An Algebraic Approach to Computations with Progress

Walter Guttmann

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

Abstract

The notion of progress appears in various computation models, for example, in the form of traces getting longer, passing of real time, incrementing a counter, going from termination to non-termination. We introduce a model of sequential computations that generalises and abstracts these examples. We generalise existing algebras for non-terminating executions and instantiate these with our model. Using these algebras we derive an approximation order for computations with time and for trace-based computations. We introduce a generalisation of omega algebras to express iteration in the new model.

Keywords: approximation, axiomatic program semantics, iteration algebras, non-termination, omega algebras, relations, sequential computations, time, traces

1. Introduction

A computation model is a mathematical description of what happens when a program is run on a computer. Such models facilitate the construction of correct programs by mathematical calculation, in addition to the experimental method of testing programs. Models differ in the kinds of computation they can represent and the precision they achieve.

In this paper we consider models for sequential computations. An example of such a computation model are binary relations: the starting state of a computation is related to the possible final states. Having several possible final states is useful for specifications that leave freedom to the programmer. Moreover, when a program is run it might abort due to an error or it might fail to terminate; extended relational models take into account these phenomena [2, 24, 10, 20, 14].

Relations describe the input-output behaviour of a computation. This provides only a simplistic model of progress: a computation will be in its starting state before it will be in its final state. How long a computation takes to go from one state to another, or which intermediate states it goes through, is not represented by the input-output relation.

Models which include a notion of time or traces take care of these phenomena [23, 20, 21, 11]. Time can be modelled by adding an extra variable to the state space; in this case, progress means that the value of this variable increases. A trace can describe the history of a computation, that is, the sequence of states from its start to the current state; in this case, progress means that the trace gets longer. Different domains of time can be used for modelling real time or an abstraction such as the number of execution steps. Traces can be combined with timing information to show when a computation passes through a state.

All of these computation models support a notion of progress. The aim of the present paper is to distil this concept. To this end we construct a general computation model that captures progress and instantiates to the various models mentioned above. Algebras and algebraic methods that have previously been developed and refined for several relational computation models provide the technical means for this work [14, 15, 16, 17, 18]. The general motivation for this work is to understand how different computation models that support a notion of progress are related and to obtain a common theory. The reason for using algebras is that they facilitate such a unification [24, 12, 14, 15, 18], are well supported by theorem proving technology [26, 27] and yet powerful enough to yield complex results which have been applied in the development and verification of programs [1, 5].

The models discussed in this paper support non-deterministic computations, which are useful for specification purposes. Accordingly, a computation is made up of a set of executions, each of which describes one possible behaviour of the computation. In previous works we have distinguished finite executions, which terminate successfully, aborting executions, which fail due to an error, and infinite executions, which do not terminate. In the present paper we refine the latter kind of executions into two classes, which we call (potentially) incomplete executions and (actually) infinite executions. While incomplete executions arise as approximations to the semantics of recursion and as non-terminating but unproductive executions, infinite executions yield actually infinite traces or unbounded progress in time.

Our computation model thus distinguishes four kinds of execution: finite, aborting, incomplete and infinite. These kinds of execution are represented by relations over a state space, which carries the values of program variables and additional information such as time or traces. Progress is modelled by a preorder on the state space; because this is also a relation, it integrates nicely with the various kinds of execution. The relations are collected in matrices, which generalise representations used in previous works [16, 21, 11, 17, 18]. The matrices facilitate the use of well-known constructions for calculating various operations that serve as the basis for program constructs [31, 19].

We prove that these operations satisfy the axioms of algebras which have previously been used to describe choice, conjunction, sequential composition and various forms of iteration [28, 5, 14, 15]. Moreover, we introduce algebras that describe the incomplete and infinite executions; they generalise algebras which have previously been used to describe the states from which such executions exist [14, 18]. By instantiating these algebras we automatically inherit hundreds of properties that have previously been derived using interactive and automated theorem provers. We also obtain an approximation order for our computations, which is the key ingredient for the semantics of recursive programs; in particular, this covers while-loops. We express the necessary fixpoints in this approximation order in terms of fixpoints in the less complex refinement order.

The contributions of the present paper are as follows:

- A new computation model that describes progress. It generalises previous models which feature progress in the form of time or traces or by distinguishing terminating and non-terminating executions.
- A distinction between potentially incomplete and actually infinite executions. The latter are represented by heterogeneous relations.
- A new algebra for incomplete and infinite executions. It generalises previous algebras that describe the states from which such executions exist.
- Instances of the new algebra and previously introduced algebras for iteration, consequences of which include separation and refinement theorems and various program transformations.
- Capped omega algebras, which generalise omega algebras to describe fixpoints of bounded affine functions.

Section 2 recalls the basic algebraic structures used in the remainder of this paper and basic properties of relations. Section 3 introduces our model of computations with progress and shows how it generalises previous models. Section 4 defines the basic operations of choice, conjunction and sequential composition. Section 5 introduces an algebra with an operation that describes the incomplete and infinite executions, and instantiates it with our model of computations with progress. Section 6 obtains an approximation order for the semantics of recursion for our computations. Section 7 introduces capped omega algebras and instantiates previous algebras for iteration.

All algebraic structures axiomatised in this paper, but not the concrete models, have been implemented in Isabelle/HOL [33], making heavy use of its integrated automated theorem provers and SMT solvers [34, 3]. The theories contain proofs of Theorems 3, 4, 6 and 8 as well as hundreds of other properties including, for example, a fixpoint calculus, separation theorems and Back's atomicity refinement theorem [14, 15, 18]. By instantiating these algebras we automatically inherit these results. The Isabelle/HOL theory files are available at http://www.csse.canterbury.ac.nz/walter.guttmann/algebra/.

2. Algebraic Structures for Sequential Computations

In this section we axiomatise the operations of non-deterministic choice, conjunction and sequential composition, and various forms of iteration featured by many computation models. Our presentation follows [17]. We also recall basic definitions and properties of relations.

2.1. Choice, Conjunction and Sequential Composition

A *bounded distributive lattice* is an algebraic structure $(S, +, \curlywedge, 0, \top)$ such that the following axioms hold:

$$\begin{aligned} x + (y + z) &= (x + y) + z & \qquad x \curlywedge (y \curlywedge z) &= (x \curlywedge y) \curlywedge z \\ x + y &= y + x & x \curlywedge y &= y \curlywedge x \\ x + x &= x & x \curlywedge x &= x \\ 0 + x &= x & \top \curlywedge x &= x \\ x + (y \curlywedge z) &= (x + y) \curlywedge (x + z) & x \curlywedge (y + z) &= (x \curlywedge y) + (x \curlywedge z) \\ x + (x \curlywedge y) &= x & x \curlywedge (x + y) &= x \end{aligned}$$

Here and in all axioms given in this paper, free variables are understood to be universally quantified. The *lattice order* $x \leq y \Leftrightarrow x + y = y \Leftrightarrow x \curlywedge y = x$ has least element 0, greatest element $\top$, least upper bound operation $+$ and greatest lower bound operation $\curlywedge$. The operations $+$ and $\curlywedge$ are $\leq$-isotone.

An idempotent semiring without a right annihilator – simply called a *semiring* in the remainder of this paper – is an algebraic structure $(S, +, \cdot, 0, 1)$ such that the following axioms hold:

$$\begin{aligned} x + (y + z) &= (x + y) + z & \qquad x \cdot (y \cdot z) &= (x \cdot y) \cdot z & \qquad x \cdot (y + z) &= (x \cdot y) + (x \cdot z) \\ x + y &= y + x & 1 \cdot x &= x & (x + y) \cdot z &= (x \cdot z) + (y \cdot z) \\ x + x &= x & x \cdot 1 &= x & 0 \cdot x &= 0 \\ 0 + x &= x \end{aligned}$$

In particular, $x \cdot 0 = 0$ is not an axiom. The operation $\cdot$ is $\leq$-isotone. We abbreviate $x \cdot y$ as xy.

A *lattice-ordered semiring* is an algebraic structure $(S, +, \curlywedge, \cdot, 0, 1, \top)$ whose reduct $(S, +, \curlywedge, 0, \top)$ is a bounded distributive lattice and whose reduct $(S, +, \cdot, 0, 1)$ is a semiring. Lattices and variants of semirings have frequently been used for modelling computations; for example, see [28, 1, 5, 38, 32, 25, 19]. This particular combination has been used in [12, 15, 17, 18].

In many computation models, the operation $+$ represents non-deterministic choice, the operation $\curlywedge$ conjunction, the operation $\cdot$ sequential composition, 0 the computation with no executions, 1 the computation that does not change the state, $\top$ the computation with all executions, and $\leq$ the refinement relation.

2.2. Fixpoints

Let S be a set partially ordered by $\leq$ and let $f : S \to S$. Provided they exist, the $\leq$-least and $\leq$-greatest fixpoints of f are denoted by μf and νf, respectively:

$$\begin{aligned} f(\mu f) &= \mu f & \qquad f(x) = x &\Rightarrow \mu f \leq x \\ f(\nu f) &= \nu f & f(x) = x &\Rightarrow \nu f \geq x \end{aligned}$$

We abbreviate $\mu(\lambda x.f(x))$ by $\mu x.f(x)$ and $\nu(\lambda x.f(x))$ by $\nu x.f(x)$. The existence of fixpoints is typically guaranteed by completeness of the structure, properties of the function or further axioms.

Some algebraic structures – for example, the Kleene algebras presented in Section 2.3 – use axioms that correspond to $\leq$-least prefixpoints, namely, $f(\mu f) \leq \mu f$ and $f(x) \leq x \Rightarrow \mu f \leq x$. Provided it exists, the $\leq$-least prefixpoint of a $\leq$-isotone function is also its $\leq$-least fixpoint. A similar remark holds for the dual case of $\leq$-greatest postfixpoints.

2.3. Iteration

The following algebras capture various fixpoints of the function $\lambda x.yx + z$, which are useful to describe iterations. For example, iterations of this kind occur in while-loops, where y represents the body of the loop and the choice between yx and z reflects continuation or termination of the loop.

A *Kleene algebra* $(S, +, \cdot, ^*, 0, 1)$ adds to a semiring an operation * with the following unfold and induction axioms [28]:

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

It follows that $y^*z = \mu x.yx + z$ and $zy^* = \mu x.xy + z$. The operation * is $\leq$-isotone.

An *omega algebra* $(S, +, \cdot, ^*, ^\omega, 0, 1)$ adds to a Kleene algebra an operation $^\omega$ with the following unfold and induction axioms [5, 32]:

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

It follows that $y^\omega + y^*z = \nu x.yx + z$ and $y^\omega = \nu x.yx$. In particular, $\top = 1^\omega$ is the $\leq$-greatest element. Moreover $y^\omega = y^\omega\top$ and the operation $^\omega$ is $\leq$-isotone. See [38] for an alternative axiomatisation of $\nu x.yx+z$.

In many computation models, the operation * represents finite iteration and $^\omega$ is used to represent infinite iteration. Kleene algebras and omega algebras axiomatise the $\leq$-least and the $\leq$-greatest fixpoints of the function $\lambda x.yx + z$, respectively. For computation models that require different fixpoints of this function, we use the following generalisations of Kleene algebras.

An *extended binary itering* $(S, +, \cdot, \star, 0, 1)$ adds to a semiring a binary operation $\star$ with the following axioms [15]:

$$\begin{aligned} (x + y) \star z &= (x \star y) \star (x \star z) & \qquad x \star (y + z) &= (x \star y) + (x \star z) \\ (xy) \star z &= z + x((yx) \star (yz)) & \qquad (x \star y)z &\leq x \star (yz) \end{aligned}$$

$$\begin{gathered} zx \leq y(y \star z) + w \Rightarrow z(x \star v) \leq y \star (zv + w(x \star v)) \\ xz \leq z(y \star 1) + w \Rightarrow x \star (zv) \leq z(y \star v) + (x \star (w(y \star v))) \\ w(x \star (yz)) \leq (w(x \star y)) \star (w(x \star y)z) \end{gathered}$$

It follows that $y \star z$ is a fixpoint of $\lambda x.yx + z$. The operation $\star$ is $\leq$-isotone. The element $y \star z$ corresponds to iterating y an unspecified number of times, followed by a single occurrence of z. This may involve an infinite number of iterations of y.

In models that satisfy $(x \star y)z = x \star (yz)$, the binary itering operation specialises to a unary operation $^\circ$ with the following simpler axioms, which are obtained by setting $x^\circ = x \star 1$. An *itering* $(S, +, \cdot, ^\circ, 0, 1)$ adds to a semiring an operation $^\circ$ with the sumstar and productstar equations of [6] and two simulation axioms [14]:

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

It follows that $y^\circ z$ is a fixpoint of $\lambda x.yx + z$ and that zy° is a fixpoint of $\lambda x.xy + z$. The operation $^\circ$ is $\leq$-isotone.

The simulation axioms of iterings are used instead of the induction axioms of Kleene algebras and omega algebras. They are still powerful enough to prove complex program transformations, but are not restricted to $\leq$-least or $\leq$-greatest fixpoints and hence general enough to hold in many different computation models.

Every Kleene algebra is an itering using $x^\circ = x^*$. Every omega algebra is an itering using $x^\circ = x^\omega 0 + x^*$. Every itering is an extended binary itering using $x \star y = x^\circ y$. Further instances and consequences of iterings and binary iterings are given in [14, 15]; they include demonic refinement algebras [38].

2.4. Relations

We finally summarise terminology and properties of relations [36]. For sets A and B a (heterogeneous) relation R of type $A \leftrightarrow B$ is a subset of the Cartesian product $A \times B$; we write $R : A \leftrightarrow B$. Relations of type $A \leftrightarrow A$ are called homogeneous. The relations of type $A \leftrightarrow B$ form a complete lattice with union $\cup$, intersection $\cap$ and partial order $\subseteq$. The least element of the partial order is the empty relation $\mathsf{O} = \emptyset$

and the greatest element is the universal relation $\mathsf{T} = A \times B$. Relational composition of $Q : A \leftrightarrow B$ and $R : B \leftrightarrow C$ yields the relation $QR : A \leftrightarrow C$ defined by

$$QR = \{(x, z) \mid \exists y \in B : (x, y) \in Q \wedge (y, z) \in R\}$$

The identity relation $\mathsf{I} : A \leftrightarrow A$ is defined by $\mathsf{I} = \{(x, x) \mid x \in A\}$. We frequently omit type information from relational expressions and assume that they are typed in the most general type-correct way. Basic properties are:

- Composition is associative, $\subseteq$-isotone and distributes over $\cup$.
- $\mathsf{O}R = \mathsf{O}$ and $R\mathsf{O} = \mathsf{O}$.
- $\mathsf{I}R = R = R\mathsf{I}$.

A relation R is a *vector* if $R\mathsf{T} = R$, *reflexive* if $\mathsf{I} \subseteq R$, *transitive* if $R^2 = RR \subseteq R$, and a *preorder* if R is reflexive and transitive. A vector $R : A \leftrightarrow B$ represents the subset of elements of A that are related by R to every element of B; the other elements of A are related to no element of B. Relational composition has higher precedence than $\cup$ and $\cap$.

Relations of type $A \leftrightarrow A$ form a Kleene algebra where R^* is the reflexive transitive closure of R. They also form an omega algebra where R^ω is a vector that represents the set of elements of A from which there is an infinite R-transition sequence.

3. Computations with Progress

In this section we describe a model of computations which have a notion of progress. Examples of progress are: traces getting longer, passing of real time, incrementing a counter, going from termination to non-termination. Technically we model progress by a preorder. The model distinguishes between computations that are infinite and those that are not. According to this distinction, the state space A of a computation is separated into two disjoint parts $A = A_{\mathrm{fin}} \cup A_\infty$. Table 1 gives examples of such a separation, where D is the set of values that program variables can take.

A_{fin}	A_∞	model
D	$\{\infty\}$	Boolean time; computation terminates in a state in D or does not terminate
$D \times \mathbb{N}$	$\{\infty\}$	abstract time; steps are counted
$D \times \mathbb{R}$	$\{\infty\}$	real time; a clock is used
D^+	D^ω	traces; finite and infinite sequences over D
$\mathbb{R} \rightarrow_{\mathrm{fin}} D$	$\mathbb{R} \rightarrow_\infty D$	timed traces; $\rightarrow_{\mathrm{fin}}$ constructs the partial functions with bounded domain; $\rightarrow_\infty$ constructs the partial functions with unbounded domain

Table 1: Separation of state space in various computation models

The Boolean-time model distinguishes only termination and non-termination. A computation either terminates in a state in D or does not terminate, which is represented by going to state ∞. The abstract-time model adds a counter which keeps track of the number of execution steps in the terminating case. The real-time model refines the integer counter to a real value representing the time when the computation is in a particular state. In trace-based models, the state space contains the current state of the computation and its history. Simple traces just keep track of a sequence of states, while timed traces add the time at which the computation was at each state in its history. This is similar to the distinction between abstract time and real time. See [23, 24, 22, 20, 21] for a discussion of these and related models. Further restrictions are imposed in some models. For example, in the real-time models underlying [20], domains of timed traces are intervals from an initial time to the current time. Without this restriction a different notion of progress is obtained.

Computations in our general model comprise executions of different kinds. We distinguish finite executions, which terminate successfully, aborting executions, which fail due to an error, incomplete executions, which are unproductive and used in approximation, and actually infinite executions. The various kinds of execution are represented by relations as described in the following.

A relation $R : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$ represents the finite executions of a computation. A pair $(x, x') \in R$ means that there is a finite execution of the computation which starts in state x and ends in state x', that is, x' is a possible output for input x. Several outputs for the same input indicate non-determinism.

Executions that abort due to an error are represented by another relation $P : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$ of the same type. The second component x' of a pair $(x, x') \in P$ might represent, for example, the state of the computation immediately before it aborts or information about the error.

Incomplete executions are represented by yet another relation $N : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$ of the same type. The second component x' of a pair $(x, x') \in N$ might provide additional information about an execution besides the fact that the computation is potentially incomplete when started in state x. Incomplete executions arise, in particular, when fixpoints of recursions are computed by approximation. In this process an incomplete execution can be replaced with finite, aborting or actually infinite executions as the approximation gets better. We understand incomplete executions to be *potentially* incomplete, that is, it is not necessary that the approximation is improved to a (complete) finite, aborting or infinite execution this way. Another purpose of incomplete executions is to deal with unproductive recursions.

Our model distinguishes between incomplete executions and actually infinite executions. This distinction is observed, for example, in trace-based models if computations are not required to add to the trace. The infinite executions are represented by a heterogeneous relation $Q : A_{\text{fin}} \leftrightarrow A_{\infty}$; it has different source and target sets.

Two particular relations $\mathsf{F} : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$ and $\mathsf{F}_{\infty} : A_{\text{fin}} \leftrightarrow A_{\infty}$ model progress. We only assume that F is a preorder, that is, $\mathsf{I} \subseteq \mathsf{F} = \mathsf{F}^2$, and that the related transitivity property $\mathsf{F}\mathsf{F}_{\infty} = \mathsf{F}_{\infty}$ holds. Table 2 shows the constants F and F_{∞} for the examples in table 1, using the prefix relation $\preceq$ on finite and infinite sequences.

$\mathsf{F} : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$	A_{fin}	$\mathsf{F}_{\infty} : A_{\text{fin}} \leftrightarrow A_{\infty}$	A_{∞}	model
T	D	T	$\{\infty\}$	Boolean time
$\{((v,t),(v',t')) \mid t \leq t'\}$	$D \times \mathbb{N}$	T	$\{\infty\}$	abstract time
$\{((v,t),(v',t')) \mid t \leq t'\}$	$D \times \mathbb{R}$	T	$\{\infty\}$	real time
$\preceq$	D^+	$\preceq$	D^ω	traces
$\subseteq$	$\mathbb{R} \rightarrow_{\text{fin}} D$	$\subseteq$	$\mathbb{R} \rightarrow_{\infty} D$	timed traces

Table 2: Progress in various computation models

Thus F_{∞} describes progress from the finite to the infinite – for example, increasing time to ∞ – and F describes progress that remains finite – for example, lengthening to a finite trace. The predicates $t \leq t'$ and $tr \preceq tr'$ relating initial and final values of time t and trace tr are used in [23, 24, 20]. In particular, the healthiness condition R1 of the Unifying Theories of Programming [24] requires $X = X \wedge (tr \preceq tr')$ for a predicate X. By translating this condition to relations and applying a property of the lattice order we obtain $X \subseteq (tr \preceq tr')$ and, further abstracting from the concrete kind of progress, $X \subseteq \mathsf{F}$. We impose this abstract progress requirement in our computation model.

We moreover impose the two closure requirements $N\mathsf{F} \subseteq N$ and $N\mathsf{F}_{\infty} \subseteq Q$. The former implies $N\mathsf{F} = N$ since F is reflexive. Consider a state x and its image under N or Q, that is, the set $\{y \mid (x,y) \in N \vee (x,y) \in Q\}$. The closure requirements state that each image set is upward closed with respect to the progress preorder. For example, for trace-based models this means that if N relates trace x to trace y, then it also relates x to each finite trace z which has the prefix y. Thus subsets of N represent better approximations, which is the purpose of this technical restriction.

The relations N, P, Q and R along with other constant relations between A_{fin} and A_{∞} are collected in a matrix. A computation is a 4×4 matrix of the following form and type, where N, P, Q, R satisfy the *progress requirements* $N, P, R \subseteq \mathsf{F}$ and $Q \subseteq \mathsf{F}_\infty$, and the *closure requirements* $N\mathsf{F} \subseteq N$ and $N\mathsf{F}_\infty \subseteq Q$:

$$\begin{pmatrix} \mathsf{I} & \mathsf{O} & \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} & \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{O} & \mathsf{I} & \mathsf{O} \\ N & P & Q & R \end{pmatrix} : \begin{pmatrix} A_{\text{fin}} \leftrightarrow A_{\text{fin}} & A_{\text{fin}} \leftrightarrow A_{\text{fin}} & A_{\text{fin}} \leftrightarrow A_\infty & A_{\text{fin}} \leftrightarrow A_{\text{fin}} \\ A_{\text{fin}} \leftrightarrow A_{\text{fin}} & A_{\text{fin}} \leftrightarrow A_{\text{fin}} & A_{\text{fin}} \leftrightarrow A_\infty & A_{\text{fin}} \leftrightarrow A_{\text{fin}} \\ A_\infty \leftrightarrow A_{\text{fin}} & A_\infty \leftrightarrow A_{\text{fin}} & A_\infty \leftrightarrow A_\infty & A_\infty \leftrightarrow A_{\text{fin}} \\ A_{\text{fin}} \leftrightarrow A_{\text{fin}} & A_{\text{fin}} \leftrightarrow A_{\text{fin}} & A_{\text{fin}} \leftrightarrow A_\infty & A_{\text{fin}} \leftrightarrow A_{\text{fin}} \end{pmatrix}$$

We denote such a matrix by $(N|P|Q|R)$.

Smaller matrices of relations have been used in [31, 19] to eliminate auxiliary variables from computations in the Unifying Theories of Programming [24]. A benefit of this approach is that many operations reduce to standard matrix constructions; for example, see the calculation of the Kleene star in Section 7.1. The matrices were subsequently generalised and applied to various other computation models [21, 11, 17, 18]. The differences between the new matrices and representations used in these previous works are as follows:

- Components N and Q are distinct, while all of [21, 11, 17, 18] used just a single component which, for the purpose of approximation, corresponds to Q in the present model. Moreover, by having component N in addition to component R the model can distinguish between incomplete executions which might be further improved, and finite executions which are maximal in the approximation order.
- None of the components are restricted to be vectors, while a number of models in [21, 11, 17, 18] had such restrictions for various components. The restrictions for each model are detailed in [17].
- All components are subjected to the progress requirements, which abstract from the kinds of progress used in [21, 11].
- The relations in the third row and column are heterogeneous relations, where matrices of homogeneous relations were used in [17, 18]. This is similar to matrices over typed Kleene algebras and typed omega algebras [30, 13] and was suggested in a different context by a referee of [18].

Using terminology of [20, 11], conscriptions, extended conscriptions, timed conscriptions and timed reactive conscriptions (a combination of extended conscriptions and timed reactive designs) are instances of the general matrix model above. See [7] for matrices over omega algebras that are similar to conscriptions but can have trace sets as entries.

The following examples illustrate the various kinds of execution represented in the four components of $(N|P|Q|R)$. First, consider three computations which contain only finite executions:

- The computation that does not change variables and does not make any progress is $\mathsf{skip} = (\mathsf{O}|\mathsf{O}|\mathsf{O}|\mathsf{I})$.
- Using abstract time, the computation that does not change variables but lets one unit of time pass is $\mathsf{pass} = (\mathsf{O}|\mathsf{O}|\mathsf{O}|R)$ where $R = \{((v,t),(v,t+1)) \mid v \in D \wedge t \in \mathbb{N}\}$.
- Using traces, the computation that increments a counter is $\mathsf{inc} = (\mathsf{O}|\mathsf{O}|\mathsf{O}|R)$ where $R = \{(tr, tr') \mid tr' = tr \mathbin{+\!\!+} [last(tr) + 1]\}$ appends a number to the sequence tr, namely the increment of the last element of tr.

Taking each of these computations as the body of an endless while-loop, we obtain the following as will be shown in Theorem 10:

- $\mathsf{while\ true\ do\ skip} = (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})$ contains incomplete and infinite executions. The incomplete executions arise during approximation and remain because no progress takes place: time does not lapse or traces do not get longer, depending on the model. The result contains F and F_∞ rather than I due to the above closure requirements.
- $\mathsf{while\ true\ do\ pass} = (\mathsf{O}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})$ contains only infinite executions. Time ticks in each iteration of the while-loop and becomes ∞ according to the semantics of the loop, given that $\mathsf{F}_\infty = \mathsf{T} : A_{\text{fin}} \leftrightarrow \{\infty\}$.

- $\mathsf{while\ true\ do\ inc} = (\mathsf{O}|\mathsf{O}|Q|\mathsf{O})$ where $Q = \{(tr, tr') \mid tr' = tr +\!\!+ \mathit{from}(\mathit{last}(tr))\}$ and $\mathit{from}(n)$ is the infinite sequence $n+1$, $n+2$, $n+3$, ... Again there are only infinite executions. The trace gets one element longer in each iteration of the while-loop and we obtain an infinite trace as the result of the loop.

Finally, the component P of $(N|P|Q|R)$ contains executions when the computation aborts, for example, because of an integer division by zero.

4. Choice, Conjunction and Sequential Composition

In this section we look at how basic program and specification constructs are represented in our computation model. In particular, we show that the operations preserve the progress requirements and the closure requirements.

- Non-deterministic choice is given by the componentwise union of the involved matrices:

$$(N_1|P_1|Q_1|R_1) + (N_2|P_2|Q_2|R_2) = (N_1 \cup N_2|P_1 \cup P_2|Q_1 \cup Q_2|R_1 \cup R_2)$$

  To verify the progress requirements, note that $N_1 \cup N_2 \subseteq \mathsf{F}$ follows from $N_1 \subseteq \mathsf{F}$ and $N_2 \subseteq \mathsf{F}$. Similarly $P_1 \cup P_2 \subseteq \mathsf{F}$ follows from $P_1 \subseteq \mathsf{F}$ and $P_2 \subseteq \mathsf{F}$. Using heterogeneous relations, $Q_1 \cup Q_2 \subseteq \mathsf{F}_\infty$ follows from $Q_1 \subseteq \mathsf{F}_\infty$ and $Q_2 \subseteq \mathsf{F}_\infty$. Finally, $R_1 \cup R_2 \subseteq \mathsf{F}$ follows from $R_1 \subseteq \mathsf{F}$ and $R_2 \subseteq \mathsf{F}$. To verify the closure requirements, note that $(N_1 \cup N_2)\mathsf{F} = N_1\mathsf{F} \cup N_2\mathsf{F} \subseteq N_1 \cup N_2$ follows from $N_1\mathsf{F} \subseteq N_1$ and $N_2\mathsf{F} \subseteq N_2$. Similarly $(N_1 \cup N_2)\mathsf{F}_\infty = N_1\mathsf{F}_\infty \cup N_2\mathsf{F}_\infty \subseteq Q_1 \cup Q_2$ follows from $N_1\mathsf{F}_\infty \subseteq Q_1$ and $N_2\mathsf{F}_\infty \subseteq Q_2$.

- Conjunction is given by the componentwise intersection of the involved matrices:

$$(N_1|P_1|Q_1|R_1) \curlywedge (N_2|P_2|Q_2|R_2) = (N_1 \cap N_2|P_1 \cap P_2|Q_1 \cap Q_2|R_1 \cap R_2)$$

  Note that $N_1 \cap N_2 \subseteq N_1 \subseteq \mathsf{F}$ and $P_1 \cap P_2 \subseteq P_1 \subseteq \mathsf{F}$ and $Q_1 \cap Q_2 \subseteq Q_1 \subseteq \mathsf{F}_\infty$ and $R_1 \cap R_2 \subseteq R_1 \subseteq \mathsf{F}$. Moreover $(N_1 \cap N_2)\mathsf{F} \subseteq N_1\mathsf{F} \cap N_2\mathsf{F} \subseteq N_1 \cap N_2$ and $(N_1 \cap N_2)\mathsf{F}_\infty \subseteq N_1\mathsf{F}_\infty \cap N_2\mathsf{F}_\infty \subseteq Q_1 \cap Q_2$.

- Sequential composition is given by the matrix product, where union and relational composition replace addition and multiplication. This elaborates as follows:

$$(N_1|P_1|Q_1|R_1) \cdot (N_2|P_2|Q_2|R_2) = (N_1 \cup R_1N_2|P_1 \cup R_1P_2|Q_1 \cup R_1Q_2|R_1R_2)$$

  Observe that the types of the involved relations match: for example, $R_1 : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$ composed with $Q_2 : A_{\text{fin}} \leftrightarrow A_\infty$ gives a relation with the same type as $Q_1 : A_{\text{fin}} \leftrightarrow A_\infty$. Note also that $N_1, N_2, P_1, P_2, R_1, R_2 \subseteq \mathsf{F}$ and $Q_1, Q_2 \subseteq \mathsf{F}_\infty$ imply $N_1 \cup R_1N_2 \subseteq \mathsf{F} \cup \mathsf{F}^2 = \mathsf{F}$ and $P_1 \cup R_1P_2 \subseteq \mathsf{F} \cup \mathsf{F}^2 = \mathsf{F}$ and $Q_1 \cup R_1Q_2 \subseteq \mathsf{F}_\infty \cup \mathsf{F}\mathsf{F}_\infty = \mathsf{F}_\infty$ and $R_1R_2 \subseteq \mathsf{F}^2 = \mathsf{F}$. Moreover the closure requirements are obtained by $(N_1 \cup R_1N_2)\mathsf{F} = N_1\mathsf{F} \cup R_1N_2\mathsf{F} \subseteq N_1 \cup R_1N_2$ and $(N_1 \cup R_1N_2)\mathsf{F}_\infty = N_1\mathsf{F}_\infty \cup R_1N_2\mathsf{F}_\infty \subseteq Q_1 \cup R_1Q_2$.

- The refinement order is the componentwise set inclusion order:

$$(N_1|P_1|Q_1|R_1) \leq (N_2|P_2|Q_2|R_2) \;\Leftrightarrow\; N_1 \subseteq N_2 \wedge P_1 \subseteq P_2 \wedge Q_1 \subseteq Q_2 \wedge R_1 \subseteq R_2$$

- The computation with no executions is

$$0 = (\mathsf{O}|\mathsf{O}|\mathsf{O}|\mathsf{O})$$

  Clearly $\mathsf{O} \subseteq \mathsf{F}$ and $\mathsf{O} \subseteq \mathsf{F}_\infty$ and $\mathsf{O}\mathsf{F} \subseteq \mathsf{O}$ and $\mathsf{O}\mathsf{F}_\infty \subseteq \mathsf{O}$. The computation 0 is a neutral element of non-deterministic choice, an annihilator of conjunction, a left annihilator of sequential composition and the least element in the refinement order.

- The computation with all executions is

$$\top = (\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F})$$

Clearly $\mathsf{F} \subseteq \mathsf{F}$ and $\mathsf{F}_\infty \subseteq \mathsf{F}_\infty$ and $\mathsf{FF} \subseteq \mathsf{F}$ and $\mathsf{FF}_\infty \subseteq \mathsf{F}_\infty$. The computation $\top$ is an annihilator of non-deterministic choice, a neutral element of conjunction and the greatest element in the refinement order.

- The computation that does not change the state is

$$1 = (\mathsf{O}|\mathsf{O}|\mathsf{O}|\mathsf{I})$$

Clearly $\mathsf{O} \subseteq \mathsf{I} \subseteq \mathsf{F}$ and $\mathsf{O} \subseteq \mathsf{F}_\infty$ and $\mathsf{OF} \subseteq \mathsf{O}$ and $\mathsf{OF}_\infty \subseteq \mathsf{O}$. The computation 1 is a neutral element of sequential composition.

- The computation with all incomplete and infinite executions is

$$\mathsf{L} = (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})$$

Clearly $\mathsf{O} \subseteq \mathsf{F}$ and $\mathsf{FF} \subseteq \mathsf{F}$ and $\mathsf{FF}_\infty \subseteq \mathsf{F}_\infty$. The computation L is a left annihilator of sequential composition. It is the least element in the approximation order which is given in Section 6.

Together with other simple matrix calculations we obtain the following basic structure using the above operations. This structure is shared by many computation models and provides, in particular, properties of basic programming constructs frequently used in reasoning about programs.

Theorem 1. *Let $S = \{(N|P|Q|R) \mid N, P, R \subseteq \mathsf{F} : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$ and $Q \subseteq \mathsf{F}_\infty : A_{\text{fin}} \leftrightarrow A_\infty$ and $N\mathsf{F} \subseteq N$ and $N\mathsf{F}_\infty \subseteq Q\}$. Then $(S, +, \curlywedge, \cdot, 0, 1, \top)$ is a lattice-ordered semiring. The lattice order is $\leq$.*

To define the semantics of recursion an approximation order is needed. The refinement order cannot be used for this purpose because L is not its least element.

5. Incomplete and Infinite Executions

In this section we describe an operation n such that $n(x)$ represents the incomplete and infinite executions of a computation x. To this end we generalise axiomatisations of n previously given in [14, 17, 18]. We first discuss these previous approaches in Section 5.1, then present the new axioms in Section 5.2 and finally instantiate these with our model of computations with progress in Section 5.3.

5.1. States with Infinite Executions

We motivate the operation n by discussing its meaning in previous works [14, 17, 18]. In these works, $n(x)$ describes the set of states from which a computation x has infinite executions. Sets are represented as tests, which are elements below or equal to 1 in the lattice order.

Consider the set S of computations. The operation n maps computations to tests; moreover we single out the computation $\mathsf{L} \in S$ that represents the endless loop, that is, the computation with all infinite executions. The general reason for looking at n and L is that they will be used to define an approximation order on computations in Section 6. The axiomatisation of n and L is based on the following Galois connection between $n(S)$ and S with lower adjoint $\lambda p.p\mathsf{L}$ and upper adjoint n:

$$n(x)\mathsf{L} \leq y \;\Leftrightarrow\; n(x) \leq n(y)$$

Its significance is that $n(y)$ is the greatest test whose composition with L is below y. The sequential composition $p\mathsf{L}$ of a test p and the computation L restricts the executions of L to those whose starting state is in the set described by p. The axioms of n are designed to enforce this characteristic Galois connection as well as properties of the approximation order on computations, which underlies the semantics of recursion.

The following axioms for n and L were used in [18], which also provides a more detailed explanation of each axiom:

$$\begin{array}{llll}
(p1) & n(x)+n(y) = n(n(x)\top + y) & (p6) & n(x) \leq n(\mathsf{L}) \curlywedge 1 \\
(p2) & n(x)n(y) = n(n(x)y) & (p7) & n(x)\mathsf{L} \leq x \\
(p3) & n(x)n(x+y) = n(x) & (p8) & n(\mathsf{L})x \leq x\,n(\mathsf{L}) \\
(p4) & n(\mathsf{L})x = (x \curlywedge \mathsf{L}) + n(\mathsf{L}0)x & (p9) & x\,n(y)\top \leq x0 + n(xy)\top \\
(p5) & x\mathsf{L} = x0 + n(x\mathsf{L})\mathsf{L} & (p10) & x\top y \curlywedge \mathsf{L} \leq x\mathsf{L}y \\
(p11) & n(\mathsf{L})x^\omega \leq x^* n(x^\omega)\top & (p12) & x\mathsf{L} \leq x\mathsf{L}x\mathsf{L}
\end{array}$$

Consequences of these axioms include properties such as $n(x)n(x) = n(x)$ and $n(x) \leq 1$ which are appropriate for tests, but not for the intended new interpretation.

The above axioms have been developed in [18] for describing non-terminating executions in a unified treatment of strict and non-strict computations; the latter can produce defined outputs from undefined inputs. An axiomatisation for strict computations has been given in [14].

The domain operation of [9] and the enabledness operator of [37] describe the set of states from which a computation has any kind of execution. The termination operator of [37] is closer to the operation n but differs in several respects. First, it is axiomatised in demonic refinement algebras [38], which is just one of the instances of n-algebras; see [14, 18] for a plethora of other computation models. The axioms for the termination operator are too strong to cover all these models; for example, they do not apply to models which distinguish aborting and incomplete executions. Second, the termination operator maps to assertions, which are duals of tests; in particular, they are elements above or equal to 1 in the lattice order. Third, the termination operator describes the set of states from which a computation is guaranteed to terminate, which is the set of states from which only finite executions exist.

5.2. Incomplete and Infinite Executions

In the present paper we intend $n(x)$ to represent the incomplete and infinite executions of computation x as a whole, not just the states from which such executions exist. For example, in trace-based models we wish to access the actual traces, not just their starting states. Accordingly, the constant L will represent the computation with all incomplete and infinite executions. To achieve this generalisation, we revise the axioms for n and L. Several axioms can be kept unchanged, a few axioms need to be weakened – that is, replaced with consequences of the previous axioms – and a few axioms need to be modified – that is, replaced with properties that hold in the previous and new computation models.

An *n-algebra* $(S, +, \curlywedge, \cdot, n, 0, 1, \mathsf{L}, \top)$ adds to a lattice-ordered semiring an operation $n : S \to S$ and a constant L with the following axioms, using the abbreviation $c(x) = n(\mathsf{L})\top \curlywedge x$:

$$\begin{array}{llll}
(n1) & n(x)+n(y) = n(n(x)\top + y) & (n6) & n(x) \leq n(\mathsf{L}) \\
(n2) & n(x)n(y) = n(n(x)y) & (n7) & n(x)\mathsf{L} \leq x \\
(n3) & n(x) \leq n(x+y) & (n8) & c(xy)z \leq c(x)y\,c(z) \\
(n4) & n(\mathsf{L})x = (x \curlywedge \mathsf{L}) + n(\mathsf{L}0)x & (n9) & x\,n(y)\top \leq x0 + n(xy)\top \\
(n5) & x\,n(y)\mathsf{L} = x0 + n(x\,n(y)\mathsf{L})\mathsf{L} & (n10) & x\top y \curlywedge \mathsf{L} \leq x\mathsf{L}y
\end{array}$$

Counterexamples generated by Nitpick [4] witness that each of the axioms $(n1)$–$(n10)$ is independent of the remaining axioms of n-algebras. While L represents the computation with all incomplete and infinite executions, constants for the computations with all aborting or all finite executions are not provided.

The axioms $(n1)$, $(n2)$, $(n4)$, $(n7)$, $(n9)$ and $(n10)$ remain unchanged. Axioms $(n3)$, $(n5)$ and $(n6)$ are consequences of the previous axiomatisation $(p1)$–$(p10)$. In particular, axiom $(p3)$ is weakened to $(n3)$, which states that n is $\leq$-isotone. The more specific axiom $(n5)$ replaces $(p5)$ which is no longer sufficient in the current weaker context. Axiom $(p6)$ is weakened to $(n6)$, which states that the incomplete and infinite executions in L are maximal.

Axiom $(n8)$ is used to propagate the restriction $c(x)$ into and through sequential compositions. This restriction is needed to handle partial-correctness computation models which do not represent infinite executions. Previously, it was implemented by $c(x) = n(\mathsf{L})x$ and the propagation was achieved by axiom

($p8$). This is no longer appropriate since n does not map to tests, so we switch to the present definition $c(x) = n(\mathsf{L})\top \curlywedge x$ to achieve the same effect. In all previously considered computation models and in the new model introduced in this paper $n(\mathsf{L})\top$ is either $\top$ or 0. Hence either $c(x) = x$ or $c(x) = 0$ holds in these models; in both cases axiom ($n8$) follows.

An *n-omega algebra* $(S, +, \curlywedge, \cdot, n, {}^*, {}^\omega, 0, 1, \mathsf{L}, \top)$ adds to an n-algebra $(S, +, \curlywedge, \cdot, n, 0, 1, \mathsf{L}, \top)$ and an omega algebra $(S, +, \cdot, {}^*, {}^\omega, 0, 1)$ the following axioms:

$$(n11)\quad c(x^\omega) \le x^* n(x^\omega)\top \qquad\qquad (n12)\quad x\mathsf{L} \le x\mathsf{L}x\mathsf{L}$$

Axiom ($n12$) remains unchanged and axiom ($n11$) is modified from the previous axiom ($p11$) as described above for axiom ($n8$).

5.3. Instance for Computations with Progress

We now instantiate the new n-algebras with our new computation model. To this end we define

$$n(N|P|Q|R) = (\mathsf{O}|\mathsf{O}|Q|N)$$

This operation extracts the incomplete executions N as well as the infinite executions Q. The aborting executions P and the finite executions R are discarded. Hence this definition matches the intuition for n given above. The incomplete executions are shifted to the component which usually represents the finite executions. This makes it possible to copy this information into the other components by sequential composition.

Note that n preserves the progress requirements since $\mathsf{O} \subseteq N \subseteq \mathsf{F}$ and $Q \subseteq \mathsf{F}_\infty$, and the closure requirements since $\mathsf{OF} \subseteq \mathsf{O}$ and $\mathsf{OF}_\infty = \mathsf{O} \subseteq Q$. Moreover, the following result shows that this definition of n satisfies the axioms of n-algebras.

Theorem 2. *Let $S = \{(N|P|Q|R) \mid N, P, R \subseteq \mathsf{F} : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$ and $Q \subseteq \mathsf{F}_\infty : A_{\text{fin}} \leftrightarrow A_\infty$ and $N\mathsf{F} \subseteq N$ and $N\mathsf{F}_\infty \subseteq Q\}$. Then $(S, +, \curlywedge, \cdot, n, 0, 1, \mathsf{L}, \top)$ is an n-algebra.*

Proof.

($n1$) The claim follows by

$$\begin{array}{cll} & n(n(N_1|P_1|Q_1|R_1)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) + (N_2|P_2|Q_2|R_2)) & \\ = & n((\mathsf{O}|\mathsf{O}|Q_1|N_1)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) + (N_2|P_2|Q_2|R_2)) & \text{definition of } n \\ = & n((N_1\mathsf{F}|N_1\mathsf{F}|Q_1 \cup N_1\mathsf{F}_\infty|N_1\mathsf{F}) + (N_2|P_2|Q_2|R_2)) & \text{definition of } \cdot \\ = & n((N_1|N_1|Q_1|N_1) + (N_2|P_2|Q_2|R_2)) & \text{closure requirements} \\ = & n(N_1 \cup N_2|N_1 \cup P_2|Q_1 \cup Q_2|N_1 \cup R_2) & \text{definition of } + \\ = & (\mathsf{O}|\mathsf{O}|Q_1 \cup Q_2|N_1 \cup N_2) & \text{definition of } n \\ = & (\mathsf{O}|\mathsf{O}|Q_1|N_1) + (\mathsf{O}|\mathsf{O}|Q_2|N_2) & \text{definition of } + \\ = & n(N_1|P_1|Q_1|R_1) + n(N_2|P_2|Q_2|R_2) & \text{definition of } n \end{array}$$

($n2$) The claim follows by

$$\begin{array}{cll} & n(n(N_1|P_1|Q_1|R_1)(N_2|P_2|Q_2|R_2)) & \\ = & n((\mathsf{O}|\mathsf{O}|Q_1|N_1)(N_2|P_2|Q_2|R_2)) & \text{definition of } n \\ = & n(N_1N_2|N_1P_2|Q_1 \cup N_1Q_2|N_1R_2) & \text{definition of } \cdot \\ = & (\mathsf{O}|\mathsf{O}|Q_1 \cup N_1Q_2|N_1N_2) & \text{definition of } n \\ = & (\mathsf{O}|\mathsf{O}|Q_1|N_1)(\mathsf{O}|\mathsf{O}|Q_2|N_2) & \text{definition of } \cdot \\ = & n(N_1|P_1|Q_1|R_1)n(N_2|P_2|Q_2|R_2) & \text{definition of } n \end{array}$$

($n3$) The claim follows by

$$\begin{array}{cll} & n(N_1|P_1|Q_1|R_1) & \\ = & (\mathsf{O}|\mathsf{O}|Q_1|N_1) & \text{definition of } n \\ \le & (\mathsf{O}|\mathsf{O}|Q_1 \cup Q_2|N_1 \cup N_2) & \text{definition of } \le \\ = & n(N_1 \cup N_2|P_1 \cup P_2|Q_1 \cup Q_2|R_1 \cup R_2) & \text{definition of } n \\ = & n((N_1|P_1|Q_1|R_1) + (N_2|P_2|Q_2|R_2)) & \text{definition of } + \end{array}$$

($n4$) The claim follows by

$$\begin{array}{rll}
& (N|P|Q|R) \curlywedge (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) + n((\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})(\mathsf{O}|\mathsf{O}|\mathsf{O}|\mathsf{O}))(N|P|Q|R) & \\
= & (N \cap \mathsf{F}|P \cap \mathsf{O}|Q \cap \mathsf{F}_\infty|R \cap \mathsf{O}) + n(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})(N|P|Q|R) & \text{definitions of } \curlywedge \text{ and } \cdot \\
= & (N|\mathsf{O}|Q|\mathsf{O}) + n(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})(N|P|Q|R) & \text{progress requirements} \\
= & (N|\mathsf{O}|Q|\mathsf{O}) + (\mathsf{O}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{F})(N|P|Q|R) & \text{definition of } n \\
= & (N|\mathsf{O}|Q|\mathsf{O}) + (\mathsf{F}N|\mathsf{F}P|\mathsf{F}_\infty \cup \mathsf{F}Q|\mathsf{F}R) & \text{definition of } \cdot \\
= & (N \cup \mathsf{F}N|\mathsf{F}P|Q \cup \mathsf{F}_\infty \cup \mathsf{F}Q|\mathsf{F}R) & \text{definition of } + \\
= & (\mathsf{F}N|\mathsf{F}P|\mathsf{F}_\infty \cup \mathsf{F}Q|\mathsf{F}R) & \mathsf{I} \subseteq \mathsf{F} \\
= & (\mathsf{O}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{F})(N|P|Q|R) & \text{definition of } \cdot \\
= & n(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})(N|P|Q|R) & \text{definition of } n
\end{array}$$

($n5$) The claim follows by

$$\begin{array}{rll}
& (N_1|P_1|Q_1|R_1)(\mathsf{O}|\mathsf{O}|\mathsf{O}|\mathsf{O}) + n((N_1|P_1|Q_1|R_1)n(N_2|P_2|Q_2|R_2)(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}))(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \\
= & (N_1|P_1|Q_1|\mathsf{O}) + n((N_1|P_1|Q_1|R_1)(\mathsf{O}|\mathsf{O}|Q_2|N_2)(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}))(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \text{definitions of } \cdot \text{ and } n \\
= & (N_1|P_1|Q_1|\mathsf{O}) + n((N_1|P_1|Q_1|R_1)(N_2\mathsf{F}|\mathsf{O}|Q_2 \cup N_2\mathsf{F}_\infty|\mathsf{O}))(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \text{definition of } \cdot \\
= & (N_1|P_1|Q_1|\mathsf{O}) + n((N_1|P_1|Q_1|R_1)(N_2|\mathsf{O}|Q_2|\mathsf{O}))(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \text{closure requirements} \\
= & (N_1|P_1|Q_1|\mathsf{O}) + n(N_1 \cup R_1N_2|P_1|Q_1 \cup R_1Q_2|\mathsf{O})(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \text{definition of } \cdot \\
= & (N_1|P_1|Q_1|\mathsf{O}) + (\mathsf{O}|\mathsf{O}|Q_1 \cup R_1Q_2|N_1 \cup R_1N_2)(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \text{definition of } n \\
= & (N_1|P_1|Q_1|\mathsf{O}) + (N_1\mathsf{F} \cup R_1N_2\mathsf{F}|\mathsf{O}|Q_1 \cup R_1Q_2 \cup N_1\mathsf{F}_\infty \cup R_1N_2\mathsf{F}_\infty|\mathsf{O}) & \text{definition of } \cdot \\
= & (N_1|P_1|Q_1|\mathsf{O}) + (N_1 \cup R_1N_2|\mathsf{O}|Q_1 \cup R_1Q_2|\mathsf{O}) & \text{closure requirements} \\
= & (N_1 \cup R_1N_2|P_1|Q_1 \cup R_1Q_2|\mathsf{O}) & \text{definition of } + \\
= & (N_1|P_1|Q_1|R_1)(N_2|\mathsf{O}|Q_2|\mathsf{O}) & \text{definition of } \cdot \\
= & (N_1|P_1|Q_1|R_1)(N_2\mathsf{F}|\mathsf{O}|Q_2 \cup N_2\mathsf{F}_\infty|\mathsf{O}) & \text{closure requirements} \\
= & (N_1|P_1|Q_1|R_1)(\mathsf{O}|\mathsf{O}|Q_2|N_2)(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \text{definition of } \cdot \\
= & (N_1|P_1|Q_1|R_1)n(N_2|P_2|Q_2|R_2)(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \text{definition of } n
\end{array}$$

($n6$) The claim follows by

$$\begin{array}{rll}
& n(N|P|Q|R) & \\
= & (\mathsf{O}|\mathsf{O}|Q|N) & \text{definition of } n \\
\leq & (\mathsf{O}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{F}) & \text{progress requirements, definition of } \leq \\
= & n(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \text{definition of } n
\end{array}$$

($n7$) The claim follows by

$$\begin{array}{rll}
& n(N|P|Q|R)(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \\
= & (\mathsf{O}|\mathsf{O}|Q|N)(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \text{definition of } n \\
= & (N\mathsf{F}|\mathsf{O}|Q \cup N\mathsf{F}_\infty|\mathsf{O}) & \text{definition of } \cdot \\
= & (N|\mathsf{O}|Q|\mathsf{O}) & \text{closure requirements} \\
\leq & (N|P|Q|R) & \text{definition of } \leq
\end{array}$$

($n8$) The claim follows since $c(x) = x$ for each $x \in S$ holds by

$$\begin{array}{rll}
& n(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) \curlywedge (N|P|Q|R) & \\
= & (\mathsf{O}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{F})(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) \curlywedge (N|P|Q|R) & \text{definition of } n \\
= & (\mathsf{FF}|\mathsf{FF}|\mathsf{F}_\infty \cup \mathsf{FF}_\infty|\mathsf{FF}) \curlywedge (N|P|Q|R) & \text{definition of } \cdot \\
= & (\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) \curlywedge (N|P|Q|R) & \mathsf{FF} = \mathsf{F} \text{ and } \mathsf{FF}_\infty = \mathsf{F}_\infty \\
= & (\mathsf{F} \cap N|\mathsf{F} \cap P|\mathsf{F}_\infty \cap Q|\mathsf{F} \cap R) & \text{definition of } \curlywedge \\
= & (N|P|Q|R) & \text{progress requirements}
\end{array}$$

$(n9)$ The claim follows by

$$\begin{aligned}
& (N_1|P_1|Q_1|R_1)n(N_2|P_2|Q_2|R_2)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) && \\
= {} & (N_1|P_1|Q_1|R_1)(\mathsf{O}|\mathsf{O}|Q_2|N_2)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) && \text{definition of } n \\
= {} & (N_1|P_1|Q_1|R_1)(N_2\mathsf{F}|N_2\mathsf{F}|Q_2 \cup N_2\mathsf{F}_\infty|N_2\mathsf{F}) && \text{definition of } \cdot \\
= {} & (N_1|P_1|Q_1|R_1)(N_2|N_2|Q_2|N_2) && \text{closure properties} \\
= {} & (N_1 \cup R_1N_2|P_1 \cup R_1N_2|Q_1 \cup R_1Q_2|R_1N_2) && \text{definition of } \cdot \\
\leq {} & (N_1 \cup R_1N_2|P_1 \cup N_1 \cup R_1N_2|Q_1 \cup R_1Q_2|N_1 \cup R_1N_2) && \text{definition of } \leq \\
= {} & (N_1|P_1|Q_1|\mathsf{O}) + (N_1 \cup R_1N_2|N_1 \cup R_1N_2|Q_1 \cup R_1Q_2|N_1 \cup R_1N_2) && \text{definition of } + \\
= {} & (N_1|P_1|Q_1|\mathsf{O}) + && \\
& (N_1\mathsf{F} \cup R_1N_2\mathsf{F}|N_1\mathsf{F} \cup R_1N_2\mathsf{F}|Q_1 \cup R_1Q_2 \cup N_1\mathsf{F}_\infty \cup R_1N_2\mathsf{F}_\infty|N_1\mathsf{F} \cup R_1N_2\mathsf{F}) && \text{closure requirements} \\
= {} & (N_1|P_1|Q_1|\mathsf{O}) + (\mathsf{O}|\mathsf{O}|Q_1 \cup R_1Q_2|N_1 \cup R_1N_2)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) && \text{definition of } \cdot \\
= {} & (N_1|P_1|Q_1|\mathsf{O}) + n(N_1 \cup R_1N_2|P_1 \cup R_1P_2|Q_1 \cup R_1Q_2|R_1R_2)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) && \text{definition of } n \\
= {} & (N_1|P_1|Q_1|R_1)(\mathsf{O}|\mathsf{O}|\mathsf{O}|\mathsf{O}) + n((N_1|P_1|Q_1|R_1)(N_2|P_2|Q_2|R_2))(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) && \text{definition of } \cdot
\end{aligned}$$

$(n10)$ The claim follows by

$$\begin{aligned}
& (N_1|P_1|Q_1|R_1)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F})(N_2|P_2|Q_2|R_2) \curlywedge (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) && \\
\leq {} & (N_1|P_1|Q_1|R_1)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F})(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) \curlywedge (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) && \text{progress requirements} \\
= {} & (N_1|P_1|Q_1|R_1)(\mathsf{F} \cup \mathsf{FF}|\mathsf{F} \cup \mathsf{FF}|\mathsf{F}_\infty \cup \mathsf{FF}_\infty|\mathsf{FF}) \curlywedge (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) && \text{definition of } \cdot \\
= {} & (N_1|P_1|Q_1|R_1)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) \curlywedge (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) && \mathsf{FF} = \mathsf{F} \text{ and } \mathsf{FF}_\infty = \mathsf{F}_\infty \\
= {} & (N_1 \cup R_1\mathsf{F}|P_1 \cup R_1\mathsf{F}|Q_1 \cup R_1\mathsf{F}_\infty|R_1\mathsf{F}) \curlywedge (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) && \text{definition of } \cdot \\
= {} & ((N_1 \cup R_1\mathsf{F}) \cap \mathsf{F}|\mathsf{O}|(Q_1 \cup R_1\mathsf{F}_\infty) \cap \mathsf{F}_\infty|\mathsf{O}) && \text{definition of } \curlywedge \\
= {} & (N_1 \cup R_1\mathsf{F}|\mathsf{O}|Q_1 \cup R_1\mathsf{F}_\infty|\mathsf{O}) && \text{progress requirements} \\
\leq {} & (N_1 \cup R_1\mathsf{F}|P_1|Q_1 \cup R_1\mathsf{F}_\infty|\mathsf{O}) && \text{definition of } \leq \\
= {} & (N_1|P_1|Q_1|R_1)(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) && \text{definition of } \cdot \\
= {} & (N_1|P_1|Q_1|R_1)(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})(N_2|P_2|Q_2|R_2) && \text{definition of } \cdot
\end{aligned}$$

□

The following result shows a number of properties which follow in n-algebras. They are used to derive more complex results, such as Theorems 4 and 8. In particular, automated theorem proving in Isabelle/HOL works better with additional properties than just with the axioms.

Theorem 3. *Let S be an n-algebra. Then the following properties hold for $x, y \in S$:*

1. $x \leq y \Rightarrow n(x) \leq n(y)$
2. $n(x) \leq n(y) \Leftrightarrow n(x)\mathsf{L} \leq y$
3. $x \leq y \Leftrightarrow x \leq y + \mathsf{L} \wedge x \leq y + n(y)\top$
4. $x \leq y \Leftrightarrow x \leq y + \mathsf{L} \wedge c(x) \leq y + n(y)\top$
5. $n(0)\mathsf{L} = 0$
6. $\mathsf{L} \leq n(\mathsf{L})\top$
7. $n(\mathsf{L}) = n(\top)$
8. $n(\mathsf{L})\mathsf{L} = \mathsf{L}$
9. $\mathsf{LL} = \mathsf{L}\top = \mathsf{L}\top\mathsf{L} = \mathsf{L}$
10. $\mathsf{L}x \leq \mathsf{L}$
11. $x\mathsf{L} \leq x0 + \mathsf{L}$
12. $x\mathsf{L} = x0 + n(x\mathsf{L})\mathsf{L}$
13. $n(x)0 \leq \mathsf{L}$
14. $n(x\top) = n(x\mathsf{L})$
15. $n(x)\mathsf{L} \leq x\mathsf{L}$
16. $n(x) \leq n(x\mathsf{L})$
17. $n(x) \leq n(\mathsf{L})n(x)$
18. $n(n(x)\mathsf{L}) = n(x)$
19. $n(x)n(\mathsf{L}) = n(x)$
20. $n(0)n(x) = n(0)$
21. $n(x)n(y) \leq n(x)$
22. $n(x\,n(y)\mathsf{L}) \leq n(xy)$
23. $x\,n(y)\mathsf{L} \leq x0 + n(xy)\mathsf{L}$
24. $x\,n(y)\top \leq xy + n(xy)\top$
25. $n(x)\top y \leq xy + n(x0)\top$
26. $n(x)\top y \leq xy + n(xy)\top$
27. $x\top \curlywedge \mathsf{L} \leq x\mathsf{L}$
28. $x\top y \curlywedge \mathsf{L} \leq x0 + \mathsf{L}y$
29. $x\top y \curlywedge \mathsf{L} = x\mathsf{L}y \curlywedge \mathsf{L}$
30. $(x \curlywedge \mathsf{L})0 \leq x0 \curlywedge \mathsf{L}$
31. $n(x)\mathsf{L} \leq x \curlywedge \mathsf{L}$
32. $x \curlywedge \mathsf{L} \leq n(\mathsf{L})x$

33. $n(x) \curlywedge \mathsf{L} \le x$
34. $(n(x) \curlywedge \mathsf{L})\top \le x$
35. $(n(x) \curlywedge \mathsf{L})\top \le n(x)\mathsf{L}$
36. $n(x) = n(x \curlywedge \mathsf{L})$
37. $n(x)\top \curlywedge \mathsf{L} = n(x)\mathsf{L}$
38. $x \le y \Rightarrow c(x) \le c(y)$
39. $c(\mathsf{L}) = \mathsf{L}$
40. $x \curlywedge \mathsf{L} \le c(x) \le x$
41. $c(x)y \le x\,c(y)$
42. $c(xy) = c(x)y = c(x)c(y)$
43. $c(n(x)y) = n(x)y$
44. $n(c(x)) = n(x)$

PROOF. See the Isabelle/HOL theory files. □

For example, property 2 is the above-mentioned Galois connection. The proof of $x \le y$ can be separated into two parts by property 3. Strict computations such as those in our model satisfy $\mathsf{L}x = \mathsf{L}$, but only the weaker property 10 holds for non-strict computations. Property 36 shows that n is concerned only with the executions contained in L, that is, the incomplete and infinite executions. Properties 41 and 42 show how c is propagated into and through sequential compositions.

6. Approximation Order

To obtain a suitable approximation order for our computation model, we follow the algebraic method developed in [14, 17, 18]. The approximation order $\sqsubseteq$ in n-algebras is

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

This modifies the order proposed in [18] by replacing $n(\mathsf{L})y$ with $c(y)$ as discussed in Section 5.2. Because $c(y) = y$ in our model, the approximation order there simplifies to

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

The part $x \le y + \mathsf{L}$ intuitively expresses that executions may be added and incomplete executions may be removed when the approximation is improved from x to y. Because of the closure requirements, an infinite execution will only be removed if it is contained in the upward closure of an incomplete execution that is removed. The intuition for $y \le x + n(x)\top$ is that where x has no incomplete executions, no executions may be added when x is improved to y, so x and y have the same executions.

For our computations the approximation order elaborates as follows. First, note that

$$\begin{array}{cll}
 & (N_2|P_2|Q_2|R_2) + (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) & \\
= & (N_2 \cup \mathsf{F}|P_2|Q_2 \cup \mathsf{F}_\infty|R_2) & \text{definition of } + \\
= & (\mathsf{F}|P_2|\mathsf{F}_\infty|R_2) & \text{progress requirements}
\end{array}$$

and

$$\begin{array}{cll}
 & (N_1|P_1|Q_1|R_1) + n(N_1|P_1|Q_1|R_1)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) & \\
= & (N_1|P_1|Q_1|R_1) + (\mathsf{O}|\mathsf{O}|Q_1|N_1)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) & \text{definition of } n \\
= & (N_1|P_1|Q_1|R_1) + (N_1\mathsf{F}|N_1\mathsf{F}|Q_1 \cup N_1\mathsf{F}_\infty|N_1\mathsf{F}) & \text{definition of } \cdot \\
= & (N_1|P_1|Q_1|R_1) + (N_1|N_1|Q_1|N_1) & \text{closure requirements} \\
= & (N_1|P_1 \cup N_1|Q_1|R_1 \cup N_1) & \text{definition of } +
\end{array}$$

Hence,

$$\begin{array}{cll}
 & (N_1|P_1|Q_1|R_1) \sqsubseteq (N_2|P_2|Q_2|R_2) & \\
\Leftrightarrow & (N_1|P_1|Q_1|R_1) \le (N_2|P_2|Q_2|R_2) + (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) \wedge & \text{definition of } \sqsubseteq \\
 & (N_2|P_2|Q_2|R_2) \le (N_1|P_1|Q_1|R_1) + n(N_1|P_1|Q_1|R_1)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) & \\
\Leftrightarrow & (N_1|P_1|Q_1|R_1) \le (\mathsf{F}|P_2|\mathsf{F}_\infty|R_2) \wedge & \text{calculations above} \\
 & (N_2|P_2|Q_2|R_2) \le (N_1|P_1 \cup N_1|Q_1|R_1 \cup N_1) & \\
\Leftrightarrow & N_1 \subseteq \mathsf{F} \wedge P_1 \subseteq P_2 \wedge Q_1 \subseteq \mathsf{F}_\infty \wedge R_1 \subseteq R_2 \wedge & \text{definition of } \le \\
 & N_2 \subseteq N_1 \wedge P_2 \subseteq P_1 \cup N_1 \wedge Q_2 \subseteq Q_1 \wedge R_2 \subseteq R_1 \cup N_1 & \\
\Leftrightarrow & N_2 \subseteq N_1 \wedge P_1 \subseteq P_2 \subseteq P_1 \cup N_1 \wedge Q_2 \subseteq Q_1 \wedge R_1 \subseteq R_2 \subseteq R_1 \cup N_1 & \text{progress requirements}
\end{array}$$

Thus finite and aborting executions can be added if they extend incomplete executions. Only incomplete and infinite executions can be removed.

Despite the modified axioms of n-algebras we retain the following properties of the approximation order from [18]. In particular, $\sqsubseteq$-least fixpoints can be represented in terms of $\leq$-least and $\leq$-greatest fixpoints. To state the properties we first specify greatest lower bounds and least fixpoints in the approximation order. Provided it exists, the $\sqsubseteq$-greatest lower bound of $x, y \in S$ in an n-algebra S is denoted by $x \sqcap y$:

$$x \sqcap y \sqsubseteq x \qquad x \sqcap y \sqsubseteq y \qquad z \sqsubseteq x \wedge z \sqsubseteq y \Rightarrow z \sqsubseteq x \sqcap y$$

Provided it exists, the $\sqsubseteq$-least fixpoint of a function f in an n-algebra is denoted by κf:

$$f(\kappa f) = \kappa f \qquad f(x) = x \Rightarrow \kappa f \sqsubseteq x$$

We abbreviate $\kappa(\lambda x.f(x))$ by $\kappa x.f(x)$.

Theorem 4. *Let S be an n-algebra.*

1. *The relation $\sqsubseteq$ is a partial order with least element L.*
2. *The operations $+$ and $\cdot$ and $\lambda x.x \curlywedge \mathsf{L}$ and $\lambda x.n(x)\mathsf{L}$ are $\sqsubseteq$-isotone.*
3. *If S is an itering, the operation $^\circ$ is $\sqsubseteq$-isotone.*
4. *If S is a Kleene algebra, the operation * is $\sqsubseteq$-isotone.*

Let $f : S \to S$ be $\leq$- and $\sqsubseteq$-isotone, and assume that μf and νf exist. Then the following are equivalent:

5. *κf exists.*
6. *κf and $\mu f \sqcap \nu f$ exist and $\kappa f = \mu f \sqcap \nu f$.*
7. *κf exists and $\kappa f = (\nu f \curlywedge \mathsf{L}) + \mu f$.*
8. *$c(\nu f) \leq (\nu f \curlywedge \mathsf{L}) + \mu f + n(\nu f)\top$.*
9. *$c(\nu f) \leq (\nu f \curlywedge \mathsf{L}) + \mu f + n((\nu f \curlywedge \mathsf{L}) + \mu f)\top$.*
10. *$(\nu f \curlywedge \mathsf{L}) + \mu f \sqsubseteq \nu f$.*
11. *$\mu f \sqcap \nu f$ exists and $\mu f \sqcap \nu f = (\nu f \curlywedge \mathsf{L}) + \mu f$.*
12. *$\mu f \sqcap \nu f$ exists and $\mu f \sqcap \nu f \leq \nu f$.*

Moreover the following are equivalent and imply conditions 5–12:

13. *κf exists and $\kappa f = n(\nu f)\mathsf{L} + \mu f$.*
14. *$c(\nu f) \leq \mu f + n(\nu f)\top$.*
15. *$n(\nu f)\mathsf{L} + \mu f \sqsubseteq \nu f$.*
16. *$\mu f \sqcap \nu f$ exists and $\mu f \sqcap \nu f = n(\nu f)\mathsf{L} + \mu f$.*

PROOF. See the Isabelle/HOL theory files. □

Conditions 8 and 14 of this result characterise the existence of κf in terms of μf and νf. Conditions 7 and 13 show how to obtain κf from μf and νf. This simplifies calculations as $\leq$ is less complex than $\sqsubseteq$.

7. Iteration

We now specialise the results about general recursions shown in Section 6 to iterations. Iterations are solved by $\sqsubseteq$-least fixpoints of the characteristic function $\lambda x.yx + z$. Using Theorem 4 the $\sqsubseteq$-least fixpoint of this function can be expressed in terms of its $\leq$-least fixpoint y^*z and its $\leq$-greatest fixpoint $y^\omega + y^*z$.

7.1. Finite Iteration

For this approach to work, our computations must form a Kleene algebra and an omega algebra. Conway's automata-based matrix construction [6] yields the Kleene star of a 2×2 matrix:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^* = \begin{pmatrix} e^* & a^*bf^* \\ d^*ce^* & f^* \end{pmatrix} \qquad \text{where} \qquad \begin{pmatrix} e \\ f \end{pmatrix} = \begin{pmatrix} a \cup bd^*c \\ d \cup ca^*b \end{pmatrix}$$

A proof similar to the one in [28] shows that this operation satisfies the Kleene algebra axioms. Our computations are 4×4 matrices, so the construction has to be applied three times. First,

$$\begin{pmatrix} \mathsf{I} & \mathsf{O} \\ Q & R \end{pmatrix}^* = \begin{pmatrix} \mathsf{I}^* & \mathsf{I}^*\mathsf{O}R^* \\ R^*Q\mathsf{I}^* & R^* \end{pmatrix} = \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ R^*Q & R^* \end{pmatrix} \qquad \text{using} \qquad \begin{pmatrix} e \\ f \end{pmatrix} = \begin{pmatrix} \mathsf{I} \cup \mathsf{O}R^*Q \\ R \cup Q\mathsf{I}^*\mathsf{O} \end{pmatrix} = \begin{pmatrix} \mathsf{I} \\ R \end{pmatrix}$$

The second application can be short-cut by instantiating the first matrix with $Q = \mathsf{O}$ and $R = \mathsf{I}$:

$$\begin{pmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} \end{pmatrix}^* = \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{I}^*\mathsf{O} & \mathsf{I}^* \end{pmatrix} = \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} \end{pmatrix}$$

Third,

$$\begin{aligned} \begin{pmatrix} \mathsf{I} & \mathsf{O} & \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} & \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{O} & \mathsf{I} & \mathsf{O} \\ N & P & Q & R \end{pmatrix}^* &= \begin{pmatrix} \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} \end{pmatrix}^* & \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} \end{pmatrix}^* \begin{pmatrix} \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{O} \end{pmatrix} \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ Q & R \end{pmatrix}^* \\ \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ Q & R \end{pmatrix}^* \begin{pmatrix} \mathsf{O} & \mathsf{O} \\ N & P \end{pmatrix} \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} \end{pmatrix}^* & \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ Q & R \end{pmatrix}^* \end{pmatrix} \\ &= \begin{pmatrix} \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} \end{pmatrix} & \begin{pmatrix} \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{O} \end{pmatrix} \\ \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ R^*Q & R^* \end{pmatrix} \begin{pmatrix} \mathsf{O} & \mathsf{O} \\ N & P \end{pmatrix} & \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ R^*Q & R^* \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \mathsf{I} & \mathsf{O} & \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} & \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{O} & \mathsf{I} & \mathsf{O} \\ R^*N & R^*P & R^*Q & R^* \end{pmatrix} \end{aligned}$$

using

$$\begin{pmatrix} e \\ f \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} \end{pmatrix} \cup \begin{pmatrix} \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{O} \end{pmatrix} \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ Q & R \end{pmatrix}^* \begin{pmatrix} \mathsf{O} & \mathsf{O} \\ N & P \end{pmatrix} \\ \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ Q & R \end{pmatrix} \cup \begin{pmatrix} \mathsf{O} & \mathsf{O} \\ N & P \end{pmatrix} \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} \end{pmatrix}^* \begin{pmatrix} \mathsf{O} & \mathsf{O} \\ \mathsf{O} & \mathsf{O} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ \mathsf{O} & \mathsf{I} \end{pmatrix} \\ \begin{pmatrix} \mathsf{I} & \mathsf{O} \\ Q & R \end{pmatrix} \end{pmatrix}$$

It follows that

$$(N|P|Q|R)^* = (R^*N|R^*P|R^*Q|R^*)$$

Note that $R \subseteq \mathsf{F}$ implies $\mathsf{I} \cup R\mathsf{F} \subseteq \mathsf{I} \cup \mathsf{F}^2 \subseteq \mathsf{F}$, whence $R^* = R^*\mathsf{I} \subseteq \mathsf{F}$ by Kleene star induction. Hence $N \subseteq \mathsf{F}$ implies $R^*N \subseteq \mathsf{F}^2 = \mathsf{F}$ and $P \subseteq \mathsf{F}$ implies $R^*P \subseteq \mathsf{F}^2 = \mathsf{F}$ and $Q \subseteq \mathsf{F}_\infty$ implies $R^*Q \subseteq \mathsf{F}\mathsf{F}_\infty = \mathsf{F}_\infty$. Thus * preserves the progress requirements. Furthermore, $R^*N\mathsf{F} \subseteq R^*N$ if $N\mathsf{F} \subseteq N$ and $R^*N\mathsf{F}_\infty \subseteq R^*Q$ if $N\mathsf{F}_\infty \subseteq Q$. Therefore * preserves the closure requirements.

The automata-based matrix construction for the omega operation does not work. Namely, the bottom right entry of $(N|P|Q|R)^\omega$ contains R^ω, but $\mathsf{I}^\omega = \mathsf{T} \nsubseteq \mathsf{F}$ despite $\mathsf{I} \subseteq \mathsf{F}$. This issue can be solved in the framework of typed omega algebras [13]. However, the omega operation has to be taken not among all relations of type $A_{\text{fin}} \leftrightarrow A_{\text{fin}}$, but only among the relations below F. To describe this, we generalise omega algebras as follows.

7.2. Capped Omega Algebras

Omega algebras axiomatise the $\leq$-greatest fixpoint of the function $\lambda x.yx + z$. Capped omega algebras deal with the function $\lambda x.(yx + z) \curlywedge v$.

A *capped omega algebra* $(S, +, \curlywedge, \cdot, ^*, ^\omega, 0, 1, \top)$ adds to a Kleene algebra $(S, +, \cdot, ^*, 0, 1)$ and a lattice-ordered semiring $(S, +, \curlywedge, \cdot, 0, 1, \top)$ a *binary* operation $^\omega$ with the following unfold and induction axioms for $v, x, y, z \in S$:

$$(\omega 1) \quad yy^\omega_v \curlywedge v = y^\omega_v \qquad\qquad (\omega 2) \quad x \le (yx + z) \curlywedge v \Rightarrow x \le y^\omega_v + y^* z$$

The element y^ω_v denotes the application of the binary $^\omega$ operation to its two arguments y and v. We use the subscript notation y^ω_v to point out that the new binary operation generalises the unary omega operation, which is obtained by setting $v = \top$. Note that y^ω_v is not necessarily the $\le$-greatest fixpoint of $\lambda x.yx \curlywedge v$; sufficient conditions for this to hold will be given in Theorem 6.

We will apply the capped omega operation to relations using $v = \mathsf{F}$. The general instance for relations is provided by the following result.

Theorem 5. *Let A be a set. The relations over A form a capped omega algebra $(2^{A\times A}, \cup, \cap, \cdot, ^*, ^\omega, \mathsf{O}, \mathsf{I}, \mathsf{T})$ using relational composition $\cdot$, reflexive transitive closure * and $R^\omega_P = \nu X.RX \cap P$.*

Proof. Axiom $(\omega 1)$ is the fixpoint property of $\nu X.RX \cap P$. For axiom $(\omega 2)$ let $X, R, Q, P : A \leftrightarrow A$ such that $X \subseteq h(X)$ where the function h is defined by $h(X) = (RX \cup Q) \cap P$. Since $2^{A\times A}$ is a complete lattice and h is $\subseteq$-isotone, the $\subseteq$-greatest postfixpoint of h exists and equals its $\subseteq$-greatest fixpoint νh. Therefore $X \subseteq \nu h$. Letting $g(X) = RX \cap P$, it remains to show

$$\nu h \subseteq \nu g \cup R^* Q$$

We prove this using ν-fusion [8]. The function $f(X) = X \cup R^* Q$ distributes over arbitrary infima in $2^{A\times A}$ and therefore possesses a lower adjoint. Moreover $(h \circ f)(X) \subseteq (f \circ g)(X)$ since

$$\begin{array}{rl} & (R(X \cup R^*Q) \cup Q) \cap P \\ = & (RX \cup RR^*Q \cup Q) \cap P \\ = & (RX \cup (RR^* \cup \mathsf{I})Q) \cap P \\ = & (RX \cup R^*Q) \cap P \\ = & (RX \cap P) \cup (R^*Q \cap P) \\ \subseteq & (RX \cap P) \cup R^*Q \end{array}$$

Hence ν-fusion yields $\nu h \subseteq f(\nu g)$ as required. □

The following result shows consequences of capped omega algebras. Note that axiom $(\omega 2)$ does not restrict the * operation to values below v; this could be achieved by generalising it to a ternary operation. However, we will apply the axioms with restricted values for y and z in which case additional properties hold as shown by the following result. It is proved in Isabelle/HOL, but we give a separate proof to cover the case of heterogeneous relations as discussed below. This is also why we distinguish between u and v.

Theorem 6. *Let S be a capped omega algebra and let $v, y \in S$. Then the following properties hold:*

1. *S is an omega algebra using $y^\omega = y^\omega_\top$.*
2. $y^\omega_v \le y^\omega_\top$
3. $y^\omega_v \le v$
4. $1^\omega_v = v$
5. $0^\omega_v = 0$

Let $s, t, u, v, x, y, z \in S$ such that $s, y \le u$ and $uu \le u$ and $z \le v$ and $uv \le v$. Then the following properties hold:

6. $y^\omega_v + y^* z = \nu x.(yx + z) \curlywedge v$
7. $yy^\omega_v = y^\omega_v$
8. $y^* y^\omega_v = y^\omega_v$
9. $y^* 0 \le y^\omega_v 0$

10. $x \le yx \curlywedge v \Rightarrow x \le y^\omega_v$
11. $y^\omega_v = \nu x.yx \curlywedge v$
12. $t \le y \Rightarrow t^\omega_v \le y^\omega_v$
13. $st \le ys \Rightarrow st^\omega_v \le y^\omega_v$
14. $s(ys)^\omega_v = (sy)^\omega_v$
15. $ys \le sy \Rightarrow (sy)^\omega_v \le s^\omega_v$
16. $(yy^*)^\omega_v = y^\omega_v$
17. $y^\omega_u z \le y^\omega_v$
18. $(y^\omega_u)^\omega_v \le y^\omega_v$

Proof.

1. The omega algebra axioms are obtained immediately by substituting $\top$ for v in axioms ($\omega1$) and ($\omega2$).
2. $y^\omega_v = yy^\omega_v \curlywedge v \le yy^\omega_v$ using ($\omega1$), whence $y^\omega_v \le y^\omega = y^\omega_\top$ by part 1.
3. $y^\omega_v = yy^\omega_v \curlywedge v \le v$ using ($\omega1$).
4. $v \le 1^\omega_v + 1^*0 = 1^\omega_v$ follows from $v \le (1v + 0) \curlywedge v$ by ($\omega2$). This shows the claim by part 3.
5. $0^\omega_v = 00^\omega_v \curlywedge v = 0 \curlywedge v = 0$ by ($\omega1$).
6. By ($\omega2$) it remains to show that $y^\omega_v + y^*z$ is a fixpoint of $\lambda x.(yx+z) \curlywedge v$. Note that $z+yv \le v+uv = v$, whence $y^*z \le v$ by Kleene star induction. Therefore
$$\begin{array}{rl} & (y(y^\omega_v + y^*z) + z) \curlywedge v \\ = & (yy^\omega_v + yy^*z + z) \curlywedge v \\ = & (yy^\omega_v + (yy^* + 1)z) \curlywedge v \\ = & (yy^\omega_v + y^*z) \curlywedge v \\ = & (yy^\omega_v \curlywedge v) + (y^*z \curlywedge v) \\ = & y^\omega_v + y^*z \end{array}$$
using ($\omega1$) in the last step.
7. $yy^\omega_v \le uv \le v$ by part 3. Hence $y^\omega_v = yy^\omega_v \curlywedge v = yy^\omega_v$ by ($\omega1$).
8. $y^\omega_v + yy^\omega_v = y^\omega_v$ by part 7, whence $y^*y^\omega_v \le y^\omega_v$ by Kleene star induction. The claim follows since $1 \le y^*$.
9. $y^*0 \le y^*y^\omega_v 0 = y^\omega_v 0$ by part 8.
10. $y^*0 \le y^\omega_v 0 \le y^\omega_v$ by part 9. Hence the claim follows by setting $z = 0$ in ($\omega2$).
11. The claim follows by ($\omega1$) and part 10.
12. $t^\omega_v = tt^\omega_v \curlywedge v \le yt^\omega_v \curlywedge v$ by ($\omega1$), whence the claim follows by part 10.
13. $st^\omega_v = s(tt^\omega_v \curlywedge v) \le stt^\omega_v \curlywedge sv \le yst^\omega_v \curlywedge uv \le yst^\omega_v \curlywedge v$ by ($\omega1$). Hence the claim follows by part 10.
14. $sy \le uu \le u$ and $ys \le uu \le u$ and $s(ys)^\omega_v \le uv \le v$ by part 3. Thus $s(ys)^\omega_v = s(ys)^\omega_v \curlywedge v = sys(ys)^\omega_v \curlywedge v$ by part 7, whence $s(ys)^\omega_v \le (sy)^\omega_v$ by part 11. This inequality holds for any $s, y \le u$, so by swapping s and y we also obtain $y(sy)^\omega_v \le (ys)^\omega_v$. Therefore $(sy)^\omega_v = sy(sy)^\omega_v \le s(ys)^\omega_v$ using part 7.
15. $sy \le uu \le u$, whence $(sy)^\omega_v = s(ys)^\omega_v \le s(sy)^\omega_v$ by parts 14 and 12. Thus $(sy)^\omega_v \le s(sy)^\omega_v \curlywedge v$ by part 3, from which the claim follows by part 10.
16. $y + uy \le u + uu = u$ implies $yy^* \le u$ by Kleene star induction. Moreover $yy^*yy^* = yyy^*y^* = yyy^*$ by properties of the Kleene star. Hence $(yy^*)^\omega_v = yy^*(yy^*)^\omega_v \le y^\omega_v$ by parts 7 and 13. Conversely, $y \le yy^*$ implies $y^\omega_v \le (yy^*)^\omega_v$ by part 12.
17. $y^\omega_u z = (yy^\omega_u \curlywedge u)z \le yy^\omega_u z \curlywedge uv \le yy^\omega_u z \curlywedge v$ by ($\omega1$), whence the claim follows by part 10.
18. $y^\omega_u y \le y^\omega_u = yy^\omega_u$ by parts 17 and 7, instantiating v with u. Moreover $y^\omega_u \le u$ by part 3, whence $(y^\omega_u)^\omega_v = (yy^\omega_u)^\omega_v \le y^\omega_v$ by parts 7 and 15. □

We also need to apply the capped omega operation using $v = \mathsf{F}_\infty$, which is a heterogeneous relation. To this end, a typed version of capped omega algebras can be introduced similarly to typed Kleene algebras, typed omega algebras or heterogeneous relation algebras [35, 30, 13]. We omit the technical machinery, but mention the key differences:

- Operations become polymorphic: there are different instances of $+$, $\curlywedge$, $\cdot$, *, $^\omega$, 0, 1 and $\top$ for different types.
- Operations become partial: $+$ and $\curlywedge$ just apply to elements of the same type, for $x \cdot y$ the target type of x must be the source type of y, and * only applies to homogeneous elements. The operation $^\omega$ takes a homogeneous element $y : a \to a$ and a heterogeneous element $v : a \to b$ and yields a heterogeneous element $y^\omega_v : a \to b$ of the same type as v. Hence the axioms ($\omega 1$) and ($\omega 2$) apply to $y : a \to a$ and $v, x, z : a \to b$.
- Theorem 5 and its proof generalise to a family of heterogeneous relations $2^{A \times B}$ indexed with sets A, B.
- Theorem 6 and its proof generalise so that $s, t, u, y : a \to a$ are homogeneous elements and $x, v, z : a \to b$ are heterogeneous elements. Applications can instantiate the result with $u = \mathsf{F}$ and $v = \mathsf{F}_\infty$ in addition to $u = v = \mathsf{F}$.

7.3. Infinite Iteration

We use the capped omega algebra structure of relations to define the unary omega operation for computations as follows:

$$(N|P|Q|R)^\omega = (R^\omega_\mathsf{F} \cup R^*N | R^\omega_\mathsf{F} \cup R^*P | R^\omega_{\mathsf{F}_\infty} \cup R^*Q | R^\omega_\mathsf{F})$$

Note that $R^\omega_\mathsf{F} \subseteq \mathsf{F}$ by Theorem 6.3, whence $N, P, R \subseteq \mathsf{F}$ also implies $R^\omega_\mathsf{F} \cup R^*N \subseteq \mathsf{F}$ and $R^\omega_\mathsf{F} \cup R^*P \subseteq \mathsf{F}$. Similarly $R^\omega_{\mathsf{F}_\infty} \subseteq \mathsf{F}_\infty$, whence $R^\omega_{\mathsf{F}_\infty} \cup R^*Q \subseteq \mathsf{F}_\infty$ if $R \subseteq \mathsf{F}$ and $Q \subseteq \mathsf{F}_\infty$. Thus $^\omega$ preserves the progress requirements. Furthermore $^\omega$ preserves the closure requirements since

$$\begin{aligned}(R^\omega_\mathsf{F} \cup R^*N)\mathsf{F} &= R^\omega_\mathsf{F}\mathsf{F} \cup R^*N\mathsf{F} \subseteq R^\omega_\mathsf{F} \cup R^*N \\ (R^\omega_\mathsf{F} \cup R^*N)\mathsf{F}_\infty &= R^\omega_\mathsf{F}\mathsf{F}_\infty \cup R^*N\mathsf{F}_\infty \subseteq R^\omega_{\mathsf{F}_\infty} \cup R^*Q\end{aligned}$$

using Theorem 6.17 and $N\mathsf{F} \subseteq \mathsf{F}$ and $N\mathsf{F}_\infty \subseteq Q$.

The following result shows that $^\omega$ and n satisfy the axioms of omega algebras and n-omega algebras given in Sections 2.3 and 5.2, respectively.

Theorem 7. *Let $S = \{(N|P|Q|R) \mid N, P, R \subseteq \mathsf{F} : A_\text{fin} \leftrightarrow A_\text{fin}$ and $Q \subseteq \mathsf{F}_\infty : A_\text{fin} \leftrightarrow A_\infty$ and $N\mathsf{F} \subseteq N$ and $N\mathsf{F}_\infty \subseteq Q\}$. Then $(S, +, \curlywedge, \cdot, n, ^*, ^\omega, 0, 1, \mathsf{L}, \top)$ is an n-omega algebra, where*

$$\begin{aligned}(N|P|Q|R)^* &= (R^*N|R^*P|R^*Q|R^*) \\ (N|P|Q|R)^\omega &= (R^\omega_\mathsf{F} \cup R^*N | R^\omega_\mathsf{F} \cup R^*P | R^\omega_{\mathsf{F}_\infty} \cup R^*Q | R^\omega_\mathsf{F})\end{aligned}$$

Proof. The Kleene star has been derived in Section 7.1. We first show the omega algebra axioms and then the n-omega algebra axioms.

- The omega unfold axiom is obtained by

$$\begin{aligned}& (N|P|Q|R)(N|P|Q|R)^\omega \\ = \;& (N|P|Q|R)(R^\omega_\mathsf{F} \cup R^*N | R^\omega_\mathsf{F} \cup R^*P | R^\omega_{\mathsf{F}_\infty} \cup R^*Q | R^\omega_\mathsf{F}) && \text{definition of } ^\omega \\ = \;& (N \cup R(R^\omega_\mathsf{F} \cup R^*N) | P \cup R(R^\omega_\mathsf{F} \cup R^*P) | Q \cup R(R^\omega_{\mathsf{F}_\infty} \cup R^*Q) | RR^\omega_\mathsf{F}) && \text{definition of } \cdot \\ = \;& (RR^\omega_\mathsf{F} \cup (RR^* \cup \mathsf{I})N | RR^\omega_\mathsf{F} \cup (RR^* \cup \mathsf{I})P | RR^\omega_{\mathsf{F}_\infty} \cup (RR^* \cup \mathsf{I})Q | RR^\omega_\mathsf{F}) && \cdot \text{ distributes over } \cup \\ = \;& (RR^\omega_\mathsf{F} \cup R^*N | RR^\omega_\mathsf{F} \cup R^*P | RR^\omega_{\mathsf{F}_\infty} \cup R^*Q | RR^\omega_\mathsf{F}) && \text{star unfold} \\ = \;& (R^\omega_\mathsf{F} \cup R^*N | R^\omega_\mathsf{F} \cup R^*P | R^\omega_{\mathsf{F}_\infty} \cup R^*Q | R^\omega_\mathsf{F}) && \text{Theorem 6.7} \\ = \;& (N|P|Q|R)^\omega && \text{definition of } ^\omega\end{aligned}$$

- For the omega induction axiom, assume

$$(N_1|P_1|Q_1|R_1) \leq (N_2|P_2|Q_2|R_2)(N_1|P_1|Q_1|R_1) + (N_3|P_3|Q_3|R_3)$$

This implies

$$\begin{array}{lll}
 & (N_1|P_1|Q_1|R_1) & \\
\le & (N_2|P_2|Q_2|R_2)(N_1|P_1|Q_1|R_1) + (N_3|P_3|Q_3|R_3) & \text{assumption} \\
= & (N_2 \cup R_2N_1|P_2 \cup R_2P_1|Q_2 \cup R_2Q_1|R_2R_1) + (N_3|P_3|Q_3|R_3) & \text{definition of } \cdot \\
= & (R_2N_1 \cup N_2 \cup N_3|R_2P_1 \cup P_2 \cup P_3|R_2Q_1 \cup Q_2 \cup Q_3|R_2R_1 \cup R_3) & \text{definition of } +
\end{array}$$

whence

$$\begin{array}{rl}
N_1 & \subseteq (R_2N_1 \cup N_2 \cup N_3) \cap \mathsf{F} \\
P_1 & \subseteq (R_2P_1 \cup P_2 \cup P_3) \cap \mathsf{F} \\
Q_1 & \subseteq (R_2Q_1 \cup Q_2 \cup Q_3) \cap \mathsf{F}_\infty \\
R_1 & \subseteq (R_2R_1 \cup R_3) \cap \mathsf{F}
\end{array}$$

by definition of $\le$ and the progress requirements. By (ω2) it follows that

$$\begin{array}{rl}
N_1 & \subseteq {R_2}^{\omega}_{\mathsf{F}} \cup R_2^*(N_2 \cup N_3) \\
P_1 & \subseteq {R_2}^{\omega}_{\mathsf{F}} \cup R_2^*(P_2 \cup P_3) \\
Q_1 & \subseteq {R_2}^{\omega}_{\mathsf{F}_\infty} \cup R_2^*(Q_2 \cup Q_3) \\
R_1 & \subseteq {R_2}^{\omega}_{\mathsf{F}} \cup R_2^*R_3
\end{array}$$

Therefore

$$\begin{array}{ll}
(N_1|P_1|Q_1|R_1) & \\
\le ({R_2}^{\omega}_{\mathsf{F}} \cup R_2^*(N_2 \cup N_3)|{R_2}^{\omega}_{\mathsf{F}} \cup R_2^*(P_2 \cup P_3)|{R_2}^{\omega}_{\mathsf{F}_\infty} \cup R_2^*(Q_2 \cup Q_3)|{R_2}^{\omega}_{\mathsf{F}} \cup R_2^*R_3) & \text{definition of } \le \\
= ({R_2}^{\omega}_{\mathsf{F}} \cup R_2^*N_2 \cup R_2^*N_3|{R_2}^{\omega}_{\mathsf{F}} \cup R_2^*P_2 \cup R_2^*P_3|{R_2}^{\omega}_{\mathsf{F}_\infty} \cup R_2^*Q_2 \cup R_2^*Q_3|{R_2}^{\omega}_{\mathsf{F}} \cup R_2^*R_3) & \cdot \text{ distributes over } \cup \\
= ({R_2}^{\omega}_{\mathsf{F}} \cup R_2^*N_2|{R_2}^{\omega}_{\mathsf{F}} \cup R_2^*P_2|{R_2}^{\omega}_{\mathsf{F}_\infty} \cup R_2^*Q_2|{R_2}^{\omega}_{\mathsf{F}}) + & \text{definition of } + \\
\quad (R_2^*N_2 \cup R_2^*N_3|R_2^*P_2 \cup R_2^*P_3|R_2^*Q_2 \cup R_2^*Q_3|R_2^*R_3) & \\
= (N_2|P_2|Q_2|R_2)^\omega + (R_2^*N_2|R_2^*P_2|R_2^*Q_2|R_2^*)(N_3|P_3|Q_3|R_3) & \text{definitions of } ^\omega,\ \cdot \\
= (N_2|P_2|Q_2|R_2)^\omega + (N_2|P_2|Q_2|R_2)^*(N_3|P_3|Q_3|R_3) & \text{definition of } ^*
\end{array}$$

(n11) Since c is the identity function in this model, the claim follows by

$$\begin{array}{lll}
 & (N|P|Q|R)^\omega & \\
= & (R^\omega_{\mathsf{F}} \cup R^*N|R^\omega_{\mathsf{F}} \cup R^*P|R^\omega_{\mathsf{F}_\infty} \cup R^*Q|R^\omega_{\mathsf{F}}) & \text{definition of } ^\omega \\
= & (\mathsf{O}|R^*P|\mathsf{O}|\mathsf{I})(R^\omega_{\mathsf{F}} \cup R^*N|R^\omega_{\mathsf{F}}|R^\omega_{\mathsf{F}_\infty} \cup R^*Q|R^\omega_{\mathsf{F}}) & \text{definition of } \cdot \\
\le & (R^*N|R^*P|R^*Q|R^*)(R^\omega_{\mathsf{F}} \cup R^*N|R^\omega_{\mathsf{F}} \cup R^*N|R^\omega_{\mathsf{F}_\infty} \cup R^*Q|R^\omega_{\mathsf{F}} \cup R^*N) & \mathsf{I} \subseteq R^*, \text{ definition of } \le \\
= & (N|P|Q|R)^*(\mathsf{O}|\mathsf{O}|R^\omega_{\mathsf{F}_\infty} \cup R^*Q|R^\omega_{\mathsf{F}} \cup R^*N)(\mathsf{I}|\mathsf{I}|\mathsf{O}|\mathsf{I}) & \text{definitions of } ^* \text{ and } \cdot \\
\le & (N|P|Q|R)^*(\mathsf{O}|\mathsf{O}|R^\omega_{\mathsf{F}_\infty} \cup R^*Q|R^\omega_{\mathsf{F}} \cup R^*N)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) & \mathsf{I} \subseteq \mathsf{F}, \text{ definition of } \le \\
= & (N|P|Q|R)^*n(R^\omega_{\mathsf{F}} \cup R^*N|R^\omega_{\mathsf{F}} \cup R^*P|R^\omega_{\mathsf{F}_\infty} \cup R^*Q|R^\omega_{\mathsf{F}})(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) & \text{definition of } n \\
= & (N|P|Q|R)^*n((N|P|Q|R)^\omega)(\mathsf{F}|\mathsf{F}|\mathsf{F}_\infty|\mathsf{F}) & \text{definition of } ^\omega
\end{array}$$

(n12) The claim follows since $(\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})(N|P|Q|R) = (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})$. □

The following result shows that despite the modified axioms of the operation n we retain many properties from [18] in n-omega algebras. This includes further representations of iteration in terms of the Kleene star and omega operations.

Theorem 8. *Let S be an n-omega algebra and let $x, y, z \in S$. Then the following properties hold:*

1. $x\mathsf{L} = x\mathsf{L}x\mathsf{L}$
2. $\mathsf{L}x^* = \mathsf{L}$
3. $(x\mathsf{L})^* = 1 + x\mathsf{L}$
4. $(x\mathsf{L})^\omega = x\mathsf{L}$
5. $(x\mathsf{L} + y)^* = y^* + y^*x\mathsf{L}$
6. $(x\mathsf{L} + y)^\omega = y^\omega + y^*x\mathsf{L}$
7. $n(x) \le n(x^\omega)$
8. $x^* + n(x^\omega)\mathsf{L} = x^* + x\,n(x^\omega)\mathsf{L} = x^* + x^*n(x^\omega)\mathsf{L}$
9. $x^*0 + n(x^\omega)\mathsf{L} = x^*0 + x^*n(x^\omega)\mathsf{L}$
10. $yx^*0 + n(yx^\omega)\mathsf{L} = yx^*0 + y\,n(x^\omega)\mathsf{L}$

11. $yx^* + n(yx^\omega)\mathsf{L} = yx^* + y\,n(x^\omega)\mathsf{L}$
12. $n(y^\omega + y^*z) = n(y^\omega) + n(y^*z)$
13. $c(x^\omega) = c(x)^\omega$
14. $c(x^\omega) \le x^*0 + n(x^\omega)\top$

Let $f : S \to S$ be given by $f(x) = yx + z$.

15. *The $\sqsubseteq$-least fixpoint of f is $\kappa f = (y^\omega \curlywedge \mathsf{L}) + y^*z = n(y^\omega)\mathsf{L} + y^*z$.*
16. *The operations $^\omega$ and $\lambda y.(\kappa x.yx + z)$ and $\lambda z.(\kappa x.yx + z)$ are $\sqsubseteq$-isotone.*
17. *S is an extended binary itering using $x \star y = (x^\omega \curlywedge \mathsf{L}) + x^*y = n(x^\omega)\mathsf{L} + x^*y$.*

Proof. See the Isabelle/HOL theory files. ☐

For example, property 14 shows how to split the executions of x^ω into the incomplete executions in $n(x^\omega)\top$ and the remaining executions in x^*0. Property 15 gives, in particular, the semantics of loops. The following result elaborates the binary itering operation of property 17 for our computations.

Theorem 9. *Let $S = \{(N|P|Q|R) \mid N, P, R \subseteq \mathsf{F} : A_{\text{fin}} \leftrightarrow A_{\text{fin}}$ and $Q \subseteq \mathsf{F}_\infty : A_{\text{fin}} \leftrightarrow A_\infty$ and $N\mathsf{F} \subseteq N$ and $N\mathsf{F}_\infty \subseteq Q\}$. Then $(S, +, \cdot, \star, 0, 1)$ is an extended binary itering and $(S, +, \cdot, ^\circ, 0, 1)$ is an itering, where*

$$\begin{aligned}(N_1|P_1|Q_1|R_1) \star (N_2|P_2|Q_2|R_2) &= ({R_1}_{\mathsf{F}}^{\omega} \cup R_1^*(N_1 \cup N_2)|R_1^*(P_1 \cup P_2)|{R_1}_{\mathsf{F}_\infty}^{\omega} \cup R_1^*(Q_1 \cup Q_2)|R_1^*R_2)\\ (N|P|Q|R)^\circ &= (R_{\mathsf{F}}^{\omega} \cup R^*N|R^*P|R_{\mathsf{F}_\infty}^{\omega} \cup R^*Q|R^*)\end{aligned}$$

Proof. The extended binary itering instance follows by

$$\begin{array}{cll} & (N_1|P_1|Q_1|R_1) \star (N_2|P_2|Q_2|R_2) & \\ = & ((N_1|P_1|Q_1|R_1)^\omega \curlywedge (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})) + (N_1|P_1|Q_1|R_1)^*(N_2|P_2|Q_2|R_2) & \text{Theorem 8.17} \\ = & (({R_1}_{\mathsf{F}}^{\omega} \cup R_1^*N_1|{R_1}_{\mathsf{F}}^{\omega} \cup R_1^*P_1|{R_1}_{\mathsf{F}_\infty}^{\omega} \cup R_1^*Q_1|{R_1}_{\mathsf{F}}^{\omega}) \curlywedge (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})) + & \text{definitions of } ^\omega,\ ^* \\ & (R_1^*N_1|R_1^*P_1|R_1^*Q_1|R_1^*)(N_2|P_2|Q_2|R_2) & \\ = & (({R_1}_{\mathsf{F}}^{\omega} \cup R_1^*N_1) \cap \mathsf{F}|\mathsf{O}|({R_1}_{\mathsf{F}_\infty}^{\omega} \cup R_1^*Q_1) \cap \mathsf{F}_\infty|\mathsf{O}) + & \text{definitions of } \curlywedge,\ \cdot \\ & (R_1^*N_1 \cup R_1^*N_2|R_1^*P_1 \cup R_1^*P_2|R_1^*Q_1 \cup R_1^*Q_2|R_1^*R_2) & \\ = & ({R_1}_{\mathsf{F}}^{\omega} \cup R_1^*N_1|\mathsf{O}|{R_1}_{\mathsf{F}_\infty}^{\omega} \cup R_1^*Q_1|\mathsf{O}) + & \text{progress requirements} \\ & (R_1^*N_1 \cup R_1^*N_2|R_1^*P_1 \cup R_1^*P_2|R_1^*Q_1 \cup R_1^*Q_2|R_1^*R_2) & \\ = & ({R_1}_{\mathsf{F}}^{\omega} \cup R_1^*(N_1 \cup N_2)|R_1^*(P_1 \cup P_2)|{R_1}_{\mathsf{F}_\infty}^{\omega} \cup R_1^*(Q_1 \cup Q_2)|R_1^*R_2) & \text{definition of } + \end{array}$$

Our computations satisfy the property $(x \star y)z = x \star (yz)$ for each $x, y, z \in S$:

$$\begin{array}{ll} ((N_1|P_1|Q_1|R_1) \star (N_2|P_2|Q_2|R_2))(N_3|P_3|Q_3|R_3) & \\ = ({R_1}_{\mathsf{F}}^{\omega} \cup R_1^*(N_1 \cup N_2)|R_1^*(P_1 \cup P_2)|{R_1}_{\mathsf{F}_\infty}^{\omega} \cup R_1^*(Q_1 \cup Q_2)|R_1^*R_2)(N_3|P_3|Q_3|R_3) & \text{definition of } \star \\ = ({R_1}_{\mathsf{F}}^{\omega} \cup R_1^*(N_1 \cup N_2 \cup R_2N_3)|R_1^*(P_1 \cup P_2 \cup R_2P_3)|{R_1}_{\mathsf{F}_\infty}^{\omega} \cup R_1^*(Q_1 \cup Q_2 \cup R_2Q_3)|R_1^*R_2R_3) & \text{definition of } \cdot \\ = (N_1|P_1|Q_1|R_1) \star (N_2 \cup R_2N_3|P_2 \cup R_2P_3|Q_2 \cup R_2Q_3|R_2R_3) & \text{definition of } \star \\ = (N_1|P_1|Q_1|R_1) \star ((N_2|P_2|Q_2|R_2)(N_3|P_3|Q_3|R_3)) & \text{definition of } \cdot \end{array}$$

Hence the itering instance $x^\circ = x \star 1$ follows, which elaborates as stated above. ☐

Therefore all consequences of iterings and binary iterings shown in [14, 15] hold for our computations. They include separation theorems generalised from omega algebras and Back's atomicity refinement theorem [1, 5].

In the following result we elaborate several while-loops in different computation models with progress. See Section 3 for a discussion of the computation models and how they represent progress. The semantics of while-loops is given by $\mathsf{while}\ p\ \mathsf{do}\ x = (px)^\circ\overline{p}$ where p is a test and $\overline{p}$ its complement [29, 14].

Theorem 10. *Consider computations with abstract time and one program variable ranging over natural numbers, that is, let $D = \mathbb{N}$ and $A_{\text{fin}} = D \times \mathbb{N}$ and $A_\infty = \{\infty\}$ and $\mathsf{F} = \{((v,t),(v',t')) \mid t \le t'\}$ and $\mathsf{F}_\infty = \mathsf{T}$.*

1. *Let* $\mathsf{true} = \mathsf{skip} = 1$. *Then* $\mathsf{while\ true\ do\ skip} = (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) = \mathsf{L}$.

2. *Let* pass $= (\mathsf{O}|\mathsf{O}|\mathsf{O}|R_2)$ *where* $R_2 = \{((v,t),(v,t+1)) \mid v \in D \wedge t \in \mathbb{N}\}$ *lets one unit of time pass without changing the value of the variable. Then* while true do pass $= (\mathsf{O}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O})$*.*

Consider computations with traces and one program variable ranging over natural numbers, that is, let $D = \mathbb{N}$ *and* $A_{\text{fin}} = D^+$ *and* $A_\infty = D^\omega$ *and* $\mathsf{F} = \preceq$ *and* $\mathsf{F}_\infty = \preceq$ *using the prefix relation* $\preceq$*.*

3. *Let* inc $= (\mathsf{O}|\mathsf{O}|\mathsf{O}|R_3)$ *where* $R_3 = \{(tr, tr') \mid tr' = tr +\!\!+ [last(tr)+1]\}$ *appends a number to the sequence* tr*, namely the increment of the last element of* tr*. Then* while true do inc $= (\mathsf{O}|\mathsf{O}|Q_3|\mathsf{O})$ *where* $Q_3 = \{(tr, tr') \mid tr' = tr +\!\!+ from(last(tr))\}$ *and* $from(n)$ *is the infinite sequence* $n+1$, $n+2$, $n+3$, $\ldots$

PROOF.

1. $(1 \cdot 1)^\circ \overline{1} = 1^\circ 0 = (\mathsf{O}|\mathsf{O}|\mathsf{O}|\mathsf{I})^\circ 0 = (\mathsf{I}_\mathsf{F}^\omega \cup \mathsf{I}^*\mathsf{O}|\mathsf{I}^*\mathsf{O}|\mathsf{I}_{\mathsf{F}_\infty}^\omega \cup \mathsf{I}^*\mathsf{O}|\mathsf{I}^*)(\mathsf{O}|\mathsf{O}|\mathsf{O}|\mathsf{O}) = (\mathsf{I}_\mathsf{F}^\omega|\mathsf{O}|\mathsf{I}_{\mathsf{F}_\infty}^\omega|\mathsf{O}) = (\mathsf{F}|\mathsf{O}|\mathsf{F}_\infty|\mathsf{O}) = \mathsf{L}$ by Theorems 9 and 6.4 using that $\mathsf{I} \subseteq \mathsf{F}$.
2. Similarly, while true do pass $= (\mathsf{O}|\mathsf{O}|\mathsf{O}|R_2)^\circ 0 = (R_{2\mathsf{F}}^{\ \omega}|\mathsf{O}|R_{2\mathsf{F}_\infty}^{\ \omega}|\mathsf{O})$. Observe that $R_2 \subseteq \mathsf{F}$. First, we show $R_{2\mathsf{F}}^{\ \omega} = \mathsf{O}$. Using $f(X) = R_2 X \cap \mathsf{F}$ we obtain $R_{2\mathsf{F}}^{\ \omega} = \nu f$ by Theorem 6.11. Moreover the greatest fixpoint νf of the $\subseteq$-isotone function f satisfies $\nu f \subseteq \bigcap_{n\in\mathbb{N}} f^n(\mathsf{T})$, which is proved by induction. Hence it remains to show $\bigcap_{n\in\mathbb{N}} f^n(\mathsf{T}) = \mathsf{O}$, which follows by noting that $t' \in \mathbb{N}$ grows without bounds in the sequence
$$\begin{aligned}
f^0(\mathsf{T}) &= \mathsf{T} \\
f^1(\mathsf{T}) &= R_2\mathsf{T} \cap \mathsf{F} = \mathsf{T} \cap \mathsf{F} = \mathsf{F} \\
f^2(\mathsf{T}) &= R_2\mathsf{F} \cap \mathsf{F} = R_2\mathsf{F} = \{((v,t),(v',t')) \mid t+1 \le t'\} \\
f^3(\mathsf{T}) &= R_2 f^2(\mathsf{T}) \cap \mathsf{F} = R_2 f^2(\mathsf{T}) = \{((v,t),(v',t')) \mid t+2 \le t'\} \\
&\vdots \\
f^n(\mathsf{T}) &= \{((v,t),(v',t')) \mid t+n-1 \le t'\}
\end{aligned}$$
Second, $R_2\mathsf{T} \cap \mathsf{F}_\infty = \mathsf{T} \cap \mathsf{F}_\infty = \mathsf{F}_\infty = \mathsf{T}$ implies $\mathsf{T} \subseteq R_{2\mathsf{F}_\infty}^{\ \omega}$ by Theorem 6.11, whence $R_{2\mathsf{F}_\infty}^{\ \omega} = \mathsf{F}_\infty$.
3. Similarly, while true do inc $= (\mathsf{O}|\mathsf{O}|\mathsf{O}|R_3)^\circ 0 = (R_{3\mathsf{F}}^{\ \omega}|\mathsf{O}|R_{3\mathsf{F}_\infty}^{\ \omega}|\mathsf{O})$. Observe that $R_3 \subseteq \preceq = \mathsf{F}$ and let $g(X) = R_3 X \cap \preceq$. Reasoning as in part 2 we obtain $R_{3\mathsf{F}}^{\ \omega} = \mathsf{O}$ by noting that $tr' \in D^+$ grows without bounds in the sequence
$$\begin{aligned}
g^0(\mathsf{T}) &= \mathsf{T} \\
g^1(\mathsf{T}) &= R_3\mathsf{T} \cap \preceq\, = \mathsf{T} \cap \preceq\, = \,\preceq \\
g^2(\mathsf{T}) &= R_3 \preceq \cap \preceq\, = R_3 \preceq\, = \{(tr,tr') \mid tr +\!\!+ [last(tr)+1] \preceq tr'\} \\
g^3(\mathsf{T}) &= R_3 g^2(\mathsf{T}) \cap \preceq\, = R_3 g^2(\mathsf{T}) = \{(tr,tr') \mid tr +\!\!+ [last(tr)+1, last(tr)+2] \preceq tr'\} \\
&\vdots \\
g^n(\mathsf{T}) &= \{(tr,tr') \mid tr +\!\!+ [last(tr)+1, last(tr)+2, \ldots, last(tr)+n-1] \preceq tr'\}
\end{aligned}$$
This converges to O because the calculation involves relations of type $A_{\text{fin}} \leftrightarrow A_{\text{fin}}$, that is, finite traces. On the other hand, for the infinite executions we calculate in the relations of type $A_{\text{fin}} \leftrightarrow A_\infty$, whence $tr' \in D^\omega$ is an infinite trace. Using $h(X) = R_3 X \cap \preceq$ where $X, \preceq\, : A_{\text{fin}} \leftrightarrow A_\infty$, the sequence converges to $\bigcap_{n\in\mathbb{N}} h^n(\mathsf{T}) = \{(tr,tr') \mid tr' = tr +\!\!+ from(last(tr))\} = Q_3$. Moreover, Q_3 is a fixpoint of h and therefore $R_{3\mathsf{F}_\infty}^{\ \omega} = Q_3$ by Theorem 6.11. □

8. Conclusion

In this paper we have unified the notion of progress for various computation models using algebraic methods. As a consequence, we inherit many results that were previously proved for the instantiated algebraic structures. Our new general model specialises to computations with or without time or traces. The obtained approximation order is suitable for models in which time is represented by a variable or by the length of a trace. In future work we will use the unified model to relate different fixpoints for the semantics of recursion in models with and without a notion of time, and to investigate Zeno effects [25].

Acknowledgements

I thank an anonymous referee of [18] for suggesting the use of heterogeneous relations in matrix models, and Roland Backhouse for suggesting the use of residuals, which were used in the first version of this paper. I thank the anonymous referees for their helpful comments on the first version, which led to the present, substantially revised paper.

References

[1] R. J. R. Back and J. von Wright. Reasoning algebraically about loops. *Acta Informatica*, 36(4):295–334, 1999.

[2] R. Berghammer and H. Zierer. Relational algebraic semantics of deterministic and nondeterministic programs. *Theoretical Computer Science*, 43:123–147, 1986.

[3] J. C. Blanchette, S. Böhme, and L. C. Paulson. Extending Sledgehammer with SMT solvers. In N. Bjørner and V. Sofronie-Stokkermans, editors, *Automated Deduction: CADE-23*, volume 6803 of *LNCS*, pages 116–130. Springer, 2011.

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

[5] E. Cohen. Separation and reduction. In R. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction*, volume 1837 of *LNCS*, pages 45–59. Springer, 2000.

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

[7] J. Cranch, M. R. Laurence, and G. Struth. Completeness results for omega-regular algebras. *Journal of Logical and Algebraic Methods in Programming*, 84(3):402–425, 2015.

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

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

[10] S. Dunne. Recasting Hoare and He's Unifying Theory of Programs in the context of general correctness. In A. Butterfield, G. Strong, and C. Pahl, editors, *5th Irish Workshop on Formal Methods*, Electronic Workshops in Computing. The British Computer Society, 2001.

[11] S. Dunne. Conscriptions: A new relational model for sequential computations. In B. Wolff, M.-C. Gaudel, and A. Feliachi, editors, *Unifying Theories of Programming, Fourth International Symposium, UTP 2012*, volume 7681 of *LNCS*, pages 144–163. Springer, 2013.

[12] W. Guttmann. Unifying recursion in partial, total and general correctness. In S. Qin, editor, *Unifying Theories of Programming, Third International Symposium, UTP 2010*, volume 6445 of *LNCS*, pages 207–225. Springer, 2010.

[13] W. Guttmann. Towards a typed omega algebra. In H. de Swart, editor, *Relational and Algebraic Methods in Computer Science*, volume 6663 of *LNCS*, pages 196–211. Springer, 2011.

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

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

[16] W. Guttmann. Extended designs algebraically. *Science of Computer Programming*, 78(11):2064–2085, 2013.

[17] W. Guttmann. Extended conscriptions algebraically. In P. Höfner, P. Jipsen, W. Kahl, and M. E. Müller, editors, *Relational and Algebraic Methods in Computer Science*, volume 8428 of *LNCS*, pages 139–156. Springer, 2014.

[18] W. Guttmann. Infinite executions of lazy and strict computations. *Journal of Logical and Algebraic Methods in Programming*, 84(3):326–340, 2015.

[19] W. Guttmann and B. Möller. Normal design algebra. *Journal of Logic and Algebraic Programming*, 79(2):144–173, 2010.

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

[21] I. J. Hayes, S. E. Dunne, and L. A. Meinicke. Linking Unifying Theories of Program refinement. *Science of Computer Programming*, 78(11):2086–2107, 2013.

[22] E. Hehner. Retrospective and prospective for Unifying Theories of Programming. In S. Dunne and W. Stoddart, editors, *Unifying Theories of Programming*, volume 4010 of *LNCS*, pages 1–17. Springer, 2006.

[23] E. C. R. Hehner. Termination is timing. In J. L. A. van de Snepscheut, editor, *Mathematics of Program Construction*, volume 375 of *LNCS*, pages 36–47. Springer, 1989.

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

[25] P. Höfner and B. Möller. An algebra of hybrid systems. *Journal of Logic and Algebraic Programming*, 78(2):74–97, 2009.

[26] P. Höfner and G. Struth. Automated reasoning in Kleene algebra. In F. Pfenning, editor, *Automated Deduction: CADE-21*, volume 4603 of *LNCS*, pages 279–294. Springer, 2007.

[27] P. Höfner, G. Struth, and G. Sutcliffe. Automated verification of refinement laws. *Annals of Mathematics and Artificial Intelligence*, 55(1–2):35–62, 2009.

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

[29] D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems*, 19(3):427–443, 1997.

[30] D. Kozen. Typed Kleene algebra. Technical Report TR98-1669, Cornell University, March 1998.

[31] B. Möller. The linear algebra of UTP. In T. Uustalu, editor, *Mathematics of Program Construction*, volume 4014 of *LNCS*, pages 338–358. Springer, 2006.

[32] B. Möller. Kleene getting lazy. *Science of Computer Programming*, 65(2):195–214, 2007.

[33] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[34] L. C. Paulson and J. C. Blanchette. Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In G. Sutcliffe, E. Ternovska, and S. Schulz, editors, *Proceedings of the 8th International Workshop on the Implementation of Logics*, pages 3–13, 2010.

[35] 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, Wien, 1997.

[36] G. Schmidt and T. Ströhlein. *Relationen und Graphen.* Springer, 1989.

[37] K. Solin and J. von Wright. Enabledness and termination in refinement algebra. *Science of Computer Programming*, 74(8):654–668, 2009.

[38] J. von Wright. Towards a refinement algebra. *Science of Computer Programming*, 51(1–2):23–45, 2004.