state. State 1 has a purely angelic choice between states 1 and 2, because each inner set is a singleton set in which the demon's choice is fixed. State 2 has a purely demonic choice between states 1 and 2, because the outer set is a singleton set in which the angel's choice is fixed. In state 3 the computation fails to progress since the demon cannot choose from the empty set. In the game interpretation this means that the angel wins; in terms of refinement this means that any specification is satisfied. In state 4 the computation fails to progress since the angel cannot choose from the empty set. In the game interpretation this means that the demon wins; in terms of refinement this means that no specification is satisfied.

The multirelation R_3 is not up-closed, but can be extended to an up-closed multirelation R_4 by adding the required supersets as shown in

$$\begin{array}{lcl} 0 & \mapsto & \uparrow\{1,2\} \cup \uparrow\{1,3,4\} \\ 1 & \mapsto & \uparrow\{1\} \cup \uparrow\{2\} \\ 2 & \mapsto & \uparrow\{1,2\} \\ 3 & \mapsto & \mathsf{P}B \\ 4 & \mapsto & \emptyset \end{array}$$

using the upward closure $\uparrow X = \{Y \mid X \subseteq Y \subseteq B\}$.

Adding a superset Y of a set X to which x is related does not change the computational interpretation. This just increases the angelic choice by options which are not interesting for the angel because they subsequently allow more choices for the demon. For example, in R_4 the state 0 is related to $\{1, 2, 3\}$ as a result of the upward closure, but angelic choice prefers $\{1, 2\}$ so as to restrict demonic choice as much as possible.

A consequence of using up-closed multirelations is a nice interplay between the outer and the inner set structures. Forming the union of up-closed multirelations simultaneously increases angelic choice and decreases demonic choice, while intersection simultaneously decreases angelic choice and increases demonic choice. Union and intersection thus provide angelic and demonic choice, respectively, at the level of computations. Moreover, these operations are duals of each other. Up-closed multirelations form a bounded distributive lattice using union and intersection as the lattice operations, the empty multirelation as the least element and the universal multirelation $\top$ as the greatest. In general, they do not form a Boolean algebra.

Consider multirelations $R \subseteq A \times \mathsf{P}B$ and $S \subseteq B \times \mathsf{P}C$. Their sequential composition $R \mathbin{;} S$ contains (x, Z) if and only if there is a set Y such that $(x, Y) \in R$ and $(y, Z) \in S$ for each $y \in Y$ [46]. This involves both existential and universal quantification signifying the angelic and demonic choices that take place. Up-closed multirelations form a monoid using sequential composition and the set membership multirelation $\in$ as neutral element. Both the empty and the universal multirelation are left annihilators of sequential

composition, but neither is a right annihilator. Up-closed multirelations satisfy the following distributivity and semidistributivity properties:

$$(R \cup S)\,;T = (R\,;T) \cup (S\,;T) \qquad (R\,;S) \cup (R\,;T) \subseteq R\,;(S \cup T)$$
$$(R \cap S)\,;T = (R\,;T) \cap (S\,;T) \qquad (R\,;S) \cap (R\,;T) \supseteq R\,;(S \cap T)$$

Sequential composition from the left in general does not distribute over union or intersection. This is in contrast to relational and matrix-based computation models, in which sequential composition from both sides distributes over union. However, union, intersection and sequential composition are $\subseteq$-isotone. Therefore, up-closed multirelations form an idempotent left semiring [39].

To represent sets of states, we generalise vectors, tests and the domain operation to up-closed multirelations. In the following we look at the homogeneous case $A = B$, which means that computations do not change the state space. For vectors, we reuse the relational definition: a multirelation R is a vector if each state is related either to all sets of states or to none, algebraically $R = R\,;\top$. In the relational case, a test is a subset of the identity relation. Because of the upward closure, we cannot simply take subsets of the set membership multirelation. Instead a multirelation is a test if it is the intersection of a vector with $\in$. This means that a state x is related either to no set or to all sets containing x. As in the case of relations, the vectors form a Boolean algebra, and the tests form a Boolean algebra in which intersection and sequential composition coincide. Our definition of tests corresponds to the assertions of [45], which have a dual notion of assumptions. The difference is that, if an assertion relates a state to no set, the corresponding assumption relates that state to all sets. The domain of a multirelation R is given by the test $d(R) = (R\,;\top) \cap \in$. This operation satisfies the axioms for domain in weaker forms of semirings given by [39, 15] and thus a modal diamond operator can be defined as $|R\rangle q = d(R\,;q)$.

Because sequential composition of multirelations already involves universal quantification, correctness claims are formalised differently from relational models. Given tests p and q and an up-closed multirelation R, the Hoare triple $p\{R\}q$ is equivalent to $p \leq (R\;;\;q\;;\;\top) \cap \in$ if we translate the definition given in [45]. Using the domain operation, this is $p \leq d(R\;;\;q)$ or equivalently $p \leq |R\rangle q$. Taken point-wise, this amounts to 'angelic correctness' of [35], whose dual 'demonic correctness' is expressed by $p \leq |R]q$. Hence both diamond and box are useful for expressing correctness of multirelations. This should be contrasted with the relational and matrix-based models, where only $p \leq |R]q$ is used and diamond represents the pre-image operator. To unify relational, matrix-based and multirelational computation models, it is therefore helpful to have a precondition operator that instantiates to both the modal diamond and the modal box operators. In particular, we no longer assume that the precondition operator distributes over intersection in its second argument. Neither $|R_5\rangle(p\;;q) = |R_5\rangle p\;;\,|R_5\rangle q$ nor $|R_6](p\;;q) = |R_6]p\;;\,|R_6]q$ holds for the up-closed multirelations R_5, R_6, p, q where $A = B = \{0,1\}$ and

$$R_5 = \begin{pmatrix} 0 & \mapsto & \uparrow\{0\} \cup \uparrow\{1\} \\ 1 & \mapsto & \emptyset \end{pmatrix} \qquad p = \begin{pmatrix} 0 & \mapsto & \uparrow\{0\} \\ 1 & \mapsto & \emptyset \end{pmatrix}$$
$$R_6 = \begin{pmatrix} 0 & \mapsto & \{\{0,1\}\} \\ 1 & \mapsto & \emptyset \end{pmatrix} \qquad q = \begin{pmatrix} 0 & \mapsto & \emptyset \\ 1 & \mapsto & \uparrow\{1\} \end{pmatrix}$$

A referee pointed out that the modal box and diamond operators can be interchanged by using the dual of a multirelation with an alternative computational interpretation, where the outer set structure describes demonic choice and the inner set structure describes angelic choice [9]. A consequence of this would be that union represents demonic choice and intersection represents angelic choice.

In the multirelational model, iteration can be represented by least or greatest fixpoints. However, the resulting operations fail to satisfy some simulation properties used for uniformly describing relational and matrix-based models. For example, consider the up-closed multirelation R_7 for $A = B = \{0,1,2\}$ given by

$$\begin{array}{lcl} 0 & \mapsto & \{\{0,1,2\}\} \\ 1 & \mapsto & \mathsf{P}B \\ 2 & \mapsto & \emptyset \end{array}$$

Neither the isolation property $R_7^\omega = (R_7^\omega \mathbin{;} \emptyset) \cup R_7^*$ nor $R_7 \mathbin{;} R_7^* \subseteq R_7^* \mathbin{;} R_7$ nor $R_7 \mathbin{;} R_7^\omega \subseteq R_7^\omega \mathbin{;} R_7$ holds, where R_7^* and R_7^ω denote the least and the greatest fixpoint of $\lambda X.(R_7 \mathbin{;} X) \cup \in$, respectively.

2.4. Summary

The presented relational, matrix-based and multirelational models exemplify the range of computation models that we wish to treat uniformly. Key changes caused by the inclusion of multirelations are:

- sequential composition no longer distributes from the left over union;
- preconditions no longer come just as modal box operators, but also as modal diamond operators;
- preconditions no longer distribute over intersection in their test argument;
- iteration no longer satisfies certain simulation properties.

In the remainder of this paper we investigate whether a uniform theory for correctness reasoning can be devised despite these changes.

3. Correctness

In this section we develop an algebraic theory of correctness suitable for relational, matrix-based and multirelational computation models. We give an axiomatic description of tests, preconditions and while-programs, which we use to obtain a correctness calculus. All results uniformly apply to various computation models and various correctness statements in each model. The presented algebras mostly generalise those in our previous work [21].

3.1. Tests

Preconditions are represented as tests, which form a Boolean subset S' of the carrier S. The subset S' is the image of a unary operation $'$ on S. The operation $'$ plays the role of negation on the subset S'; its effect on elements of $S \setminus S'$ is of no concern. Similarly, a binary operation $\cdot$ on S plays the role of meet on S'. The purpose of the following axiomatisation is to introduce a Boolean algebra of tests without adding a new sort as in [34] or imposing additional axioms as for the domain operation. To achieve this, we apply Huntington's Boolean algebra axioms to S' [31]. The restriction to S' is essential as up-closed multirelations and several matrix-based computation models do not form a Boolean algebra. Thus a *test algebra* [24, 21, 20] is a structure $(S, \cdot, ')$ satisfying the axioms

$$\begin{array}{cc} x'(y'z') = (x'y')z' & x' = (x''y')'(x''y'')' \\ x'y' = y'x' & x'y' = (x'y')'' \end{array}$$

As here, the symbol $\cdot$ is frequently omitted. It follows that $S' = \{x' \mid x \in S\}$ is a Boolean algebra with meet $\cdot$, complement $'$, order $x' \leq y' \Leftrightarrow x'y' = x'$, least element $0 = x'x''$ for any x, and greatest element $1 = 0'$. The operation $x' + y' = (x''y'')'$ is the join in S'. The extension $(S, +, \cdot, ', 0, 1)$ is also called a test algebra; elements of S' are *tests*. The equation $x = x''$ holds if and only if x is a test.

A test algebra is *complete* if S' is a complete Boolean algebra. Then every set of tests has a supremum in S' and the meet operation $\cdot$ on S' distributes over suprema. We denote suprema by $\sum$ and infima by $\prod$. An *ascending chain* in a complete partial order is a sequence x_i such that $x_i \leq x_{i+1}$ for each $i \in \mathbb{N}$, while $x_i \geq x_{i+1}$ is required for a *descending chain*.

3.2. Preconditions

A *precondition algebra* $(S, \cdot, «, ')$ is a test algebra $(S, \cdot, ')$ expanded with a binary operation « satisfying the axioms

$$\begin{array}{cc} x«q = (x«q)'' & xy«q = x«y«q \\ q \leq 1«q & x«pq \leq x«q \end{array}$$

for $x, y \in S$ and $p, q \in S'$. We assume that « associates to the right and has a lower precedence than $\cdot$.

The first axiom states that the result of « is a test, making « an operation which takes an element and a test and yields a test, hence right-associativity. The remaining axioms express the effect of « on 1, the sequential composition of elements and the conjunction of postconditions. They are weaker than our previous axioms of [24, 21] in two respects. First, distributivity $x«pq = (x«p)(x«q)$ is replaced with $x«pq \leq x«q$ which expresses that « is isotone: this accommodates the modal diamond operator in addition to the modal box operator. Second, previous axioms about the precondition of tests implied $q = 1«q$; these are replaced with $q \leq 1«q$ which suffices for the following development.

In many computation models, the precondition $x«q$ represents the set of states from which execution of x is guaranteed to establish postcondition q. Properties of « are recorded in the following result.

Theorem 1. *Let S be a precondition algebra and $x, y \in S$ and $p, q, r \in S'$. Then*

- $x«\_$ *is isotone,*
- $x«pq \leq (x«p)(x«q)$,
- $xy«1 \leq x«1$,
- $x«q \leq x«1 \leq 1$,
- $p \leq x«q \wedge q \leq y«r \Rightarrow p \leq xy«r$.

For example, the last property amounts to soundness of the rule for sequential composition in the correctness calculus of Section 3.4.

3.3. While-programs

We describe the semantics of while-programs by specifying the effect on postconditions similarly to [17]. A *while-algebra* $(S, \lhd\rhd, \cdot, «, \star, ')$ is a precondition algebra $(S, \cdot, «, ')$ expanded with a ternary operation $\lhd\rhd$ and a binary operation $\star$ satisfying the axioms

$$\begin{aligned}(x \lhd p \rhd y)«q &= p(x«q) + p'(y«q)\\ (p \star x)«q &= p(x«(p \star x)«q) + p'q\\ (p \star x)«q &= (p \star x)«p'q\end{aligned}$$

for $x, y \in S$ and $p, q \in S'$. The element $x \lhd p \rhd y$ represents the conditional statement if p then x else y and the corresponding axiom characterises the two branches under a postcondition; see [28, 29, 32] for more comprehensive axiomatisations. The element $p \star x$ represents the while-loop while p do x and the second axiom describes its fixpoint unfolding, again under a postcondition. The third axiom describes that at the end of a loop its condition is false. All three axioms are equations of tests. Some of their consequences are recorded in the following result.

Theorem 2. *Let S be a while-algebra and $x, y \in S$ and $p, q, r \in S'$. Then*

- $pq \leq x«r \wedge p'q \leq y«r \Rightarrow q \leq (x \lhd p \rhd y)«r$,
- $p((x \lhd p \rhd y)«q) = p(x«q)$,
- $p'((x \lhd p \rhd y)«q) = p'(y«q)$,
- $p((p \star x)«q) = p(x«(p \star x)«q)$,
- $p'((p \star x)«q) = p'q$,
- $(p \star x)«q \leq (x(p \star x) \lhd p \rhd 1)«q$,
- $(p \star x)«q \leq (x \lhd p \rhd 1)«(p \star x)«q$,
- $p' \leq (p \star x)«p' \leq (p \star x)«1$,
- $q \leq (p \star x)«1 \Leftrightarrow pq \leq (p \star x)«1$.

For example, the first property amounts to soundness of the rule for conditionals in the correctness calculus of Section 3.4. The next four properties show how to reduce conditionals and while-loops if their conditions are known to hold or not to hold.

In Section 3.4, the test $\ell = (1 \star 1)\text{«}1$ helps us to treat claims of total-correctness and claims that do not involve termination in a uniform way. The element $1 \star 1$ represents the endless loop $\mathsf{while\ true\ do\ skip}$. It establishes the postcondition true if and only if the infinite executions are ignored. Claims which do not involve termination are thus obtained in instances with $\ell = 1$, whereas instances with $\ell = 0$ yield total correctness. In particular, a convenient way to obtain partial correctness is to add the axiom $x\text{«}1 = 1$, a characteristic property of wlp [17].

Consider a while-algebra S, a subset $A \subseteq S$ of atomic programs and a subset $T \subseteq S'$ of atomic tests. We assume that $1 \in A$ and $0 \in T$, that is, skip is an atomic program and false is an atomic test. There are no further requirements on A and T; in concrete models they typically contain basic statements, such as assignments, and basic conditions.

Test expressions are constructed from atomic tests by the operations $'$ for negation and $\cdot$ for conjunction. Hence they are tests and closed under 0, 1, finite sums and finite products. *While-programs* are constructed from atomic programs and test expressions by the operations $\cdot$ for sequential composition, $\lhd\rhd$ for conditionals and $\star$ for while-loops. Hence they are closed under 1 and finite products. *Precondition expressions* are test expressions extended by preconditions; they are constructed from test expressions and while-programs by the operations $'$ for negation, $\cdot$ for conjunction and « for preconditions. Hence they are tests and closed under 0, 1, ℓ, finite sums and finite products.

3.4. Correctness calculus

In the following we introduce correctness statements and a sound and complete correctness calculus. To this end, a *correctness algebra* S is a complete while-algebra satisfying the additional axioms

$$\begin{array}{ll} pq \leq x\text{«}q \Rightarrow \ell q \leq (p \star x)\text{«}q & (x\text{«}q)\ell \leq x\text{«}q\ell \\ p(x\text{«}q) \leq q \Rightarrow (p \star x)\text{«}q \leq q + \ell & y\text{«}\sum t_i = \sum y\text{«}t_i \end{array}$$

for $x, y \in S$ and $p, q, t_i \in S'$ such that y is a while-program and t_i is an ascending chain of tests. The first axiom is sufficient to prove soundness of the correctness calculus below, while the remaining three suffice for proving correctness. The axiom $(x\text{«}q)\ell \leq x\text{«}q\ell$ imports the constant ℓ into the postcondition and clearly holds for $\ell = 0$ or $\ell = 1$. The axiom $y\text{«}\sum t_i = \sum y\text{«}t_i$ states that the function $y\text{«}\_$ is continuous if y is a while-program. For up-closed multirelations, this follows if y is boundedly non-deterministic as we will explain in Section 4.5.

A *correctness statement* $p\{x\}q$ is composed of a while-program x and precondition expressions p and q. The statement $p\{x\}q$ is *valid* if and only if $p \leq x\text{«}q$. Its meaning depends on the model and the interpretation of the precondition operation «. In some models, $p \leq x\text{«}q$ amounts to partial correctness, that is, all finite executions of x starting in p establish the postcondition q. In other models, $p \leq x\text{«}q$ amounts to total correctness, which additionally requires that all executions of x starting in p are finite. In yet other models, $p \leq x\text{«}q$ requires that no execution of x starting in p aborts.

To *derive* correctness statements, we use a calculus with the following rules, for atomic program z, while-programs x and y, test expression p, precondition expressions q, r, s and t, and tests t_i:

$$\text{(atom)}\quad \frac{}{z\text{«}q\{z\}q}$$

$$\text{(seq)}\quad \frac{q\{x\}r \quad r\{y\}s}{q\{xy\}s}$$

$$\text{(cond)}\quad \frac{pq\{x\}r \quad p'q\{y\}r}{q\{x \lhd p \rhd y\}r}$$

$$\text{(while)}\quad \frac{q \leq t_{<\infty} \quad t_0pq\{x\}\ell q \quad \forall n > 0 : t_npq\{x\}t_{<n}q}{q\{p \star x\}p'q}$$

$$\text{(cons)}\quad \frac{q \leq r \quad r\{x\}s \quad s \leq t}{q\{x\}t}$$

Rule (while) is abstracted from the Hoare calculus for total correctness [1], but a change has to be made to accommodate multirelational models. The test $t_{<n}$ is defined by $t_{<n} = \sum_{0 \leq i < n} t_i$ for $n \in \mathbb{N} \cup \{\infty\}$. If $\ell = 0$, the sequence of tests t_i describes the bound function; each test t_n represents a set of states from which the loop terminates after at most n iterations. The inequality $q \leq t_{<\infty}$ expresses that the bound is non-negative while the invariant q holds. By $t_n pq\{x\}t_{<n}q$ every iteration decreases the bound and preserves the loop invariant q at the same time. These two tasks cannot be separated as in previous calculi [1, 21] because « is not distributive so as to capture multirelational models. The triple $t_0 pq\{x\}\ell q$ makes sure that the iteration terminates. If $\ell = 1$, the premises simplify to $pq\{x\}q$ by setting $t_i = q$, which expresses that the loop invariant q is preserved by the loop body for partial-correctness statements. Rule (while) concludes that the invariant is preserved by the while-loop.

A related rule for while-programs appears in [2]. It can be used for showing total correctness, but not partial correctness. In that rule, invariant and bound function are conflated, for which the authors quote practical reasons. Our investigation suggests that this is necessary due to the multirelational model.

The following result shows that our calculus is sound and complete. As usual, completeness is relative to having all true inequalities $p \leq q$ available in the calculus. We sketch the proof and provide some technical details because the support for multirelations substantially increases its complexity.

Theorem 3. *In a correctness algebra, the above calculus is sound and complete, that is, all valid and only valid correctness statements can be derived.*

Proof (Sketch). The additional difficulty arises because decreasing the bound and preserving the loop invariant are not separated. Technically, the triple $t_n pq\{x\}t_{<n}q$ implies but is not equivalent to the conjunction of $t_n pq\{x\}t_{<n}$ and $t_n pq\{x\}q$. Namely, in terms of preconditions we have

$$t_n pq \leq x \mathbin{«} t_{<n} q \leq (x \mathbin{«} t_{<n})(x \mathbin{«} q)$$

by Theorem 1, but the latter inequality is not an equality in general.

Because of the inequality, the triple $t_n pq\{x\}t_{<n}q$ implies both $t_n pq\{x\}t_{<n}$ and $t_n pq\{x\}q$. This means that the soundness part is not affected and can be adapted from our previous proof; see [21]. First, $pq \leq x \mathbin{«} q$ is proved from the premises of rule (while). Then $t_n q \leq (p \star x) \mathbin{«} p'q$ and $t_{<n} q \leq (p \star x) \mathbin{«} p'q$ are proved simultaneously by induction on n. The first axiom of correctness algebras is used in the base case.

Completeness is proved by induction over while-programs and the difficulty arises for while-loops, in which case the triple $w\{p \star x\}p'w$ must be derived where $w = (p \star x) \mathbin{«} q$. This is done by applying rule (while) of the calculus, for which we invent the following bound function given by the sequence t_n:

$$\begin{aligned} t_n &= f^n(p' + (x \mathbin{«} w\ell)) \\ f(r) &= p' + (x \mathbin{«} wr) \end{aligned}$$

Its main feature is that the invariant w is wired into each unfolding of the function f, because it can no longer be established separately from decreasing the bound. This makes it relatively easy to establish the triples required for applying rule (while), but complicates proof of the remaining assumption $w \leq t_{<\infty}$. For this, we invent a second bound function given by the sequence s_n:

$$\begin{aligned} s_n &= g^n(p'q + p(x \mathbin{«} w\ell)) \\ g(r) &= p'q + p(x \mathbin{«} wr) \end{aligned}$$

Again the invariant w is wired into each unfolding of the function g. Because $s_n \leq t_n$ for each $n \in \mathbb{N}$, it remains to show $w \leq s_{<\infty}$. For this, we observe that $w\ell \leq s_0 \leq s_{<\infty}$ by unfolding w and importing ℓ into the postcondition, whence it suffices to show $w \leq s_{<\infty} + \ell$. By a consequence of two correctness algebra axioms, this reduces to $p(x \mathbin{«} s_{<\infty}) + p'q \leq s_{<\infty}$ and hence to $p(x \mathbin{«} s_{<\infty}) \leq s_{<\infty}$. Now s_n is an ascending chain by induction, again using $w\ell \leq s_0$. We therefore apply continuity of « and are left with $p(x \mathbin{«} s_i) \leq s_{<\infty}$ for each $i \in \mathbb{N}$. But this follows since $s_i \leq w$ holds, which implies

$$p(x \mathbin{«} s_i) = p(x \mathbin{«} ws_i) \leq p'q + p(x \mathbin{«} ws_i) = g(s_i) = s_{i+1} \leq s_{<\infty} \; .$$

Note that $t_i \leq w$ does not hold, so this argument cannot be made using the bound function t_n instead of s_n. Moreover, the bound function s_n cannot be used instead of t_n to establish the triples required for applying rule (while). Thus the main idea and difficult part is to carefully craft two bound functions $s_n \leq t_n$ such that s_n applies to one assumption of rule (while) and t_n applies to the remaining two assumptions. □

Our calculus unifies and generalises previous algebraic calculi for partial, total and general correctness [34, 40, 41, 18, 19, 24, 21]. In particular, it applies to further computation models, including multirelations.

Some axioms of precondition algebras and while-algebras appear to be necessary to prove soundness and completeness of the above calculus. Instantiating $r = y\mathbin{«}s$ and $q = x\mathbin{«}r = x\mathbin{«}y\mathbin{«}s$ in rule (seq) implies $x\mathbin{«}y\mathbin{«}s \leq xy\mathbin{«}s$ by Theorem 3. Likewise, instantiating $q = r = x\mathbin{«}s$ in rule (cons) shows $s \leq t \Rightarrow x\mathbin{«}s \leq x\mathbin{«}t$ and hence $x\mathbin{«}st \leq x\mathbin{«}t$. Furthermore, if the triple $q\{1\}q$ is assumed to be valid, as expected if 1 represents the skip program, we obtain $q \leq 1\mathbin{«}q$. Instantiating $q = p(x\mathbin{«}r)+p'(y\mathbin{«}r)$ in rule (cond) implies $p(x\mathbin{«}r)+p'(y\mathbin{«}r) \leq (x \lhd p \rhd y)\mathbin{«}r$. Whether the remaining inequalities are necessary is an open question.

As an example for applying the calculus in a multirelational setting we revisit a simplified version of the Nim game [5, 2, 35]. A pile of matches is given, from which the angel and the demon take turns in removing either one or two. The last player to remove a match loses and the angel starts. The state of the game is given by the value of the variable $v \in \mathbb{N}$. The angel's move A, the demon's move D and their composition $R = A\,;\,D$ are described by the following three multirelations:

0	$\mapsto$	$\mathsf{P}\mathbb{N}$	0	$\mapsto$	$\emptyset$	0	$\mapsto$	$\mathsf{P}\mathbb{N}$
1	$\mapsto$	$\uparrow\{0\}$	1	$\mapsto$	$\uparrow\{0\}$	1	$\mapsto$	$\emptyset$
2	$\mapsto$	$\uparrow\{0\} \cup \uparrow\{1\}$	2	$\mapsto$	$\uparrow\{0,1\}$	2	$\mapsto$	$\uparrow\{0\}$
3	$\mapsto$	$\uparrow\{1\} \cup \uparrow\{2\}$	3	$\mapsto$	$\uparrow\{1,2\}$	3	$\mapsto$	$\uparrow\{0\}$
4	$\mapsto$	$\uparrow\{2\} \cup \uparrow\{3\}$	4	$\mapsto$	$\uparrow\{2,3\}$	4	$\mapsto$	$\uparrow\{0,1\} \cup \uparrow\{1,2\}$
5	$\mapsto$	$\uparrow\{3\} \cup \uparrow\{4\}$	5	$\mapsto$	$\uparrow\{3,4\}$	5	$\mapsto$	$\uparrow\{1,2\} \cup \uparrow\{2,3\}$
	$\dots$			$\dots$			$\dots$	

The image $\mathsf{P}\mathbb{N}$ means that the angel wins, while $\emptyset$ means that the demon wins. The game is specified by the while-loop while true do R, which is algebraically represented by the element $1 \star R$. A win for the angel can be guaranteed from a state v if $v \bmod 3 \neq 1$. Let the multirelation q describe these states as a test, and consider the sequence of tests $t_n = (v = n)$ for which $t_{<\infty} = 1$. Then $t_n q\{R\}t_{<n}q$ holds for every $n > 0$, that is, q is an invariant of R and the bound t is decreased by R at the same time. Moreover, $t_0 q\{R\}0$ holds since R maps state 0 to $\mathsf{P}\mathbb{N}$. We apply the total-correctness instance of rule (while), where $\ell = 0$ and $p = 1$ and $x = R$. The conclusion $q\{1 \star R\}0$ shows that the angel can guarantee a win by establishing postcondition false from starting states in q.

At the same time, the calculus permits the derivation of partial-correctness claims and other kinds of correctness statements for relational and matrix-based models as described in [21].

4. Modal semirings

Modal semirings are semirings extended by a domain operation [14, 40]. In [21] we relativised the domain operation so as to ignore executions contained in a given constant Z. In this section we show that these relativised modal semirings instantiate the various algebras for correctness introduced in Section 3. We furthermore extend modal semirings by a general iteration operator to define the semantics of while-programs for relational, matrix-based and multirelational computation models in a uniform way.

4.1. Idempotent left semirings

Because left distributivity fails for up-closed multirelations, we work in idempotent left semirings [39]. An *idempotent left semiring* is an algebraic structure $(S, +, \cdot, 0, 1)$ satisfying the axioms

$$\begin{array}{ccc} x + (y + z) = (x + y) + z & xy \leq x(y + z) & x(yz) = (xy)z \\ x + y = y + x & xz + yz = (x + y)z & 1x = x \\ x + x = x & 0x = 0 & x1 = x \\ 0 + x = x & & \end{array}$$

In particular, neither $x(y+z) = xy + xz$ nor $x0 = 0$ is an axiom. The *semilattice order* $x \leq y \Leftrightarrow x + y = y$ has least element 0, least upper bound + and isotone operations + and $\cdot$.

In computation models, the operation + represents non-deterministic choice, the operation $\cdot$ sequential composition, 0 the computation with no executions, 1 the program which does not change the state, and $\leq$ the refinement relation. All computation models of Section 2 are idempotent left semirings and, except for the relational model, they do not satisfy the law $x0 = 0$. The relational and matrix-based models satisfy left distributivity $xy + xz = x(y + z)$, but up-closed multirelations do not.

When instantiating an idempotent left semiring with multirelations, the operation + may represent angelic or demonic choice as we will show in Section 5. We use an algebra with just one choice operation because it should also capture the relational and matrix-based models, which offer only a single kind of choice. This is consistent, for example, with general refinement algebras [48]; the algebras of monotonic Boolean transformers [45] are an example with two choice operations.

4.2. Relative domain

Tests can be introduced in idempotent left semirings as elements in the image of an operation d that describes the domain of a computation. The domain $d(x)$ of the computation x is a test representing the set of states from which x has executions. Its Boolean complement, the antidomain $a(x)$, represents the set of states from which x has no executions. These operations are taken relative to an element Z which captures executions that are ignored, whence they satisfy the characteristic properties

$$\begin{array}{rcl} x \leq d(y)x + \mathsf{Z} & \Leftrightarrow & d(x) \leq d(y) \\ a(y)x \leq \mathsf{Z} & \Leftrightarrow & a(y) \leq a(x) \end{array}$$

Hence $d(x)$ is the least test p such that all executions of x start in p, except those executions that are in Z. Similarly, $a(x)$ is the greatest test p such that x has no executions starting in p, except perhaps executions in Z. Setting $\mathsf{Z} = 0$ gives the usual domain and antidomain operations.

A *relative domain semiring* is an algebraic structure $(S, +, \cdot, d, 0, 1, \mathsf{Z})$ such that the reduct $(S, +, \cdot, 0, 1)$ is an idempotent left semiring and the axioms

$$\begin{array}{llr} d(\mathsf{Z}) = 0 & d(x + y) = d(x) + d(y) & x \leq d(x)x + \mathsf{Z} \\ d(x) \leq 1 & d(d(x)y) = d(x)d(y) & d(xy) = d(xd(y)) \end{array}$$

are satisfied. Setting $\mathsf{Z} = 0$ gives the domain semiring axioms of [16]. The following result records properties of the domain operation, which carry over from [21] to the present setting of idempotent left semirings that is needed to capture the multirelational models.

Theorem 4. *Let S be a relative domain semiring and $x, y \in S$. Then*

- $(d(S), +, \cdot, 0, d(1))$ *is a bounded distributive lattice [13] with least element* 0 *and greatest element* $d(1)$,
- d *is isotone,*
- $d(0) = 0$,
- $d(d(x)) = d(x)$,
- $d(xy) \leq d(x) \leq d(1)$,
- $\mathsf{Z}x \leq \mathsf{Z}$,
- $x + \mathsf{Z} = d(x)x + \mathsf{Z}$,
- $d(x) = 0 \Leftrightarrow x \leq \mathsf{Z}$,
- $xy \leq \mathsf{Z} \Leftrightarrow xd(y) \leq \mathsf{Z}$,
- $d(x)y \leq z \Leftrightarrow d(x)y \leq d(x)z$,
- $d(x)y \leq yd(z) \Leftrightarrow d(x)y = d(x)yd(z)$,
- $x \leq d(y)x + \mathsf{Z} \Leftrightarrow d(x) \leq d(y)$.

For example, the property $d(x) = 0 \Leftrightarrow x \leq \mathsf{Z}$ shows that precisely the computations below Z have an empty relative domain as all their executions are ignored. The last property is the characteristic property of relative domain mentioned above.

Each computation model of Section 2 is a relative domain semiring where Z is any one of the values 0, loop, abort or $\mathsf{loop} + \mathsf{abort}$ available in the model. Here loop represents the endless loop and abort the computation which has every aborting execution. In these models the relative domain $d(x)$ is given by first omitting the executions of x that are in Z and then taking the usual domain.

A *relative antidomain semiring* is a structure $(S, +, \cdot, a, d, 0, 1, \mathsf{Z})$ such that the reduct $(S, +, \cdot, 0, 1)$ is an idempotent left semiring, $d(x) = a(a(x))$ and the axioms

$$\begin{array}{lll} a(\mathsf{Z}) = 1 & a(x + y) = a(x)a(y) & a(x)x \leq \mathsf{Z} \\ a(x)d(x) = 0 & a(d(x)y) = a(x) + a(y) & a(xy) = a(xd(y)) \end{array}$$

are satisfied. Setting $\mathsf{Z} = 0$ gives axioms which are equivalent to the antidomain axioms of Boolean domain semirings [16]. Most properties of antidomain generalise from [21] to the present setting of idempotent left semirings. In particular, the following result shows that tests and their Boolean complements can be represented by domain elements $d(x)$ and their antidomain $a(x)$.

Theorem 5. *Let S be a relative antidomain semiring and $x, y, z \in S$. Then*

- $(S, +, \cdot, a, 0, 1)$ *is a test algebra with* $S' = d(S)$,
- $(S, +, \cdot, d, 0, 1, \mathsf{Z})$ *is a relative domain semiring,*
- $(a(S), +, \cdot, a, 0, 1)$ *is a Boolean algebra with complement* a,
- $a(S) = d(S)$,
- $d(a(x)) = a(d(x)) = a(x)$,
- $a(x) + d(x) = 1$,
- $a(x) \leq a(xy)$,
- $a(x) = 1 \Leftrightarrow x \leq \mathsf{Z}$,
- $ya(z) \leq a(x)y \Leftrightarrow ya(z) = a(x)ya(z) \Leftrightarrow d(x)ya(z) = 0$,
- $a(y)x \leq \mathsf{Z} \Leftrightarrow a(y) \leq a(x)$.

The last property is the characteristic property of antidomain mentioned above. A property that does not follow is left distributivity for domain elements $d(x)(y+z) = d(x)y+d(x)z$. For example, it would imply the shunting rule $d(x)y \leq z \Leftrightarrow y \leq z + a(x)\top$ if the greatest element $\top$ exists. Because left distributivity of tests holds for up-closed multirelations, it could be added as an axiom without affecting the validity of the models of Section 2.

Using the Boolean complement of the relative domain, each computation model of Section 2 is a relative antidomain semiring where Z is any of the values 0, loop, abort or $\mathsf{loop} + \mathsf{abort}$ available in the model and different from $\top$.

In [21] we have shown that many correctness claims in relational and matrix-based models take the form $p \cdot R \cdot q' \leq \mathsf{Z}$ using tests p and q. In a relative antidomain semiring such claims take the form $d(x)ya(z) \leq \mathsf{Z}$ where $d(x)$, y and $d(z)$ play the roles of p, R and q, respectively. By Theorem 5 this is equivalent to $d(x) \leq a(ya(z))$. On the other hand, correctness claims for up-closed multirelations take the form $d(x) \leq d(yd(z))$. In the following we show that these different claims can be unified.

4.3. Relative modal operators

Preconditions such as those in correctness statements can be expressed by modal diamond and box operators. These are defined in terms of the relative domain and antidomain operations.

In a relative domain semiring the binary *diamond* operator is defined by $|x\rangle y = d(xy)$, which is the same as $d(xd(y))$. In computation models, the element $|x\rangle y$ represents the set of states from which x has an execution that is not in Z and leads to a state in $d(y)$. The diamond operator satisfies many properties known from the unrelativised setting as the following result shows.

Theorem 6. *Let S be a relative domain semiring and $x, y, z \in S$ and $p, q \in d(S)$. Then*

- $|\_\rangle\_$ *is isotone,*
- $|x + y\rangle z = |x\rangle z + |y\rangle z$,
- $|x\rangle y + |x\rangle z \leq |x\rangle(y + z)$,
- $|xy\rangle z = |x\rangle(yz) = |x\rangle|y\rangle z$,
- $|x\rangle(pq) \leq |x\rangle p \cdot |x\rangle q$,
- $|px\rangle y = p|x\rangle y = p|px\rangle y$,
- $|xp\rangle q = |xp\rangle(pq)$,
- $p \leq |x\rangle y \Leftrightarrow p \leq |px\rangle y$,
- $pq \leq |x\rangle y \Leftrightarrow pq \leq |qx\rangle y$,
- $|p\rangle q = pq$,
- $|p\rangle 0 = |0\rangle y = 0$,
- $|p\rangle 1 = |p\rangle p = |1\rangle p = p$,
- $p|x\rangle q \leq \mathsf{Z} \Leftrightarrow pxq \leq \mathsf{Z}$,
- $|x\rangle q \leq p \Leftrightarrow xq \leq px + \mathsf{Z}$.

For example, the last two properties show how to eliminate the diamond operator on the left-hand side of an inequality, while $|px\rangle y = p|x\rangle y = p|px\rangle y$ shows that tests can be imported to and exported from diamonds. Because left distributivity does not hold in idempotent left semirings, we no longer have $|x\rangle(y + z) = |x\rangle y + |x\rangle z$.

In a relative antidomain semiring the dual *box* operator is defined by $|x]y = a(xa(y))$. In computation models, this represents the set of states from which all executions of x that are not in Z lead to $d(y)$. Also the box operator satisfies many properties known from the unrelativised setting. Moreover, as shown in the following result, both box and diamond can express preconditions.

Theorem 7. *Let S be a relative antidomain semiring and $x, y, z \in S$ and $p, q \in d(S)$. Then*

- S *is a precondition algebra with* $x \ll q = |x]q$,
- S *is a precondition algebra with* $x \ll q = |x\rangle q$,
- $|x]y = a(|x\rangle a(y))$,
- $|x\rangle y = a(|x]a(y))$,
- $|x]\_$ *is isotone,*
- $|\_]x$ *is antitone,*
- $|x + y]z = |x]z \cdot |y]z$,
- $|xy]z = |x]|y]z$,
- $|x](pq) \leq |x]p \cdot |x]q$,
- $|px]y = a(p) + |x]y$,
- $p|x]y = p|px]y$,
- $|xp]q = |xp](pq)$,
- $p \leq |x]y \Leftrightarrow p \leq |px]y$,
- $pq \leq |x]y \Leftrightarrow pq \leq |qx]y$,
- $|p]q = a(p) + q$,
- $|p]0 = a(p)$,
- $|p]1 = |p]p = |0]y = 1$,
- $|1]q = q \leq |p]q$,
- $(|x]y)xa(y) \leq \mathsf{Z}$,
- $|x]1 = 1 \Leftrightarrow x0 \leq \mathsf{Z}$,
- $|x\rangle q \leq p \Leftrightarrow a(p)xq \leq \mathsf{Z}$,
- $p \leq |x]q \Leftrightarrow pxa(q) \leq \mathsf{Z}$.

The property $p|x]y = p|px]y$ shows how to import a test into a box. For another example, $(|x]y)xa(y) \leq \mathsf{Z}$ states that in a state in $|x]y$, where all executions that are not in Z lead to $d(y)$, there are no executions to $a(y)$ except perhaps those in Z. The last two properties give ways to eliminate diamond and box. The last property applies to correctness statements. Again because idempotent left semirings lack left distributivity, we no longer have $|x](pq) = |x]p \cdot |x]q$. We also lose the demodalisation property $p \leq |x]q \Rightarrow px \leq xq + \mathsf{Z}$. As explained in Section 2, the correctness claim for relational and matrix-based models is formalised by $p \leq |x]q$, and the correctness claim for multirelational models is formalised by $p \leq |x\rangle q$.

A relative antidomain semiring is *complete* if $a(\sum x_i) = \prod a(x_i)$ for every ascending chain x_i and $a(\prod y_i) = \sum a(y_i)$ for every descending chain y_i. It follows that d distributes over suprema of ascending chains and infima of descending chains.

4.4. Iteration

We give axioms for operations that describe iteration in various computation models. They are useful for giving a semantics to while-programs, for example. In relational models, iteration is characterised by the induction axioms of Kleene algebras. In some matrix-based models, co-induction axioms such as those of omega algebras or demonic refinement algebras are used instead. Unifying these models, we have introduced weaker simulation axioms in [20]. However, even some of the simulation properties fail in multirelational models; see the counterexamples in Section 2, though this calls for further study.

We therefore rely on the equational properties of iteration [7, 8]. Bloom and Ésik define 'Conway semirings' which are semirings expanded by an operation * satisfying the sumstar axiom $(x+y)^* = (x^*y)^*x^*$ and the productstar axiom $(xy)^* = 1 + x(yx)^*y$ of Conway [12]. In the following we generalise them to reflect the absence of left distributivity in idempotent left semirings.

An *idempotent left Conway semiring*, briefly ILC-semiring, is an idempotent left semiring expanded by an operation $^\circ$ satisfying the axioms

$$\begin{aligned} (x+y)^\circ &= x^\circ(yx^\circ)^\circ \\ 1 + xx^\circ &= x^\circ \\ (xy)^\circ x &\leq x(yx)^\circ \end{aligned}$$

Properties of the operation $^\circ$ are shown in the following result.

Theorem 8. *Let S be an ILC-semiring and $x, y, z \in S$. Then $^\circ$ is isotone and*

- $0^\circ = 1 \leq (x0)^\circ = 1 + x0 \leq x^\circ$,
- $1 + x^\circ x \leq x^\circ = x^\circ x^\circ = (x^\circ x)^\circ = (xx^\circ)^\circ = 1 + x + x^\circ x^\circ$,
- $x \leq x^\circ x \leq xx^\circ \leq x^\circ$,
- $x^\circ \leq x^\circ 1^\circ \leq (1+x)^\circ = x^{\circ\circ} = x^{\circ\circ\circ}$,
- $x^\circ \leq 1^\circ x^\circ \leq x^{\circ\circ}$,
- $1^\circ = 1^{\circ\circ}$,
- $x^\circ y^\circ \leq (x+y)^\circ \leq (x^\circ y^\circ)^\circ = (y^\circ x^\circ)^\circ = x^\circ(y^\circ x^\circ)^\circ = (x^\circ y^\circ)^\circ x^\circ$,
- $1 + x(yx)^\circ y \leq (xy)^\circ \leq (x^\circ y)^\circ x^\circ \leq (x+y)^\circ = x^\circ(1 + y(x+y)^\circ)$,
- $(yx^\circ)^\circ = (yy^\circ x^\circ)^\circ$.

Moreover, $y^\circ z$ is a fixpoint of $\lambda x.yx + z$ and zy° is a prefixpoint of $\lambda x.xy + z$.

Counterexamples generated by Isabelle's Nitpick [6] witness that none of the following properties hold in an ILC-semiring:

$$\begin{array}{cccc} x^\circ = 1 + x^\circ x & x(yx)^\circ = (xy)^\circ x & 1^\circ x^\circ \leq x^\circ 1^\circ = x^{\circ\circ} & (x+y)^\circ = (x^\circ y)^\circ x^\circ \\ (xy)^\circ = 1 + x(yx)^\circ y & xx^\circ = x^\circ x & x^\circ 1^\circ \leq 1^\circ x^\circ = x^{\circ\circ} & (x+y)^\circ = x^\circ + x^\circ y(x+y)^\circ \end{array}$$

These properties hold in the relational and matrix-based models, but have to be omitted as we generalise to multirelational computation models. Other properties, such as $1^\circ = 1$, fail already in matrix-based models where $^\circ$ involves infinite iteration.

A *left Kleene algebra* [33, 39] is an idempotent left semiring expanded by an operation * satisfying the axioms

$$1 + yy^* \leq y^* \qquad z + yx \leq x \Rightarrow y^*z \leq x$$

It follows that y^*z is the least fixpoint of $\lambda x.yx + z$. The Kleene star describes finite iteration.

In addition to being an ILC-semiring, many computation models are left Kleene algebras or can easily be extended by an operation * satisfying the left Kleene algebra axioms. The operations * and $^\circ$ may be identical as in the relational and some multirelational models or different as in other computation models. Similarly, many models either already support or can be extended by a relative antidomain operation. This is reflected in the following structures.

A *modal Conway semiring* is a structure $(S, +, \cdot, a, d, ^\circ, 0, 1, \mathsf{Z})$ such that the reduct $(S, +, \cdot, a, d, 0, 1, \mathsf{Z})$ is a relative antidomain semiring and the reduct $(S, +, \cdot, ^\circ, 0, 1)$ is an ILC-semiring. A *modal Conway algebra* is a structure $(S, +, \cdot, a, d, ^*, ^\circ, 0, 1, \mathsf{Z})$ such that the reduct $(S, +, \cdot, a, d, ^\circ, 0, 1, \mathsf{Z})$ is a modal Conway semiring and the reduct $(S, +, \cdot, ^*, 0, 1)$ is a left Kleene algebra.

We give several multirelational instances of these structures in Section 5. As the following result shows, the operations $\lhd\rhd$ and $\star$ can be defined in modal Conway semirings, giving a unified semantics of while-programs.

Theorem 9. *Let S be a modal Conway semiring and $x, y \in S$ and $p \in d(S)$. Then S is a while-algebra with $x \lhd p \rhd y = px + a(p)y$ and $p \star x = (px)^\circ a(p)$.*

By Theorem 7, the underlying precondition operation can be either $x\mathbin{«}q = |x\rangle q$ or $x\mathbin{«}q = |x]q$.

4.5. Correctness

The following result shows how to obtain a sound and complete correctness calculus in modal Conway algebras subjected to additional conditions.

Theorem 10. *Let S be a modal Conway algebra – which is a test algebra, a precondition algebra and a while-algebra according to Theorems 5, 7 and 9 – with a complete relative antidomain semiring, $\ell \in \{0, 1\}$ and $x(y + \mathsf{Z}) \leq xy + \mathsf{Z}$ for each $x, y \in S$. Then the following hold.*

- *Let $x^\circ = x^*$ for each $x \in S$ and $y \sum t_i = \sum yt_i$ for every while-program y and ascending chain of tests t_i. Then S is a correctness algebra using $x\mathbin{«}q = |x\rangle q$.*
- *Let $\ell x^\circ \leq x^*$ and $p|x]q \leq q \Rightarrow |x^\circ]p \leq q + \ell$ for each $x \in S$ and $p, q \in d(S)$. Let $y \prod t_i = \prod yt_i$ for every while-program y and descending chain of tests t_i. Then S is a correctness algebra using $x\mathbin{«}q = |x]q$.*

In both cases, Theorem 3 yields a sound and complete correctness calculus.

The property $x(y+\mathsf{Z}) \leq xy+\mathsf{Z}$ holds for multirelational models where $\mathsf{Z} = 0$ and for relational and matrix-based models which satisfy left distributivity and $x\mathsf{Z} \leq x0 + \mathsf{Z}$ [21]. The property $p|x]q \leq q \Rightarrow |x^\circ]p \leq q + \ell$ is similar to the induction axiom for the convergence operation of [41]. For up-closed multirelations, the continuity property $y \sum t_i = \sum yt_i$ holds if the demonic choices in y are finite [44]; see [36] for continuity in probabilistic Kleene algebras. A related argument shows that $y \prod t_i = \prod yt_i$ holds for an up-closed multirelation y if the angelic choices in y are finite.

Pre-post specifications in the style of [38, 43, 42, 48] can be introduced in relative antidomain semirings by a Galois connection as elaborated in [21]. This generalises to multirelational models; see [45] for pre-post specifications in algebras of monotonic Boolean transformers.

5. Instances

In this section we instantiate the algebras of Section 4 to demonstrate the range of computation models covered by our theory of correctness. We first recall the algebras of monotonic Boolean transformers [45] that capture up-closed multirelations and play a prominent role below.

An *algebra of monotonic Boolean transformers* is a structure $(S, \sqcup, \sqcap, \cdot, ^\smile, ^\omega, \bot, \top, 1)$ such that the reduct $(S, \sqcup, \sqcap, \bot, \top)$ is a bounded distributive lattice, the reduct $(S, \cdot, 1)$ is a monoid and the axioms

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl} (x \sqcap y) \cdot z &=& (x \cdot z) \sqcap (y \cdot z) & x^{\smile\smile} &=& x & (x \cdot \top) \sqcap (x^\smile \cdot \bot) &=& \bot \\ \top \cdot x &=& \top & (x \cdot y)^\smile &=& x^\smile \cdot y^\smile & x \cdot x^\omega \sqcap 1 &=& x^\omega \\ x \le y \Rightarrow z \cdot x \le z \cdot y & & & x \le y \Leftrightarrow y^\smile \le x^\smile & & & x \cdot z \sqcap y \le z \Rightarrow x^\omega \cdot y \le z & \end{array}$$

are satisfied, where $x \le y \Leftrightarrow x \sqcup y = y \Leftrightarrow x \sqcap y = x$ is the lattice order. It follows that both $(S, \sqcup, \cdot, \bot, 1)$ and $(S, \sqcap, \cdot, \top, 1)$ are idempotent left semirings. An *assertion* is an element $p \in S$ such that $p \le 1$ and $p = p \cdot \top \sqcap 1$. An *assumption* is the dual $p^\smile$ of an assertion p.

In multirelational models, $\sqcup$ and $\sqcap$ are angelic and demonic choice, $\bot$ and $\top$ are the empty and the universal multirelations, $\cdot$ is sequential composition, 1 is the set membership multirelation $\in$, and $^\smile$ is the dual of multirelations. Assertions represent tests.

As a consequence of the above axioms, $x^\omega \cdot y$ is the least fixpoint of the function $\lambda z.x \cdot z \sqcap y$. Algebras of monotonic Boolean transformers can be extended by the weak iteration operator * of [48] with the axioms

$$x \cdot x^* \sqcap 1 = x^* \qquad z \le x \cdot z \sqcap y \Rightarrow z \le x^* \cdot y$$

Then $x^* \cdot y$ is the greatest fixpoint of $\lambda z.x \cdot z \sqcap y$. Note that * and $^\omega$ denote other fixpoints for monotonic Boolean transformers than for Kleene and omega algebras, which use the function $\lambda z.x \cdot z \sqcup y$.

The following result gives numerous instances of ILC-semirings which cover all computation models of Section 2.

Theorem 11. *ILC-semirings have the following models:*

1. *Every left Kleene algebra is an ILC-semiring using $x^\circ = x^*$.*
   In particular, so is every probabilistic Kleene algebra [37].
2. *Every left omega algebra [11, 39] is an ILC-semiring using $x^\circ = x^*(x^\omega 0 + 1)$.*
   In particular, every left omega algebra with $\top x = \top$ is an ILC-semiring using $x^\circ = x^*(x^\omega + 1)$.
3. *Every general refinement algebra [48] is an ILC-semiring using $x^\circ = x^\omega$.*
   In particular, so is every demonic refinement algebra.
4. *Extended designs [25, 23] form an ILC-semiring using $x^\circ = d(x^\omega)\mathsf{loop} + x^*$.*
5. *The model of [20] that represents finite, infinite and aborting executions independently forms an ILC-semiring using $x^\circ = n(x^\omega)\mathsf{loop} + x^*$, where $n(x)$ captures the infinite executions of x as a test.*
6. *Every itering [20] is an ILC-semiring.*
7. *Every extended binary itering [22] is an ILC-semiring using $x^\circ = x \star 1$.*
8. *Every algebra of monotonic Boolean transformers is an ILC-semiring in two ways:*
   - *using $x^\circ = x^\omega$ and $x + y = x \sqcap y$ and $0 = \top$, and*
   - *using $x^\circ = x^{\smile\omega\smile}$ and $x + y = x \sqcup y$ and $0 = \bot$.*
9. *Every algebra of monotonic Boolean transformers extended by the weak iteration operator * of [48] is an ILC-semiring in two further ways:*
   - *using $x^\circ = x^*$ and $x + y = x \sqcap y$ and $0 = \top$, and*
   - *using $x^\circ = x^{\smile *\smile}$ and $x + y = x \sqcup y$ and $0 = \bot$.*

Up-closed multirelations form a general refinement algebra and an algebra of monotonic Boolean transformers. Hence they form an ILC-semiring in four different ways as stated in the last two facts.

Idempotent left semirings can be extended with domain and antidomain operations as shown in [39, 15]. This generalises to our relative domain and antidomain operations, which can extend each of the above models. Because of their duality, up-closed multirelations actually form relative antidomain semirings in two different ways as shown by the following result.

Theorem 12. *Every algebra of monotonic Boolean transformers is a relative antidomain semiring in two ways:*

- *using* $a(x) = x^\smile\top \sqcup 1$ *and* $d(x) = x\bot \sqcup 1$ *and* $x + y = x \sqcap y$ *and* $0 = \mathsf{Z} = \top$*, and*
- *using* $a(x) = x^\smile\bot \sqcap 1$ *and* $d(x) = x\top \sqcap 1$ *and* $x + y = x \sqcup y$ *and* $0 = \mathsf{Z} = \bot$*.*

In the second instance, tests are assertions, while they are assumptions in the first instance. Given a relative antidomain semiring, we can choose either diamond or box to represent preconditions by Theorem 7.

Altogether we obtain eight different instances in which up-closed multirelations form a modal Conway semiring. We describe these instances in terms of monotonic Boolean transformers. The options are summarised as follows:

- choose operations or their duals,
- choose least or greatest fixpoint for loops,
- choose diamond or box for preconditions.

All eight instances have the same operations for sequential composition $\cdot$ and 1. Depending on the choice of operations or their duals we obtain the instances

$+$	0	$\leq$	Z	$d(x)$	$\lvert x\rangle y$	$a(x)$	$\lvert x]y$	$x \lhd p \rhd y$
$\sqcup$	$\bot$	$\leq$	$\bot$	$x\top \sqcap 1$	$xy\top \sqcap 1$	$\neg x$	$\neg(x\neg y)$	$px \sqcup \neg py$
$\sqcap$	$\top$	$\geq$	$\top$	$x\bot \sqcup 1$	$xy\bot \sqcup 1$	$\sim x$	$\sim(x\sim y)$	$px \sqcap \sim py$

where $\neg x = x^\smile\bot \sqcap 1$ complements assertions and $\sim x = x^\smile\top \sqcup 1$ complements assumptions. In each of these two cases, we can choose the least or the greatest fixpoint for loops, which yields

$+$	x°	$p \star x$
$\sqcup$	$x^{\smile *\smile} = \mu y.xy \sqcup 1$	$(px)^{\smile *\smile}\neg p = \mu y.pxy \sqcup \neg p$
$\sqcup$	$x^{\smile \omega\smile} = \nu y.xy \sqcup 1$	$(px)^{\smile \omega\smile}\neg p = \nu y.pxy \sqcup \neg p$

$+$	x°	$p \star x$
$\sqcap$	$x^\omega = \mu y.xy \sqcap 1$	$(px)^\omega \sim p = \mu y.pxy \sqcap \sim p$
$\sqcap$	$x^* = \nu y.xy \sqcap 1$	$(px)^* \sim p = \nu y.pxy \sqcap \sim p$

where $\mu x.f(x)$ is the least fixpoint of f and $\nu x.f(x)$ the greatest. Finally, in each of these four cases, we can choose the diamond or the box operator for preconditions.

$+$	$x \lhd p \rhd y$	$p \star x$	$x \ll q$	ℓ
$\sqcup$	$px \sqcup \neg py$	$(px)^{\smile *\smile}\neg p$	$xq\top \sqcap 1$	$\bot$
$\sqcup$	$px \sqcup \neg py$	$(px)^{\smile *\smile}\neg p$	$x^\smile q\top \sqcap 1$	1
$\sqcup$	$px \sqcup \neg py$	$(px)^{\smile \omega\smile}\neg p$	$xq\top \sqcap 1$	1
$\sqcup$	$px \sqcup \neg py$	$(px)^{\smile \omega\smile}\neg p$	$x^\smile q\top \sqcap 1$	$\bot$

$+$	$x \lhd p \rhd y$	$p \star x$	$x \ll q$	ℓ
$\sqcap$	$px \sqcap \sim py$	$(px)^\omega \sim p$	$xq\bot \sqcup 1$	1
$\sqcap$	$px \sqcap \sim py$	$(px)^\omega \sim p$	$x^\smile q\bot \sqcup 1$	$\top$
$\sqcap$	$px \sqcap \sim py$	$(px)^* \sim p$	$xq\bot \sqcup 1$	$\top$
$\sqcap$	$px \sqcap \sim py$	$(px)^* \sim p$	$x^\smile q\bot \sqcup 1$	1

The instances are collected in the following result, which shows that we obtain a sound and complete correctness calculus for four of them. We obtain at least soundness for the remaining four, with completeness being an open problem.

Theorem 13. *Every complete algebra of monotonic Boolean transformers is a modal Conway algebra with a precondition operation in eight ways:*

$+$	x^*	x°	$x \ll q$	*sound*	*complete*
$\sqcup$	$x^{\smile *\smile}$	$x^{\smile *\smile}$	$\lvert x\rangle q$	✓	✓
$\sqcup$	$x^{\smile *\smile}$	$x^{\smile *\smile}$	$\lvert x]q$	✓	
$\sqcup$	$x^{\smile *\smile}$	$x^{\smile \omega\smile}$	$\lvert x\rangle q$	✓	
$\sqcup$	$x^{\smile *\smile}$	$x^{\smile \omega\smile}$	$\lvert x]q$	✓	✓

$+$	x^*	x°	$x \ll q$	*sound*	*complete*
$\sqcap$	x^*	x^ω	$\lvert x\rangle q$	✓	
$\sqcap$	x^*	x^ω	$\lvert x]q$	✓	✓
$\sqcap$	x^*	x^*	$\lvert x\rangle q$	✓	✓
$\sqcap$	x^*	x^*	$\lvert x]q$	✓	

Instances marked sound and complete satisfy the conditions of Theorem 10 yielding a sound and complete correctness calculus. Instances marked sound satisfy conditions that yield a sound correctness calculus.

For constructing while-programs, these instances use the set of all assertions/assumptions as atomic tests and the set of all continuous/co-continuous elements as atomic programs. These are described in the following.

Let S be a complete algebra of monotonic Boolean transformers. Recall that a subset $T \subseteq S$ is *directed* if it is not empty and every pair of elements in T has an upper bound in T. Dually, lower bounds are required for *co-directed* sets. We call an element $x \in S$ *continuous* if $x \sum_{y \in T} y = \sum_{y \in T} xy$ for every directed set $T \subseteq S$. Dually, $x \in S$ is *co-continuous* if $x \prod_{y \in T} y = \prod_{y \in T} xy$ for every co-directed set $T \subseteq S$.

Theorem 14. *Let S be a complete algebra of monotonic Boolean transformers. Let $x, y \in S$ be continuous, let $u, v \in S$ be co-continuous and let p be an assertion or an assumption. Then*

- *p and 1 and xy and $x \sqcup y$ and $x \sqcap y$ and $x^{\smile *\smile}$ and x^ω and $u^\smile$ are continuous;*
- *p and 1 and uv and $u \sqcup v$ and $u \sqcap v$ and $u^{\smile \omega \smile}$ and u^* and $x^\smile$ are co-continuous.*

6. Conclusion

Preconditions, correctness statements and correctness calculi can be unified for relational, matrix-based and multirelational computation models. These models support various combinations of finite, infinite and aborting executions and demonic/angelic choice. The symmetries of multirelations go beyond a simple exchange of the lattice structure for its dual and allow a uniform treatment of instances with different precondition and iteration operations. Many useful properties of iteration can be derived in a weak setting of equational axioms without left distributivity of sequential composition.

Our investigation offers various opportunities for further research. In Section 3 we observe several properties of preconditions and while-programs that are necessary for our correctness calculus, but for some axioms this question is open. The two implicational axioms of correctness algebras are quite symmetric, which should be explored further given that one is used for soundness and the other for completeness of the calculus. The relative domain semirings of Section 4 unify matrix-based computation models with finite, infinite and aborting executions, but so far multirelational computation models do not distinguish between infinite and aborting executions. We also observe that some simulation properties do not generalise from relational and matrix-based models to multirelations, and it remains to find implicational properties that uniformly hold for the various iteration operations supported by multirelations. Multirelations can be boundedly non-deterministic in two ways, with respect to angelic choices or with respect to demonic choices, but it is not clear how these two notions are connected.

Acknowledgement

I thank the anonymous referees and Jeremy Gibbons for their insightful remarks. I am indebted to an anonymous referee of [21] who asked whether that approach can be extended to work for general refinement algebras.

Corrections

In my paper [21] the following statements are erroneous and should be replaced as indicated. The Isabelle theories contain the correct statements.

- In Theorem 3 on page 208, replace $|x\rangle y + z = |x\rangle y + |x\rangle z$ with $|x\rangle(y + z) = |x\rangle y + |x\rangle z$.
- In Theorem 3 on page 208, replace $|x\rangle(pq) = |x\rangle p \cdot |x\rangle q$ with $|x\rangle(pq) \leq |x\rangle p \cdot |x\rangle q$.
- In Theorem 10 on page 212, replace $p((p \star x)\mathbin{«} q) = p(x\mathbin{«}(p \star x)q)$ with $p((p \star x)\mathbin{«} q) = p(x\mathbin{«}(p \star x)\mathbin{«} q)$.

References

[1] K. R. Apt, F. S. de Boer, and E.-R. Olderog. *Verification of Sequential and Concurrent Programs*. Springer, third edition, 2009.

[2] R.-J. Back and J. von Wright. *Refinement Calculus*. Springer, New York, 1998.

[3] R. J. R. Back and J. von Wright. Duality in specification languages: A lattice-theoretical approach. *Acta Informatica*, 27(7):583–625, July 1990.

[4] R. J. R. Back and J. von Wright. Combining angels, demons and miracles in program specifications. *Theoretical Computer Science*, 100(2):365–383, June 1992.

[5] R. J. R. Back and J. von Wright. Games and winning strategies. *Information Processing Letters*, 53(3):165–172, February 1995.

[6] J. C. Blanchette and T. Nipkow. Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In M. Kaufmann and L. C. Paulson, editors, *Interactive Theorem Proving*, volume 6172 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2010.

[7] S. L. Bloom and Z. Ésik. *Iteration Theories: The Equational Logic of Iterative Processes*. Springer, 1993.

[8] S. L. Bloom and Z. Ésik. Matrix and matricial iteration theories, part I. *Journal of Computer and System Sciences*, 46(3):381–408, June 1993.

[9] A. Cavalcanti, J. Woodcock, and S. Dunne. Angelic nondeterminism in the unifying theories of programming. *Formal Aspects of Computing*, 18(3):288–307, September 2006.

[10] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981.

[11] E. Cohen. Separation and reduction. In R. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction*, volume 1837 of *Lecture Notes in Computer Science*, pages 45–59. Springer, 2000.

[12] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

[13] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, second edition, 2002.

[14] J. Desharnais, B. Möller, and G. Struth. Kleene algebra with domain. *ACM Transactions on Computational Logic*, 7(4):798–833, October 2006.

[15] J. Desharnais and G. Struth. Domain axioms for a family of near-semirings. In J. Meseguer and G. Roşu, editors, *Algebraic Methodology and Software Technology*, volume 5140 of *Lecture Notes in Computer Science*, pages 330–345. Springer, 2008.

[16] J. Desharnais and G. Struth. Internal axioms for domain semirings. *Science of Computer Programming*, 76(3):181–203, March 2011.

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

[18] W. Guttmann. General correctness algebra. In R. Berghammer, A. M. Jaoua, and B. Möller, editors, *Relations and Kleene Algebra in Computer Science*, volume 5827 of *Lecture Notes in Computer Science*, pages 150–165. Springer, 2009.

[19] W. Guttmann. Partial, total and general correctness. In C. Bolduc, J. Desharnais, and B. Ktari, editors, *Mathematics of Program Construction*, volume 6120 of *Lecture Notes in Computer Science*, pages 157–177. Springer, 2010.

[20] W. Guttmann. Algebras for iteration and infinite computations. *Acta Informatica*, 49(5):343–359, August 2012.

[21] W. Guttmann. Unifying correctness statements. In J. Gibbons and P. Nogueira, editors, *Mathematics of Program Construction*, volume 7342 of *Lecture Notes in Computer Science*, pages 198–219. Springer, 2012.

[22] W. Guttmann. Unifying lazy and strict computations. In W. Kahl and T. G. Griffin, editors, *Relational and Algebraic Methods in Computer Science*, volume 7560 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2012.

[23] W. Guttmann. Extended designs algebraically. *Science of Computer Programming*, 2013. In press; available from http://dx.doi.org/10.1016/j.scico.2012.07.009.

[24] W. Guttmann, G. Struth, and T. Weber. Automating algebraic methods in Isabelle. In S. Qin and Z. Qiu, editors, *Formal Methods and Software Engineering*, volume 6991 of *Lecture Notes in Computer Science*, pages 617–632. Springer, 2011.

[25] I. J. Hayes, S. E. Dunne, and L. Meinicke. Unifying theories of programming that distinguish nontermination and abort. In C. Bolduc, J. Desharnais, and B. Ktari, editors, *Mathematics of Program Construction*, volume 6120 of *Lecture Notes in Computer Science*, pages 178–194. Springer, 2010.

[26] W. H. Hesselink. Multirelations are predicate transformers. Available from http://www.cs.rug.nl/~wim/pub/whh318.pdf, February 2004.

[27] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580/583, October 1969.

[28] C. A. R. Hoare. A couple of novelties in the propositional calculus. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 31(9–12):173–178, October 1985.

[29] C. A. R. Hoare, I. J. Hayes, J. He, C. C. Morgan, A. W. Roscoe, J. W. Sanders, I. H. Sorensen, J. M. Spivey, and B. A. Sufrin. Laws of programming. *Communications of the ACM*, 30(8):672–686, 1987.

[30] C. A. R. Hoare and J. He. *Unifying theories of programming*. Prentice Hall Europe, 1998.

[31] E. V. Huntington. Boolean algebra. A correction. *Transactions of the American Mathematical Society*, 35(2):557–558, 1933.

[32] M. Jackson and T. Stokes. Semigroups with if-then-else and halting programs. *International Journal of Algebra and Computation*, 19(7):937–961, 2009.

[33] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, May 1994.

[34] D. Kozen. On Hoare logic and Kleene algebra with tests. *ACM Transactions on Computational Logic*, 1(1):60–76, July 2000.

[35] C. E. Martin, S. A. Curtis, and I. Rewitzky. Modelling angelic and demonic nondeterminism with multirelations. *Science of Computer Programming*, 65(2):140–158, March 2007.
[36] A. McIver, T. M. Rabehaja, and G. Struth. On probabilistic Kleene algebras, automata and simulations. In H. de Swart, editor, *Relational and Algebraic Methods in Computer Science*, volume 6663 of *Lecture Notes in Computer Science*, pages 264–279. Springer, 2011.
[37] A. K. McIver and T. Weber. Towards automated proof support for probabilistic distributed systems. In G. Sutcliffe and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 3835 of *Lecture Notes in Computer Science*, pages 534–548. Springer, 2005.
[38] L. Meertens. Abstracto 84: The next generation. In A. L. Martin and J. L. Elshoff, editors, *ACM '79: Proceedings of the 1979 annual conference*, pages 33–39. ACM Press, 1979.
[39] B. Möller. Kleene getting lazy. *Science of Computer Programming*, 65(2):195–214, March 2007.
[40] B. Möller and G. Struth. Algebras of modal operators and partial correctness. *Theoretical Computer Science*, 351(2):221–239, February 2006.
[41] B. Möller and G. Struth. WP is WLP. In W. MacCaull, M. Winter, and I. Düntsch, editors, *Relational Methods in Computer Science 2005*, volume 3929 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2006.
[42] C. Morgan. The specification statement. *ACM Transactions on Programming Languages and Systems*, 10(3):403–419, July 1988.
[43] J. M. Morris. A theoretical basis for stepwise refinement and the programming calculus. *Science of Computer Programming*, 9(3):287–306, December 1987.
[44] K. Nishizawa, N. Tsumagari, and H. Furusawa. The cube of Kleene algebras and the triangular prism of multirelations. In R. Berghammer, A. M. Jaoua, and B. Möller, editors, *Relations and Kleene Algebra in Computer Science*, volume 5827 of *Lecture Notes in Computer Science*, pages 276–290. Springer, 2009.
[45] V. Preoteasa. Algebra of monotonic Boolean transformers. In A. Simao and C. Morgan, editors, *Formal Methods: Foundations and Applications*, volume 7021 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 2011.
[46] I. Rewitzky. Binary multirelations. In H. de Swart, E. Orłowska, G. Schmidt, and M. Roubens, editors, *Theory and Applications of Relational Structures as Knowledge Instruments*, volume 2929 of *Lecture Notes in Computer Science*, pages 256–271. Springer, 2003.
[47] I. Rewitzky and C. Brink. Monotone predicate transformers as up-closed multirelations. In R. Schmidt, editor, *Relations and Kleene Algebra in Computer Science*, volume 4136 of *Lecture Notes in Computer Science*, pages 311–327. Springer, 2006.
[48] J. von Wright. Towards a refinement algebra. *Science of Computer Programming*, 51(1–2):23–45, May 2004.