Algebras for Iteration and Infinite Computations

Walter Guttmann

Abstract We give axioms for an operation that describes iteration in various relational models of computations. The models differ in their treatment of finite, infinite and aborting executions, covering partial, total and general correctness and extensions thereof. Based on the common axioms we derive separation, refinement and program transformation results hitherto known from particular models, henceforth recognised to hold in many different models. We introduce a new model that independently describes the finite, infinite and aborting executions of a computation, and axiomatise an operation that extracts the infinite executions in this model and others. From these unifying axioms we derive explicit representations for recursion and iteration. We show that also the new model is an instance of our general theory of iteration. All results are verified in Isabelle heavily using automated theorem provers.

1 Introduction

Sequential composition, non-deterministic choice, iteration and recursion are fundamental operations in many models of computations. Some models describe a computation by relating the system states before and after its execution. The precision of these relational models varies concerning finite, infinite and aborting executions (which terminate due to an error, such as integer division by zero; 'finite' means 'normally terminating'). Some ignore infinite executions, others describe infinite executions but not aborting ones. Some models unify aborting executions with infinite executions, and some ignore finite executions in the presence of infinite ones. Another model discussed in this paper is more precise by independently representing finite, infinite and aborting executions.

We use algebra for giving structure to this diversity of models and for unifying existing approaches. The general idea is to investigate key aspects of the computation models, to describe them by operations of algebraic structures whose elements represent computations, to find properties of these operations that hold in several models, and to install such properties as axioms of the algebraic structures. Results derived from these common axioms hold in several computation models. Individual models can be characterised by adding

Walter Guttmann
Universität Ulm, Germany
E-mail: walter.guttmann@uni-ulm.de

more specific properties as axioms, resulting in a hierarchy of algebras. Lying at the core of our development, semirings are used to represent non-deterministic choice and sequential composition. They are extended in various ways to describe iteration, recursion and infinite computations. Our contributions are as follows.

In Section 3 we expand semirings by an operation for iteration with axioms that are general enough to capture the semantics of while-loops in a variety of models including partial, total and general correctness, yet powerful enough to derive complex results including program transformations and refinement theorems for all these models. Besides finding the right balance, the difficulty of giving suitable axioms comes from the fact that in different models iteration is captured by either least fixpoints, greatest fixpoints or various combinations of the two. We therefore cannot use the induction and co-induction axioms on which Kleene algebra and omega algebra are based [35, 12] but replace them by simulation properties. Together with Conway's sumstar and productstar equations [13] we obtain a new characterisation of iteration which is valid in several models. Existing iteration operations are instances of this unified iteration, which allows us to transfer results known from particular models to many different ones.

In Section 4 we introduce a relational model of computations with independent finite, infinite and aborting executions. It is the most precise model considered in this paper and useful for situations that need to distinguish infinite and aborting executions reflecting their observable difference. We present an axiomatic semantics for this model. In particular, we axiomatise an operation that yields the infinite executions in this and several other models. The operation is used to derive the semantics of recursive computations. For loops this specialises to a semantics which satisfies the axioms of iteration given in Section 3. All results obtained there are instantiated by the new model. Similarly to the domain operation of modal Kleene algebras [41], our new operation induces modal diamond and box operators, which we use to reason about the infinite executions occurring in finite and infinite iterations.

The above theory is implemented in Isabelle/HOL, making heavy use of the integrated automatic theorem provers and SMT solvers [45, 7]. The technical contents of this paper are taken from verified Isabelle theory files and adapted only to improve presentation. From more than 750 new facts contained in the implementation only the most relevant ones are shown here. Their proofs are omitted and can be found in the theory files, which are available at http://www.uni-ulm.de/en/in/pm/staff/guttmann/algebra/.

2 Semirings

An idempotent semiring without right zero – called *semiring* in the remainder of this paper – is an algebraic structure $(S, +, \cdot, 0, 1)$ satisfying the axioms

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

where $x \cdot y$ is conventionally abbreviated as xy. In particular, the operation $+$ is idempotent and 0 is not required to be a right annihilator, that is, $x0 = 0$ is not an axiom. The *semilattice order* $x \leq y \Leftrightarrow x+y = y$ has least element 0, least upper bound $+$ and isotone operations $+$ and $\cdot$. A semiring is *bounded* if it has a greatest element $\top$ satisfying $x+\top = \top$.

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, $\top$ the computation with all possible executions, and $\leq$ the refinement relation where $x \leq y$ means that x refines y. Omission of the axiom $x0 = 0$ is essential as it does not hold in many models.

3 Iteration

An iteration describes the repeated sequential execution of a computation. Being a special case of recursion, this is typically modelled by fixpoints.

The equational properties of the fixpoint operation, hence in particular iterations, are thoroughly investigated in [9]. The authors 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 [13]. The latter is equivalent to the conjunction of sliding $x(yx)^* = (xy)^*x$ and either left unfold $1+xx^* = x^*$ or right unfold $1+x^*x = x^*$. In Kleene algebras, these well-known regular identities are augmented by induction axioms to the effect that y^*z is the least fixpoint of $\lambda x.yx+z$ and zy^* that of $\lambda x.xy+z$.

Because we aim for models which vary in the fixpoints used for iteration, we cannot settle for the least fixpoint or any other particular fixpoint. This rules out the use of the induction axioms of Kleene algebra. But we can take from that setting Conway's suggestion of using simulation axioms instead [13]. Our new iteration generalises the Kleene star * and is denoted by $^\circ$ to avoid confusion.

A well-known simulation law in Kleene algebra is $zx \leq yz \Rightarrow zx^\circ \leq y^\circ z$. It is obtained by setting $w = 0$ in the 'iteration theorem' $zx \leq yz+w \Rightarrow zx^\circ \leq y^\circ(z+wx^\circ)$ of omega algebra [12]. We further generalise this by weakening the antecedent to $zx \leq yy^\circ z+w$. The resulting, first simulation axiom is $zx \leq yy^\circ z+w \Rightarrow zx^\circ \leq y^\circ(z+wx^\circ)$.

The dual simulation law $xz \leq zy \Rightarrow x^\circ z \leq zy^\circ$ holds in Kleene algebra, but it implies $1^\circ 0 \leq 0$ which fails in other target models. First of all, we therefore weaken its consequent to $x^\circ z \leq zy^\circ + x^\circ 0$. Two generalisations make the outcome nearly symmetric to the first axiom; the resulting, second simulation axiom is $xz \leq zy^\circ + w \Rightarrow x^\circ z \leq (z+x^\circ w)y^\circ$. The symmetric antecedent $zx \leq y^\circ z+w$ cannot be used in the first axiom because this would imply $1^\circ \leq 0^\circ$ which again fails in some target models.

Additionally to these two simulation axioms we adopt the sumstar and productstar equations. Thus an *itering* $(S,+,\cdot,^\circ,0,1)$ is a semiring $(S,+,\cdot,0,1)$ expanded by an operation $^\circ$ satisfying the four axioms

$$\begin{array}{ll} (x+y)^\circ = (x^\circ y)^\circ x^\circ & zx \leq yy^\circ z+w \Rightarrow zx^\circ \leq y^\circ(z+wx^\circ) \\ (xy)^\circ = 1+x(yx)^\circ y & xz \leq zy^\circ + w \Rightarrow x^\circ z \leq (z+x^\circ w)y^\circ \end{array}$$

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

Theorem 1 *Let S be an itering and $x,y,z \in S$. Then $^\circ$ is isotone and*

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

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

Counterexamples generated by Isabelle's Nitpick [8] witness that none of the inequalities can be strengthened to an equation and that the induction axioms of Kleene algebra do not follow. In particular, $0^\circ = 1 \neq 1^\circ = 0^{\circ\circ}$ in some target models. In a bounded itering, $\top^\circ = \top$.

3.1 Applications: Separation and Refinement

Four examples show the versatility of iterings. The first generalises two 'separation theorems' of omega algebra [12].

Theorem 2 *Let S be an itering and $x,y \in S$ such that $yx \leq xy^\circ$ or $yx \leq xx^\circ(1+y)$. Then $y^\circ x^\circ \leq x^\circ y^\circ = (x+y)^\circ$.*

In particular, the assumption is satisfied if $yx \leq xy$. The weaker bounds xy° and $xx^\circ(1+y)$ include infinite iterations in some models.

The second example is Back's atomicity refinement theorem. Our formulation is adapted from [46].

Theorem 3 *Let S be an itering and $a,b,l,r,q,s \in S$ such that*

$$\begin{array}{lllll} s = sq & rb \leq br & rl \leq lr & bl \leq lb & r^\circ q \leq qr^\circ \\ a = qa & qb = 0 & al \leq la & ql \leq lq & q \leq 1 \end{array}$$

Then $s(a+b+r+l)^\circ q \leq s(ab^\circ q + r + l)^\circ$.

The continuity assumption of [3] is expressed by $r^\circ q \leq qr^\circ$ in our setting.

The other two examples deal with while-programs. To define their semantics we first introduce tests.

3.2 Tests

In semirings, tests are elements ≤ 1 which represent conditions occurring in computations. For an arbitrary set S, they are introduced by means of two operations $\cdot$ and $'$ with axioms making the image $S' = \{x' \mid x \in S\}$ a Boolean algebra with greatest lower bound $\cdot$ and complement $'$ [30]. Any axiomatisation of Boolean algebras can be applied to this end; for concision we use Huntington's axioms which lead to the following definition. A *test algebra* is a structure $(S,\cdot,')$ satisfying the axioms

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

The last axiom states that S' is closed under the operation $\cdot$ and the remaining ones are associativity, commutativity and Huntington's special axiom. Then $(S',+,\cdot,',0,1)$ is a Boolean algebra with the order $x' \leq y' \Leftrightarrow x'y' = x'$, the least upper bound $x' + y' = (x''y'')'$, the least element $0 = x'x''$ for any x, and the greatest element $1 = 0'$. The extension $(S,+,\cdot,',0,1)$ thus obtained is also called a test algebra; elements of S' are *tests*.

A benefit of this axiomatisation is that it avoids introducing a separate sort for Boolean elements as, for example, in Kleene algebra with tests [36] without imposing additional constraints as, for example, in antidomain semirings [17].

For representing conditional statements and while-loops it is necessary to take the complement of tests. We do not make the whole set S a Boolean algebra because some models of programs are not closed under complements [32,25].

The Isabelle implementation defines a class for test algebras including the definite description $0 = (THE\ x.(\forall y.x = y'y''))$. It introduces 0 without imposing the uniqueness property $x'x'' = y'y''$ which can be derived from the axioms above. The defining property $0 = x'x''$ is then proved from the definite description and uniqueness.

In the remainder of this paper, the symbols p, q, r denote tests. A *test itering* is a structure $(S,+,\cdot,^\circ,',0,1)$ whose reduct $(S,+,\cdot,^\circ,0,1)$ is an itering and whose reduct $(S,+,\cdot,',0,1)$ is a test algebra. It follows that tests are preserved by and can be imported into iterations.

Theorem 4 *Let S be a test itering and let $x \in S$ and $p \in S'$ such that $px \leq xp$. Then $px^\circ = px^\circ p = p(px)^\circ$.*

The assumption $px \leq xp$ is equivalent to $px = pxp$.

3.3 Applications: Transformation of While-Programs

In test iterings we define the semantics of while-programs by

$$\begin{aligned} x\,;y &= xy \\ \mathsf{if}\ p\ \mathsf{then}\ x\ \mathsf{else}\ y &= px + p'y \\ \mathsf{if}\ p\ \mathsf{then}\ x &= px + p' \\ \mathsf{while}\ p\ \mathsf{do}\ x &= (px)^\circ p' \end{aligned}$$

A while-program is in *normal form* if it has the form x ; while p do y with while-free x and y. An element x *preserves* the test p if both $px \leq xp$ and $p'x \leq xp'$ hold. An element x *assigns p to q* if $x = x(pq + p'q')$ holds.

The third example is a split/merge loop theorem of Back and von Wright [3].

Theorem 5 *Let S be a test itering and let $x,y \in S$ and $p,q \in S'$ such that $p'y \leq yp'$. Then* while $p+q$ do (if p then x else y) = (while p do x) ; (while q do y).

While the original proof takes two pages of calculation, the Isabelle proof boils down to only two calls to the SMT solver Z3.

The fourth example is Kozen's algebraic version of the while-program normal form theorem [36]. Its proof uses the following program transformations to move while-programs out of each kind of program construct and hence into normal form.

Theorem 6 *Let S be a test itering and let $x_1,x_2,y_1,y_2,z_1,z_2 \in S$ and $p,q,r_1,r_2 \in S'$.*

- *Let z_1 assign p to q and let x_1, x_2, y_1, y_2 preserve q. Then*

$$\begin{aligned} &z_1\ ;\ \mathsf{if}\ p\ \mathsf{then}\ (x_1\ ;\ \mathsf{while}\ r_1\ \mathsf{do}\ y_1)\ \mathsf{else}\ (x_2\ ;\ \mathsf{while}\ r_2\ \mathsf{do}\ y_2) = \\ &z_1\ ;\ (\mathsf{if}\ q\ \mathsf{then}\ x_1\ \mathsf{else}\ x_2)\ ;\ \mathsf{while}\ qr_1 + q'r_2\ \mathsf{do}\ (\mathsf{if}\ q\ \mathsf{then}\ y_1\ \mathsf{else}\ y_2). \end{aligned}$$

- *Let z_1 assign p to q and let x_1, y_1 preserve q. Then*

$$\begin{aligned} &z_1\ ;\ \mathsf{while}\ p\ \mathsf{do}\ (x_1\ ;\ \mathsf{while}\ r_1\ \mathsf{do}\ y_1) = \\ &z_1\ ;\ (\mathsf{if}\ q\ \mathsf{then}\ x_1)\ ;\ \mathsf{while}\ q(p+r_1)\ \mathsf{do}\ (\mathsf{if}\ r_1\ \mathsf{then}\ y_1\ \mathsf{else}\ x_1). \end{aligned}$$

– *Let z_1 assign r_1 to q and let z_2 assign q to p and let $z_1z_2 = z_2z_1$. Let x_2, y_2, z_2 preserve q and let y_1, z_1, x_2, y_2 preserve p. Then*

$$\begin{array}{l} x_1 \,;\, z_1 \,;\, z_2 \,;\, (\mathsf{while}\ r_1\ \mathsf{do}\ y_1 \,;\, z_1) \,;\, x_2 \,;\, (\mathsf{while}\ r_2\ \mathsf{do}\ y_2) = \\ x_1 \,;\, z_1 \,;\, z_2 \,;\, (\mathsf{if}\ q\ \mathsf{then}\ (y_1 \,;\, z_1 \,;\, \mathsf{if}\ q'\ \mathsf{then}\ x_2)\ \mathsf{else}\ x_2) \,; \\ \mathsf{while}\ q + r_2\ \mathsf{do}\ (\mathsf{if}\ q\ \mathsf{then}\ (y_1 \,;\, z_1 \,;\, \mathsf{if}\ q'\ \mathsf{then}\ x_2)\ \mathsf{else}\ y_2). \end{array}$$

By repeatedly applying these program transformations it can be shown that every while-program, suitably augmented with assigning elements, is equivalent to a while-program in normal form under certain preservation assumptions.

The transformations explicitly state the source and target programs, the positions where assigning elements are inserted in them and the preservation assumptions. Also included is the commutativity assumption $z_1z_2 = z_2z_1$ for the assigning elements, which is not obvious in previous proofs. The Isabelle proofs are fairly extensive, requiring 40 calls to the SMT solver Z3 and the automatic theorem prover Metis.

3.4 Models

The generality of iterings is demonstrated by giving six models. We describe these models and the kinds of executions they can represent. The axiom $x0 = 0$ is again omitted in the following semiring-based structures.

The first model is that of *Kleene algebras* [35] which are semirings expanded by an operation * satisfying the axioms

$$1 + yy^* \leq y^* \qquad \begin{array}{l} z + yx \leq x \Rightarrow y^*z \leq x \\ z + xy \leq x \Rightarrow zy^* \leq x \end{array}$$

The operation * thus axiomatised satisfies the axioms of $^\circ$. In this model, which is typical for partial-correctness approaches [37,41], the operation $^\circ$ is characterised as a least fixpoint. Partial-correctness semantics describes only finite executions; infinite and aborting ones are ignored.

To benefit from our general results about test iterings and while-programs, Kleene algebras can be extended by tests using either the test algebras of Section 3.2, a second sort [36] or (anti)domain operations [14], see below.

All subsequent models are based on Kleene algebras and therefore share the above instance of $^\circ$, but each of them has another instance of $^\circ$, too. This implies that $^\circ$ is not determined uniquely by its axioms.

The second model is that of *omega algebras* [12] which are Kleene algebras expanded by an operation $^\omega$ satisfying the axioms

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

Every omega algebra is bounded by $\top = 1^\omega$ and the law $x^\omega\top = x^\omega$ holds. The itering axioms follow by instantiating $x^\circ = x^\omega 0 + x^*$. In this model, which is typical for general-correctness approaches [42,40,20], the operation $^\circ$ is a combination of a least and a greatest fixpoint [21–23]. General-correctness semantics describes finite and infinite executions independently; aborting executions are ignored or treated as non-terminating ones.

For handling two kinds of iteration in a uniform way, the notation $x \circ y$ is used in [12] to denote either x^*y or $x^\omega + x^*y$. This corresponds to $x^\circ y$ and illustrates that two different

itering structures are present in omega algebras. Our subsequent instances show that itering structures are present in other models as well.

The third model is that of omega algebras extended by the axiom $\top x = \top$ making $\top$ a left annihilator. In this case, every element x^ω is a left annihilator by $x^\omega y = x^\omega \top y = x^\omega \top = x^\omega$. Thus the itering axioms follow by instantiating $x^\circ = x^\omega + x^*$. In this model, which is typical for total-correctness approaches [11,28,39,29], the operation $^\circ$ is characterised as a greatest fixpoint. Total-correctness semantics ignores finite executions in the presence of infinite ones; aborting executions are again ignored or treated as non-terminating ones.

The fourth model is that of *demonic refinement algebras* [46] which are Kleene algebras expanded by an operation $^\Omega$ satisfying the axioms

$$1 + yy^\Omega = y^\Omega \qquad x \leq z + yx \Rightarrow x \leq y^\Omega z$$
$$y^\Omega 0 + y^* = y^\Omega$$

The operation $^\Omega$ thus axiomatised satisfies the axioms of $^\circ$. Demonic refinement algebras also model total correctness; in fact they are interdefinable with the previous model of omega algebras extended with $\top x = \top$ [33].

The fifth model is that of *extended designs* [31,25] which are elements of omega algebras expanded by *(anti)domain* operations a and d satisfying the axioms [17]

$$d(x) = a(a(x)) \qquad a(x)x = 0$$
$$a(xd(y)) = a(xy) \qquad d(x) + a(x) = 1$$

and an element L satisfying the axioms

$$d(x0)\mathsf{L} \leq x \qquad d(\mathsf{L}0) = 1 \qquad x\mathsf{L} \leq x0 + \mathsf{L}$$

The domain operation d represents the states in which a computation is enabled, that is, the states from which it can be executed [14]. These states are tests and the antidomain a is the Boolean complement of d. (The absence of executions from a state may be interpreted as an error in some models, but in general indicates partial relations, such as those described by [43].) The element L represents the endless loop, that is, the computation which has only infinite executions.

The test itering axioms follow by instantiating $x^\circ = d(x^\omega)\mathsf{L} + x^*$ and $x' = a(x)$. Extended designs go beyond general correctness by distinguishing finite, infinite and aborting executions, but like total correctness they ignore finite and infinite executions in the presence of aborting ones.

The sixth model, which we introduce in Section 4, lifts this restriction by treating finite, infinite and aborting executions independently. The following result summarises the above discussion.

Theorem 7 *Iterings have the following models:*

1. *Every Kleene algebra is an itering using $x^\circ = x^*$.*
2. *Every omega algebra is an itering using $x^\circ = x^\omega 0 + x^*$.*
3. *Every omega algebra with $\top x = \top$ is an itering using $x^\circ = x^\omega + x^*$.*
4. *Every demonic refinement algebra is an itering using $x^\circ = x^\Omega$.*
5. *Extended designs form a test itering using $x^\circ = d(x^\omega)\mathsf{L} + x^*$.*

All these instances have been proved in Isabelle using the subclass and sublocale mechanisms. As a consequence all results derived for iterings are automatically available in all of these models. In particular, this includes Theorems 1–6.

4 Finite, Infinite and Aborting Executions

In this section we discuss a model of computations which independently describes finite, infinite and aborting executions. This model reflects the observable difference between infinite and aborting executions, facilitates the statement of separate preconditions for these executions [26] and supports reasoning in situations where aborting executions are acceptable but infinite ones are not or vice versa. An operational semantics of computations with independent finite, infinite and aborting executions is given in [2]. The present paper contributes a relational model and an axiomatic semantics. We show that it satisfies the axioms of iteration given in Section 3, whence all results derived in that general setting carry over.

4.1 Relational Models of Computations

First we introduce a relational model of computations having independent finite, infinite and aborting executions. More precisely, a computation is represented as a matrix of relations, following [39,20,29,25]. We describe in this model the operations $+$, $\cdot$, * and $^\omega$ of omega algebra, an approximation order for the semantics of recursion, and an operation n that extracts the infinite executions of a computation. We also discuss connections to the models of Section 3.4 which impose restrictions on the executions they can represent.

Let Z be the set of states a computation can be in, for example, given by the possible values of program variables. A simple model of a computation is a relation over Z, that is, a subset R of the Cartesian product $Z \times Z$. The intuition is that $(x,y) \in R$ if and only if there is an execution of the computation R that starts in the state x and terminates in the state y. Relations are used instead of functions for the purpose of non-deterministic computations. This simple model represents finite executions but not infinite or aborting ones. It is therefore suitable for a partial-correctness semantics. Relations with an operation for reflexive transitive closure form a Kleene algebra.

To model finite, infinite and aborting executions, we use 3×3 matrices whose entries are relations over Z. The matrices have the following, fixed structure:

$$(P|Q|R) = \begin{pmatrix} \top & 0 & 0 \\ 0 & \top & 0 \\ P & Q & R \end{pmatrix}$$

The first two rows contain as constant entries the universal relation $\top = Z \times Z$ and the empty relation $0 = \emptyset$. Moreover the relations P and Q in the third row must be *vectors*, that is, $P;\top = P$ and $Q;\top = Q$ where ; denotes relational composition. A vector relates a state $x \in Z$ either to all states or to none, whence it corresponds to a set of states. No restriction is placed on the relation R in the matrix.

The matrix $(P|Q|R)$ represents the following computation. The entry P captures the set of states from which there are aborting executions. The entry Q does likewise for infinite executions. The entry R contains the state transitions effected by finite executions.

The matrix is structured so that many program constructs can be defined in terms of familiar matrix operations. For example, sequential composition is obtained by the matrix product, and the Kleene star is obtained by the standard automata-based construction [13]. These and further operations are elaborated in the following.

The non-deterministic choice between two computations $(P_1|Q_1|R_1)$ and $(P_2|Q_2|R_2)$ is given by componentwise union of the involved relations:

$$(P_1|Q_1|R_1) + (P_2|Q_2|R_2) = (P_1 \cup P_2 \mid Q_1 \cup Q_2 \mid R_1 \cup R_2)$$

By taking the componentwise subset order we obtain the semilattice order that models refinement of computations. Thus $(P_1|Q_1|R_1)$ refines $(P_2|Q_2|R_2)$ if and only if

$$(P_1|Q_1|R_1) \leq (P_2|Q_2|R_2) \Leftrightarrow P_1 \subseteq P_2 \wedge Q_1 \subseteq Q_2 \wedge R_1 \subseteq R_2$$

The least element of the semilattice order is the computation $(0|0|0)$ and $(\top|\top|\top)$ is its greatest element.

Sequential composition is given by the matrix product resulting in

$$(P_1|Q_1|R_1) \cdot (P_2|Q_2|R_2) = (P_1 \cup R_1;P_2 \mid Q_1 \cup R_1;Q_2 \mid R_1;R_2)$$

The precedence of ; is higher than that of $\cup$. For example, $P_1 \cup R_1;P_2$ describes that a sequential composition aborts if the first computation aborts or the first computation terminates in a state from which the second computation aborts. Thus aborting and infinite executions are not affected by subsequent computations. In particular, sequential composition has a number of left annihilators, one of which is $\mathsf{L} = (0|\top|0)$ representing the computation that has only infinite executions, the endless loop. The neutral element is the computation $(0|0|1)$, using the identity relation 1 over Z, which always terminates without changing the state.

Part of the structure of relations is thus lifted to the matrices. A crucial difference is that sequential composition of computations has no right annihilator, despite $R;0 = 0$ for every relation R. For example, $\mathsf{L} \cdot (0|0|0) = \mathsf{L}$. Nevertheless the computations with the operations $+$ and $\cdot$ form a semiring thanks to omission of the axiom $x0 = 0$. They even form a Kleene algebra and an omega algebra using

$$\begin{aligned}(P|Q|R)^* &= (R^*;P \mid R^*;Q \mid R^*)\\ (P|Q|R)^\omega &= (R^\omega \cup R^*;P \mid R^\omega \cup R^*;Q \mid R^\omega)\end{aligned}$$

These operations are derived using standard and typed matrix constructions [13,24]. On the right-hand side, R^* is the reflexive transitive closure of R and R^ω is the greatest fixpoint of $\lambda X.R;X$. The latter is a vector representing the states from which infinite R-transition paths exist.

For recursion we need the approximation order on computations, which is the following variant of the Egli-Milner order:

$$(P_1|Q_1|R_1) \sqsubseteq (P_2|Q_2|R_2) \Leftrightarrow Q_2 \subseteq Q_1 \wedge P_1 \subseteq P_2 \subseteq P_1 \cup Q_1 \wedge R_1 \subseteq R_2 \subseteq R_1 \cup Q_1$$

Thus infinite executions may be removed and aborting and finite executions may be added provided there are infinite executions. Instantiating this order with $P_1 = P_2$ gives the original Egli-Milner relation $Q_2 \subseteq Q_1 \wedge R_1 \subseteq R_2 \subseteq R_1 \cup Q_1$ [4,43,18] which does not consider aborting executions and thus applies to the general-correctness model discussed below.

For algebraically capturing the approximation order we use the operation n that extracts the infinite executions Q of a computation $(P|Q|R)$. The set of states in the vector Q shall be represented as a test, that is, we want $n((P|Q|R)) = (0|0|Q \cap 1)$. Recall that tests are elements below the identity $(0|0|1)$. In particular, sequential composition of a test $n(x)$ with a computation y restricts the executions of y to those starting in the set represented by $n(x)$. A reason why we choose the test $(0|0|Q \cap 1)$ instead of the vector $(0|Q|0)$ is that the latter can be obtained from the former by sequential composition with L, but not vice versa without an additional operation such as domain.

In the above model, the aborting, infinite and finite executions of a computation, given by the entries P, Q and R of its matrix, can be chosen independently of each other. Another model, namely extended designs, is obtained by replacing the constants in each matrix as in

$$\begin{pmatrix} \top & \top & \top \\ 0 & \top & 0 \\ P & Q & R \end{pmatrix}$$

and imposing the restrictions $P \subseteq Q$ and $P \subseteq R$ [31,25]. The entries P and Q are again required to be vectors. The main effect of the additional restrictions is that an aborting execution cannot be enforced, but entails all finite and infinite executions starting in the same state. For example, the computation $(\top|0|0)$ which has only aborting executions cannot be represented as an extended design.

Yet another model is obtained by restriction to computations of the form $(0|Q|R)$, that is, by imposing $P = 0$. This is general correctness, which has no notion of aborting executions, but can describe finite and infinite executions independently [4,10,44,34,5,43,18,19,42, 39,20]. In this case, computations may also be represented by the 2×2 matrix

$$\begin{pmatrix} \top & 0 \\ Q & R \end{pmatrix}$$

A total-correctness model [32,39,29] is obtained by replacing the constants in such matrices as in

$$\begin{pmatrix} \top & \top \\ Q & R \end{pmatrix}$$

and imposing the restriction $Q \subseteq R$. Then an infinite execution entails all finite executions starting in the same state.

The intuition underlying the choice of matrices for the individual models is explained in [39,20,29,25].

4.2 Axioms for the Infinite Executions

We now axiomatise the relational models of Section 4.1, in particular the infinite executions of a computation. An *n-semiring* $(S,+,\cdot,n,0,1,\mathsf{L},\top)$ is a bounded semiring $(S,+,\cdot,0,1,\top)$ expanded by an operation $n : S \to S$ and a constant L satisfying the axioms

$$\begin{array}{lrlll}
(n1) & n(0) = 0 & & (n5) & n(x) = n(x0)n(x) \\
(n2) & n(\top) = 1 & & (n6) & xn(y)\mathsf{L} = x0 + n(xy)\mathsf{L} \\
(n3) & n(x+y) = n(x)+n(y) & & (n7) & x \leq x0 + n(x\mathsf{L})\top \\
(n4) & n(n(x)y) = n(x)n(y) & &
\end{array}$$

The constants 0, 1 and $\top$ of the semiring represent the computations $(0|0|0)$, $(0|0|1)$ and $(\top|\top|\top)$, respectively. The constant L represents the endless loop, that is, the computation $(0|\top|0)$ that has only infinite executions. The element $n(x)$ represents the infinite executions of the computation x as a test, that is, as an element ≤ 1.

The n-semiring axioms hold in all models described in Section 4.1 except the simple partial-correctness model, and have the following rationale. Axioms $(n1)$ and $(n2)$ express that the computation 0 has no infinite executions and the computation $\top$ has infinite executions starting from each state, respectively. By axiom $(n3)$, the infinite executions of a

non-deterministic choice between two computations are given as the union of the individual infinite executions. Axiom $(n4)$ expresses that the infinite executions of a computation restricted to starting states $n(x)$ are given by intersecting with the set $n(x)$.

By axioms $(n1)$–$(n4)$ the image $n(S) = \{n(x) \mid x \in S\}$ of the operation n is closed under 0, 1, $+$ and $\cdot$. Moreover n is $\leq$-isotone since it is additive, whence $n(x) \leq 1$ and $n(x)0 = 0$ hold. Therefore $(n(S), +, \cdot, 0, 1)$ is a bounded semiring with right annihilator 0.

By adding axiom $(n5)$ we obtain $n(x)n(x) = n(x)$. Hence $(n(S), +, \cdot, 0, 1)$ is a bounded distributive lattice by Theorem II.10 in Birkhoff's book [6]. Moreover $n(1) = 0$ follows.

A consequence of adding axiom $(n6)$ is $n(x)\mathsf{L} \leq x$. The element $n(x)\mathsf{L}$ contains the infinite executions of x, see the characterisation below. By adding axiom $(n7)$ we also obtain $n(\mathsf{L}) = 1$. It follows that there is a Galois connection $n(x)\mathsf{L} \leq y \Leftrightarrow n(x) \leq n(y)$ between $n(S)$ and S with lower adjoint $\lambda p.p\mathsf{L}$ and upper adjoint n. Its significance is that $n(y)$ is the greatest test that, sequentially composed with L, is below y. This is the characterising property of the infinite executions of y in relational models, which any axiomatisation of n must satisfy. These and further consequences of n-semirings are summarised in the following result.

Theorem 8 *Let S be an n-semiring and $x, y \in S$. Then $(n(S), +, \cdot, 0, 1)$ is a bounded semiring with right annihilator 0 and a bounded distributive lattice. Moreover n is isotone, $\mathsf{L}x = \mathsf{L}$ and*

- $n(0) = n(1) = 0$,
- $n(\mathsf{L}) = n(\top) = 1$,
- $n(x\mathsf{L}) = n(x\top)$,
- $n(xy) = n(xn(y)\mathsf{L})$,
- $n(xn(y)) = n(x) = n(x0)$,
- $x0 + n(x\mathsf{L})\mathsf{L} = x\mathsf{L} \leq x0 + \mathsf{L}$,
- $n(x) \leq n(y) \Leftrightarrow n(x)\mathsf{L} \leq y$.

Axioms $(n6)$ and $(n7)$ are also used to establish that $\cdot$ is isotone with respect to the approximation order we introduce in Section 4.3. Counterexamples generated by Mace4 [38] witness that each of the axioms $(n1)$–$(n7)$ is independent of the others and the underlying semiring axioms.

The operation n facilitates two tasks: to access the infinite executions of a computation and to represent tests. For several of the models discussed in Section 3.4 we could serve these tasks by using the domain operation instead [20–22, 25]. However, domain semirings are not sufficient to describe computations having independent aborting, finite and infinite executions. This is because sequential composition, non-deterministic choice and domain cannot distinguish between aborting and infinite executions, but it is necessary to access the infinite executions of a computation to define the approximation order used for recursion. (The necessary information happens to be available for extended designs since in that model aborting executions entail infinite ones.) So while domain could still be used to induce tests, another operation has to be added for the infinite executions.

4.3 Approximation and Recursion

The approximation order $\sqsubseteq$ on computations, which instantiates to the variant of the Egli-Milner order in the model of Section 4.1, is defined as follows:

$$x \sqsubseteq y \Leftrightarrow x \leq y + n(x)\mathsf{L} \wedge y \leq x + n(x)\top$$

To obtain an intuition for this definition, consider a computation x that has infinite executions starting in every state. Then $n(x) = 1$ and $x \sqsubseteq y$ reduces to $x \leq y + \mathsf{L}$, meaning that y must have at least the aborting and finite executions of x and may have any infinite executions. On the other hand, if x has no infinite executions, then $n(x) = 0$ and $x \sqsubseteq y$ reduces to $x = y$: no executions may be added or removed.

Theorem 9 *Let S be an n-semiring. Then $\sqsubseteq$ is a partial order on S with least element L. Moreover the operations $+$ and $\cdot$ and $\lambda x.n(x)\mathsf{L}$ are $\sqsubseteq$-isotone. If S is an itering, the operation $^\circ$ is $\sqsubseteq$-isotone. If S is an omega algebra, the operation $^\omega$ is $\sqsubseteq$-isotone.*

The approximation order is suitable for defining the semantics of recursion in our new model as well as the extended-design and general-correctness models. Hence any results derived below also hold in these models. Different approximation orders are required for partial and total correctness, namely $\leq$ and $\geq$, respectively. A unified treatment of these approximation orders and recursion which covers partial, total and general correctness is given in [22] and generalised to extended designs and a model of non-strict computations in [27].

For the semantics of recursion we use a fixpoint theory which derives properties of fixpoints from their existence rather than from completeness of the underlying structure [16]. Let $f : S \to S$ be a function. Then μf, νf and ξf denote the $\leq$-least, $\leq$-greatest and $\sqsubseteq$-least fixpoints of f, provided they exist. In that case, the elements μf, νf and ξf satisfy the following characterising properties:

$$\begin{array}{ll} f(\mu f) = \mu f & f(x) = x \Rightarrow \mu f \leq x \\ f(\nu f) = \nu f & f(x) = x \Rightarrow \nu f \geq x \\ f(\xi f) = \xi f & f(x) = x \Rightarrow \xi f \sqsubseteq x \end{array}$$

In our model of computations, the semantics of the recursion specified by $f(x) = x$ is ξf, that is, the least fixpoint of f in the approximation order.

Including an axiom like $f(\mu f) = \mu f$ in Isabelle would force the existence of μf which is not guaranteed without completeness. Instead we provide the definite description $\mu f = (THE\ y. f(y) = y \wedge (\forall x. f(x) = x \Rightarrow y \leq x))$ as a definition. It introduces μf without imposing its existence. Uniqueness and the characterising properties of μf are then proved from the definite description and the assumption of existence. A similar handling applies to the other fixpoints. From these definitions a calculus is developed for least (pre)fixpoints and greatest (post)fixpoints including diagonal, exchange, fusion, rolling and square rules in the style of [1] but based on existence rather than completeness.

Denote by $x \sqcap y$ the $\sqsubseteq$-greatest lower bound of x and y, provided it exists. The following result gives conditions on the existence of $\sqsubseteq$-least fixpoints and shows how to calculate them. It generalises corresponding results for general correctness and extended designs [21, 22, 25].

Theorem 10 *Let S be an n-semiring and let $f : S \to S$ be $\leq$- and $\sqsubseteq$-isotone such that μf and νf exist. Then the following are equivalent:*

1. *ξf exists.*
2. *ξf and $\mu f \sqcap \nu f$ exist and $\xi f = \mu f \sqcap \nu f$.*
3. *ξf exists and $\xi f = \mu f + n(\nu f)\mathsf{L}$.*
4. *$\nu f \leq \mu f + n(\nu f)\top$.*
5. *$\mu f + n(\nu f)\mathsf{L} \sqsubseteq \nu f$.*
6. *$\mu f \sqcap \nu f$ exists and $\mu f \sqcap \nu f = \mu f + n(\nu f)\mathsf{L}$.*
7. *$\mu f \sqcap \nu f$ exists and $\mu f \sqcap \nu f \leq \nu f$.*

Condition 3 reduces the calculation of ξf to that of μf and νf, which are often easier to obtain as the semilattice order $\leq$ is less complex than the approximation order $\sqsubseteq$. It is typically inferred by establishing condition 4 that characterises the existence of ξf in terms of μf and νf.

4.4 Iteration

A special kind of recursion is the while-loop $\mathsf{while}\ p\ \mathsf{do}\ w$, which is the solution to its unfolding equation $x = pwx + p'$ using the complement p' of the condition p. We solve the more general equation $x = yx + z$ by calculating the $\sqsubseteq$-least fixpoint ξf of the function $f : S \to S$ given by $f(x) = yx + z$ for $y, z \in S$. To instantiate Theorem 10 for while-loops an additional assumption is needed that captures the interaction of * and $^\omega$ with n.

Theorem 11 *Let S be an n-semiring and omega algebra with $x^\omega \leq x^* n(x^\omega)\top$ for each $x \in S$. Let $y, z \in S$ and $f(x) = yx + z$. Then $\xi f = n(y^\omega)\mathsf{L} + y^* z$.*

The proof uses that f is $\leq$- and $\sqsubseteq$-isotone, $\mu f = y^* z$ and $\nu f = y^\omega + y^* z$. As yet it is unknown whether the additional assumption $x^\omega \leq x^* n(x^\omega)\top$ is independent of the other axioms.

Suggested by the previous result we let $y^\circ = n(y^\omega)\mathsf{L} + y^*$ and obtain $\xi f = y^\circ z$. The operation $^\circ$ thus defined satisfies the itering axioms, whence the new model is another instance of our general theory of iteration described in Section 3.

Theorem 12 *Let S be an n-semiring and omega algebra. Then S is an itering using $x^\circ = n(x^\omega)\mathsf{L} + x^*$.*

As a consequence all results derived for iterings are available in this model, too. The additional assumption of Theorem 11 is not required here, just to derive the semantics of iteration as a special case of recursion.

Further consequences about the interaction of * and $^\omega$ with n are summarised in the following result.

Theorem 13 *Let S be an n-semiring and omega algebra and let $x, y, z \in S$. Then*

- $x^* n(x)\mathsf{L} = x^* 0$,
- $x^* n(x^\omega)\top = x^* 0 + n(x^\omega)\top$,
- $n(x) \leq n(x^*) \leq n(x^\omega) = n(x^\omega y)$,
- $n(x) \leq n(z + yx) \Rightarrow n(x) \leq n(y^\omega + y^* z)$.

The last implication is similar to the induction axiom of omega algebra. Whether the dual $n(z + yx) \leq n(x) \Rightarrow n(y^* z) \leq n(x)$ follows is as yet unknown. This is reversed in omega algebras with domain, where $d(z + yx) \leq d(x) \Rightarrow d(y^* z) \leq d(x)$ holds [14] but the dual $d(x) \leq d(z + yx) \Rightarrow d(x) \leq d(y^\omega + y^* z)$ has not been derived, see the additional axiom for the divergence operation [15]. The reason for this asymmetry is the characterisation of $n(x)$ as a greatest test and of $d(x)$ as a least test satisfying certain properties.

4.5 Boolean Tests

The axioms of n-semirings induce a set of tests $n(S)$ which is a bounded distributive lattice. To make it a Boolean algebra, so as to inherit our general results about while-programs, we

add the Boolean complement $\bar{n}$ of n. This is achieved by introducing the operation $\bar{n} : S \to S$ such that $\bar{n}(x) + n(x) = 1$ and $\bar{n}(x)n(x) = 0$. Hence $\bar{n}(x)$ represents the set of states from which the computation x has no infinite executions.

After adding the complementing axioms to those of n, it is possible to reduce their number, similarly to the case of (anti)domain [17]. Thus an *$\bar{n}$-semiring* $(S,+,\cdot,\bar{n},n,0,1,\mathsf{L},\top)$ is a bounded semiring $(S,+,\cdot,0,1,\top)$ expanded by operations $\bar{n},n : S \to S$ and a constant L satisfying the axioms

$$\begin{array}{lll} (a1) & \bar{n}(x)+n(x) = 1 & (a4) \quad \bar{n}(x) = \bar{n}(x0) \\ (a2) & \bar{n}(x+y) = \bar{n}(x)\bar{n}(y) & (a5) \quad xn(y)\mathsf{L} = x0+n(xy)\mathsf{L} \\ (a3) & \bar{n}(\bar{n}(x)y) = n(x)+\bar{n}(y) & (a6) \quad \bar{n}(x\mathsf{L})x \leq x0 \end{array}$$

The following result shows that $\bar{n}$-semirings provide the intended structure.

Theorem 14 *Let S be an $\bar{n}$-semiring and $x \in S$. Then S is an n-semiring and a test algebra with complement $x' = \bar{n}(x\mathsf{L})$. In particular, $n(S) = \bar{n}(S)$ is a Boolean algebra, $\bar{n}(x)n(x) = 0$ and $\bar{n}$ is antitone. Moreover $n(x) = \bar{n}(\bar{n}(x)\mathsf{L})$ and $\bar{n}(x) = n(\bar{n}(x)\mathsf{L})$.*

By the following consequence, while-programs can be defined as in Section 3.3 and all of the results shown there also hold in the new model.

Theorem 15 *Let S be an $\bar{n}$-semiring and omega algebra. Then S is a test itering using $x^\circ = n(x^\omega)\mathsf{L}+x^*$ and $x' = \bar{n}(x\mathsf{L})$.*

4.6 Modal Operators

For the domain operation it is possible to define modal diamond and box operators [41]. Given an element x and a test p, the diamond operator yields $d(xp)$. In partial-correctness models, this captures the set of states from which there is an execution of x to a state in p, that is, the preimage of p under x. Dually, the box operator yields the states from which all executions of x go to p, and hence corresponds to the weakest liberal precondition. In other computation models, box describes the weakest precondition or variants thereof [42, 26].

We introduce modal diamond and box operators in $\bar{n}$-semirings. They are defined by

$$\begin{aligned} |x\rangle y &= n(xy\mathsf{L}) \\ |x]y &= \bar{n}(x\bar{n}(y\mathsf{L})\mathsf{L}) \end{aligned}$$

Typically the second argument of these operators is a test p. Then $|x\rangle p$ yields the set of states from which x has infinite executions or finite executions terminating in p. Moreover $|x]p$ yields the set of states from which all finite executions of x terminate in p and there are no infinite executions. Diamond and box satisfy the following distribution, duality, induction and unfold properties.

Theorem 16 *Let S be an $\bar{n}$-semiring and let $x,y,z \in S$ and $p,q \in n(S)$. Then*

$$\begin{array}{rlrl} |x+y\rangle z &= |x\rangle z+|y\rangle z & |x+y]z &= |x]z \cdot |y]z \\ |x\rangle(y+z) &= |x\rangle y+|x\rangle z & |x](pq) &= |x]p \cdot |x]q \\ |xy\rangle z &= |x\rangle|y\rangle z = |x\rangle(yz) & |xy]z &= |x]|y]z \\ |x\rangle y &= (|x]y')' & |x]y &= (|x\rangle y')' \end{array}$$

using $y' = \bar{n}(y\mathsf{L})$ from the induced test algebra. If S is an omega algebra,

$$\begin{aligned} |x]p \cdot q \leq p &\Rightarrow |x^\omega + x^*]q \leq p \\ |x^\omega + x^*]p &= p \cdot |px]|x^\omega + x^*]p \\ |x^*]p &= p \cdot |px]|x^*]p \\ |x^\omega + x^*]p &= p \cdot |x^\omega + x^*](p' + |x]p) \end{aligned}$$

The latter facts can be dualised as well. The last property is a version of Segerberg's formula with infinite iterations.

We give several applications that show how to reason about programs using the modal operators. They are instances of the following general result.

Theorem 17 *Let S be an $\bar{n}$-semiring and omega algebra and let $x \in S$ and $p,q,r \in n(S)$. Then*

- $|x^\circ\rangle p = |(p'x)^\circ\rangle p = |\mathsf{while}\ p'\ \mathsf{do}\ x\rangle p$,
- $qp\mathsf{L} \leq xp\mathsf{L} \wedge p \leq q + r \Rightarrow p \leq |\mathsf{while}\ q\ \mathsf{do}\ x\rangle r$,

using $x^\circ = n(x^\omega)\mathsf{L} + x^*$ and $p' = \bar{n}(p\mathsf{L})$ and while-programs from the induced test itering.

The first property has the following interpretation. Its left-hand side $|x^\circ\rangle p$ describes a nondeterministic iteration to reach a state in p. This can be optimised to the deterministic loop $|\mathsf{while}\ p'\ \mathsf{do}\ x\rangle p$ which stops as soon as p is reached.

One instance of the second property is given by the following program x and conditions p, q, r with two integer variables a and b:

$$\begin{aligned} x &= (a := a/b) & \qquad q &= (a \geq 1) \\ p &= (b \geq 1) & \qquad r &= (a < 1) = q' \end{aligned}$$

Then $qp\mathsf{L} \leq p\mathsf{L} \leq pxp\mathsf{L} \leq xp\mathsf{L}$ holds because $p\mathsf{L}$ has only infinite executions starting in a state with $b \geq 1$, and so has $pxp\mathsf{L}$ since the assignment $a := a/b$ terminates and does not change the value of b. Moreover clearly $p \leq 1 = q + q' = q + r$. By Theorem 17 the execution of the program $\mathsf{while}\ a \geq 1\ \mathsf{do}\ a := a/b$ in a state with $b \geq 1$ does not terminate or terminates in a state satisfying $a < 1$. Because the loop is deterministic, this implies that it does not abort.

Another instance of the second property uses $p = (a \geq 1 \wedge b = 1)$ and $r = 0$ while x and q remain as above. Then $qp\mathsf{L} \leq p\mathsf{L} = pxp\mathsf{L} \leq xp\mathsf{L}$ because in a state with $b = 1$ the assignment $a := a/b$ has no effect. Moreover clearly $p \leq q = q + r$. By Theorem 17 the execution of the program $\mathsf{while}\ a \geq 1\ \mathsf{do}\ a := a/b$ in a state with $b = 1$ does not terminate.

5 Conclusion

The investigated generalisation of Kleene algebra facilitates the description of iteration and the unification of program transformations for several relational computation models. This is achieved by replacing the induction axioms with simulation axioms that can be instantiated by various fixpoints, yet suffice for general laws refining and transforming iterations and while-programs. Results derived in this setting hold in six computation models including partial, total and general correctness and extensions, which represent different kinds of executions with varying precision.

In one of these models, a computation comprises finite, infinite and aborting executions in an orthogonal way described by a matrix of relations. This representation makes it possible to define program constructs in terms of standard matrix operations. Abstracting from

this model, approximation, recursion and iteration are algebraically described using an operation that extracts the infinite executions of a computation. Its axioms apply to several computation models enabling a unified treatment. At the same time, this operation is an alternative to the domain operation for inducing tests, which represent conditions in programs. Related modal operators describe variants of preimage and weakest preconditions.

Acknowledgements I thank Bernhard Möller, Georg Struth and the anonymous referees for providing helpful comments.

References

1. C. J. Aarts, R. C. Backhouse, E. A. Boiten, H. Doornbos, N. van Gasteren, R. van Geldrop, P. F. Hoogendijk, E. Voermans, and J. van der Woude. Fixed-point calculus. *Information Processing Letters*, 53(3):131–136, 1995.
2. K. R. Apt, F. S. de Boer, and E.-R. Olderog. *Verification of Sequential and Concurrent Programs*. Springer, third edition, 2009.
3. R. J. R. Back and J. von Wright. Reasoning algebraically about loops. *Acta Informatica*, 36(4):295–334, 1999.
4. J. W. de Bakker. Semantics and termination of nondeterministic recursive programs. In S. Michaelson and R. Milner, editors, *Automata, Languages and Programming: Third International Colloquium*, pages 435–477. Edinburgh University Press, 1976.
5. R. Berghammer and H. Zierer. Relational algebraic semantics of deterministic and nondeterministic programs. *Theoretical Computer Science*, 43:123–147, 1986.
6. G. Birkhoff. *Lattice Theory*, volume XXV of *Colloquium Publications*. American Mathematical Society, third edition, 1967.
7. J. C. Blanchette, S. Böhme, and L. C. Paulson. Extending Sledgehammer with SMT solvers. In N. Bjørner and V. Sofronie-Stokkermans, editors, *Automated Deduction: CADE-23*, volume 6803 of *Lecture Notes in Computer Science*, pages 116–130. Springer, 2011.
8. 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.
9. S. L. Bloom and Z. Ésik. *Iteration Theories: The Equational Logic of Iterative Processes*. Springer, 1993.
10. M. Broy, R. Gnatz, and M. Wirsing. Semantics of nondeterministic and noncontinuous constructs. In F. L. Bauer and M. Broy, editors, *Program Construction*, volume 69 of *Lecture Notes in Computer Science*, pages 553–592. Springer, 1979.
11. J.-L. De Carufel and J. Desharnais. Demonic algebra with domain. In R. Schmidt, editor, *Relations and Kleene Algebra in Computer Science*, volume 4136 of *Lecture Notes in Computer Science*, pages 120–134. Springer, 2006.
12. 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.
13. J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.
14. J. Desharnais, B. Möller, and G. Struth. Kleene algebra with domain. *ACM Transactions on Computational Logic*, 7(4):798–833, 2006.
15. J. Desharnais, B. Möller, and G. Struth. Algebraic notions of termination. *Logical Methods in Computer Science*, 7(1:1):1–29, 2011.
16. J. Desharnais, B. Möller, and F. Tchier. Kleene under a modal demonic star. *Journal of Logic and Algebraic Programming*, 66(2):127–160, 2006.
17. J. Desharnais and G. Struth. Internal axioms for domain semirings. *Science of Computer Programming*, 76(3):181–203, 2011.
18. H. Doornbos. A relational model of programs without the restriction to Egli-Milner-monotone constructs. In E.-R. Olderog, editor, *Programming Concepts, Methods and Calculi*, pages 363–382. North-Holland Publishing Company, 1994.
19. S. Dunne. Recasting Hoare and He's Unifying Theory of Programs in the context of general correctness. In A. Butterfield, G. Strong, and C. Pahl, editors, *5th Irish Workshop on Formal Methods*, Electronic Workshops in Computing. The British Computer Society, 2001.

20. 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.
21. 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.
22. W. Guttmann. Unifying recursion in partial, total and general correctness. In S. Qin, editor, *Unifying Theories of Programming, Third International Symposium, UTP 2010*, volume 6445 of *Lecture Notes in Computer Science*, pages 207–225. Springer, 2010.
23. W. Guttmann. Fixpoints for general correctness. *Journal of Logic and Algebraic Programming*, 80(6):248–265, 2011.
24. W. Guttmann. Towards a typed omega algebra. In H. de Swart, editor, *Relational and Algebraic Methods in Computer Science*, volume 6663 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 2011.
25. W. Guttmann. Extended designs algebraically. *Science of Computer Programming*, 2012. To appear.
26. 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.
27. W. Guttmann. Unifying lazy and strict computations. In T. G. Griffin and W. Kahl, editors, *Relational and Algebraic Methods in Computer Science*, Lecture Notes in Computer Science. Springer, 2012. To appear.
28. W. Guttmann and B. Möller. Modal design algebra. In S. Dunne and W. Stoddart, editors, *Unifying Theories of Programming*, volume 4010 of *Lecture Notes in Computer Science*, pages 236–256. Springer, 2006.
29. W. Guttmann and B. Möller. Normal design algebra. *Journal of Logic and Algebraic Programming*, 79(2):144–173, 2010.
30. 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.
31. I. J. Hayes, S. E. Dunne, and L. Meinicke. Unifying theories of programming that distinguish non-termination 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.
32. C. A. R. Hoare and J. He. *Unifying theories of programming*. Prentice Hall Europe, 1998.
33. P. Höfner, B. Möller, and K. Solin. Omega algebra, demonic refinement algebra and commands. In R. Schmidt, editor, *Relations and Kleene Algebra in Computer Science*, volume 4136 of *Lecture Notes in Computer Science*, pages 222–234. Springer, 2006.
34. D. Jacobs and D. Gries. General correctness: A unification of partial and total correctness. *Acta Informatica*, 22(1):67–83, 1985.
35. D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.
36. D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, 19(3):427–443, 1997.
37. D. Kozen. On Hoare logic and Kleene algebra with tests. *ACM Transactions on Computational Logic*, 1(1):60–76, 2000.
38. W. McCune. Mace4 reference manual and guide. Technical Memorandum ANL/MCS-TM-264, Mathematics and Computer Science Division, Argonne National Laboratory, 2003. Mace4 is available at http://www.cs.unm.edu/~mccune/mace4/.
39. B. Möller. The linear algebra of UTP. In T. Uustalu, editor, *Mathematics of Program Construction*, volume 4014 of *Lecture Notes in Computer Science*, pages 338–358. Springer, 2006.
40. B. Möller. Kleene getting lazy. *Science of Computer Programming*, 65(2):195–214, 2007.
41. B. Möller and G. Struth. Algebras of modal operators and partial correctness. *Theoretical Computer Science*, 351(2):221–239, 2006.
42. 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.
43. G. Nelson. A generalization of Dijkstra's calculus. *ACM Transactions on Programming Languages and Systems*, 11(4):517–561, 1989.
44. D. Parnas. A generalized control structure and its formal definition. *Communications of the ACM*, 26(8):572–581, 1983.
45. L. C. Paulson and J. C. Blanchette. Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In G. Sutcliffe, E. Ternovska, and S. Schulz, editors, *Proceedings of the 8th International Workshop on the Implementation of Logics*, pages 3–13, 2010.
46. J. von Wright. Towards a refinement algebra. *Science of Computer Programming*, 51(1–2):23–45, 2004.