Normal Design Algebra

Walter Guttmann[a], Bernhard Möller*,[b]

[a] *Institut für Programmiermethodik und Compilerbau, Universität Ulm, D-89069 Ulm, Germany*
[b] *Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany*

Abstract

We generalise the designs of the Unifying Theories of Programming (UTP) by defining them as matrices over semirings with ideals. This clarifies the algebraic structure of designs and considerably simplifies reasoning about them, for example, since they form a Kleene and omega algebra and a test semiring. We apply our framework to investigate symmetric linear recursion and its relation to tail-recursion. This substantially involves Kleene and omega algebra as well as additional algebraic formulations of determinacy, invariants, domain, pre-image, convergence and Noetherity. Due to the uncovered algebraic structure of UTP designs, all our general results also directly apply to UTP.

Key words: Unifying Theories of Programming, semantics, semiring, Kleene algebra, omega algebra, fixpoint, linear recursion

1. Introduction

The Unifying Theories of Programming (UTP), developed in [26], model the termination behaviour of programs using two special variables ok and ok' that express whether a program has been started and has terminated. Specifications and programs are identified with predicates relating the initial values v of variables to their final values v'; moreover, ok and ok' may occur freely in predicates. Using these variables, Hoare and He introduce *designs*, that is, predicates of the form

$$P \vdash Q \;\Leftrightarrow_{\mathrm{def}}\; ok \wedge P \Rightarrow ok' \wedge Q,$$

with ok and ok' not occurring in P or Q. The intended use is an assumption/commitment style of specification: if the assumption P holds then every computation admitted by the design will eventually terminate so that the commitment Q holds. In particular, designs reflect a total correctness view.

In the general case, UTP allows the assumption P to involve both the initial and final values of the program variables. A subclass that is interesting for a number of reasons is that of *normal* designs in which P is a *condition*, only allowed to depend on the input values of variables.

Our preceding work [21, 35] has established a general algebraic treatment of designs and of the more liberal predicates known as *prescriptions* [15] that reflect a general correctness view. Specifically, our approach no longer mentions the 'unobservable' variables ok and ok'; in fact it is even completely variable-free and hence does not need to work with substitutions. This makes calculations not only easier, but also less error-prone and more compact.

In its first part, the present paper continues and extends that research by developing the simpler algebraic framework of *ideal semirings*, tailored to the particular case of (normal) designs. While still strictly more general than the original, purely relational UTP semantics, it exhibits the algebraic structure of designs more clearly and allows a much simpler derivation of explicit characterisations of operators on designs.

*Corresponding author
Email addresses: walter.guttmann@uni-ulm.de (W. Guttmann), bernhard.moeller@informatik.uni-augsburg.de (B. Möller)

In particular, the generalised normal designs again form an ideal semiring and can be extended by operators for finite and infinite iteration to a weak Kleene and omega algebra. Moreover, we show that they can be made into a test algebra and enriched by the modal operators diamond and box. For termination analysis we deal with the convergence and divergence operators that characterise the program states from which either no or at least one infinite computation is possible. The operators are defined in a way that enables the use of first-order theorem provers to obtain results automatically, see [28].

In the second part of the paper the benefits of our algebraic approach are further demonstrated by a number of investigations on transformations between special linear recursions, including both non-tail and tail recursions. Such investigations have hitherto been performed neither for the concrete case of UTP nor in the more general semiring setting. Since this work is carried out at the more abstract level of Kleene and omega algebras, the transformations are not only valid for UTP but also, for instance, for predicate transformer models, trace-based models and demonic refinement algebra [45]. Furthermore, algebraic formulations of determinacy, domain and invariants are applied during this task. The investigation may also be seen as a continuation of classical work on schematic program transformation, such as [17], carried out with full algebraic rigour.

The paper is organised as follows. In Section 2 we propose axioms for (normal) designs and show that they induce a weak Kleene and omega algebra and an ideal semiring. In contrast to previous axiomatisations, the new axioms are based on the established ring-theoretic concept of ideals and perfectly suited for the calculation with designs using their matrix representation. The axioms together with the matrix representation allow a concise derivation of the Kleene star and omega operations for normal designs. We then generalise [26, Theorem 3.1.6] to our setting and use it to extend star and omega to general designs.

Most facts derived in Section 2 have already been established in previous works, such as [26, 21]. Our contribution is to generalise them to the new axioms and to substantially simplify the proofs. In contrast, most facts derived in the later sections are new.

After the foundations have been laid, in Section 3 we show how to apply the framework of Kleene algebra and omega algebra to relate tail-recursion to linear recursion and different kinds of linear recursions. By using the model of Section 2, our results are considerably more general than in plain UTP and hence apply to a wider range of structures. By replacing conditions with tests, our calculations also become more simple. We moreover deal with both the least and the greatest fixpoints.

In Section 4 we apply our algebraic techniques to symmetric linear recursion. Our general framework, hence also UTP, is extended by algebraic notions of domain, pre-image, determinacy, invariants and convergence. We show how to implement both the recursion's least and greatest fixpoints by two consecutive while-loops each. We finally discuss axioms for symmetric linear recursion along the lines of the axioms for Kleene/omega algebra.

In Section 5, we use the convergence operator to discuss Noetherity. Assuming an atomistic test algebra, we show that progressive boundedness and progressive finiteness coincide for deterministic elements.

This paper is a revised and enhanced edition of [22] including also results from [19, Chapter 2].

2. Star and omega designs

In this section we derive the Kleene star and omega operations for generalised designs. While the result already appears in [21], the present generalisation is based on a modified set of axioms, and the new derivation is significantly shortened by using the matrix representation of [35]. The star and omega operations are important since they enable the application of general algebraically derived results to UTP, as shown in Sections 3 and 4.

To model the essence of normal designs, condition semirings have been introduced in [21]. They are more general than the algebras of predicates or relations used in [26], that is, they impose fewer axioms. This has provided insight into the general properties of designs and their operators, connections between UTP and other theories of total correctness of programs, and concise closed expressions for the semantics of the demonic while loop.

In Section 2.1 we propose a modified set of axioms that reflects the traditional nomenclature from ring theory, and investigate the connection to the former definition. We then define designs and normal designs

as matrices of elements governed by our new axioms and point out why the new axiomatisation is adequate for this purpose.

Since it is well-known how to lift the Kleene star operation to matrix algebras [30], the matrix model also lends itself to showing that normal designs form a weak Kleene algebra in Section 2.3. Using a similar lifting in Section 2.4, we show that normal designs also form a weak omega algebra. The Kleene star and omega operations are given explicitly. In Section 2.6 we generalise [26, Theorem 3.1.6] that describes the least and greatest fixpoints of functions on (not necessarily normal) designs. Our result is finally applied to derive the star and omega operations for these designs.

2.1. Axioms for conditions

In [21] normal designs are modelled as commands over condition semirings, adapting the axioms of commands over test semirings studied in [36]. In the following, we base our axiomatisation on the established ring-theoretic concept of ideals, generalised to semirings as in [23]. Semirings with their operations $+$ and $\cdot$ axiomatise the central concepts of choice and sequential composition that occur in many branches of computer science.

Definition 2.1.

1. A *weak semiring* is a structure $(S, +, 0, \cdot, 1)$ such that
   - $(S, +, 0)$ is a commutative monoid,
   - $(S, \cdot, 1)$ is a monoid,
   - the operation $\cdot$ distributes over $+$ in both arguments and
   - 0 is a left annihilator, that is, $0 \cdot x = 0$.
2. A weak semiring is *idempotent* if $+$ is, that is, if $x + x = x$.
3. In an idempotent weak semiring the relation $x \leq y \Leftrightarrow_{\text{def}} x + y = y$ is a partial order, called the *natural order* on S.
4. A *semiring* is a weak semiring in which 0 is also a right annihilator, that is, $x \cdot 0 = 0$.

The $\cdot$ operation is extended elementwise to sets $A, B \subseteq S$ by $A \cdot B =_{\text{def}} \{a \cdot b \mid a \in A \wedge b \in B\}$ and $A \cdot b =_{\text{def}} A \cdot \{b\}$ for $b \in S$. We frequently omit the $\cdot$ notation and abbreviate $a \cdot b$ to ab.

In an idempotent weak semiring $+$ can be interpreted as (angelic) choice where 0 models the program with no transitions, and $\cdot$ as sequential composition where 1 models the program *skip*. The natural order $x \leq y$ expresses that x refines y, since y offers all the possible choices of x but maybe more. Its least element is 0; moreover $+$ and $\cdot$ are isotone with respect to $\leq$ and $a + b$ is the least upper bound or *join* of a and b with respect to $\leq$.

A prominent example of an idempotent semiring is given by the set of binary relations $A \leftrightarrow A =_{\text{def}} \mathscr{P}(A \times A)$ on a set A with union $\cup$ as $+$, the empty relation $\emptyset$ as 0, relational composition ; as $\cdot$ and the identity relation as 1. Another example is $(A^*, \cup, \emptyset, \cdot, \{\varepsilon\})$, the set of formal languages over A with language concatenation $\cdot$ and empty word ε. In both cases the natural order $\leq$ coincides with the subset relation $\subseteq$.

As in the relational interpretation, we can model the commitment parts of designs by general semiring elements. To model their assumption parts, we need special semiring elements that play the role of input conditions. In UTP these are represented as *vectors*, that is, as right-universal relations having the form $R_B =_{\text{def}} B \times A$ for a subset $B \subseteq A$ of the base set A; if the elements of A are thought of as states then R_B characterises exactly the input states in B. To prepare our axiomatisation of conditions, let us list some properties that are typical of conditions represented as vectors:

1. The sequential composition of an arbitrary relation and a condition yields a condition again.
2. An arbitrary relation is input-restricted by conjoining it, that is, forming its *meet* $\sqcap$ or greatest lower bound, with a condition.
3. This restriction distributes over union in both arguments.
4. Restriction by the universal condition is no restriction at all.

5. Conditions can be used for Boolean reasoning, since they form a Boolean algebra.
6. Conditions are invariant under post-composition with the universal relation $A \times A$.

It turns out that these properties are sufficient for an abstract axiomatisation of conditions. Since for many results already properties 1–5 suffice, we first deal with these only; property 6 will be added later. Property 1 can be rephrased by saying that the set of all conditions is a left ideal of the semiring under consideration. This motivates the following definition in which T abstracts the set of conditions.

Definition 2.2. A structure $(S, T, +, 0, \cdot, 1, \sqcap, \top, \overline{\ })$ is an *ideal semiring* if the following properties hold.

- $(S, +, 0, \cdot, 1)$ is an idempotent weak semiring having a greatest element $\top$.
- T is a left ideal of S, that is,
  - $(T, +, 0)$ is a submonoid of $(S, +, 0)$ and
  - $S \cdot T \subseteq T$.
- The *restriction operation* $\sqcap : T \times S \to S$ distributes over $+$, that is,
  - $\forall a \in S : \forall t, u \in T : (t + u) \sqcap a = (t \sqcap a) + (u \sqcap a)$ and
  - $\forall a, b \in S : \forall t \in T : t \sqcap (a + b) = (t \sqcap a) + (t \sqcap b)$.
- $\forall a \in S : \top \sqcap a = a$.
- $(T, +, 0, \sqcap, \top, \overline{\ })$ is a Boolean algebra; in particular, $0 \in T$ and $\top \in T$.

In the remainder we abbreviate ideal semiring structures to (S, T). An ideal semiring is *strict* if the underlying weak semiring is a semiring, that is, if 0 is both a left and a right annihilator. In the remainder we consider $\sqcap$ to bind stronger than $+$.

Over an ideal semiring, the assumption part of a design will be taken from the set T and its commitment part from the set S. This makes sense because of the following concrete instance: let A be a set, then the relations $S =_{\text{def}} A \leftrightarrow A$ form a (strict) ideal semiring with $\top = A \times A$, the set $S \cdot \top$ of vectors as the left ideal and intersection $\cap$ as $\sqcap$. Another example of an ideal semiring is given in Corollary 2.9 by the normal designs.

Consider an ideal semiring (S, T). Since T is a left ideal of S, it follows that T is also a subsemiring (without 1) of S. But T has another semiring structure, by virtue of being a Boolean algebra, with $+$ as addition and $\sqcap$ as composition. We can therefore relate T to the concept of a module from ring theory [23]. The following lemma also gives a few properties of restriction and shows that $t \sqcap a$ is the meet of t and a.

Lemma 2.3. *In an ideal semiring (S, T),*

1. *$\sqcap$ is isotone in both arguments,*
2. *$\sqcap$ is the greatest lower bound operation on $T \times S$ ordered by $\leq$ componentwise,*
3. *the shunting rule $t \sqcap a \leq b \Leftrightarrow a \leq \overline{t} + b$ holds for all $t \in T$ and $a, b \in S$ and*
4. *S is a unitary left T-semimodule with scalar multiplication $\sqcap$.*

Proof. Let $a, b \in S$ and $t, u \in T$.

1. The claim follows immediately from the distributivity axioms of $\sqcap$.
2. First, $t \sqcap a \leq t \sqcap \top = t$ by Part 1 and Boolean algebra. Second, $t \sqcap a \leq \top \sqcap a = a$ by Part 1 and an axiom. Third, if $b \leq t$ and $b \leq a$, then $b = \top \sqcap b = (\overline{t} + t) \sqcap b = \overline{t} \sqcap b + t \sqcap b \leq \overline{t} \sqcap t + t \sqcap a = t \sqcap a$ by axioms, Boolean algebra and Part 1.
3. The part ($\Rightarrow$) follows by $a = (\overline{t} + t) \sqcap a = \overline{t} \sqcap a + t \sqcap a \leq \overline{t} + b$ using Part 2. The part ($\Leftarrow$) follows by $t \sqcap a \leq t \sqcap (\overline{t} + b) = t \sqcap \overline{t} + t \sqcap b \leq b$ using Parts 1 and 2.
4. It remains to show the associative law $(t \sqcap u) \sqcap a = t \sqcap (u \sqcap a)$ since the distributive and unitary laws are already axioms. The calculations use Parts 1 and 2 several times.

$*$ $(t \sqcap u) \sqcap a \leq t \sqcap u \leq t$ and $(t \sqcap u) \sqcap a \leq u \sqcap a$, hence $(t \sqcap u) \sqcap a \leq t \sqcap (u \sqcap a)$.

$*$ $t \sqcap (u \sqcap a) \leq t \sqcap u$ and $t \sqcap (u \sqcap a) \leq u \sqcap a \leq a$, hence $t \sqcap (u \sqcap a) \leq (t \sqcap u) \sqcap a$. □

Thus the elements of T in an ideal semiring (S, T) can be seen as scalars acting via $\sqcap$ on the elements of the semiring S. At the same time the associative law reflects a behaviour typical of a restriction operation: iterated restriction is equivalent to a single restriction with the conjoined restricting conditions. As a consequence of Lemma 2.3.4, (S, T) may be characterised by stating that T is a Boolean algebra and a left ideal of S, and S is a unitary left T-semimodule, all with respect to the appropriate operations.

Ideal semirings are sufficient for representing designs, whereas for *normal* designs, that is, designs satisfying the healthiness condition H3 of [26], we need the subclass of ideal condition semirings. As stated above, in the relational calculus any condition t is a right-universal relation and hence invariant under post-composition with the universal relation $\top$, that is, $t \cdot \top = t$. As will be shown below, it suffices to require that $t = u \cdot \top$ for some $u \in T$. This motivates the following definition.

Definition 2.4. An *ideal condition semiring* is an ideal semiring (S, T) where additionally $T \subseteq T \cdot \top$ holds. In this case the elements of T are called *conditions*. A semiring S with greatest element $\top$ is *replete* if $S \cdot \top$ is a Boolean algebra.

Two examples of ideal condition semirings are the previously mentioned ideal semirings, namely the relations (with vectors as conditions) and the normal designs. An example satisfying the condition property $T \subseteq T \cdot \top$, but not the ideal property $S \cdot T \subseteq T$ are the normal prescriptions of [15] used in general correctness semantics.

Observe that in an ideal condition semiring $S \cdot \top \subseteq S \cdot T \subseteq T$ holds, since $\top \in T$. Moreover, the set $S \cdot \top$ consists of all elements that are invariant under right composition with $\top$. If $S \cdot \top$ is a Boolean algebra, it could be a proper subset of T and certainly is another candidate for the condition set of an ideal semiring over S. In the context of Kleene algebra with tests, one similarly considers taking all elements ≤ 1 or just a Boolean subalgebra of them as tests [31]. We show below that such a question does not arise here since in an ideal condition semiring $S \cdot \top$ coincides with the full set T of conditions, whence we choose the term *replete*. In [21] the now inappropriate term 'ideal-closed' is used.

Lemma 2.5. *Let (S, T) be an ideal condition semiring.*

1. *$S \cdot \top = T$, hence S is replete.*
2. *$\forall t \in T : t \cdot \top = t$.*
3. *$\forall t \in T : \forall a, b \in S : (t \sqcap a) \cdot b = t \sqcap (a \cdot b)$.*
   In particular, for $a = 1$ we obtain $\forall t \in T : \forall b \in S : (t \sqcap 1) \cdot b = t \sqcap b$.

Proof. Let $a, b \in S$ and $t \in T$.

1. $S \cdot \top \subseteq S \cdot T \subseteq T \subseteq T \cdot \top \subseteq S \cdot \top$, hence $S \cdot \top = T$ is a Boolean algebra.
2. Since $T \subseteq T \cdot \top$, there is a $u \in T$ such that $t = u \cdot \top$. Thus $t \cdot \top = u \cdot \top \cdot \top = u \cdot \top = t$, using $\top \cdot \top \geq \top \cdot 1 = \top$.
3. The part $(\leq)$ follows by isotony since $(t \sqcap a) \cdot b \leq a \cdot b$ and $(t \sqcap a) \cdot b \leq t \cdot b \leq t \cdot \top = t$ using Lemma 2.3.2 and Part 2. As a consequence, the part $(\geq)$ follows by Boolean algebra since $t \sqcap (a \cdot b) = t \sqcap ((t \sqcap a) \cdot b + (\overline{t} \sqcap a) \cdot b) \leq t \sqcap ((t \sqcap a) \cdot b + \overline{t} \sqcap (a \cdot b)) = t \sqcap ((t \sqcap a) \cdot b) \leq (t \sqcap a) \cdot b$. □

Therefore, and in contrast to the condition semirings of [21], every ideal condition semiring is already replete. The cause for this restriction is the axiom $S \cdot T \subseteq T$, since the other prerequisites of Lemma 2.5 also hold in condition semirings. As we will point out in Section 2.2, however, this axiom is necessary for the representation of normal designs as matrices and, subsequently, to obtain a Kleene and omega algebra. Nevertheless, the new axioms are less restrictive than those in [35] since they do not require S to be a Boolean semiring.

2.2. Designs and normal designs as matrices

We define (normal) designs as 2×2 matrices over a weak semiring, similarly to [35]. The difference is that we do not demand a Boolean semiring but take the elements from an ideal semiring. Our goal is to lift the semiring structure to designs which is a prerequisite for further structures introduced in the following sections.

The matrix representation of designs eliminates the auxiliary variables ok and ok' that are introduced to deal with non-termination in [26]. It is motivated by the following observation. Let $\mathbb{B} =_{\text{def}} \{\mathit{false}, \mathit{true}\}$ denote the Boolean values, which ok and ok' can take. A design R is a particular relation of type $\mathbb{B} \times X \leftrightarrow \mathbb{B} \times X'$, where the sets X and X' consist of the possible states before and after the execution of the computation modelled by R. We decompose such relations according to the possible combinations of the values for ok and ok' by forming the residual relations

$$R(ok, ok') =_{\text{def}} \{(x, x') \mid ((ok, x), (ok', x')) \in R\}$$

for $ok, ok' \in \mathbb{B}$. This corresponds to a restriction or selection followed by projection in the formalism of [6]. Hence the relation R may be represented as a matrix containing its four residual relations:

$$R = \begin{pmatrix} R(\mathit{false}, \mathit{false}) & R(\mathit{false}, \mathit{true}) \\ R(\mathit{true}, \mathit{false}) & R(\mathit{true}, \mathit{true}) \end{pmatrix}.$$

Operations on relations thus reduce to standard matrix operations; in particular, relational union and composition correspond to matrix addition and multiplication, respectively. Moreover, reasoning is completely component-free, without ok and ok' and variable substitutions. The matrix representation is further elaborated in [35].

In the following we concentrate on the case of homogeneous relations, that is, relations that leave the state space $X = X'$ unchanged. The heterogeneous case is mildly more complicated: composition becomes a partial operation with differing left and right identities, according to [41].

Designs are a subclass of relations, characterised in [26] by the healthiness conditions H1 and H2. They translate to the matrix representation as follows. A relation R satisfies H1 if both residuals in the top row of R's matrix are the universal relation $\top$. It satisfies H2 if in both rows of its matrix the left element is a subset of the right one. This motivates our definition of designs in the abstract setting, where we use 2×2 matrices with elements from an ideal semiring S as entries.

Definition 2.6. Let (S, T) be an ideal semiring. The set of *designs* over (S, T) is

$$\mathrm{D}(S, T) =_{\text{def}} \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in S^{2\times 2} \;\middle|\; a = b = \top \;\wedge\; c \in T \;\wedge\; c \leq d \right\}.$$

For $t \in T$ and $a \in S$, we define the *design*

$$t \vdash a =_{\text{def}} \begin{pmatrix} \top & \top \\ \overline{t} & \overline{t} + a \end{pmatrix}.$$

If, additionally, (S, T) is an ideal condition semiring, we call its designs *normal* and set $\mathrm{ND}(S, T) =_{\text{def}} \mathrm{D}(S, T)$.

The design $t \vdash a$ represents a program whose execution is guaranteed to terminate when started in one of the states described by t, and whose possible transitions are described by a. From the definitions it follows that *every* design can be denoted in abbreviated form, since

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{D}(S, T) \Rightarrow \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \overline{c} \vdash d.$$

Due to the imposed restrictions, designs could also be represented by two-dimensional vectors or pairs of semiring elements [21]. The full matrix representation allows us to reuse the standard matrix operations

and constructions, such as the Kleene star in Section 2.3. One well-known result [8] is that matrices over a semiring can be made into a semiring themselves by defining addition $+$ and multiplication $\cdot$ as in linear algebra; this generalises in a straightforward way to matrices over a weak semiring.

As the following theorem shows, the normal designs are closed under these operations and hence form a weak semiring of their own. Observe that the left ideal property is crucial for the closure under $\cdot$ and that a strict ideal condition semiring is needed for the right unit law.

Theorem 2.7. *Let (S, T) be a strict ideal condition semiring. Then the structure of normal designs $(\mathrm{ND}(S,T), +, \top \vdash 0, \cdot, \top \vdash 1)$ is an idempotent weak semiring.*

Proof. Let $a, b \in S$ and $t, u \in T$ such that $t \leq a$ and $u \leq b$.

- $+$ is total since $t + u \in T$ and $t + u \leq a + b$ and
$$\begin{pmatrix} \top & \top \\ t & a \end{pmatrix} + \begin{pmatrix} \top & \top \\ u & b \end{pmatrix} = \begin{pmatrix} \top & \top \\ t+u & a+b \end{pmatrix}.$$
- $+$ is associative, commutative and idempotent since it is defined componentwise.
- $\top \vdash 0$ is neutral with respect to $+$ since
$$\begin{pmatrix} \top & \top \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \top & \top \\ t & a \end{pmatrix} = \begin{pmatrix} \top & \top \\ t & a \end{pmatrix} = \begin{pmatrix} \top & \top \\ t & a \end{pmatrix} + \begin{pmatrix} \top & \top \\ 0 & 0 \end{pmatrix}.$$
- $\cdot$ is total since $t\top + au = t + au \in T$ by Lemma 2.5.2 and $t + au \leq t + ab$ and
$$\begin{pmatrix} \top & \top \\ t & a \end{pmatrix} \cdot \begin{pmatrix} \top & \top \\ u & b \end{pmatrix} = \begin{pmatrix} \top\top + \top u & \top\top + \top b \\ t\top + au & t\top + ab \end{pmatrix} = \begin{pmatrix} \top & \top \\ t+au & t+ab \end{pmatrix}.$$
- $\cdot$ is associative since matrix multiplication is associative.
- $\top \vdash 1$ is neutral with respect to $\cdot$ since
$$\begin{pmatrix} \top & \top \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \top & \top \\ t & a \end{pmatrix} = \begin{pmatrix} \top & \top \\ 0+1t & 0+1a \end{pmatrix} = \begin{pmatrix} \top & \top \\ t & a \end{pmatrix}$$
and, using $a0 = 0$,
$$\begin{pmatrix} \top & \top \\ t & a \end{pmatrix} \cdot \begin{pmatrix} \top & \top \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \top & \top \\ t+a0 & t+a1 \end{pmatrix} = \begin{pmatrix} \top & \top \\ t & a \end{pmatrix}.$$
- $\cdot$ distributes over $+$ since it does so for matrices.
- $\top \vdash 0$ is a left annihilator of $\cdot$ since
$$\begin{pmatrix} \top & \top \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \top & \top \\ t & a \end{pmatrix} = \begin{pmatrix} \top & \top \\ 0+0t & 0+0a \end{pmatrix} = \begin{pmatrix} \top & \top \\ 0 & 0 \end{pmatrix}.$$ □

Remark. With two modifications, Theorem 2.7 generalises to designs over ideal semirings: the composition of designs is the more verbose
$$\begin{pmatrix} \top & \top \\ t & a \end{pmatrix} \cdot \begin{pmatrix} \top & \top \\ u & b \end{pmatrix} = \begin{pmatrix} \top & \top \\ t\top + au & t\top + ab \end{pmatrix},$$
and the right unit law fails. (End of remark)

The proofs of this theorem and the next lemma show the advantages of the semiring formalisation. The calculations are concise and straightforward; moreover, they do not need to reason about variables and substitutions, in particular, not about the non-observables ok and ok'.

Now we investigate the natural order on (normal) designs more closely. Observe that it reflects the implication order and not the refinement order of UTP, which is the reverse.

Lemma 2.8. *Assume an ideal (condition) semiring.*

1. *The natural order on (normal) designs is*

$$t \vdash a \leq u \vdash b \quad \Leftrightarrow \quad u \leq t \ \wedge \ u \sqcap a \leq b \quad \Leftrightarrow \quad u \leq t \ \wedge \ a \leq \overline{u} + b.$$

   In particular, $0 \vdash 0$ *(corresponding to true in UTP) is the greatest element.*
2. $t \vdash a = u \vdash b \Leftrightarrow t = u \ \wedge \ t \sqcap a = u \sqcap b.$
3. $t \vdash a \ = \ t \vdash \overline{t} + a \ = \ t \vdash t \sqcap a.$
4. *The sum of designs is* $(t \vdash a) + (u \vdash b) = t \sqcap u \vdash a + b.$
5. *The composition of designs is* $(t \vdash a) \cdot (u \vdash b) = \overline{\overline{t}\top + a\overline{u}} \vdash ab$ *which simplifies to* $t \sqcap \overline{a\overline{u}} \vdash ab$ *for normal designs.*
6. *If* $a \leq \overline{t}$ *then the normal design* $t \vdash a$ *is a left annihilator. In particular,* $t \vdash \overline{t}$ *and* $t \vdash 0$ *(hence also the greatest element* $0 \vdash 0$*) are left annihilators.*

Proof. Let $a, b \in S$ and $t, u \in T$.

1. By the componentwise matrix order and the shunting rule of Lemma 2.3.3,

$$\begin{aligned} t \vdash a \leq u \vdash b \quad &\Leftrightarrow \quad \begin{pmatrix} \top & \top \\ \overline{t} & \overline{t} + a \end{pmatrix} \leq \begin{pmatrix} \top & \top \\ \overline{u} & \overline{u} + b \end{pmatrix} \\ &\Leftrightarrow \quad \overline{t} \leq \overline{u} \ \wedge \ \overline{t} + a \leq \overline{u} + b \\ &\Leftrightarrow \quad \overline{t} \leq \overline{u} \ \wedge \ \overline{t} \leq \overline{u} + b \ \wedge \ a \leq \overline{u} + b \\ &\Leftrightarrow \quad u \leq t \ \wedge \ u \sqcap a \leq b. \end{aligned}$$

2. By Part 1,

$$\begin{aligned} t \vdash a = u \vdash b \quad &\Leftrightarrow \quad t \vdash a \leq u \vdash b \ \wedge \ u \vdash b \leq t \vdash a \\ &\Leftrightarrow \quad u \leq t \ \wedge \ u \sqcap a \leq b \ \wedge \ t \leq u \ \wedge \ t \sqcap b \leq a \\ &\Leftrightarrow \quad t = u \ \wedge \ t \sqcap a = u \sqcap b. \end{aligned}$$

3. This is immediate from Part 2.
4. The sum of designs is given by

$$\begin{aligned} (t \vdash a) + (u \vdash b) \ &= \ \begin{pmatrix} \top & \top \\ \overline{t} & \overline{t} + a \end{pmatrix} + \begin{pmatrix} \top & \top \\ \overline{u} & \overline{u} + b \end{pmatrix} \ = \ \begin{pmatrix} \top & \top \\ \overline{t} + \overline{u} & \overline{t} + \overline{u} + a + b \end{pmatrix} \\ &= \ \overline{\overline{t} + \overline{u}} \vdash a + b \ = \ t \sqcap u \vdash a + b. \end{aligned}$$

5. The composition of designs is given by

$$\begin{aligned} (t \vdash a) \cdot (u \vdash b) \ &= \ \begin{pmatrix} \top & \top \\ \overline{t} & \overline{t} + a \end{pmatrix} \cdot \begin{pmatrix} \top & \top \\ \overline{u} & \overline{u} + b \end{pmatrix} \\ &= \ \begin{pmatrix} \top & \top \\ \overline{t}\top + (\overline{t} + a)\overline{u} & \overline{t}\top + (\overline{t} + a)(\overline{u} + b) \end{pmatrix} \\ &= \ \begin{pmatrix} \top & \top \\ \overline{t}\top + a\overline{u} & \overline{t}\top + a\overline{u} + ab \end{pmatrix} \ = \ \overline{\overline{t}\top + a\overline{u}} \vdash ab, \end{aligned}$$

   and $\overline{\overline{t}\top + a\overline{u}} = t \sqcap \overline{a\overline{u}}$ for normal designs.
6. By Part 3, $t \vdash a = t \vdash t \sqcap a = t \vdash 0$ and $(t \vdash 0) \cdot (u \vdash b) = t \sqcap \overline{0\overline{u}} \vdash 0b = t \vdash 0$ by Part 5. □

Theorem 2.7 and Lemma 2.8 show that normal designs behave just as expected from [26], also in their abbreviated forms. In particular, our generalised normal designs satisfy the healthiness conditions H1–H3. Let us briefly discuss healthiness condition H4, formally $(t \vdash a) \cdot (0 \vdash 0) = (0 \vdash 0)$, that characterises *totality* of $t \vdash a$. A simple calculation using Parts 5 and 2 of Lemma 2.8 shows that it reduces to $t \leq a\top$ for a normal design $t \vdash a$. In the relational semiring the vector $a\top$ represents the domain of a, that is, the set of its initial states. Hence totality $t \leq a\top$ means that all states admitted by the assumption t actually enable a-transitions.

On one hand, the subset of total designs is interesting for computation purposes. On the other hand, it is possible to set up a correspondence between total normal designs and the elements of the underlying semiring S, see [21]. This connection abstractly expresses a previous result of [18] and can be used to derive demonic programming operators and a demonic refinement order on S that reflects total correctness. Using the star and omega operations of Sections 2.3 and 2.4 below, closed representations of the least and greatest fixpoints of the recursion defining the demonic while-loop on S can be calculated, too.

For the remainder of this section and the three following ones, we restrict our attention to normal designs, assuming a strict ideal condition semiring. In Section 2.6 we return to the more general case of designs over an ideal semiring.

Since we have seen that normal designs form a semiring, it is a consequent next step to check whether they even form a condition semiring. This would embed the set of conditions into the set of matrices and hence allow a more uniform treatment. The next corollary shows that this is indeed possible; it will be useful for representing UTP-conditions as tests in Section 3. The $\sqcap$ operation is lifted componentwise to designs and the lifted $^-$ acts on the assumption part only, as detailed in the proof.

Corollary 2.9. *Let (S, T) be a strict ideal condition semiring and*

$$C =_{\text{def}} \{t \vdash 0 \mid t \in T\} = \left\{ \begin{pmatrix} \top & \top \\ t & t \end{pmatrix} \;\middle|\; t \in T \right\}.$$

Then the structure $(\mathrm{ND}(S,T), C, +, \top \vdash 0, \cdot, \top \vdash 1, \sqcap, 0 \vdash 0, ^-)$ is an ideal condition semiring.

Proof. Theorem 2.7 shows that $\mathrm{ND}(S,T)$ forms an idempotent weak semiring. Its greatest element is $0 \vdash 0$ by Lemma 2.8.1. It is easily calculated that C is a submonoid of $\mathrm{ND}(S,T)$, and the left ideal property follows since

$$(t \vdash a) \cdot (u \vdash 0) = t \sqcap \overline{a\overline{u}} \vdash a0 = t \sqcap \overline{a\overline{u}} \vdash 0.$$

As a special case, we obtain the condition property $(t \vdash 0) \cdot (0 \vdash 0) = t \vdash 0$. We define the restriction $\sqcap$ as the componentwise restriction on the matrix representation:

$$\begin{pmatrix} \top & \top \\ t & t \end{pmatrix} \sqcap \begin{pmatrix} \top & \top \\ u & b \end{pmatrix} =_{\text{def}} \begin{pmatrix} \top & \top \\ t \sqcap u & t \sqcap b \end{pmatrix}.$$

Immediate consequences are distributivity over $+$ and neutrality of the universal condition $0 \vdash 0$. The Boolean algebra structure also follows using the complement

$$\overline{\overline{t} \vdash 0} = \overline{\begin{pmatrix} \top & \top \\ t & t \end{pmatrix}} =_{\text{def}} \begin{pmatrix} \top & \top \\ \bar{t} & \bar{t} \end{pmatrix} = t \vdash 0. \qquad \square$$

2.3. Star designs

So far, we have only dealt with the non-iterative programming constructs. Now we introduce finite iteration in terms of the Kleene star operation. It will be used in Sections 3 and 4 for concrete transformations of recursive programs. That normal designs have a star entails that the results of these sections are applicable to UTP. We use the classical axiomatisation of the star operation from [30].

Definition 2.10. A *Kleene algebra* is a structure $(S, ^*)$ such that S is an idempotent semiring and the operation star * satisfies the unfold and induction laws

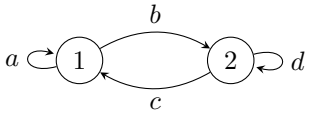
$$1 + a \cdot a^* \leq a^* \qquad b + a \cdot c \leq c \Rightarrow a^* \cdot b \leq c$$
$$1 + a^* \cdot a \leq a^* \qquad b + c \cdot a \leq c \Rightarrow b \cdot a^* \leq c$$

for $a, b, c \in S$. In a *weak* Kleene algebra, S is only required to be an idempotent weak semiring.

It follows from these axioms that a^*b is the least fixpoint of the function $\lambda x.ax+b$ and that the operation * is isotone with respect to the natural order. Moreover, the unfold law can be strengthened to an equality $1 + aa^* = a^*$. Further consequences are $a^*a^* = a^*$ and $a(ba)^* = (ab)^*a$. Examples of Kleene algebras are again the relations, where R^* is the reflexive transitive closure of R, and the formal languages where $L^* = \bigcup_{n\in\mathbb{N}} L^n$ is the finitely iterated language concatenation.

Remark. A related example is formed by path sets in a directed graph where $\cdot$ is path concatenation. The relation, language and path algebras can be used to derive graph and pointer algorithms from high-level specifications [33]. In the context of graph theory also other Kleene algebras arise, for example, to calculate the shortest paths between all pairs of nodes using the Floyd-Warshall algorithm. The underlying instance is $(\mathbb{R} \cup \{\pm\infty\}, \min, \infty, +, 0, ^*)$ where a^* is $-\infty$ for negative a and 0 otherwise. Further examples and a framework for such problems are given in [9] using a variant called *closed semiring*; see [29] for the slight difference to Kleene algebra. (End of remark)

There is also a generic way to extend the (weak) matrix semiring over a (weak) semiring into a Kleene algebra. We use the construction presented in [16]; similar ones appear in [8, 30]. By iterative application they can handle general $n \times n$ matrices, but we only need the case $n = 2$. Intuitively, the construction describes the finite execution paths in a two-state automaton with transitions labelled according to the matrix entries a, b, c and d:



(1)

The entry at row i and column j of the matrix contains a regular expression for the paths from node i to node j.

Definition 2.11. The *Kleene star* of a 2×2 matrix is given by

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^* =_{\text{def}} \begin{pmatrix} f^* & f^*bd^* \\ e^*ca^* & e^* \end{pmatrix},$$

where $f = a + bd^*c$ and $e = d + ca^*b$.

The Kleene star of a normal design hence is

$$\begin{pmatrix} \top & \top \\ t & a \end{pmatrix}^* = \begin{pmatrix} \top^* & \top a^* \\ a^*t\top & a^* \end{pmatrix} = \begin{pmatrix} \top & \top \\ a^*t & a^* \end{pmatrix},$$

since $f = \top + \top a^*t = \top$ and $e = a + t\top^*\top = a + t = a$ and $\top a^* \geq \top 1 = \top$. The result is a normal design since $t \in T \Rightarrow a^*t \in T$ and $t \leq a \Rightarrow a^*t \leq a^*a \leq a^*$. Observe that the left ideal property is crucial again. We therefore obtain the following result.

Theorem 2.12. *Let (S, T) be an ideal condition semiring such that S is a Kleene algebra. Then the structure $(\mathrm{ND}(S, T), +, \top \vdash 0, \cdot, \top \vdash 1, ^*)$ is a weak Kleene algebra.*

Proof. Because of Theorem 2.7, it remains to show that the star unfold and induction axioms are satisfied. But this follows, since they are valid in the encompassing full matrix algebra and $\mathrm{ND}(S, T)$ is closed under star, as just shown. □

The benefit of deriving the star operation via the matrix construction becomes manifest when comparing with the previous approach in [21] where the operation had to be 'guessed' and verified by an extensive proof.

We finally derive the Kleene star of a normal design in the abbreviated representation used by [26]. As a prerequisite we prove the following distribution lemma.

Lemma 2.13. *Consider an ideal condition semiring (S, T) such that S is a Kleene algebra and let $a \in S$ and $t \in T$.*

1. $(t + a)^* = a^*t + a^*$.
2. $(t + a)^*t = a^*t$.

Proof.

1. First, we have $1 + (t + a)(a^*t + a^*) \leq 1 + t\top + a(a^*t + a^*) = t + aa^*t + a^* = a^*t + a^*$, and therefore $(t + a)^* \leq a^*t + a^*$ by star induction. Second, by isotony and star unfold, $a^*t + a^* \leq (t + a)^*(t + a) + (t + a)^* \leq (t + a)^*$.
2. By Part 1, we have $(t + a)^*t = (a^*t + a^*)t \leq a^*t\top + a^*t = a^*t$. The other inequality $a^*t \leq (t + a)^*t$ follows by isotony of star. □

Corollary 2.14. *Let (S, T) be an ideal condition semiring such that S is a Kleene algebra. Let $t \vdash a$ be a normal design over (S, T). Then $(t \vdash a)^* = (\overline{a^*\bar{t}} \vdash a^*)$.*

Proof. By Theorem 2.12 and Lemma 2.13,

$$(t \vdash a)^* = \begin{pmatrix} \top & \top \\ \bar{t} & \bar{t} + a \end{pmatrix}^* = \begin{pmatrix} \top & \top \\ (\bar{t} + a)^*\bar{t} & (\bar{t} + a)^* \end{pmatrix} = \begin{pmatrix} \top & \top \\ a^*\bar{t} & a^*\bar{t} + a^* \end{pmatrix} = \overline{a^*\bar{t}} \vdash a^*. \qquad \square$$

This result intuitively describes the finite iterations of a program $t \vdash a$ as a program $\overline{a^*\bar{t}} \vdash a^*$ whose transitions are described by finitely iterating the transitions in a, and whose execution terminates if, performing these transitions, it cannot reach one of the states in $\bar{t}$ (that do not guarantee termination).

2.4. Omega designs

Next, we add the operation omega for infinite iteration. It will again be used in program manipulations in Sections 3 and 4. The results are applicable to UTP due to the fact that normal designs have an omega operation. We basically follow the axiomatisation from [7].

Definition 2.15. An *omega algebra* is a structure $(S, {}^\omega)$ such that S is a Kleene algebra and the operation omega ${}^\omega$ satisfies the unfold and co-induction laws

$$a^\omega \leq a \cdot a^\omega \qquad\qquad c \leq a \cdot c + b \Rightarrow c \leq a^\omega + a^* \cdot b$$

for $a, b, c \in S$. In a *weak* omega algebra, S is only required to be a weak Kleene algebra, but the unfold law is strengthened to $a^\omega = a \cdot a^\omega$ since the inequality $(\geq)$ need not hold in absence of the right annihilation axiom [34].

It follows from these axioms that $a^\omega + a^*b$ is the greatest fixpoint of the function $\lambda x.ax + b$ and that the operation ${}^\omega$ is isotone. Moreover, $a^*a^\omega = a^\omega$ holds. For the special case $b = 0$ we therefore obtain $c \leq ac \Rightarrow c \leq a^\omega + a^*0 = a^*a^\omega + a^*0 = a^*a^\omega = a^\omega$, hence a^ω is the greatest fixpoint of $\lambda x.ax$. Furthermore, there exists a greatest element $\top = 1^\omega$ that additionally satisfies $a^\omega = a^\omega\top$ for each $a \in S$.

An example of an omega algebra are again the relations where the vector R^ω characterises those points from which infinite paths of R-transitions emerge. The formal languages also form an omega algebra, but L^ω is either the set of all elements or empty, depending on whether $\varepsilon \in L$ or not. This becomes more useful if one includes as elements the infinite words over the base set.

As with Kleene algebras, we can form matrices with elements of omega algebras as entries. The omega operation can be lifted to matrices by another construction, presented in [32]. It can also handle general $n \times n$ matrices by iterative application, but again we only need the case $n = 2$. Intuitively, the construction describes the infinite execution paths in the two-state automaton shown in diagram (1).

Definition 2.16. The *omega* of a 2×2 matrix is given by

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^\omega =_{\text{def}} \begin{pmatrix} f^\omega + a^* b e^\omega & f^\omega + a^* b e^\omega \\ d^* c f^\omega + e^\omega & d^* c f^\omega + e^\omega \end{pmatrix},$$

where $f = a + bd^*c$ and $e = d + ca^*b$ as in Definition 2.11.

The omega of a normal design hence is

$$\begin{pmatrix} \top & \top \\ t & a \end{pmatrix}^\omega = \begin{pmatrix} \top^\omega + \top a^\omega & \top^\omega + \top a^\omega \\ a^* t \top^\omega + a^\omega & a^* t \top^\omega + a^\omega \end{pmatrix} = \begin{pmatrix} \top & \top \\ a^* t + a^\omega & a^* t + a^\omega \end{pmatrix},$$

since, as before, $f = \top$, $e = a$ and $\top^\omega = \top$. The result is a normal design since $t \in T \Rightarrow a^*t + a^\omega = a^*t + a^\omega \top \in T$. Observe that the left ideal property is crucial again. We therefore obtain the following result.

Theorem 2.17. *Let (S, T) be an ideal condition semiring such that S is an omega algebra. Then the structure $(\mathrm{ND}(S,T), +, \top \vdash 0, \cdot, \top \vdash 1, ^*, ^\omega)$ is a weak omega algebra.*

Proof. Because of Theorem 2.12, it remains to show that the omega co-induction and unfold axioms are satisfied. But this follows, since they are valid in the encompassing full matrix algebra and $\mathrm{ND}(S,T)$ is closed under omega, as just shown. □

Again this may be compared to the previous approach of [21] to see the advantage, as pointed out in Section 2.3.

We finally derive the omega of a normal design in the abbreviated representation. As a prerequisite we prove the following distribution lemma.

Lemma 2.18. *Consider an ideal condition semiring (S, T) such that S is an omega algebra and let $a \in S$ and $t \in T$. Then $(t + a)^\omega + (t + a)^* t = a^\omega + a^* t$.*

Proof. The part $(\geq)$ is immediate by isotony. For $(\leq)$, after application of Lemma 2.13.2 it suffices to show $(t+a)^\omega \leq a^\omega + a^*t$. But this follows by omega co-induction from $(t+a)^\omega \leq (t+a)(t+a)^\omega \leq t\top + a(t+a)^\omega = t + a(t+a)^\omega$. □

Corollary 2.19. *Let (S, T) be an ideal condition semiring such that S is an omega algebra. Let $t \vdash a$ be a normal design over (S, T). Then $(t \vdash a)^\omega = (\overline{a^\omega + a^* \overline{t}} \vdash 0)$.*

Proof. By Theorem 2.17 and Lemma 2.18,

$$\begin{pmatrix} \top & \top \\ \overline{t} & \overline{t} + a \end{pmatrix}^\omega = \begin{pmatrix} \top & \top \\ (\overline{t} + a)^\omega + (\overline{t} + a)^* \overline{t} & (\overline{t} + a)^\omega + (\overline{t} + a)^* \overline{t} \end{pmatrix} = \begin{pmatrix} \top & \top \\ a^\omega + a^* \overline{t} & a^\omega + a^* \overline{t} \end{pmatrix}.$$ □

This result intuitively describes the infinite iterations of a program $t \vdash a$ as a program $\overline{a^\omega + a^* \overline{t}} \vdash 0$ whose execution is not guaranteed to terminate if started from a state in a^ω (where a can be iterated infinitely) or from a state in $a^* \overline{t}$ (where it can reach a state that does not guarantee termination).

2.5. UTP algebras

As we have seen in the previous sections, normal designs form an idempotent weak semiring, a weak Kleene algebra and a weak omega algebra. The qualifier 'weak' means that the right annihilation law $\forall x : x \cdot 0 = 0$ is not required to hold. In a semiring with greatest element $\top$ this axiom may be restated as $\top \cdot 0 = 0$ by isotony.

As stated in Corollary 2.9, (normal) designs have the greatest element $\top_{\mathrm{D}} =_{\text{def}} 0 \vdash 0$ that corresponds to the predicate *true* in the instance of UTP. The least element $0_{\mathrm{D}} =_{\text{def}} \top \vdash 0$ is not a right annihilator of designs; indeed $\top_{\mathrm{D}} \cdot 0_{\mathrm{D}} = \top_{\mathrm{D}}$ since

$$(0 \vdash 0) \cdot (\top \vdash 0) \;=\; \overline{\overline{0} \cdot \top + 0 \cdot \overline{\top}} \vdash 0 \cdot 0 \;=\; 0 \vdash 0.$$

Omitting the right annihilation law gives us the freedom to impose this left annihilation law instead. A theory without a left absorbing greatest element is investigated in [20].

Definition 2.20. A *UTP semiring/Kleene algebra/omega algebra* is a weak semiring/Kleene algebra/omega algebra with greatest element $\top$ such that $\top \cdot 0 = \top$ holds or, equivalently, $\forall x : \top \cdot x = \top$.

An immediate consequence is that normal designs form a UTP omega algebra. The axiom $\top \cdot 0 = \top$ is typical of total correctness frameworks, not only of UTP. For example, it also holds in the demonic refinement algebras of [45]. Actually the latter are equivalent to UTP omega algebras as is shown by [27] using the name *top-left-strict weak omega algebras*. Once $\top$ is a left annihilator, further elements are, too.

Lemma 2.21. *Let a be an element of a UTP omega algebra. Then $a\top$ and a^ω are left annihilators.*

Proof. $(a\top)x = a(\top x) = a\top$. The claim about a^ω follows since $a^\omega = a^\omega\top$. □

2.6. Fixpoints and designs

In Sections 2.3 and 2.4 we have shown how to calculate the least and the greatest fixpoints of the iteration function $\lambda x.ax + b$ on normal designs. We now use our algebraic techniques to extend this by considering all designs instead of just normal designs. Moreover, we take up the idea of [26] to consider a more general function

$$H(P \vdash Q) =_{\text{def}} F(P \vdash Q) \vdash G(P \vdash Q)$$

on designs, in which F and G define for a design $P \vdash Q$ the assumption and commitment parts of the result design $H(P \vdash Q)$ separately. In [26, Theorem 3.1.6] it is shown how to calculate the assumption and commitment parts of the greatest fixpoint of H explicitly.

In this section, we additionally deal with the least fixpoint and generalise that result to our setting, that is, we do not assume that P and Q are relations. Instead, we only require that $P \vdash Q$ is a design over an ideal semiring and that certain fixpoints exist. Our treatment is based on the following basic definitions about fixpoints taken from [13].

Definition 2.22. Let f be a function on a partial order. An element a is a *fixpoint* of f if it satisfies the fixpoint law $f(a) = a$. The element μf is the *least prefixpoint* of f if the following unfold and induction laws hold:

$$f(\mu f) \leq \mu f \qquad \forall x : f(x) \leq x \Rightarrow \mu f \leq x$$

The element νf is the *greatest postfixpoint* of f if the following unfold and co-induction laws hold:

$$\nu f \leq f(\nu f) \qquad \forall x : x \leq f(x) \Rightarrow x \leq \nu f$$

We abbreviate $\mu(\lambda x.f(x))$ by $\mu x.f(x)$ and $\nu(\lambda x.f(x))$ by $\nu x.f(x)$.

In this paper, by writing μf and νf we assume that these elements exist. If f is isotone and the underlying partial order is complete, this is a consequence of Tarski's fixpoint theorem [44]. We furthermore recall the following facts about fixpoints from [13]. If f is isotone, then μf is the least fixpoint of f, and νf the greatest. If f and g are isotone and $f \leq g$ it follows that $\mu f \leq \mu g$ and $\nu f \leq \nu g$. If the partial order is a Boolean algebra with complement $\neg$ it follows that $\mu f = \neg\nu x.\neg f(\neg x)$ and $\nu f = \neg\mu x.\neg f(\neg x)$.

2.6.1. The greatest fixpoint

In the remainder of this section, let $H(t \vdash a) =_{\text{def}} F(t \vdash a) \vdash G(t \vdash a)$ be an isotone function of designs over an ideal semiring (S, T) such that $F(t \vdash a) \in T$ for all $t \in T$ and $a \in S$. The following definitions of P_ν, R_ν and Q_ν are taken from [26]. Note that μ and ν are swapped relative to the original definitions, since we use the implication order and not the refinement order.

Definition 2.23.

1. Define $P_\nu : S \to T$ by $P_\nu(a) =_{\text{def}} \mu t.F(t \vdash a)$.
2. Define $R_\nu : S \to S$ by $R_\nu(a) =_{\text{def}} \overline{P_\nu(a)} + G(P_\nu(a) \vdash a)$.
3. Define $Q_\nu =_{\text{def}} \nu R_\nu$.

We first prove a few isotony statements supporting the existence of the used fixpoints. If T is complete, then $P_\nu(a)$ and hence $R_\nu(a)$ exist by Lemma 2.24.1. If additionally S is complete, then Q_ν exists by Lemma 2.24.3. Afterwards we generalise [26, Theorem 3.1.6] to our setting.

Lemma 2.24.

1. *Let $a, b \in S$ and $t, u \in T$ such that $a \leq b$ and $u \leq t$. Then $F(u \vdash b) \leq F(t \vdash a)$ and $F(u \vdash b) \sqcap G(t \vdash a) \leq G(u \vdash b)$. In particular, $\lambda t.F(t \vdash a)$ is isotone and $\lambda a.F(t \vdash a)$ is antitone.*
2. *P_ν is antitone.*
3. *R_ν is isotone.*

Proof.

1. Since $u \leq t$ and $u \sqcap a \leq a \leq b$ by Lemma 2.3.2 we have $t \vdash a \leq u \vdash b$ by Lemma 2.8.1. Since H is isotone we conclude $F(t \vdash a) \vdash G(t \vdash a) \leq F(u \vdash b) \vdash G(u \vdash b)$ which, again by Lemma 2.8.1, is equivalent to the claim.
2. Assume $a \leq b$. By Part 1 we have $\lambda t.F(t \vdash a) \geq \lambda t.F(t \vdash b)$, from which we obtain $\mu t.F(t \vdash a) \geq \mu t.F(t \vdash b)$ by isotony of μ.
3. Let $a, b \in S$ such that $a \leq b$. By Part 2 we have $P_\nu(b) \leq P_\nu(a)$. Now Part 1 shows $F(P_\nu(b) \vdash b) \sqcap G(P_\nu(a) \vdash a) \leq G(P_\nu(b) \vdash b)$. By Definition 2.23.1, $F(P_\nu(b) \vdash b) = P_\nu(b)$ and shunting shows $G(P_\nu(a) \vdash a) \leq \overline{P_\nu(b)} + G(P_\nu(b) \vdash b) = R_\nu(b)$. Since $\overline{P_\nu(a)} \leq \overline{P_\nu(b)} \leq R_\nu(b)$, we have $R_\nu(a) \leq R_\nu(b)$ by the join property. □

Theorem 2.25. $\nu H = P_\nu(Q_\nu) \vdash Q_\nu$.

Proof. First, we prove that $P_\nu(Q_\nu) \vdash Q_\nu$ is a fixpoint of H. By Definition 2.23.1 we have $P_\nu(Q_\nu) = F(P_\nu(Q_\nu) \vdash Q_\nu)$. Hence, by Lemma 2.8.3,

$$\begin{aligned} H(P_\nu(Q_\nu) \vdash Q_\nu) &= F(P_\nu(Q_\nu) \vdash Q_\nu) \vdash G(P_\nu(Q_\nu) \vdash Q_\nu) = P_\nu(Q_\nu) \vdash G(P_\nu(Q_\nu) \vdash Q_\nu) \\ &= P_\nu(Q_\nu) \vdash \overline{P_\nu(Q_\nu)} + G(P_\nu(Q_\nu) \vdash Q_\nu) = P_\nu(Q_\nu) \vdash R_\nu(Q_\nu) = P_\nu(Q_\nu) \vdash Q_\nu. \end{aligned}$$

Second, we prove that $P_\nu(Q_\nu) \vdash Q_\nu$ is the greatest postfixpoint of H. Assume $t \vdash a \leq H(t \vdash a)$ which by Lemma 2.8.1 is equivalent to

$$F(t \vdash a) \leq t \text{ and } F(t \vdash a) \sqcap a \leq G(t \vdash a).$$

Hence $P_\nu(a) \leq t$ by Definition 2.23.1, and therefore also $P_\nu(a) = F(P_\nu(a) \vdash a) \leq F(t \vdash a)$ by Lemma 2.24.1. As a consequence $P_\nu(a) \sqcap a \leq G(t \vdash a)$, and therefore

$$P_\nu(a) \sqcap a \leq F(P_\nu(a) \vdash a) \sqcap G(t \vdash a) \leq G(P_\nu(a) \vdash a)$$

by Lemma 2.24.1. By shunting we obtain $a \leq \overline{P_\nu(a)} + G(P_\nu(a) \vdash a) = R_\nu(a)$, hence $a \leq Q_\nu$ by Definition 2.23.3. Therefore $P_\nu(Q_\nu) \sqcap a \leq Q_\nu$, and $P_\nu(Q_\nu) \leq P_\nu(a) \leq t$ by Lemma 2.24.2. Altogether $t \vdash a \leq P_\nu(Q_\nu) \vdash Q_\nu$ by Lemma 2.8.1. □

In Section 2.4 we have derived the omega operation on normal designs. As an example of using Theorem 2.25, let us now characterise the omega operation on designs over an ideal semiring (S, T) such that S is a weak omega algebra. Recall that a^ω is the greatest fixpoint of the function $\lambda x.ax$; this motivates the definition of H in the following result.

Corollary 2.26. *Let $t \vdash a$ be a design and define H by setting $H(u \vdash b) =_{\text{def}} (t \vdash a)(u \vdash b)$. Then $\nu H = \overline{a^\omega + a^* \bar{t} \top} \vdash 0$.*

Proof. Observe that H is isotone and, by Lemma 2.8.5,

$$H(u \vdash b) \;=\; (t \vdash a)(u \vdash b) \;=\; \overline{\bar{t}\top + a\bar{u}} \vdash ab \;=\; F(u \vdash b) \vdash G(u \vdash b),$$

where $F(u \vdash b) =_{\text{def}} \overline{\bar{t}\top + a\bar{u}}$ and $G(u \vdash b) =_{\text{def}} \bar{t}\top + a\bar{u} + ab$. By Definition 2.23.1,

$$P_\nu(b) = \mu u.F(u \vdash b) = \mu u.\overline{\bar{t}\top + a\bar{u}} = \overline{\nu u.\bar{t}\top + au} = \overline{a^\omega + a^*\bar{t}\top}.$$

Since $P_\nu(b)$ is constant, let $P_\nu = P_\nu(b)$. By Definitions 2.23.2 and 2.23.3, as well as omega and star properties,

$$\begin{aligned} Q_\nu &= \nu b.\overline{P_\nu(b)} + G(P_\nu(b) \vdash b) = \nu b.\overline{P_\nu} + \bar{t}\top + a\overline{P_\nu} + ab = a^\omega + a^*(\overline{P_\nu} + \bar{t}\top + a\overline{P_\nu}) \\ &= a^\omega + a^*\overline{P_\nu} + a^*\bar{t}\top = a^\omega + a^*(a^\omega + a^*\bar{t}\top) + a^*\bar{t}\top = a^\omega + a^*\bar{t}\top = \overline{P_\nu}. \end{aligned}$$

By Theorem 2.25 and Lemma 2.8.3, $\nu H \;=\; P_\nu(Q_\nu) \vdash Q_\nu \;=\; P_\nu \vdash \overline{P_\nu} \;=\; P_\nu \vdash 0$. □

Observe how this result generalises Corollary 2.19 from normal designs to designs. It replaces the term $a^*\bar{t}$ with $a^*\bar{t}\top$, since $\bar{t} = \bar{t}\top$ does not necessarily hold in an ideal semiring. In an ideal condition semiring both terms are equal by Lemma 2.5.2.

2.6.2. The least fixpoint

The least fixpoint of a function on designs can be calculated in a similar way. To this end, we swap μ and ν in the definitions of P_ν, R_ν and Q_ν, and use $P_\mu(a) =_{\text{def}} \nu t.F(t \vdash a)$ and $R_\mu(a) =_{\text{def}} \overline{P_\mu(a)} + G(P_\mu(a) \vdash a)$ and $Q_\mu =_{\text{def}} \mu R_\mu$ in the following. Lemma 2.24 and its proof remain unchanged except for swapping μ and ν. We only need to restate Theorem 2.25. Note that we cannot use duality as an argument since the underlying structure is an ideal semiring (S, T) but not a lattice. Its dual need not be an ideal semiring as witnessed, for example, by the restriction operation $\sqcap$ that is defined only on $T \times S$.

Although the semantics of recursion is defined by the greatest fixpoint in UTP and other total correctness approaches, the least fixpoint is useful to show that recursive equations have unique solutions or to prove the termination of a recursion [26, Section 2.7].

Theorem 2.27. $\mu H = P_\mu(Q_\mu) \vdash Q_\mu$.

Proof. The proof that $P_\mu(Q_\mu) \vdash Q_\mu$ is a fixpoint of H proceeds exactly as for Theorem 2.25. We now prove that $P_\mu(Q_\mu) \vdash Q_\mu$ is the least prefixpoint of H. To this end, assume $H(t \vdash a) = F(t \vdash a) \vdash G(t \vdash a) \leq t \vdash a$, which by Lemma 2.8.1 is equivalent to

$$t \leq F(t \vdash a) \text{ and } t \sqcap G(t \vdash a) \leq a.$$

Since $t \leq F(t \vdash a) = F(t \vdash \bar{t} + a)$, we have $t \leq P_\mu(\bar{t} + a)$ by definition of P_μ as greatest fixpoint. Moreover,

$$t \sqcap G(P_\mu(\bar{t} + a) \vdash \bar{t} + a) = t \sqcap F(t \vdash \bar{t} + a) \sqcap G(P_\mu(\bar{t} + a) \vdash \bar{t} + a) \leq t \sqcap G(t \vdash \bar{t} + a) = t \sqcap G(t \vdash a) \leq a$$

by Lemma 2.24.1. By shunting, $R_\mu(\bar{t}+a) = \overline{P_\mu(\bar{t} + a)} + G(P_\mu(\bar{t}+a) \vdash \bar{t}+a) \leq \bar{t}+a$, hence $Q_\mu = \mu R_\mu \leq \bar{t}+a$. This implies $t \sqcap Q_\mu \leq a$, and $t \leq P_\mu(\bar{t} + a) \leq P_\mu(Q_\mu)$ by antitony of P_μ shown in Lemma 2.24.2. Therefore $P_\mu(Q_\mu) \vdash Q_\mu \leq t \vdash a$ by Lemma 2.8.1. □

In Section 2.3 we have derived the star operation on normal designs. As an example of using Theorem 2.27, let us now characterise the Kleene star on designs over an ideal semiring (S, T) such that S is a weak Kleene algebra. Recall that a^* is the least fixpoint of the function $\lambda x.ax + 1$; this motivates the definition of H in the following result.

Corollary 2.28. *Let $t \vdash a$ be a design and define $H(u \vdash b) =_{\text{def}} (t \vdash a)(u \vdash b) + (\top \vdash 1)$. Then $\mu H = \overline{a^*\bar{t}\top} \vdash a^*$.*

Proof. Observe that H is isotone and, by Lemma 2.8,

$$H(u \vdash b) \;=\; (t \vdash a)(u \vdash b) + (\top \vdash 1) \;=\; \overline{\bar{t}\top + a\bar{u}} \vdash (ab+1) \;=\; F(u \vdash b) \vdash G(u \vdash b),$$

where $F(u \vdash b) =_{\mathrm{def}} \overline{\bar{t}\top + a\bar{u}}$ and $G(u \vdash b) =_{\mathrm{def}} \bar{t}\top + a\bar{u} + ab + 1$. By the definition of P_μ,

$$P_\mu(b) = \nu u.F(u \vdash b) = \nu u.\overline{\bar{t}\top + a\bar{u}} = \overline{\mu u.\bar{t}\top + au} = \overline{a^*\bar{t}\top}.$$

Since $P_\mu(b)$ is constant, let $P_\mu = P_\mu(b)$. By the definitions of R_μ and Q_μ, as well as star properties,

$$\begin{aligned} Q_\mu &= \mu b.\overline{P_\mu(b)} + G(P_\mu(b) \vdash b) = \mu b.\overline{P_\mu} + \bar{t}\top + a\overline{P_\mu} + ab + 1 = a^*(\overline{P_\mu} + \bar{t}\top + a\overline{P_\mu} + 1) \\ &= a^*\overline{P_\mu} + a^*\bar{t}\top + a^* = a^*a^*\bar{t}\top + a^*\bar{t}\top + a^* = a^*\bar{t}\top + a^* = \overline{P_\mu} + a^*. \end{aligned}$$

By Theorem 2.27 and Lemma 2.8.3, $\mu H \;=\; P_\mu(Q_\mu) \vdash Q_\mu \;=\; P_\mu \vdash \overline{P_\mu} + a^* \;=\; P_\mu \vdash a^*$. □

Again, this result generalises Corollary 2.14 from normal designs to designs by replacing $a^*\bar{t}$ with $a^*\bar{t}\top$.

3. Relating recursive definitions

In this section we investigate the relation between different kinds of linear recursions. We first show how concrete recursions can be modelled in our general framework developed in Section 2. By instantiation we are then able to apply the results obtained there to derive properties of the recursions studied. To obtain a more convenient notation, we use tests [31] instead of conditions.

Besides the concrete investigation, a major goal is to establish the applicability of the general results of the previous section to UTP. Once this procedure is clear we can conduct further development at the abstract level without referring to concrete programs, and this is done in Section 4. In this sense, the present section may be seen as a preparation for the next, which also features extensive use of tests.

3.1. Tail-recursion and tests

As a motivating example we derive three variants of the computation of the factorial. Only one of the implementations is tail-recursive, which leads to considerable difficulties when trying to show their equivalence. Let us begin with the tail-recursive variant. We assume that the variables x and y have natural numbers as their values.

Example 1. We start with the specification $P_1 =_{\mathrm{def}} x, y := 0, yx!$ and derive, using the notations of [26], namely $S \lhd C \rhd T$ for the conditional $\mathsf{if}\ C\ \mathsf{then}\ S\ \mathsf{else}\ T$ and ; for sequential composition and $\mathbb{I}$ for skip,

$$\begin{aligned} P_1 &= x, y := 0, yx! \\ &= (x, y := 0, yx!) \lhd x = 0 \rhd (x, y := 0, yx!) \\ &= (y := y \cdot 1) \lhd x = 0 \rhd (x, y := 0, yx(x-1)!) \\ &= (y := y) \lhd x = 0 \rhd (y := yx \,;\, x, y := 0, y(x-1)!) \\ &= \mathbb{I} \lhd x = 0 \rhd (y := yx \,;\, x := x - 1 \,;\, x, y := 0, yx!) \\ &= \mathbb{I} \lhd x = 0 \rhd (y := yx \,;\, x := x - 1 \,;\, P_1). \end{aligned}$$

The calculation of the factorial is thus realised by successively multiplying the numbers $x, x-1, \ldots, 1$ in decreasing order. In each recursive step, the variable x is decremented after the multiplication but before the recursive call. The recursion terminates when $x = 0$; the initial value of x is lost after this procedure.

In UTP, the solution to such a recursive equation is defined as a least fixpoint, so we obtain

$$\begin{aligned} P_1 &= \mu X \bullet \mathbb{I} \lhd x = 0 \rhd (y := yx \,;\, x := x - 1 \,;\, X) \\ &= (x \neq 0) * (y := yx \,;\, x := x - 1). \end{aligned}$$

using the notation of [26]. However, in UTP the least fixpoint is taken with respect to the refinement order, which is the reverse of the implication order we are using in our model of UTP designs. So we shall have to investigate the greatest fixpoint with respect to the natural order.

The first goal is to describe this type of recursion in our framework of ideal semirings. It is clear that sequential composition is modelled by $\cdot$ and skip by 1. To represent the conditional algebraically, we use special semiring elements called *tests* [31]. They are similar to conditions, but work symmetrically on the input and output sides of the semiring elements and hence can express pre- and post-restrictions in a uniform way. Rather than Kleene algebras with tests, we use the more liberal test semirings [34]. In the case of relations, tests are subsets of the identity relation, occasionally called co-reflexives or partial identities.

Definition 3.1. A *test semiring* is an idempotent weak semiring $(S, +, 0, \cdot, 1)$ with a distinguished set of elements $\mathrm{test}(S) \subseteq S$ called *tests* and a *negation* operation $\neg$ such that $(\mathrm{test}(S), +, 0, \cdot, 1, \neg)$ is a Boolean algebra.

In particular, $p \leq 1$ for each test $p \in \mathrm{test}(S)$. In the relational semiring, where 1 is the identity relation, tests therefore take the form $\{(x, x) \mid x \in B\}$ for some subset $B \subseteq A$ of the base set A. Hence from a test p one can construct a corresponding vector (condition) by passing to $p \cdot \top$, where $\top$ is the universal relation on A. Conversely, one can extract a test from a vector by intersecting it with the identity relation.

The following lemma shows that this generalises and provides a way to turn an ideal condition semiring into a test semiring. Actually, conditions are isomorphic to tests: a pair of isomorphisms is given by $\lambda t.t \sqcap 1$ mapping conditions to tests and $\lambda p.p\top$ mapping tests to conditions.

Lemma 3.2. *In an ideal condition semiring (S, T) define $\mathrm{test}(S, T) =_{\mathrm{def}} \{t \sqcap 1 \mid t \in T\} \subseteq S$ and $\neg p =_{\mathrm{def}} \overline{p\top} \sqcap 1$. Then the structures $(T, +, 0, \sqcap, \top, \bar{\ })$ and $(\mathrm{test}(S, T), +, 0, \cdot, 1, \neg)$ are isomorphic Boolean algebras. Moreover, for $t \in T$ we have $\neg(t \sqcap 1) = \bar{t} \sqcap 1$.*

Proof. Let $f : T \to \mathrm{test}(S, T)$ and $g : \mathrm{test}(S, T) \to T$ be given by $f(t) = t \sqcap 1$ and $g(p) = p\top$. We first show that f and g are inverse to each other, hence bijections. One direction is $g(f(t)) = (t \sqcap 1)\top = t \sqcap (1\top) = t \sqcap \top = t$ by Lemma 2.5.3. The other part is a consequence since, letting $p = t \sqcap 1$, we have $f(g(p)) = p\top \sqcap 1 = (t \sqcap 1)\top \sqcap 1 = t \sqcap 1 = p$.

We next show that f is a homomorphism. Meet, join, complement, bottom and top are preserved since

- $f(t \sqcap u) = (t \sqcap u) \sqcap 1 = t \sqcap (u \sqcap 1) = (t \sqcap 1)(u \sqcap 1) = f(t) \cdot f(u)$ by Lemmas 2.3.4 and 2.5.3,
- $f(t + u) = (t + u) \sqcap 1 = (t \sqcap 1) + (u \sqcap 1) = f(t) + f(u)$ by distributivity,
- $f(\bar{t}) = \bar{t} \sqcap 1 = \overline{(t \sqcap 1)\top} \sqcap 1 = \neg(t \sqcap 1) = \neg f(t)$ as above,
- $f(0) = 0 \sqcap 1 = 0$ and $f(\top) = \top \sqcap 1 = 1$.

The claim follows since the structure $(T, +, 0, \sqcap, \top, \bar{\ })$ is a Boolean algebra. □

We can apply this result to designs. Recall from Corollary 2.9 that the normal designs over (S, T) have $\{t \vdash 0 \mid t \in T\}$ as conditions and $\top \vdash 1$ as the identity of composition. We therefore obtain as tests the normal designs

$$(t \vdash 0) \sqcap (\top \vdash 1) = \begin{pmatrix} \top & \top \\ \bar{t} & \bar{t} \end{pmatrix} \sqcap \begin{pmatrix} \top & \top \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \top & \top \\ 0 & \bar{t} \sqcap 1 \end{pmatrix} = \top \vdash (\bar{t} \sqcap 1).$$

Let t be a condition and $p = t \sqcap 1$ the corresponding test. Using Lemma 2.5.3 we can then represent UTP's conditional as

$$(a \lhd t \rhd b) = t \sqcap a + \bar{t} \sqcap b = (t \sqcap 1)a + (\bar{t} \sqcap 1)b = pa + \neg(t \sqcap 1)b = pa + \neg pb.$$

For a test p and an element a the term pa is called the *input restriction* of a by p. The *output restriction* is obtained symmetrically as ap.

In the special case where $a = 1$ and $b = 0$ the conditional simplifies to the test p. Thus tests play the role of UTP's assumptions [26, Definition 2.8.3] and the guards of [45].

Thanks to Lemma 3.2 we can entirely replace the conditions of a semiring by tests in our general framework of Section 2, and we will do so in the remainder of this paper. We have chosen to start with

conditions, since they are closer to the original, relational interpretation of UTP and since conditions (or another set satisfying the ideal property) are necessary in the underlying semiring if designs are to be represented by matrices.

First, for abbreviation, we use the identifiers m ($m$ultiply) for $y := yx$ and d ($d$ecrement) for $x := x - 1$ and p ($p$ositive?) for the test $x \neq 0$. Then the UTP semantics of the recursion we have derived for P_1 is the fixpoint

$$P_1^\nu =_{\text{def}} \nu x.pmdx + \neg p.$$

In a second step, we abstract from the concrete program parts and investigate the properties of this fixpoint for arbitrary semiring elements m, d and test p. We furthermore investigate the least fixpoint $P_1^\mu =_{\text{def}} \mu x.pmdx + \neg p$ that may be of interest in other theories. If we assume that the underlying semiring is a Kleene algebra or even an omega algebra, as are UTP designs, the fixpoints can be represented as $P_1^\mu = (pmd)^* \neg p$ and $P_1^\nu = (pmd)^\omega + (pmd)^* \neg p$.

3.2. Linear recursion

Let us introduce a second computation of the factorial. It is no longer tail-recursive but the recursion is still linear. We then relate it to the first example.

Example 2. We now start with the specification $P_2 =_{\text{def}} y := yx!$ and derive

$$\begin{array}{lcl} P_2 & = & y := yx! \\ & = & (y := yx!) \lhd x = 0 \rhd (y := yx!) \\ & = & (y := y \cdot 1) \lhd x = 0 \rhd (y := yx(x-1)!) \\ & = & (y := y) \lhd x = 0 \rhd (y := yx \,;\, y := y(x-1)!) \\ & = & \mathbb{I} \lhd x = 0 \rhd (y := yx \,;\, x := x - 1 \,;\, y := yx! \,;\, x := x + 1) \\ & = & \mathbb{I} \lhd x = 0 \rhd (y := yx \,;\, x := x - 1 \,;\, P_2 \,;\, x := x + 1). \end{array}$$

The computation proceeds as in the first example, but the variable x is incremented after returning from the recursive call. Therefore the value of x after this procedure is the same as its initial value.

We reason similarly to the first example, and using i ($i$ncrement) for $x := x+1$ we obtain by abstraction the semiring fixpoint formulations

$$P_2^\nu =_{\text{def}} \nu x.pmdxi + \neg p$$

and $P_2^\mu =_{\text{def}} \mu x.pmdxi + \neg p$. Since these are not tail-recursions, there is no obvious fixpoint-free representation of P_2^ν and P_2^μ using the Kleene star or the omega operation.

At the concrete UTP level, both specifications can be related as follows:

$$(P_2 \,;\, x := 0) \;=\; (y := yx! \,;\, x := 0) \;=\; (x, y := 0, yx!) \;=\; P_1. \tag{2}$$

To extend this relation to the implementations, both derivations above would have to be produced independently. Moreover, it is unclear how to mimic (2) at the abstract algebraic level. Our objective in the following is, therefore, to algebraically relate both implementations by transforming one into the other. Using z (set to $z$ero) for $x := 0$ we would like to obtain

$$P_2^\nu z \;=\; (\nu x.pmdxi + \neg p)z \;=\; (\nu x.pmdx + \neg p) \;=\; P_1^\nu$$

and similarly $P_2^\mu z = P_1^\mu$. This result can then be used, for example, to transform the implementation P_2^ν to the tail-recursive P_1^ν.

3.3. Relating tail-recursion and linear recursion

We have considered two versions of the computation of the factorial and expressed their connection using the semiring formalism. This relation can now be proved in a completely algebraic way. We start with the least fixpoints P_1^μ and P_2^μ.

Theorem 3.3. *Let a, b, c, d be elements of a weak Kleene algebra such that $bd = d$ and $cd = c$. Then $(\mu x.axb + c)d = a^*c$.*

Proof. Let $g(x) =_{\text{def}} axb + c$. If the Kleene algebra is complete and composition distributes over arbitrary sums, we can apply μ-fusion [2]. Let $f(x) =_{\text{def}} xd$ and $h(x) =_{\text{def}} ax + c$, then

$$f(g(x)) = (axb + c)d = axbd + cd = axd + c = h(f(x)),$$

from which the claim $f(\mu g) = \mu h$ follows.

Without these additional assumptions, we prove the claim as follows. For the part $(\geq)$ note first that $c = cd = (\mu x.c)d \leq (\mu x.axb + c)d = (\mu g)d$. Second, using the fixpoint law in the last step,

$$a(\mu g)d = a(\mu g)bd \leq a(\mu g)bd + cd = (a(\mu g)b + c)d = (\mu g)d.$$

Therefore $c + a(\mu g)d \leq (\mu g)d$, which implies $a^*c \leq (\mu g)d$ by star induction. To show the converse inequality, from $a(a^*cb^*)b + c \leq a^*cb^*$ we infer $\mu g \leq a^*cb^*$ by the fixpoint induction law. Therefore $(\mu g)d \leq a^*cb^*d = a^*cd = a^*c$, since $bd = d \Rightarrow b^*d = d$ by star axioms. □

Instantiating $d = 1$ in Theorem 3.3, and therefore also $b = 1$, we obtain the special case $(\mu x.ax + c) = a^*c$, the least fixpoint representation of a^*c. Another consequence is the equality of the above two implementations of the factorial.

Corollary 3.4. $P_2^\mu z = P_1^\mu$.

Proof. Observe that $iz = z$ holds, since incrementing x before setting it to 0 is superfluous. Furthermore, $\neg pz = \neg p$ since setting x to 0 can be omitted if it already is 0. Therefore the assumptions of Theorem 3.3 are satisfied and we conclude

$$P_2^\mu z = (\mu x.pmdxi + \neg p)z = (pmd)^*\neg p = (\mu x.pmdx + \neg p) = P_1^\mu.$$ □

To relate the implementations P_1^ν and P_2^ν, we have to restrict ourselves to UTP algebras. Note that ν-fusion cannot be directly applied since composition does not distribute over meets.

Theorem 3.5. *Let a, b, c, d be elements of a UTP omega algebra.*

1. $\nu x.axb = a^\omega$.
2. *If $bd = d$ and $cd = c$, then $(\nu x.axb + c)d = a^\omega + a^*c$.*

Proof.

1. Let $e(x) =_{\text{def}} axb$. By Lemma 2.21 and omega unfold, $aa^\omega b = aa^\omega = a^\omega$, which shows that a^ω is a fixpoint of e and hence $a^\omega \leq \nu e$. For the converse inequality observe that for an arbitrary fixpoint e° of e we have $e^\circ\top = ae^\circ b\top \leq ae^\circ\top$, so that $e^\circ\top \leq a^\omega$ by omega co-induction. But $e^\circ \leq e^\circ\top$ and we are done.
2. Let $g(x) =_{\text{def}} axb + c$. Using the fixpoint law in the first step,

   $$(\nu g)d = (a(\nu g)b + c)d = a(\nu g)bd + cd = a(\nu g)d + c.$$

   This implies the part $(\leq)$ by omega co-induction. By star induction, this also implies $a^*c \leq (\nu g)d$. Hence it remains to show $a^\omega \leq (\nu g)d$ for the part $(\geq)$. But this holds, since $a^\omega = a^\omega d = (\nu x.axb)d \leq (\nu g)d$ by Lemma 2.21, Part 1 and isotony. □

Compared with the original aim to relate the two implementations of the factorial, the solution provided by Theorem 3.5.2 is considerably more general due to its algebraic nature. It abstracts from the concrete recursion to the recursion pattern $\nu x.axb + c$, from the concrete program statements to their properties $bd = d$ and $cd = c$, and from designs to any structure satisfying the axioms of a UTP omega algebra.

Corollary 3.6. *In a UTP algebra, $P_2^\nu z = P_1^\nu$.*

Proof. We proceed similarly to the proof of Corollary 3.4 and using Theorem 3.5.2 we obtain

$$P_2^\nu z = (\nu x.pmdxi + \neg p)z = (pmd)^\omega + (pmd)^*\neg p = (\nu x.pmdx + \neg p) = P_1^\nu.$$ □

This kind of reasoning is generalised in Section 4.

3.4. Relating linear recursions

Let us derive a third implementation of the factorial, again not tail-recursive.

Example 3. We now start with the specification $P_3 =_{\text{def}} y := x!$ and derive

$$\begin{array}{lcl} P_3 & = & y := x! \\ & = & (y := x!) \lhd x = 0 \rhd (y := x!) \\ & = & (y := 1) \lhd x = 0 \rhd (y := x(x-1)!) \\ & = & (y := 1) \lhd x = 0 \rhd (y := (x-1)! \,;\, y := yx) \\ & = & (y := 1) \lhd x = 0 \rhd (x := x-1 \,;\, y := x! \,;\, x := x+1 \,;\, y := yx) \\ & = & (y := 1) \lhd x = 0 \rhd (x := x-1 \,;\, P_3 \,;\, x := x+1 \,;\, y := yx). \end{array}$$

This computation successively multiplies the numbers $1, 2, \ldots, x$ in increasing order. The reversed order is achieved by accumulating the multiplications after returning from the recursive calls instead of before. As a consequence, the variable y has to be initialised in the base case. Again, the value of x at the start and at the end of the procedure are the same.

We reason similarly to the first example, and using o (set to *o*ne) for $y := 1$ we thus obtain the semiring fixpoint expressions

$$P_3^\nu =_{\text{def}} \nu x.pdxim + \neg po$$

and $P_3^\mu =_{\text{def}} \mu x.pdxim + \neg po$.

The specifications P_2 and P_3 can be related as follows:

$$(y := 1 \,;\, P_2) \;=\; (y := 1 \,;\, y := yx!) \;=\; (y := 1 \,;\, y := x!) \;=\; (y := x!) \;=\; P_3.$$

Again our objective is to algebraically relate the implementations, that is, we would like to obtain

$$oP_2^\nu \;=\; o(\nu x.pmdxi + \neg p) \;=\; (\nu x.pdxim + \neg po) \;=\; P_3^\nu \tag{3}$$

and similarly $oP_2^\mu = P_3^\mu$.

In this case, not even μ-fusion can be applied directly, since as a prerequisite we would need $pmdxi = pdxim$ for arbitrary x, which is not true. We will, however, show that under certain side conditions the finite parts of the following two recursions are equivalent:

$$\mu x.pmdxi + \neg p \qquad\qquad \mu x.pdxim + \neg p$$

We first ignore the tests and show that the finite approximations then coincide. These are, respectively, $(md)^k i^k$ and $d^k (im)^k$ if termination occurs after k steps.

As an abbreviation, for arbitrary a and $n \in \mathbb{N}$ we set $a^{(n)} =_{\text{def}} d^n a i^n$. This corresponds to executing a in a state after n operations of type d and restoring the initial state afterwards using i repeatedly. In the concrete case where $m = (y := yx)$, $d = (x := x - 1)$ and $i = (x := x + 1)$ we have $m^{(n)} = (y := y(x - n))$. An assumption $m^{(j)} m^{(k)} = m^{(k)} m^{(j)}$ then expresses a special case of the commutativity of multiplication.

One central assumption is $di \leq 1 = id$ which implies $d^k i^k \leq 1 = i^k d^k$ for all $k \in \mathbb{N}$. Hence $p^{(n)} = d^n p i^n \leq d^n i^n \leq 1$ for each test p, and we assume that $p^{(n)}$ is itself a test. In particular, $1^{(n)}$ is the test $x \geq n$, and therefore $m1^{(n)} = 1^{(n)} m$ for all $n \in \mathbb{N}$. For each $k \leq n$ this implies

$$\begin{array}{l} m^{(k)} 1^{(n)} = d^k m i^k d^n i^n = d^k m d^{n-k} i^{n-k} i^k = d^k m 1^{(n-k)} i^k = d^k 1^{(n-k)} m i^k \\ \qquad = d^k d^{n-k} i^{n-k} i^k d^k m i^k = d^n i^n m^{(k)} = 1^{(n)} m^{(k)}. \end{array}$$

Lemma 3.7.

1. $(md)^k = m^{(0)} \cdots m^{(k-1)} d^k$.
2. $(im)^k = i^k m^{(k-1)} \cdots m^{(0)}$.
3. *If the* $m^{(j)}$ *in the formulas above commute, then* $(md)^k i^k = d^k (im)^k$.

Proof.

1. The proof is by induction on k. The base case $k = 0$ is obvious. For the induction step let $e =_{\text{def}} m^{(0)} \cdots m^{(k-1)}$. Then
$$(md)^{k+1} = (md)^k md = ed^k md = ed^k mi^k d^k d = em^{(k)} d^{k+1} = m^{(0)} \cdots m^{(k)} d^{k+1}.$$
2. Symmetrically to Part 1.
3. By the commutativity assumption, $e =_{\text{def}} m^{(0)} \cdots m^{(k-1)} = m^{(k-1)} \cdots m^{(0)}$. Hence, using Parts 1 and 2,
$$(md)^k i^k = ed^k i^k = e1^{(k)} = 1^{(k)} e = d^k i^k e = d^k (im)^k.$$ □

We now include tests into our considerations. The informal idea behind the next lemma is to move in an iteration $(pmd)^k$ all occurrences of p to the left so that on the right a pure iteration of md remains and we can apply the previous lemma. Consider a sequence mdp in which p is tested *after* md. Suppose now that m does not influence p (which holds for the concrete case above when p is the test $x \neq 0$, so that we have again programs that compute the factorial). Then we can also first change the state according to d, test p and restore the original state using i. After that we execute m and d and can omit the test of p, since it has already been tested 'beforehand'. This reasoning is captured by the formula $mdp = dpimd$ or, using our above abbreviation, $mdp^{(0)} = p^{(1)} md$, where the required independence of p from m is expressed as the commutativity requirement $pm = mp$.

If we have that property then
$$(pmd)^2 = pmdpmd = p^{(0)} mdp^{(0)} md = p^{(0)} p^{(1)} mdmd = p^{(0)} p^{(1)} (md)^2$$
and we have achieved our goal in this special case. The general case is covered by the following lemma.

Lemma 3.8. *Let q be a test and a an element that commutes with all tests $q^{(j)}$ and $(\neg q)^{(j)}$. Denote by $r =_{\text{def}} \prod_{j=0}^{k-1} q^{(j)}$ the conjunction of the tests q as performed in the states reached from the initial one in at most $k-1$ steps of type d.*

1. $adq^{(j)} = q^{(j+1)} ad$.
2. $(ad)^k q^{(j)} = q^{(j+k)} (ad)^k$.
3. $(qad)^k = r(ad)^k$.
4. $(qad)^k \neg q = r(\neg q)^{(k)} (ad)^k$.
5. $(qd)^k \neg q = r(\neg q)^{(k)} d^k$.

Proof.

1. $q^{(j+1)} ad = aq^{(j+1)} d = ad^{j+1} qi^{j+1} d = add^j qi^j = adq^{(j)}$.
2. Induction on k using Part 1.
3. Induction on k using Part 1.
4. Follows from Parts 2 and 3.
5. This is the special case $a = 1$ of Part 4. □

Now we are ready for the main result which implies that the finite parts of oP_2^μ and P_3^μ, respectively oP_2^ν and P_3^ν coincide (since o commutes with p and d).

Theorem 3.9. *Assume that m commutes with all tests $p^{(j)}$ and $(\neg p)^{(j)}$ and that the $m^{(j)}$ involved in the formulas below commute. Then*
$$(pmd)^k \neg pi^k = (pd)^k \neg p(im)^k.$$

Proof. Set $r =_{\text{def}} \prod_{j=0}^{k-1} p^{(j)}$. By Lemmas 3.8.4, 3.7.3 and 3.8.5,
$$(pmd)^k \neg pi^k = r(\neg p)^{(k)} (md)^k i^k = r(\neg p)^{(k)} d^k (im)^k = (pd)^k \neg p(im)^k.$$ □

A general investigation of the infinite parts is postponed to the next section. In the current case, there is no infinite part, since the recursions P_2 and P_3 always terminate. We therefore obtain $oP_2^\nu = P_3^\nu$ and achieve our goal of establishing the algebraic relation (3) between our fixpoint expressions corresponding to the recursions we have derived for the original UTP specifications P_2 and P_3.

4. Symmetric linear recursion

We further investigate fixpoints of the function $f(x) =_{\mathrm{def}} axb + c$ using now modal Kleene algebras, that is, Kleene algebras with domain, diamond and box operators. The investigation proceeds by separately considering the finite and the infinite parts of the fixpoints in Sections 4.2 and 4.4, respectively. One goal is to implement the recursion described by f by two consecutive while-loops, as achieved in Section 4.5. However, the structures and techniques introduced in this section are also applicable in further contexts.

The elements a, b, c in the definition of the function f can be instantiated to normal designs due to the development of Section 2. The results in this section therefore also apply to UTP. This shows that one can reason about the programs of UTP and related theories in a purely algebraic manner.

Since certain results hold only if a is deterministic, we need to characterise determinacy algebraically. For this, we can again employ tests, together with the domain operation on which we can base the modal operators. We also use tests for the algebraic representation of (co)invariants that will simplify subsequent arguments. In Section 4.3 further properties are formulated algebraically, namely convergence and divergence.

4.1. Domain, modal operators, determinacy and invariants

The domain of a semiring element a characterises the initial states of a, that is, the states from which corresponding output states may be reached under a. We use the equational axiomatisation of [12].

Definition 4.1. Let S be a test semiring. The domain operation $\ulcorner : S \to \mathrm{test}(S)$ is characterised by the axioms

$$\begin{array}{rcll} a & \leq & \ulcorner a \cdot a & \text{(d1)} \\ \ulcorner(p \cdot a) & \leq & p & \text{(d2)} \\ \ulcorner(a \cdot \ulcorner b) & \leq & \ulcorner(a \cdot b) & \text{(d3)} \end{array}$$

for $a, b \in S$ and $p \in \mathrm{test}(S)$.

Observe that $\ulcorner aa \leq 1a = a$ by isotony. Therefore the axiom (d1) can be strengthened to $a = \ulcorner aa$ which states that restriction to the full domain has no effect. Axiom (d2) formalises that the domain of a restricted element indeed reflects the restriction by satisfying the restricting test. It can be shown that (d1) $\wedge$ (d2) is equivalent to the domain elimination law

$$\ulcorner a \leq p \quad \Leftrightarrow \quad a \leq p \cdot a \qquad \text{(dom)}$$

This implies that the domain operation is unique if it exists. Furthermore, $\ulcorner$ is isotone, distributes over $+$, and satisfies $a \leq 0 \Leftrightarrow \ulcorner a \leq 0$ and $\ulcorner(pa) = p\ulcorner a$ and $\ulcorner p = p$ for each test p. Using (d1) and (d2), axiom (d3) can be strengthened to an equality. This intuitively means that in the interaction of a and b only their 'boundary' matters but not their inner structure, at least to obtain the domain of the composition. See [12] for further properties.

Remark. In a semiring having a greatest element $\top$ there is another equivalent characterisation $\ulcorner a \leq p \Leftrightarrow a \leq p\top$ in the form of a Galois connection [1]. This certainly applies if we use the test semiring induced by Lemma 3.2 from an ideal condition semiring (S, T). But in the latter case, we can even do better and explicitly characterise the domain operation. Adapting a result of [21] we can then show that $\ulcorner a = a\top \sqcap 1$ satisfies the axioms:

- (d1) follows since $\ulcorner aa = (a\top \sqcap 1)a = a\top \sqcap a \geq a$ by Lemmas 2.5.3 and 2.3.2.
- (d2) follows since $\ulcorner(pa) = pa\top \sqcap 1 \leq p\top \sqcap 1 = p$ by Lemmas 2.3.1 and 3.2.
- (d3) follows since $\ulcorner(a\ulcorner b) = a\ulcorner b\top \sqcap 1 = a(b\top \sqcap 1)\top \sqcap 1 \leq ab\top\top \sqcap 1 = ab\top \sqcap 1 = \ulcorner(ab)$ by Lemmas 2.3.1 and 2.3.2.

This fits in well with Lemma 3.2 and means that $a\top$ is the condition representation of $\lceil a$. Moreover, it entails $\lceil a\top = (a\top \sqcap 1)\top = a\top$ for all a.

As a consequence, we can calculate the domain of normal designs over strict ideal condition semirings due to Corollary 2.9:

$$\lceil\begin{pmatrix} \top & \top \\ t & a \end{pmatrix} = \begin{pmatrix} \top & \top \\ t & a \end{pmatrix}\begin{pmatrix} \top & \top \\ \top & \top \end{pmatrix} \sqcap \begin{pmatrix} \top & \top \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \top & \top \\ (t+a)\top & (t+a)\top \end{pmatrix} \sqcap \begin{pmatrix} \top & \top \\ 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} \top & \top \\ 0 & a\top \sqcap 1 \end{pmatrix} = \begin{pmatrix} \top & \top \\ 0 & \lceil a \end{pmatrix},$$

since for designs we have $t \leq a$. The corresponding result using condition semirings appears in [35], where also a connection to predicate transformer algebras, such as demonic refinement algebra [45, 43], is established. (End of remark)

With the help of the domain operation we define the forward modal operators diamond and box (of semiring elements) as test transformers. In their terms we can also characterise determinacy.

Definition 4.2. Assume a test semiring S with domain. The operation *diamond of* a is given by $\langle a\rangle p =_{\text{def}} \lceil(a \cdot p)$. Its dual operation *box of* a is $[a]p =_{\text{def}} \neg\langle a\rangle\neg p$. For each $a \in S$ both operations $\langle a\rangle$ and $[a]$ map tests to tests. An element $a \in S$ is *deterministic* if $\langle a\rangle \leq [a]$.

Thus $\langle a\rangle p$ characterises those states for which *some* a-successor state satisfies p, whereas $[a]p$ characterises those states for which *all* a-successor states satisfy p. The box operator is the abstract counterpart of the wlp operator [14], and diamond is an abstract pre-image operator. The operations $\langle\text{-}\rangle$, $\langle a\rangle$ and $[a]$ are isotone and enjoy many useful algebraic properties [12].

Intuitively, determinacy of a can be understood as follows: if, in a given state, there is an a-transition to some target set p, then all a-transitions lead to p. Since this holds for every p, even for the 'finest' ones that represent singleton states, there is at most one a-transition from any given initial state. This characterisation is well-known from modal logic [39] and has been transferred to the semiring context in [10], where also other notions of determinacy are investigated. The latter work shows that a is deterministic if and only if for all tests p_1 and p_2 with $p_1p_2 = 0$ also $\langle a\rangle p_1 \cdot \langle a\rangle p_2 = 0$.

To reason about the interaction of an element and a test, we introduce the notion of a (co)invariant. To prepare it, we first note that the characterising property (dom) of domain entails the following characterisations of diamond and box as well as equivalent test propagation properties:

$$\langle a\rangle p \leq q \;\Leftrightarrow\; \lceil(ap) \leq q \;\Leftrightarrow\; ap \leq qap \;\Leftrightarrow\; ap \leq qa \qquad \text{(dia)}$$
$$p \leq [a]q \;\Leftrightarrow\; pa\neg q \leq 0 \;\Rightarrow\; pa \leq paq \;\Leftrightarrow\; pa \leq aq \qquad \text{(box)}$$

If 0 is a right annihilator, the implication in (box) becomes an equivalence. In the particular case where $q = p$, the two formulas at the very right of (dia) and (box) mean that p propagates backwards or forwards through a, respectively. Intuitively, $p \leq [a]p$ means that all a-transitions originating in states satisfying p lead to states that satisfy p. This motivates the following definition, calling p an invariant of a.

Definition 4.3. A test p is an *invariant* of a if $pa \leq ap$, and a *co-invariant* of a if $ap \leq pa$. More generally, we say that two elements a and b *semi-commute* if $ab \leq ba$, and *commute* if $ab = ba$.

The following lemma shows a close relation between invariants and co-invariants.

Lemma 4.4. *Consider a test semiring S and let $a, b \in S$ and $p \in \text{test}(S)$.*

1. *If p is a co-invariant of a, then $\neg p$ is an invariant of a.*
2. *If 0 is a right annihilator, then p is a co-invariant of a if and only if $\neg p$ is an invariant of a.*
3. *If S has a domain operation and a and b semi-commute, then $\neg\lceil b$ is an invariant of a.*

Proof.

1. Assuming $ap \leq pa$, we have $\neg pa = \neg pap + \neg pa\neg p \leq \neg ppa + \neg pa\neg p = \neg pa\neg p \leq a\neg p$.
2. Assuming $pa \leq ap$, we have $a\neg p = pa\neg p + \neg pa\neg p \leq ap\neg p + \neg pa\neg p = \neg pa\neg p \leq \neg pa$. The claim follows together with Part 1.
3. $\langle a\rangle \lceil b = \lceil(a\lceil b) = \lceil(ab) \leq \lceil(ba) = \lceil(b\lceil a) \leq \lceil b$, hence $\lceil b$ is a co-invariant of a by (dia). The claim follows by Part 1. □

We freely use the equivalent characterisations (dia) of co-invariants. Several sufficient criteria for co-invariance under a deterministic a are given by the following lemma.

Lemma 4.5. *Let a be deterministic.*

1. *If p is contracted by $[a]$, that is, $[a]p \leq p$, then p is a co-invariant of a.*
2. *If p is expanded by $\langle a\rangle$, that is, $p \leq \langle a\rangle p$, then $\neg p$ is a co-invariant of a.*

Proof.

1. $\langle a\rangle p \leq [a]p \leq p$.
2. $p \leq \langle a\rangle p \Leftrightarrow \neg\langle a\rangle p \leq \neg p \Leftrightarrow [a]\neg p \leq \neg p$, and apply Part 1. □

Let us explain the intuition underlying Part 2: p being expanded by $\langle a\rangle$ means pointwise that every point in p has an a-successor in p. Dually, $\neg p$ being a co-invariant of a means that all a-predecessors of points in $\neg p$ are in $\neg p$ again. Now assume that a is deterministic and some point x in $\neg p$ has an a-predecessor y in p. Then y has an a-successor in p. But by determinacy of a the only a-successor of y is x, which is in $\neg p$; we thus obtain a contradiction. Part 1 can be interpreted in a similar manner.

4.2. The finite part

Let us return to the discussion of our function $f(x) = axb+c$. We say that *the choice in f is deterministic* if $\lceil a\lceil c = 0$, so that for every initial state it is clear which of the two branches (if any) can be taken from it. Our main goal in the following is to derive an implementation of f by two consecutive while-loops. To prove the correctness of the implementation, we look at the finite part and at the infinite part of the recursion, in turn. The separate correctness results are then combined in Section 4.5.

We use elements l from the underlying semiring to describe the left context in which the computation modelled by a terminates after at most (or exactly) n recursive steps. In the special case of l being a test, it describes the states from which such a termination occurs.

Definition 4.6. Let $n \in \mathbb{N}$ and l be a semiring element. Then *l terminates a after at most n steps* if

$$la^n = la^n\neg\lceil a$$

and *l terminates a after exactly n steps* if, additionally,

$$\forall k < n : la^k = la^k\lceil a.$$

Intuitively, this means that after starting with l and repeating a, if possible, n times we are out of the domain of a, hence a cannot be repeated any more. As a consequence we obtain $\forall k > n : la^k = la^n aa^{k-n-1} = la^n\neg\lceil a\lceil aaa^{k-n-1} = la^n 0$. Exact termination additionally requires that indeed n repetitions are possible, since the first $n-1$ iterations preserve the domain of a. The following lemma simplifies the investigated recursion in such terminating contexts.

Lemma 4.7. *Let f° be a fixpoint of f.*

1. *If l terminates a after at most n steps, then $lf^\circ = \sum_{k=0}^{n} la^k cb^k$.*
2. *If the choice in f is deterministic and l' terminates a after exactly n steps, then $l'f^\circ = l'a^n cb^n$.*

Proof.

1. We first show by induction that $\forall n \in \mathbb{N} : f^\circ = a^n f^\circ b^n + \sum_{k=0}^{n-1} a^k c b^k$. For $n = 0$ this is clear, and for $n \geq 0$ we obtain

$$\begin{array}{l} a^{n+1} f^\circ b^{n+1} + \sum_{k=0}^{n} a^k c b^k = a^n (a f^\circ b) b^n + a^n c b^n + \sum_{k=0}^{n-1} a^k c b^k \\ = a^n (a f^\circ b + c) b^n + \sum_{k=0}^{n-1} a^k c b^k = a^n f^\circ b^n + \sum_{k=0}^{n-1} a^k c b^k = f^\circ \end{array}$$

   using the fixpoint property and the induction hypothesis in the last two steps. As shown above, $l a^{n+1} = l a^n 0$, and therefore

$$l f^\circ = l(a^{n+1} f^\circ b^{n+1} + \sum_{k=0}^{n} a^k c b^k) = l a^n 0 + l a^n c b^n + l \sum_{k=0}^{n-1} a^k c b^k = \sum_{k=0}^{n} l a^k c b^k.$$

2. Since the choice in f is deterministic, $\forall k < n : l' a^k c = l' a^k \urcorner a \urcorner c c = l' a^k 0$. Therefore, continuing the proof of Part 1,

$$l' f^\circ = \sum_{k=0}^{n} l' a^k c b^k = l' a^n c b^n + \sum_{k=0}^{n-1} l' a^k c b^k = l' a^n c b^n + \sum_{k=0}^{n-1} l' a^k 0 = l' a^n c b^n,$$

   since $l' a^k 0 \leq l' a^n c b^n$ for $k < n$. □

Under additional assumptions, we can represent the finite part of a fixpoint of f by two consecutive while-loops. This may be compared to the construct 'dojustasoften' of [3].

Remark. The essential reason why such a transformation is possible is that the recursion structure can be linearised, for example, using the property that sequential composition $\cdot$ is associative with 1 as right identity [24]. The procedure can also be seen as an abstract variant of a standard technique for eliminating linear recursive calls in applicative programs. There one replaces a linear recursion by the composition of two tail-recursive functions, the first one recursing until the termination case is reached and the second one performing the pending post-processing operations. In terms of applicative programming constructs this is described in detail as the transformation rule *Recursion simplification II* in [38, page 299]. (End of remark)

The assumptions concern four special elements o, i, d, z of the semiring which together implement an abstract counter. Intuitively,

- o initialises a new counter to 0,
- i increments that counter,
- d decrements the counter if its value is greater than 0 and
- z tests whether the counter is 0 and, if so, removes it.

These elements count the number of iterations, provided they do not interfere with the loop constituents which is ensured by commutativity conditions.

The idea now is to represent the recursion given by f as two consecutive while-loops. The first one performs the a operations and increases the counter as many times as recursive calls occur. After that it performs the termination action c. The second loop performs the b operations and decreases the counter till it becomes zero again. As usual, the while-loops are represented by fixpoints of tail-recursive functions.

Definition 4.8. Let $f(x) = axb + c$. Let o, i commute with a, b, c; let d, z semi-commute with a, and let

$$oz = id = 1 \ \wedge \ od = iz = 0.$$

In this case we refer to o, i, d, z as *counter elements* and say that the *counter assumptions* are satisfied. We then define $g(x) =_{\text{def}} aix + c$ and $h(x) =_{\text{def}} dbx + z$.

The assumptions formalise that a new counter has 0 as its value and cannot be decremented, and that after incrementing a counter it is not 0 and decrementing restores its value. Observe that o has z and i has d as a right-inverse element.

Example 4. One way to implement the counter elements o, i, d, z is using a stack st of natural numbers. Let

$$\begin{array}{lll} p =_{\text{def}} \ \neg(\mathit{isempty}\ \mathit{st}) & q =_{\text{def}} \ (\mathit{top}\ \mathit{st} = 0) & o =_{\text{def}} \ (\mathit{push}\ 0\ \mathit{st}) \\ & \hat{\imath} =_{\text{def}} \ (\mathit{top}\ \mathit{st} := \mathit{top}\ \mathit{st} + 1) & i =_{\text{def}} \ (\hat{\imath} \lhd p \rhd 1) \\ & \hat{d} =_{\text{def}} \ (\mathit{top}\ \mathit{st} := \mathit{top}\ \mathit{st} - 1) & d =_{\text{def}} \ (\neg q \cdot \hat{d} \lhd p \rhd 1) \\ & \hat{z} =_{\text{def}} \ (\mathit{pop}\ \mathit{st}) & z =_{\text{def}} \ pq\hat{z} \end{array}$$

The elements o, i, d, z satisfy the (semi-)commutativity assumptions of Definition 4.8 provided a, b, c describe programs that do not refer to the stack st. Moreover, o establishes the postconditions p and q, whereas $p\hat{\imath}$ preserves p and establishes $\neg q$, thus

$$\begin{array}{ccc} o = op = opq & o\neg q = 0 & o\hat{z} = 1 \\ pi = p\hat{\imath} = p\hat{\imath}p = p\hat{\imath}p\neg q & p\hat{\imath}q = 0 & p\hat{\imath}\hat{d} = p \end{array}$$

Hence the counter assumptions follow by

$$\begin{aligned} oz &= opq\hat{z} = o\hat{z} = 1 \\ od &= opd = op\neg q\hat{d} = o\neg q\hat{d} = 0\hat{d} = 0 \\ iz &= piz + \neg piz = p\hat{\imath}pq\hat{z} + \neg ppq\hat{z} = p\hat{\imath}q\hat{z} + 0q\hat{z} = 0\hat{z} + 0 = 0 \\ id &= pid + \neg pid = p\hat{\imath}pd + \neg pd = p\hat{\imath}p\neg q\hat{d} + \neg p = p\hat{\imath}\hat{d} + \neg p = p + \neg p = 1 \end{aligned}$$

For this interpretation the choice in the function h of Definition 4.8 is deterministic, since $\ulcorner(db)\urcorner z = \ulcorner(p\neg q\hat{d}b + \neg pb)\urcorner(pq\hat{z}) \leq (p\neg q + \neg p)(pq) = 0$. Therefore, the recursion h may be implemented by the loop $(\mathsf{while}\ \neg(pq)\ \mathsf{do}\ d\ ;\ b)\ ;\ \hat{z}$. It simplifies to $(\mathsf{while}\ \neg q\ \mathsf{do}\ \hat{d}\ ;\ b)\ ;\ \hat{z}$ if, as intended, p holds before (and hence during) the loop. If also the choice in f (and hence in g) is deterministic, the recursion g may be implemented by the loop $(\mathsf{while}\ \ulcorner a\ \mathsf{do}\ a\ ;\ i)\ ;\ c$. (Continued in Section 4.5)

The least fixpoints of g and h are the loops $(ai)^*c$ and $(db)^*z$, respectively, just as described above. However, we reason about arbitrary fixpoints of these functions in the following lemma.

Lemma 4.9. *Under the counter assumptions, let f°, g° and h° be fixpoints of f, g and h, respectively. Let l terminate a after at most n steps. Then $lf^\circ = log^\circ h^\circ$.*

Proof. The proof proceeds in three steps.

- We show that the first loop g correctly realises the iterations of a while accumulating a counting right context, that is, that $log^\circ = \sum_{k=0}^{n} la^k coi^k$. Observe that lo terminates ai after at most n steps, since

  $$lo(ai)^n = la^n oi^n = la^n \neg\ulcorner aoi^n \leq la^n oi^n \neg\ulcorner a = lo(ai)^n \neg\ulcorner a \leq lo(ai)^n \neg\ulcorner(ai)$$

  by Lemma 4.4.3. Hence $log^\circ = \sum_{k=0}^{n} lo(ai)^k c = \sum_{k=0}^{n} la^k coi^k$ by Lemma 4.7.1.

- We show that in the accumulated left context oi^k the second loop h correctly realises the iterations of b, that is, that $oi^k h^\circ = b^k$. Observe that oi^k terminates db after at most k steps, because $o\ulcorner d = 0$ and

  $$oi^k(db)^k = ob^k = b^k o = b^k o\neg\ulcorner d = ob^k \neg\ulcorner d = oi^k(db)^k \neg\ulcorner d \leq oi^k(db)^k \neg\ulcorner(db).$$

  Now for $0 \leq j < k$ we obtain

  $$\begin{aligned} oi^k(db)^j z &= oi^{k-j-1} ii^j (db)^j z = oi^{k-j-1} ib^j z = oi^{k-j-1} b^j iz = oi^{k-j-1} b^j 0 \\ &\leq oi^{k-j-1} b^{j+1} z = oi^{k-j-1} i^{j+1} (db)^{j+1} z = oi^k (db)^{j+1} z. \end{aligned}$$

  Hence, again by Lemma 4.7.1,

  $$oi^k h^\circ = \textstyle\sum_{j=0}^{k} oi^k (db)^j z = oi^k (db)^k z = ob^k z = b^k oz = b^k.$$

$*$ We combine both facts obtained above to conclude

$$log^\circ h^\circ = \sum_{k=0}^{n} la^k coi^k h^\circ = \sum_{k=0}^{n} la^k cb^k = lf^\circ$$

by Lemma 4.7.1. □

We finally show how terminating (test) elements can be chosen. Intuitively, the test q_n describes that a can be iterated n times but not $n+1$ times. It only terminates a after *at most* n steps because a may be non-deterministic and some transition paths could be shorter. That cannot happen if a is deterministic and then q'_n describes that a can be iterated exactly n times along the single transition path.

Lemma 4.10. *Let $n \in \mathbb{N}$ and let $q_n =_{\text{def}} \ulcorner(a^n)\neg\ulcorner(a^{n+1})$ and $q'_n =_{\text{def}} \ulcorner(a^n\neg\ulcorner a)$. Note that $q'_n = \langle a^n\rangle\neg\ulcorner a = \neg[a^n]\ulcorner a$.*

1. *$q_n \leq q'_n$, and if a is deterministic, then $q_n = q'_n$.*
2. *The test q_n terminates a after at most n steps.*
3. *If a is deterministic, the test q'_n terminates a after exactly n steps.*

Proof. Let us first state an observation that is used in Parts 1 and 3 of the proof: if a is deterministic, then a^j is deterministic for every $j \in \mathbb{N}$. This follows by induction since determinacy is preserved under composition and the element 1 is deterministic [10].

1. The part $(\leq)$ follows from $\ulcorner(a^n) = \ulcorner(a^n\ulcorner a + a^n\neg\ulcorner a) = \ulcorner(a^{n+1}) + \ulcorner(a^n\neg\ulcorner a)$ by shunting. For $(\geq)$ observe $\ulcorner(a^n\neg\ulcorner a) \leq \ulcorner(a^n)$ and $\ulcorner(a^n\neg\ulcorner a) = \langle a^n\rangle\neg\ulcorner a \leq [a^n]\neg\ulcorner a = \neg\langle a^n\rangle\ulcorner a = \neg\ulcorner(a^{n+1})$.
2. Since $\ulcorner(q_n a^n\ulcorner a) = q_n\ulcorner(a^n\ulcorner a) = \ulcorner(a^n)\neg\ulcorner(a^{n+1})\ulcorner(a^{n+1}) = 0$ we have $q_n a^n\ulcorner a = 0$, hence $q_n a^n \leq q_n a^n\neg\ulcorner a$.
3. By Parts 1 and 2, the test $q'_n = q_n$ terminates a after at most n steps. It remains to show that termination does not occur before n steps. Let $k < n$, then a^k is deterministic, and we have

$$q'_n = \langle a^n\rangle\neg\ulcorner a = \langle a^k\rangle\langle a\rangle\langle a^{n-k-1}\rangle\neg\ulcorner a \leq \langle a^k\rangle\langle a\rangle 1 = \langle a^k\rangle\ulcorner a \leq [a^k]\ulcorner a.$$

Therefore $q'_n a^k \leq q'_n a^k\ulcorner a$ by (box). The converse inequality is trivial. □

4.3. Convergence and divergence

Before we can treat the infinite part of the investigated recursion, we have to develop algebraic means to describe convergence. That is, we need to characterise the initial states of an element a from which no infinite transition paths emerge. This set is represented as the test Δa [36], which is axiomatised as follows.

Definition 4.11. Let S be a weak Kleene algebra with tests. The *convergence operation* $\Delta : S \to \text{test}(S)$ satisfies the unfold and induction laws

$$[a](\Delta a) = \Delta a \qquad\qquad p \cdot [a]q \leq q \Rightarrow \Delta a \cdot [a^*]p \leq q$$

for $a \in S$ and $p, q \in \text{test}(S)$.

Thus $\Delta a \cdot [a^*]p$ is the least fixpoint of $\lambda q.p \cdot [a]q$ and Δa is the least fixpoint of $[a]$. Moreover, $\neg\ulcorner a \leq \Delta a \leq \neg\ulcorner(a^\omega)$ and hence $\Delta a \cdot a^\omega = 0$ provided the omega operation exists. Finally, Δ is antitone and $[a^*](\Delta a) = [a \cdot a^*](\Delta a) = [a](\Delta a) = \Delta a$. Except for $\Delta a = \mu[a]$ the proofs of these properties can be translated from [21] to our present setting of weak semirings. We prove the missing claim without assuming that 0 is a right annihilator in Lemma 4.12.4.

In addition to the convergence of an element a we consider its *divergence* $\nabla a =_{\text{def}} \neg\Delta a$. It abstracts the set of points from which infinite a-transition paths start. By standard fixpoint theory and the de Morgan duality between box and diamond we have $\nabla a = \nu\langle a\rangle$. The following lemma uses (co)invariants to show the interaction between an element and its convergence and divergence. It also relates divergence to finite deterministic iteration.

Lemma 4.12.

1. Δa *is an invariant of* a.
2. ∇a *is a co-invariant of* a.
3. *If* a *and* b *semi-commute, then so do* a^* *and* b, *as well as* a *and* b^*. *In particular, a (co)invariant of* a *is also one of* a^*.
4. Δa *is the least fixpoint of* $[a]$.
5. *If* a *is deterministic, then* Δa *and* ∇a *commute with* a.
6. *If* a *is deterministic and* $p \leq \nabla a$, *then* $pa^* \neg \lceil a = 0$.

Proof.

1. The claim follows immediately from the definition of Δa and (box).
2. The claim follows immediately from the definition of ∇a and (dia) by duality of box and diamond.
3. Let $ab \leq ba$. Then $b + aba^* \leq b + baa^* = b(1 + aa^*) \leq ba^*$ by the star unfold law, hence the star induction law implies $a^*b \leq ba^*$. Symmetrically one can show $ab^* \leq b^*a$.
4. The least fixpoint of $[a]$ is $\Delta a \cdot [a^*]1$, but in absence of the right annihilation law we cannot show $[a^*]1 = 1$. However, we show $\Delta a \leq [a^*]1$ which is sufficient. By Parts 2 and 3, ∇a is a co-invariant of a^*, hence $\langle a^* \rangle(\nabla a) \leq \nabla a$ by (dia). Using duality and isotony of $[a^*]$ we obtain $\Delta a \leq [a^*](\Delta a) \leq [a^*]1$.
5. The commutativity of Δa with a is immediate from Lemma 4.5.1 and Part 1. The commutativity of ∇a with a now follows by Lemma 4.4.1 and Part 2.
6. Observe that $\neg \lceil a \leq \neg \lceil (a \nabla a) = \neg \langle a \rangle (\nabla a) = [a](\Delta a) = \Delta a$. Hence $pa^* \neg \lceil a \leq \nabla a a^* \Delta a = \nabla a \Delta a a^* = 0$ by Parts 5 and 3. □

For example, the intuition underlying Part 2, that states $a\nabla a \leq \nabla a a$, is that a transition that leads to a divergent path is the beginning of a divergent path itself. Part 6 also has a nicely intuitive, pointwise interpretation: by starting in a divergent state and following the unique transition path we can never reach a dead end of a.

Remark. The assumption $p \leq \nabla a$ of Lemma 4.12.6, also used in Lemma 4.14 below, restricts p to divergent states. It is, in particular, satisfied by $p = \lceil (a^\omega)$. To see this, let us restate [11, Lemma 7.6] for weak omega algebras:

1. $\lceil (a^\omega) \leq \nabla a$, which follows from $\nabla a = \nu \langle a \rangle$ since $\langle a \rangle \lceil (a^\omega) = \lceil (a \lceil (a^\omega)) = \lceil (a a^\omega) = \lceil (a^\omega)$.
2. $(\forall a : a\top = \lceil a \top) \Rightarrow (\forall a : \nabla a \leq \lceil (a^\omega))$, which follows using $\nabla a = \lceil (\nabla a \top) \leq \lceil (a^\omega)$ since $\nabla a \top = (\langle a \rangle \nabla a)\top = \lceil (a \nabla a)\top = a \nabla a \top$ implies $\nabla a \top \leq a^\omega$ by omega co-induction.

The premise of the second claim is certainly satisfied if we use the test semiring induced by Lemma 3.2 from an ideal condition semiring (S, T). This follows using the property $\lceil a \top = (a \top \sqcap 1)\top = a \top$ stated in Section 4.1. For normal designs we thus obtain divergence explicitly as $\nabla a = \lceil (a^\omega)$. We can conversely express omega by divergence as $a^\omega = a^\omega \top = (a^\omega \top \sqcap 1)\top = \lceil (a^\omega)\top = \nabla a \top$.

If a is deterministic, the second claim may be simplified to $\lceil a \leq a\top \Rightarrow \nabla a \leq \lceil (a^\omega)$. To see this, note that ∇a commutes with a by Lemma 4.12.5. Hence $\nabla a \top = \nabla a \lceil a \top \leq \nabla a a \top = a \nabla a \top$, from which the proof is completed as above. (End of remark)

4.4. The infinite part

Let us again return to the discussion of our function $f(x) = axb + c$, now treating its infinite part. We first show the following lemma concerning invertible elements of weak omega algebras. The intended application is using the right-inverse counter elements z and d of o and i, respectively.

Lemma 4.13. *Let* y *and* x *be semi-commuting elements of a weak omega algebra.*

1. $yx^\omega \leq x^\omega$.
2. $y(xy)^\omega \leq (xy)^\omega \leq x^\omega$.
3. *If* y *has a right-inverse* r *that semi-commutes with* x, *then* $yx^\omega = x^\omega$ *and* $(xy)^\omega = x^\omega$.

Proof. Let $yx \leq xy$.

1. By omega unfold, isotony and semi-commutativity $yx^\omega \leq yxx^\omega \leq xyx^\omega$. The claim follows by omega co-induction.
2. Since y and xy semi-commute by $y(xy) = (yx)y \leq (xy)y$ we have $y(xy)^\omega \leq (xy)^\omega$ by Part 1. Therefore $(xy)^\omega \leq xy(xy)^\omega \leq x(xy)^\omega$ by omega unfold and isotony. Now the second inequality follows by omega co-induction.
3. Let $yr = 1$ and $rx \leq xr$. Then $x^\omega = yrx^\omega \leq yx^\omega \leq x^\omega$ by Part 1. For the second claim, observe that $rx^\omega \leq x^\omega \leq xx^\omega = xyrx^\omega$ by Part 1 and omega unfold. Hence $rx^\omega \leq (xy)^\omega$ by omega co-induction, which entails $x^\omega = yrx^\omega \leq y(xy)^\omega \leq (xy)^\omega \leq x^\omega$ by isotony and Part 2. □

We can apply the algebraic characterisation of divergence and the results of Section 4.3 to both representations of our non-tail-recursion f, the fixpoint and the while-loops. The result parallels Lemmas 4.7 and 4.9.

Lemma 4.14. *Let a and the choice in f be deterministic and assume $p \leq \nabla a$.*

1. $pa^*c = 0 = p(\mu f)$.
2. *Under the counter assumptions,* $po(\mu g) = 0 = po(\mu g)(\mu h)$.
3. *In a UTP omega algebra,* $p(\nu f) = pa^\omega$.
4. *In a UTP omega algebra, under the counter assumptions,* $po(\nu g)(\nu h) = p(\nu f)$.

Proof.

1. $pa^*c = pa^*\lceil cc \leq pa^*\neg\lceil ac = 0$ by determinacy of the choice and Lemma 4.12.6. Since $f(a^*cb^*) = aa^*cb^*b + c \leq a^*cb^*$ we have $\mu f \leq a^*cb^*$, hence $p(\mu f) \leq pa^*cb^* = 0$.
2. Observe that $1 + aia^*i^* \leq 1 + aa^*ii^* \leq a^*i^*$ by Lemma 4.12.3 using the counter assumptions and star unfold. By star induction we obtain $(ai)^* \leq a^*i^*$, and therefore $po(\mu g) = po(ai)^*c \leq poa^*i^*c = pa^*coi^* = 0$ repeatedly using the counter assumptions, Lemma 4.12.3 and Part 1. Hence $po(\mu g)(\mu h) = 0(\mu h) = 0$.
3. Let $e(x) = axb$. We show $p(\nu f) = p(\nu e)$ which implies the claim by Theorem 3.5.1. The part $(\geq)$ is obvious by isotony, and for $(\leq)$ note that by Lemma 4.12.5,

$$\nabla a(\nu f) = \nabla a(a(\nu f)b + c) = \nabla aa(\nu f)b + \nabla ac = a\nabla a(\nu f)b$$

   since $\nabla a \leq \lceil a \leq \neg\lceil c$. By the greatest fixpoint property, $\nabla a(\nu f) \leq \nu e$. Therefore also $p(\nu f) = p\nabla a(\nu f) \leq p(\nu e)$.
4. $po(\nu g)(\nu h) = po(\mu g + (ai)^\omega)(\nu h) = po(\mu g)(\nu h) + po(ai)^\omega(\nu h) = po(ai)^\omega$ by Part 2 and Lemma 2.21. By the counter assumptions, o has right-inverse z semi-commuting with a, and i has right-inverse d semi-commuting with a. We thus obtain $po(ai)^\omega = poa^\omega = pa^\omega = p(\nu f)$ by Lemma 4.13.3 and Part 3. □

Remark. The assumptions of Lemma 4.14 may be relaxed by noting that the essential property $pa^*c \leq 0$ is equivalent to $p \leq \neg\langle a^*\rangle\lceil c = [a^*]\neg\lceil c$, which also suffices to prove Part 3. Intuitively, this characterises the states from which no a-transition paths into the domain of c exist. These are just the states where proper termination does not occur. (End of remark)

4.5. Putting the finite and infinite parts together

We can finally combine the results for the finite and infinite parts of the investigated recursion obtained in Sections 4.2 and 4.4. The next lemma shows how to split up the initial states according to these two parts. In the following we assume that certain countable sums of tests exist. This is the case, for instance, when the Boolean test algebra is complete. Distribution of arbitrary elements over these sums is also assumed.

Lemma 4.15.

1. *Let* $q_n = \lceil(a^n)\neg\lceil(a^{n+1})$ *as defined in Lemma 4.10 and assume that* $r =_{\text{def}} \sum_{n\in\mathbb{N}} q_n$ *exists. Then* $a^\infty =_{\text{def}} \neg\sum_{n\in\mathbb{N}} \neg\lceil(a^n)$ *exists and* $a^\infty + r = 1$.

2. *Let* $q'_n = \langle a^n\rangle\neg\lceil a$ *as defined in Lemma 4.10 and assume that* $r' =_{\mathrm{def}} \sum_{n\in\mathbb{N}} q'_n$ *exists and* a *distributes over this sum. Then* $r' = \langle a^*\rangle\neg\lceil a$ *and* $\nabla a + r' = 1$.
3. $\nabla a \leq a^\infty$, *and if* a *is deterministic, then* $\nabla a = a^\infty$.

Proof.

1. We show that $\neg a^\infty = \sum_{n\in\mathbb{N}} \neg\lceil(a^n) = r$. Let x be an upper bound of the tests in the sum, that is, $\forall n \in \mathbb{N} : \neg\lceil(a^n) \leq x$. Since $q_n \leq \neg\lceil(a^{n+1})$, we have $\forall n \in \mathbb{N} : q_n \leq x$. Therefore x is also an upper bound of r.
   It remains to show that r is an upper bound of the tests in the sum, or $\neg\lceil(a^n) \leq r$ for all $n \in \mathbb{N}$. This follows from $\sum_{i=0}^{n-1} q_i = \sum_{i=0}^{n} \neg\lceil(a^i)$ which we prove by induction. For the induction base $n = 0$ this is clear since $0 = \neg 1 = \neg\lceil 1 = \neg\lceil(a^0)$. For the induction step $n \geq 0$ we use the Boolean law $\neg p + pq = \neg p + q$ to calculate

$$\begin{aligned}\textstyle\sum_{i=0}^{n} q_i &= \textstyle\sum_{i=0}^{n-1} q_i + q_n = \sum_{i=0}^{n} \neg\lceil(a^i) + q_n = \sum_{i=0}^{n-1} \neg\lceil(a^i) + \neg\lceil(a^n) + \lceil(a^n)\neg\lceil(a^{n+1})\\ &= \textstyle\sum_{i=0}^{n-1} \neg\lceil(a^i) + \neg\lceil(a^n) + \neg\lceil(a^{n+1}) = \sum_{i=0}^{n+1} \neg\lceil(a^i).\end{aligned}$$

2. Since $q'_n \leq \langle a^*\rangle\neg\lceil a$ for each $n \in \mathbb{N}$ by isotony of diamond, we have $r' \leq \langle a^*\rangle\neg\lceil a$. The reverse inequality reduces by diamond star induction [12] to $\neg\lceil a + \langle a\rangle r' \leq r'$, which is equivalent to $\neg\lceil a \leq r'$ and $\langle a\rangle r' \leq r'$. The first property holds by definition of r' since $\neg\lceil a = q'_0$. The second property is equivalent to r' being a co-invariant of a. To show it, observe that $\langle a\rangle q'_n = \langle a\rangle\langle a^n\rangle\neg\lceil a = \langle a^{n+1}\rangle\neg\lceil a = q'_{n+1}$, hence $aq'_n \leq q'_{n+1}a$ by (dia). Therefore, and since $q'_0 a = \neg\lceil a\lceil aa = 0$,

$$ar' = \textstyle\sum_{n\in\mathbb{N}} aq'_n \leq \sum_{n\in\mathbb{N}} q'_{n+1}a = \sum_{n\in\mathbb{N}} q'_n a = r'a.$$

   The existence of the intermediate sums is guaranteed by the existence of r' and the distributivity of a. For the second claim, observe that $1 = \lceil a + \neg\lceil a = \langle a\rangle 1 + \neg\lceil a$ holds, which entails $1 \leq \nabla a + \langle a^*\rangle\neg\lceil a = \nabla a + r'$ by divergence co-induction.
3. The part $(\leq)$ follows since

$$\nabla a \leq \langle a^n\rangle\nabla a \leq \lceil(a^n) \Rightarrow \neg\lceil(a^n) \leq \neg\nabla a \Rightarrow \textstyle\sum_{n\in\mathbb{N}} \neg\lceil(a^n) \leq \neg\nabla a \Rightarrow \nabla a \leq a^\infty.$$

   For the part $(\geq)$ observe that determinacy of a implies $q_n = q'_n$ by Lemma 4.10.1, and therefore $r = r'$. In Part 1 we have shown $r = \neg a^\infty$, hence $\nabla a + \neg a^\infty = 1$ by Part 2, from which $\nabla a \geq a^\infty$ follows by shunting. □

Note that Part 1 gives a statement about q_n, which has been used in Section 4.2 to state the results about the finite part for non-deterministic a and choice in f. However, Lemma 4.14 does not carry on this generality to the infinite part. Part 3 shows that both ways of partitioning given in Lemma 4.15 coincide when a is deterministic.

Remark. This result has the following interpretation in terms of [42]: $\neg\nabla a = \Delta a$ describes the *progressively finite* states, that is, the states from which no infinite a-transition paths emerge, also known as the *initial part* of a. The test $\neg a^\infty$ describes the *progressively bounded* states, that is, those having an upper bound on the lengths of the emerging a-transition paths. Every progressively bounded state is progressively finite, that is, $\neg a^\infty \leq \neg\nabla a$. As detailed in Section 5, with deterministic a the progressively bounded and the progressively finite states are the same. The result can be seen as a special case of König's Infinity Lemma. (End of remark)

Using Lemma 4.15.2 we partition the states according to the finite and infinite parts. The finite parts are described by restricting the input states to q'_n for each $n \in \mathbb{N}$ and the infinite part is described by restriction to ∇a. Applying the results of Sections 4.2 and 4.4 we thus obtain an implementation of the recursion specified by f using two while-loops.

Theorem 4.16. *Let* a *and the choice in* f *be deterministic. Under the counter assumptions,* $\mu f = o(\mu g)(\mu h)$ *and* $\nu f = o(\nu g)(\nu h)$.

Proof. By Lemma 4.15.2, distributivity, Lemmas 4.14, 4.10.3 and 4.9, again distributivity, and again Lemma 4.15.2,

$$\begin{aligned}\mu f &= (\nabla a + \textstyle\sum_{n\in\mathbb{N}} q'_n)(\mu f) = \nabla a(\mu f) + \sum_{n\in\mathbb{N}} q'_n(\mu f)\\ &= \nabla a o(\mu g)(\mu h) + \textstyle\sum_{n\in\mathbb{N}} q'_n o(\mu g)(\mu h) = (\nabla a + \sum_{n\in\mathbb{N}} q'_n) o(\mu g)(\mu h) = o(\mu g)(\mu h).\end{aligned}$$

The calculation for ν proceeds similarly. □

Using Theorem 4.16 we can transform each of the recursive implementations of the factorial derived in Section 3 into two consecutive while-loops. However, our result also works for other recursions.

Example 4 (continued from Section 4.2). Consider the following program that prepends the list xs to the list ys. During the recursion an auxiliary list ts accumulates the reverse of xs, just to be consumed afterwards in prepending:

$$\begin{aligned}P_4 \quad = \quad & \mathbb{I} \lhd \mathit{isempty}\ xs \rhd (\mathit{push}\ (\mathit{top}\ xs)\ ts\ ;\ \mathit{pop}\ xs\ ;\ P_4\ ;\\ & \qquad\qquad \mathit{push}\ (\mathit{top}\ ts)\ ys\ ;\ \mathit{push}\ (\mathit{top}\ ts)\ xs\ ;\ \mathit{pop}\ ts).\end{aligned}$$

With the semiring elements

$$\begin{aligned}s &=_{\mathrm{def}} \neg(\mathit{isempty}\ xs)\\ a &=_{\mathrm{def}} (\mathit{push}\ (\mathit{top}\ xs)\ ts\ ;\ \mathit{pop}\ xs)\\ b &=_{\mathrm{def}} (\mathit{push}\ (\mathit{top}\ ts)\ ys\ ;\ \mathit{push}\ (\mathit{top}\ ts)\ xs\ ;\ \mathit{pop}\ ts)\end{aligned}$$

we obtain the fixpoint $P_4^\nu =_{\mathrm{def}} \nu x.saxb + \neg s$. Since $\ulcorner(sa)\ulcorner(\neg s) = s\ulcorner a\neg s = 0$, the choice is deterministic, as is sa. Moreover, the counter elements o, i, d, z chosen above obviously (semi-)commute with sa, b and $\neg s$ as required by Definition 4.8. Thus Theorem 4.16 yields

$$P_4^\nu = o(\nu x.saix + \neg s)(\nu x.dbx + z).$$

After being established by o, the test p is true during both loops (as can be shown using commutativity). Hence we obtain the simplified

$$P_4^\nu = o(\nu x.sa\hat{i}x + \neg s)(\nu x.\neg q\hat{d}bx + q)\hat{z}$$

which immediately translates to the program

$\mathit{push}\ 0\ \mathit{st}$
$\mathsf{while}\ \neg(\mathit{isempty}\ xs)\ \mathsf{do}$
    $\mathit{push}\ (\mathit{top}\ xs)\ ts\ ;\ \mathit{pop}\ xs\ ;\ \mathit{top}\ \mathit{st} := \mathit{top}\ \mathit{st} + 1$
$\mathsf{while}\ \mathit{top}\ \mathit{st} \neq 0\ \mathsf{do}$
    $\mathit{top}\ \mathit{st} := \mathit{top}\ \mathit{st} - 1\ ;\ \mathit{push}\ (\mathit{top}\ ts)\ ys\ ;\ \mathit{push}\ (\mathit{top}\ ts)\ xs\ ;\ \mathit{pop}\ ts$
$\mathit{pop}\ \mathit{st}$

Note that termination of the second while-loop is controlled by the counter on top of st.

If the underlying partial order is complete, we can generalise the μ-part of Theorem 4.16 to the non-deterministic case using μ-fusion, provided b commutes with both d and z.

Theorem 4.17. *Let S be a weak Kleene algebra in which arbitrary sums of elements exist and composition distributes over such sums. Assume that the counter assumptions hold and that $bdx + bzy = dbx + zby$ for arbitrary right contexts x and y. Then $\mu f = o(\mu g)(\mu h)$.*

Proof. For reasons that will become clear later, we restrict the domain of g. Let $C \subseteq S$ contain the elements that commute with i, that is, $C = \{x \in S \mid xi = ix\}$. Note that C is complete by the assumptions. Let $g' : C \to S$ be the restriction of g to C. By the commutativity assumptions,

$$g'(x)i = (aix + c)i = aixi + ci = iaix + ic = i(aix + c) = ig'(x),$$

hence the type of g' is even $g' : C \to C$. Closing our introductory remark, observe that $\mu g' = \mu g \in C$, since

$$(\mu g)i = (ai)^* ci = (ia)^* ic = i(ai)^* c = i(\mu g).$$

Let $e : C \to S$ be given by $e(x) = ox(\mu h)$. If we can prove $e \circ g' = f \circ e$, the claim follows by μ-fusion. Observe that

$$\begin{aligned} f(e(x)) &= ae(x)b + c = aox(\mu h)b + c, \\ e(g'(x)) &= og'(x)(\mu h) = o(aix + c)(\mu h) = oaix(\mu h) + oc(\mu h). \end{aligned}$$

But the latter equals $aoxi(\mu h) + co(\mu h)$ by the commutativity assumptions and the restriction to C. It therefore suffices to show $i(\mu h) = (\mu h)b$ and $o(\mu h) = 1$. To this end, observe that $(bd)^* bz \leq (db)^* zb$ follows by star induction from

$$bz + bd(db)^* zb = zb + db(db)^* zb = (1 + db(db)^*)zb = (db)^* zb.$$

A symmetric argument shows $(db)^* zb \leq (bd)^* bz$, hence

$$\begin{aligned} i(\mu h) &= i(db)^* z = i(1 + d(bd)^* b)z = iz + id(bd)^* bz = 0 + (db)^* zb = (\mu h)b, \\ o(\mu h) &= o(db)^* z = o(1 + d(bd)^* b)z = oz + od(bd)^* bz = 1 + 0 = 1, \end{aligned}$$

using star properties and the counter assumptions. □

4.6. Axiomatisation of symmetric linear recursion

Although we have now obtained quite a number of results on the function $f(x) =_{\text{def}} axb + c$, we still have no closed representation of its extremal fixpoints. This is no surprise, since it abstracts the context-free grammar $x ::= axb|c$. In the semiring of formal languages over an alphabet, with operations union and concatenation as $+$ and $\cdot$, its least fixpoint is the prototypical non-regular language $\{a^n cb^n \mid n \in \mathbb{N}\}$. Hence we cannot hope to express this least fixpoint using the star operation of plain Kleene algebra; we need something else.

One possibility is to axiomatise the fixpoints of f directly. We follow the pattern of Kleene and omega algebras; the recursions there are the special cases $a = 1$ or $b = 1$. An axiomatic treatment of least fixpoints of general context-free recursions can be found in [16]. The description of context-free languages by an iterated substitution operation is studied, for example, in [40].

To have a simple notation we denote the intended least fixpoint of f by $(a|c|b)$ and axiomatise it, following the pattern of Definition 2.22, by

$$a(a|c|b)b + c \leq (a|c|b) \qquad\qquad axb + c \leq x \Rightarrow (a|c|b) \leq x$$

Putting $b = 1$ and $a^* c = (a|c|1)$ we obtain the axioms of a weak Kleene algebra.

Lemma 4.18. *If the underlying weak semiring admits countable sums and composition distributes over them, then* $(a|c|b) = \sum_{n \in \mathbb{N}} a^n cb^n$.

Proof. Clearly, $\sum_{n \in \mathbb{N}} a^n cb^n$ is contracted by f, which shows the part $(\leq)$. The converse inequality follows, since for an arbitrary fixpoint f° of f a straightforward induction shows $a^n cb^n \leq f^\circ$ for all $n \in \mathbb{N}$. □

For the special case $c = 0$ we can prove a least fixpoint result without any assumptions on countable sums.

Lemma 4.19. *Assume a weak Kleene algebra and define* $e(x) =_{\text{def}} axb$. *Then* $\mu e = a^* 0$.

Proof. We first show $(\mu e)0 = a^*(\mu e)0$. The part $(\leq)$ holds by $1 \leq a^*$ and $(\geq)$ reduces by star induction to $(\mu e)0 + a(\mu e)0 \leq (\mu e)0$, that is, $a(\mu e)0 \leq (\mu e)0$. But this holds, since $(\mu e)0 = a(\mu e)b0 \geq a(\mu e)0$.

Now we have $a^* 0 \leq a^*(\mu e)0 = (\mu e)0 \leq \mu e$. The reverse inequality holds since $e(a^* 0) = aa^* 0b \leq a^* 0$. □

The greatest fixpoint of f can be axiomatised together with a^ω. For a demonic setting, suitable axioms are the following:

$$a^\omega = aa^\omega b \qquad\qquad x \leq axb + c \Rightarrow x \leq a^\omega + (a|c|b)$$

Putting again $b = 1$ we obtain the axioms of a weak omega algebra. By the omega unfold axiom all elements a^ω and hence, in particular, $\top = 1^\omega$ are left zeros; hence the above characterises demonic settings such as UTP or demonic refinement algebra [27].

Using the new axioms we can generalise Theorem 4.17 as follows.

Theorem 4.20. *Let S be a weak semiring in which countable sums of elements exist and composition distributes over such sums. Under the counter assumptions, $\mu f = o(\mu g)(\mu h)$.*

Proof. By Lemma 4.18,

$$o(\mu g)(\mu h) = o(ai|c|1)(db|z|1) = o(\textstyle\sum_{n\in\mathbb{N}}(ai)^n c1^n)(\sum_{m\in\mathbb{N}}(db)^m z1^m) = \sum_{m,n\in\mathbb{N}} o(ai)^n c(db)^m z.$$

Consider the terms of the sum:

- If $n < m$, then $o(ai)^n c(db)^m z = a^n coi^n(db)^n db(db)^{m-n-1}z = a^n cb^n 0 \leq a^n cb^n$.
- If $n = m$, then $o(ai)^n c(db)^m z = a^n coi^n(db)^n z = a^n cb^n oz = a^n cb^n$.
- If $n > m$, then $o(ai)^n c(db)^m z = a^n coi^{n-m-1}ib^m z = a^n cb^m oi^{n-m-1}0$
  $\leq a^n cb^m oi^{n-m-1}d^{n-m-1}zb^{n-m} = a^n cb^m b^{n-m} = a^n cb^n$.

Hence $\sum_{m,n\in\mathbb{N}} o(ai)^n c(db)^m z = \sum_{n\in\mathbb{N}} a^n cb^n = (a|c|b) = \mu f$ by Lemma 4.18. □

5. Noetherity and deterministic termination

We have already employed the convergence and divergence operators to good advantage. In this section we use them to discuss Noetherian elements, that is, elements that do not admit infinite transition paths. Subsequently this is used in the termination analysis of deterministic programs. In particular, we show that for these the difference between progressive finiteness and progressive boundedness does not arise.

5.1. Noetherian elements

The absence of infinite transition paths can be characterised as follows.

Definition 5.1. An element a of a convergence semiring is *Noetherian* if $\Delta a = 1$.

It is known [12] that a is Noetherian if and only if 0 is the only test expanded by $\langle a\rangle$, that is,

$$\forall p : p \leq \langle a\rangle p \Rightarrow p \leq 0 \qquad\qquad \text{(noe)}$$

We now develop some further useful properties of Noetherity and the convergence operator.

Lemma 5.2.

1. *a is Noetherian if and only if $\ulcorner a \leq \Delta a$.*
2. *qa is Noetherian if and only if $q\ulcorner a \leq \Delta(qa)$.*
3. *If $q \leq \Delta a$ then qa is Noetherian.*
4. *$\Delta(qa) \geq [q](\Delta a) = \neg q + \Delta a$.*
5. *$[qa] \geq [aq] \Rightarrow \Delta(qa) \leq [q](\Delta a)$.*

Proof.

1. $\ulcorner a \leq \Delta a \Leftrightarrow 1 \leq \neg\ulcorner a + \Delta a = [\ulcorner a]\Delta a = [\ulcorner a][a]\Delta a = [\ulcorner aa]\Delta a = [a]\Delta a = \Delta a$.
2. The claim follows immediately from Part 1.

3. The claim follows from Part 2, since $q\lceil a \leq q \leq \Delta a \leq \Delta(qa)$ by antitony of Δ.
4. By the rolling rule of fixpoint calculus and antitony of Δ,

$$\Delta(qa) = [q](\Delta(aq)) \geq [q](\Delta a) = \neg\lceil(q\neg\Delta a) = \neg q + \Delta a.$$

5. We show that $[q](\Delta a)$ is contracted by $[qa]$. By convergence unfold, the idempotence of tests, box composition, and the assumption,

$$[q](\Delta a) = [q][a](\Delta a) = [q][qa](\Delta a) \geq [q][aq](\Delta a) = [qa][q](\Delta a). \qquad \square$$

The premise of Lemma 5.2.5 reads more nicely in diamond form, namely $\langle qa\rangle \leq \langle aq\rangle$, meaning that extensionally q is an invariant of a.

5.2. Atomic tests and their images

To prepare our result about deterministic termination we need to consider atomic tests; they abstract singleton sets of states.

Definition 5.3. Consider a partial order with least element 0. Then a is an *atom* if it is a minimal non-zero element, that is,

$$a \neq 0 \ \wedge\ \forall b : b < a \Rightarrow b = 0.$$

An element b is a *subatom* if it is below an atom, that is, if it is 0 or an atom.

In a test semiring S an *atomic test* is an atom in the Boolean algebra $\mathrm{test}(S)$. We call S *test-atomistic* if $\mathrm{test}(S)$ is an atomistic Boolean algebra, that is, if arbitrary sums of atomic tests exist, every test is the sum of the atomic tests below it, and composition distributes through arbitrary sums of atomic tests.

For the remainder of this section we assume an ideal semiring with domain and a symmetrically axiomatised co-domain operation $\lceil$.

Definition 5.4. The *image* and *inverse image* of a test p under an element a are, respectively,

$$p : a =_{\mathrm{def}} (pa)\rceil \qquad\qquad a : p =_{\mathrm{def}} \lceil(ap)$$

Note that $a : p = \langle a\rangle p$.

Lemma 5.5.

1. *Consider a test-atomistic semiring. Let q be an atomic test and a be deterministic. Then $q : a$ is a subatom.*
2. *If $q \leq \lceil a$ then $q \leq a : (q : a)$.*
3. *Let q be an atomic test with $qa \neq 0$. Then $q \leq a : (q : a)$.*

Proof.

1. If $q : a = 0$ the claim is trivial. So assume $p =_{\mathrm{def}} q : a \neq 0$, hence $qa \neq 0$. Let $\mathrm{At}(p)$ be the set of atomic tests below p, hence $p = \sum_{r\in\mathrm{At}(p)} r$. We have

$$qa = qa(qa)\rceil = qap = qa\textstyle\sum_{r\in\mathrm{At}(p)} r = \sum_{r\in\mathrm{At}(p)} qar.$$

   Since determinacy is downward closed [10, Lemma 17], also qa is deterministic. Hence for all $r_1, r_2 \in \mathrm{At}(p)$ with $r_1 \neq r_2$ we have $\langle qa\rangle r_1 \cdot \langle qa\rangle r_2 = 0$. By atomicity of q and domain axiom (d2) we have $\langle qa\rangle r_i \in \{0, q\}$. Therefore for at most one $r \in \mathrm{At}(p)$ we have $\langle qa\rangle r \neq 0$, that is, $qar \neq 0$. In this case, $0 \neq (qar)\rceil = (qa)\rceil r \leq r$ and $qa = qar$, hence, using atomicity of r,

$$q : a = (qa)\rceil = (qar)\rceil = r.$$

2. $a : (q : a) = \lceil(a(qa)\rceil) \geq \lceil(qa(qa)\rceil) = \lceil(qa) = q\lceil a = q$.
3. Since $qa \neq 0$, we have $0 \neq \lceil(qa) \leq q$ and therefore $q = \lceil(qa) = q\lceil a$, that is, $q \leq \lceil a$, and Part 2 applies. $\square$

5.3. Deterministic termination

Using atoms we can sharpen our statements about Noetherity and restriction.

Lemma 5.6. *Let qa be Noetherian and q an atom. Then $qaq = 0$ and hence $(qa)^n = 0$ for all $n \geq 2$.*

Proof. Suppose $qaq \neq 0$, hence $\ulcorner(qaq) \neq 0$. By domain axiom (d2), $\ulcorner(qaq) \leq q$. By atomicity of q therefore $\ulcorner(qaq) = q$. Hence $q = \langle qa\rangle q$ and $q \neq 0$, a contradiction to Noetherity of qa. □

The final result shows that for deterministic elements the difference between progressive boundedness and progressive finiteness does not arise. Extending Lemma 4.15.3 we can now characterise single states as atoms and thus, in the Noetherian case, show the existence of an upper bound on the number of iterations starting from such a state.

Theorem 5.7. *Consider a test-atomistic convergence semiring. If a is deterministic and Noetherian and q is an atomic test, then there is an $n \in \mathbb{N}$ with $qa^n = 0$.*

Proof. Assume that $qa^n \neq 0$ for all $n \in \mathbb{N}$ and set $r_n =_{\text{def}} q : a^n$. Since determinacy is closed under composition [10, Lemma 22], all a^n are deterministic, too, and hence by Lemma 5.5.1 all r_n are atoms. Moreover, a straightforward calculation shows $r_{n+1} = r_n : a$. Hence, by Lemma 5.5.3, we have

$$r_n \leq a : r_{n+1}.$$

We now show that $p =_{\text{def}} \sum_{n\geq 0} r_n$ is expanded by $\langle a\rangle$. By universal distributivity of test multiplication and of domain, an index shift and the above inequality,

$$a : p = \textstyle\sum_{n\geq 0} a : r_n \geq \sum_{n>0} a : r_n = \sum_{n\geq 0} a : r_{n+1} \geq \sum_{n\geq 0} r_n \geq p.$$

Moreover, $p \neq 0$ since $q = r_0 \leq p$. By (noe) this contradicts Noetherity of a. □

6. Conclusion

This paper shows that the standard theory of normal designs carries over to our more general algebraic setting. It should be noted that the operations of complement and meet are not required for all semiring elements but only on the conditions. Additionally, we have pointed out normal designs as instances of various algebraic structures and thereby established connections to existing theories, enabling the application of well-known mathematical results. For example, the combination of the approach using ideal semirings with the matrix calculus of [35] leads to considerably simpler reasoning, since results about the star and omega iterations of matrices can be reused.

We have shown that normal designs can be equipped with box and diamond operators. While the box on the underlying semiring is the abstract counterpart of the wlp operator, the one on designs corresponds to wp. Hence the general soundness and completeness proof for the associated Hoare logic, originally developed for a partial correctness framework, can directly be applied to normal designs [36].

Our algebraic treatment can be carried over to the prescriptions of [15] that are similar to designs but reflect a general correctness view. For example, the proof of [35] showing that the normal prescriptions form a weak semiring can be adapted to our setting. The main drawback of using prescriptions is that their natural order cannot be used as the refinement order since they model erratic non-determinism. Indeed, neither $\mu x.x$ nor $\nu x.x$ is a suitable model of the infinitely looping program since $\mu x.x$ is neutral with respect to choice and $\nu x.x$ is not a left annihilator of the composition of prescriptions. To solve this problem one has to use the Egli-Milner order, see [36].

Connections to related algebraic approaches have been mentioned throughout this paper. Additional relations to theories of total and general correctness, such as [14, 5, 4, 37, 25, 13], are detailed in [21].

Further applications of our generalised results could be handling trace semantics and other semantical models, thus dealing with healthiness conditions such as R1–R3 of UTP in a purely algebraic fashion. The presented method could also guide the extension of the basic UTP model by parameters that describe further observations as proposed in [26].

Acknowledgement. We are grateful to Roland Glück, Peter Höfner and an anonymous referee for valuable remarks.

References

[1] C. J. Aarts. Galois connections presented calculationally. Master's thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1992.
[2] 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, 10 February 1995.
[3] F. L. Bauer. Programming as an evolutionary process. In *Proceedings of the 2nd International Conference on Software Engineering*, pages 223–234. IEEE Computer Society Press, 1976.
[4] R. Berghammer and H. Zierer. Relational algebraic semantics of deterministic and nondeterministic programs. *Theoretical Computer Science*, 43:123–147, 1986.
[5] 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-Verlag, 1979.
[6] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.
[7] 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-Verlag, 2000.
[8] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.
[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
[10] J. Desharnais and B. Möller. Characterizing determinacy in Kleene algebras. *Information Sciences*, 139(3):253–273, December 2001.
[11] J. Desharnais, B. Möller, and G. Struth. Algebraic notions of termination. Report 2006-23, Institut für Informatik, Universität Augsburg, October 2006.
[12] J. Desharnais, B. Möller, and G. Struth. Kleene algebra with domain. *ACM Transactions on Computational Logic*, 7(4):798–833, October 2006.
[13] J. Desharnais, B. Möller, and F. Tchier. Kleene under a modal demonic star. *Journal of Logic and Algebraic Programming*, 66(2):127–160, February–March 2006.
[14] E. W. Dijkstra. *A Discipline of Programming*. Prentice Hall, 1976.
[15] 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, July 2001.
[16] Z. Ésik and H. Leiß. Algebraically complete semirings and Greibach normal form. *Annals of Pure and Applied Logic*, 3(1–3):173–203, May 2005.
[17] S. A. Greibach. *Theory of Program Structures: Schemes, Semantics, Verification*, volume 36 of *Lecture Notes in Computer Science*. Springer-Verlag, 1975.
[18] W. Guttmann. Non-termination in Unifying Theories of Programming. In W. MacCaull, M. Winter, and I. Düntsch, editors, *Relational Methods in Computer Science 2005*, volume 3929 of *Lecture Notes in Computer Science*, pages 108–120. Springer-Verlag, 2006.
[19] W. Guttmann. *Algebraic Foundations of the Unifying Theories of Programming*. Dissertation, Universität Ulm, December 2007.
[20] W. Guttmann. Lazy relations. In R. Berghammer, B. Möller, and G. Struth, editors, *Relations and Kleene Algebra in Computer Science*, volume 4988 of *Lecture Notes in Computer Science*, pages 138–154. Springer-Verlag, 2008.
[21] 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-Verlag, 2006.
[22] W. Guttmann and B. Möller. Normal design algebra. Report 2006-28, Institut für Informatik, Universität Augsburg, December 2006.
[23] U. Hebisch and H. J. Weinert. *Halbringe*. Teubner, 1993.
[24] F. W. von Henke. Formal transformations and the development of programs. In J. Gruska, editor, *Mathematical Foundations of Computer Science 1977*, volume 53 of *Lecture Notes in Computer Science*, pages 288–296. Springer-Verlag, 1977.
[25] W. H. Hesselink. *Programs, Recursion and Unbounded Choice*. Cambridge University Press, 1992.
[26] C. A. R. Hoare and J. He. *Unifying theories of programming*. Prentice Hall Europe, 1998.
[27] 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-Verlag, 2006.
[28] P. Höfner and G. Struth. Automated reasoning in Kleene algebra. In F. Pfenning, editor, *Automated Deduction: CADE-21*, volume 4603 of *Lecture Notes in Computer Science*, pages 279–294. Springer-Verlag, 2007.
[29] D. Kozen. On Kleene algebras and closed semirings. In B. Rovan, editor, *Mathematical Foundations of Computer Science 1990*, volume 452 of *Lecture Notes in Computer Science*, pages 26–47. Springer-Verlag, 1990.
[30] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, May 1994.

[31] D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, 19(3):427–443, May 1997.
[32] V. Mathieu and J. Desharnais. Verification of pushdown systems using omega algebra with domain. In W. MacCaull, M. Winter, and I. Düntsch, editors, *Relational Methods in Computer Science 2005*, volume 3929 of *Lecture Notes in Computer Science*, pages 188–199. Springer-Verlag, 2006.
[33] B. Möller. Derivation of graph and pointer algorithms. In B. Möller, H. A. Partsch, and S. A. Schuman, editors, *Formal Program Development*, volume 755 of *Lecture Notes in Computer Science*, pages 123–160. Springer-Verlag, 1993.
[34] B. Möller. Lazy Kleene algebra. In D. Kozen, editor, *Mathematics of Program Construction*, volume 3125 of *Lecture Notes in Computer Science*, pages 252–273. Springer-Verlag, 2004.
[35] 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-Verlag, 2006.
[36] 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-Verlag, 2006.
[37] G. Nelson. A generalization of Dijkstra's calculus. *ACM Transactions on Programming Languages and Systems*, 11(4):517–561, October 1989.
[38] H. A. Partsch. *Specification and Transformation of Programs: A Formal Approach to Software Development*. Springer-Verlag, 1990.
[39] S. Popkorn. *First Steps in Modal Logic*. Cambridge University Press, 1994.
[40] V. N. Red'ko. Some aspects of the theory of languages. *Cybernetics and Systems Analysis*, 1(4):15–26, July 1965.
[41] G. Schmidt, C. Hattensperger, and M. Winter. Heterogeneous relation algebra. In C. Brink, W. Kahl, and G. Schmidt, editors, *Relational Methods in Computer Science*, chapter 3, pages 39–53. Springer-Verlag, Wien, 1997.
[42] G. Schmidt and T. Ströhlein. *Relationen und Graphen*. Springer-Verlag, 1989.
[43] K. Solin and J. von Wright. Refinement algebra with operators for enabledness and termination. In T. Uustalu, editor, *Mathematics of Program Construction*, volume 4014 of *Lecture Notes in Computer Science*, pages 397–415. Springer-Verlag, 2006.
[44] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.
[45] J. von Wright. Towards a refinement algebra. *Science of Computer Programming*, 51(1–2):23–45, May 2004.