Convolution as a Unifying Concept: Applications in Separation Logic, Interval Calculi and Concurrency

BRIJESH DONGOL, Brunel University London, United Kingdom
IAN J. HAYES, The University of Queensland, Australia
GEORG STRUTH, University of Sheffield, United Kingdom

A notion of convolution is presented in the context of formal power series together with lifting constructions characterising algebras of such series, which usually are quantales. A number of examples underpin the universality of these constructions, the most prominent ones being separation logics, where convolution is separating conjunction in an assertion quantale; interval logics, where convolution is the chop operation; and stream interval functions, where convolution is proposed for analysing the trajectories of dynamical or real-time systems. A Hoare logic can be constructed in a generic fashion on the power series quantale, which applies to each of these examples. In many cases, commutative notions of convolution have natural interpretations as concurrency operations.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal methods*; D.3.1 [**Programming Languages**]: Formal Definitions and Theory—*Semantics*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*Parallelism and concurrency*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*Logics of programs*; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*Algebraic approaches to semantics*

General Terms: Algorithms, Languages, Theory, Verification

Additional Key Words and Phrases: Concurrency, convolution, formal power series, formal semantics, Hoare logics, interval logics, quantales, semigroups, separation logics, systems verification

ACM Reference Format:
Brijesh Dongol, Ian J. Hayes and Georg Struth. 2014. Convolution, Separation and Concurrency. *ACM Trans. Comput. Logic* V, N, Article A (January YYYY), 25 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Algebraic approaches are fundamental to mathematics and computing. Universal constructions, such as products, quotients or adjunctions, can, for instance, be presented and investigated in algebra or category theory in simple generic ways. We investigate the notion of *convolution* or *Cauchy product* from formal language theory [Droste et al. 2009; Berstel and Reutenauer 1984] as such a universal algebraic construct. It supports a generic development of objects and calculi interesting to mathematics and computing, and provides a unified structural view on various computational models.

The research reported here was supported in part by Australian Research Council Grant DP130102901 and EPSRC Grant EP/J003727/1.

Authors' addresses: B. Dongol, Department of Computer Science, Brunel University, UK; G. Struth, Department of Computer Science, University of Sheffield, Sheffield, UK; I. J. Hayes, School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, 4072, Australia.

DOI:http://dx.doi.org/10.1145/0000000.0000000

This is interesting both conceptually and for designing modular mathematical components in theorem proving environments.

The operational content of convolution is simple: an entity is decomposed in all possible ways into two parts, two functions are applied separately or in parallel to each of these pairs, their outputs are composed, and all possible compositions are summed.

Hence suppose f and g are functions from a suitable algebra S with composition $\circ$ into algebra Q with composition $\odot$ and summation Σ. Then, for any $x \in S$, the convolution of f and g is given by

$$(f \otimes g)\, x \; = \sum_{x = y \circ z} f\, y \odot g\, z.$$

This language-theoretic form of convolution differs slightly from convolution in other fields of mathematics, and it is an algebraic analogue of the Day convolution of functors from a monoidal category into the category Set [Day 1970]. In formal language theory, total functions of type $S \to Q$ are called (*formal* or *rational*) *power series*. They map elements of the free monoid $S = X^*$ with alphabet X into a semiring $(Q, +, \odot, 0)$. Because finite words can only be split into finitely many prefix/suffix pairs, the sum in convolution is finite, and hence well defined. A standard example of Q is the boolean semiring, where $+$ is disjunction and $\odot$ is conjunction. Power series then become characteristic functions; they indicate whether or not a word is in a language. Convolution in this case specialises to the language product. More generally, Q can model probabilities or weights associated to words; a Handbook is devoted to the subject [Droste et al. 2009].

We complement this body of work by generalising the type of power series, rebalancing the assumptions on source algebras S and target algebras Q, while avoiding the machinery of enriched categories [Kelly 1982]. This shifts the focus to other applications. Among those are separation logic (cf. [Ishtiaq and O'Hearn 2001; Reynolds 2002]), where convolution becomes separating conjunction, interval temporal logics [Moszkowski 2000], where convolution becomes chop, and combinations of these with new notions of spatial, temporal and concurrent composition. The power series approach captures the algebra of convolutions and allows derivation of compositional inference systems for each of these applications in a uniform manner. Compared to approaches based on enriched categories it is conceptually simpler and more suitable for mechanised reasoning [Dongol et al. 2015].

Our main contributions are as follows:

— For power series from partial semigroups into quantales, we present a generic lifting construction to power series quantales with convolution as multiplication.

— For non-commutative convolution we construct, as examples, the quantales of languages, binary relations, matrices, sets of traces or interval functions as instances.

— For commutative convolution we construct assertion quantales for separation logic from resource monoids, multisets, heaplets or finite vectors.

— We generalise the lifting construction to partial quantales, to bi-semigroups and bi-quantales, as well as two-dimensional power series. Based on this we outline a new algebraic approach to the interval-based analysis of dynamic and real-time systems, where convolution yields spatial, temporal and concurrency operators as an example.

— We explain how predicate transformer algebras and Hoare calculi arise in the power series approach in generic fashion and discuss some extensions and some ramifications of deriving concurrency rules in this setting.

Our lifting constructions are simple and generic: after setting up a suitable partial semigroup S (for instance words under concatenation, closed intervals under concatenation, resources under aggregation operations) the space of functions from S into a quantale Q itself forms a quantale with convolution as multiplication. When the target quantale Q is the two-element boolean quantale $\mathbb{B}$, power series are predicates and multiplication in the booleans is conjunction $\sqcap$. Convolution then becomes

$$(f \otimes g)\, x \;=\; \sum_{x = y \circ z} f\, y \sqcap g\, z$$

where $\sum$ denotes join or existential quantification in $\mathbb{B}$. If S is a set of resources and $\circ$ a commutative operation of resource aggregation, then convolution is separating conjunction. If S is a set of closed intervals and $\circ$ concatenates adjoining intervals, then convolution is chop. Our lifting result then guarantees that the predicates over S form an assertion quantale — of separation logic, interval logics and so forth. But it covers models beyond the booleans as well, such as probabilistic or weighted predicates or linear transformations on vectors. In general, convolution admits spatial or concurrent interpretations whenever both compositions $\circ$ and $\odot$ are commutative.

The remainder of this article is organised as follows. Section 2 recalls the basic algebraic structures needed. Section 3 introduces our approach to power series with partial semigroups as source algebras and quantales as target algebras; it also proves our basic lifting result. Section 4 discusses the case of power series into the boolean quantale, when convolution becomes a possibly non-commutative notion of separating conjunction. Sections 5 and 6 present non-commutative and commutative instances of our lifting lemma, respectively; Section 5 discussing, among others, the chop operation over intervals and Section 6 focusing on variants of separating conjunction. Section 7 presents a lifting result for power series into partial quantales with an example, while Section 8 considers formal power series that are partial functions, with applications to sparse and non-square matrices. Section 9 generalises the lifting result to bi-semigroups and bi-quantales and presents two examples. Section 10 generalises the result to power series from two semigroups into a bi-quantale; and presents in particular the quantale of stream interval functions, which is based on this generalisation. Section 11 further generalises the approach to applications with finite and infinite behaviours. Section 12 shows that the interchange laws of concurrent Kleene algebras fail in general power series quantales. Section 13 discusses extensions and applications of the approach, including how predicate transformer algebras and generic Hoare logics with a frame rule can be obtained from power series quantales. Section 14 concludes the article.

Additional material, including more detailed proofs and examples, can be found in an extended report [Dongol et al. 2014b].

2. ALGEBRAIC PRELIMINARIES

This section briefly recalls the most important algebraic structures used in this article: partial semigroups and monoids, their commutative variants, semirings and dioids as well as quantales. We also consider such structures with two operations of composition or multiplication: bi-semigroups, bi-monoids, bi-semirings and bi-quantales.

A *partial semigroup* is a structure $(S, D, \cdot)$ such that S is a set, $D \subseteq S \times S$ is the domain over which the binary operation $\cdot$ is defined, and it is an operation of type $D \to S$ that satisfies the usual associativity law in the sense that if either side is defined then so is the other side and both are equal [Bergelson et al. 1994]. A partial semigroup can be embedded into a (total) semigroup $(S, \odot, \bot)$ with the adjoined element $\bot \notin S$ denoting undefined, where $x \odot y = x \cdot y$ if $(x, y) \in D$, and $x \odot y = \bot$ otherwise.

A *partial monoid* is a partial semigroup with multiplicative unit 1. A *generalised partial monoid* is a structure $(S, D, I, \cdot)$ such that $(S, D, \cdot)$ is a partial semigroup, $I \subseteq S$

is a set of generalised units such that for each $x \in S$ there exist $e, e' \in I$ such that $e \cdot x = x = x \cdot e'$. A very similar axiomatisation has been used by Jónsson and Tarski [1952] for generalised Brand groupoids. Alternatively, generalised partial monoids can be identified with small categories whereas partial semigroups have been called semigroupoids in category theory.

We often write $(S, \cdot)$ for partial semigroups and $(S, \cdot, 1)$ for partial monoids, leaving D and $\perp$ implicit. A (partial) semigroup S is *commutative* if $x \cdot y = y \cdot x$ for all $x, y \in S$. Henceforth, we write $\cdot$ for a general multiplication and $*$ for a commutative one.

An important property of semigroups is *opposition duality*. For every partial semigroup $(S, \cdot)$, the structure $(S, \odot)$ with $x \odot y = y \cdot x$ for all $x, y \in S$ forms a partial semigroup; the *opposite* of S. Similarly, the opposite of a partial monoid is a partial monoid.

The definitions of semigroups and monoids generalise to n operations, but we are mainly interested in the case $n = 2$. A *partial bi-semigroup* is a structure $(S, \circ, \bullet)$ such that $(S, \circ)$ and $(S, \bullet)$ are partial semigroups. *Partial bi-monoids* $(S, \circ, \bullet, 1^\circ, 1^\bullet)$ can be defined as usual by postulating a unit element for each semigroup operation.

A *semiring* is a structure $(S, +, \cdot, 0)$ such that $(S, +, 0)$ is a commutative monoid, $(S, \cdot)$ a semigroup, and the distributivity laws $x \cdot (y + z) = x \cdot y + x \cdot z$ and $(x + y) \cdot z = x \cdot z + y \cdot z$ as well as the annihilation laws $0 \cdot x = 0$ and $x \cdot 0 = 0$ hold. A semiring is *unital* if the multiplicative reduct is a monoid (with unit 1). A *dioid* is an additively idempotent semiring S, that is, $x + x = x$ holds for all $x \in S$. The additive reduct of a dioid thus forms a semilattice with order defined by $x \leq y \Leftrightarrow x + y = y$. Obviously, the classes of semirings and dioids are closed under opposition duality.

A *bi-semiring* is a structure $(S, +, \circ, \bullet, 0)$ such that $(S, +, \circ, 0)$ and $(S, +, \bullet, 0)$ are semirings; a *trioid* is an additively idempotent bi-semiring. A bi-semiring or trioid is *unital* if the underlying bi-semigroup is a bi-monoid.

A *quantale* is a structure $(Q, \leq, \cdot)$ such that $(Q, \leq)$ is a complete lattice, $(Q, \cdot)$ is a semigroup and the distributivity axioms

$$x \cdot (\sum_{i \in I} y_i) = \sum_{i \in I} (x \cdot y_i) \qquad \text{and} \qquad (\sum_{i \in I} x_i) \cdot y = \sum_{i \in I} (x_i \cdot y)$$

hold, where $\sum X$ denotes the supremum of a set $X \subseteq Q$. Similarly, we write $\prod X$ for the infimum of X, and $x + y$ and $x \sqcap y$ for the supremum and infimum of $\{x, y\}$, respectively. The distributivity laws imply, in particular, $\cdot$ is isotone in both arguments:

$$x \leq y \Rightarrow z \cdot x \leq z \cdot y \qquad \text{and} \qquad x \leq y \Rightarrow x \cdot z \leq y \cdot z.$$

A quantale is *commutative* and *partial* whenever the underlying semigroup is; *unital* if the underlying semigroup is a monoid; and *distributive* if the distributivity laws

$$x \sqcap (\sum_{i \in I} y_i) = \sum_{i \in I} (x \sqcap y_i) \qquad \text{and} \qquad x + (\prod_{i \in I} y_i) = \prod_{i \in I} (x + y_i)$$

hold. A *boolean quantale* is a distributive quantale in which every element has a complement. The boolean unital quantale $\mathbb{B}$ of the booleans, where multiplication $\cdot$ coincides with meet, plays an important role in this article.

A *bi-quantale* is a structure $(Q, \leq, \circ, \bullet)$ such that $(Q, \leq, \circ)$ and $(Q, \leq, \bullet)$ are quantales. It is unital if the two underlying semigroups are monoids.

It is easy to see that every (unital) quantale is a (unital) dioid and every (unital) bi-quantale a (unital) trioid. In addition, $0 = \sum_{i \in \emptyset} x_i$ and annihilation laws as in dioids follow from this as special cases of distributivity.

3. POWER SERIES QUANTALES

Formal (or rational) power series [Berstel and Reutenauer 1984] have been studied in formal language theory for decades. For brevity, we call them *power series* in this article. In formal language theory, a power series is simply a total function from the free monoid X^* over a finite alphabet X into a suitable algebra Q, usually a semiring or dioid $(Q,+,\cdot,0,1)$.

Operations on $f,g : X^* \to Q$ are defined as follows. Addition is lifted pointwise, that is, $(f+g)\,x = f\,x + g\,x$. Multiplication is given by the *convolution* or *Cauchy product*

$$(f \cdot g)\,x = \sum_{x=yz} f\,y \cdot g\,z,$$

where yz denotes word concatenation and the sum in the convolution is finite since finite words can only be split in finitely many ways into prefix/suffix pairs. Furthermore, the *empty power series* $\mathbb{0}$ maps every word to 0, whereas the *unit power series* $\mathbb{1}$ maps the empty word to 1 and all other words to 0.

We write Q^{X^*} for the set of power series from X^* to Q and, more generally, Q^S for the class of functions of type $S \to Q$. The following lifting result is well known.

PROPOSITION 3.1. *If* $(Q,+,\cdot,0,1)$ *is a semiring (dioid), then so is* $(Q^{X^*},+,\cdot,\mathbb{0},\mathbb{1})$.

This construction generalises from free monoids over finite alphabets to arbitrary partial semigroups or monoids. The sums in convolutions then become infinite due to infinitely many possible decompositions of elements. Here, due to potential divergence, these sums may not exist. However, we usually consider target algebras in which addition is idempotent and sums correspond to suprema. The existence of arbitrary suprema can then be covered by completeness assumptions.

We fix algebraic structures S and Q. First, we merely assume that S is a set, but for more powerful lifting results it is required to be a partial semigroup or partial monoid.

For a family of functions $f_i : S \to Q$ and $i \in I$ we define

$$(\sum_{i \in I} f_i)\,x = \sum_{i \in I} f_i\,x,$$

whenever the supremum in Q on the right-hand side exists. This comprises

$$(f+g)\,x = f\,x + g\,x.$$

Another special case is

$$(\sum_{i \in \emptyset} f_i)\,x = \sum_{i \in \emptyset} f_i\,x = 0.$$

Hence, in particular, $\sum_{i \in \emptyset} f_i = \lambda x.\ 0$ and we write $\mathbb{0}$ for this function.

We define the convolution

$$(f \otimes g)\,x = \sum_{x = y \circ z} f\,y \odot g\,z,$$

with multiplication at the levels of S, Q and Q^S. Again, this requires that the supremum on the right-hand side exists in Q. Hereafter, for simplicity, we overload $\otimes$, $\circ$ and $\odot$ with a single multiplication symbol $\cdot$. Finally, whenever S is endowed with suitable generalised units, we define $\mathbb{1} : S \to Q$ as

$$\mathbb{1}\,x = \begin{cases} 1, & \text{if } x \in I, \\ 0, & \text{otherwise.} \end{cases}$$

In the case of partial monoids, the condition $x \in I$ specialises to $x = 1$.

Note that the pointwise lifting $(f + g)\,x = f\,x + g\,x$ can be seen as a special case of convolution with partial semigroup $(S, \cdot)$ and composition defined by $x \cdot y = x$ whenever $x = y$ and undefined if $x \neq y$, for all $x, y \in S$.

Theorem 3.4, the main result in this section, shows that quantale laws lift from the algebra Q to the function space Q^S of power series under these definitions. On the way to this result, we recall that semilattice and lattice structures lift to function spaces, a fundamental result of domain theory [Abramsky and Jung 1994]. The proofs of the next two lemmas are straightforward exercises and may be found in [Dongol et al. 2014b].

LEMMA 3.2. *Let S be a set. If $(L, +, 0)$ is a semilattice with least element 0, then so is $(L^S, +, \mathbb{0})$. If L is a complete lattice, then so is L^S.*

Infima, if they exist, are defined like suprema by pointwise lifting as

$$(\prod_{i \in I} f_i)\,x = \prod_{i \in I} f_i\,x,$$

thus $(f \sqcap g)\,x = f\,x \sqcap g\,x$. Lemma 3.2 can then be strengthened as follows.

LEMMA 3.3. *Let S be a set. If $(L, +, \sqcap, 0)$ is a (distributive) lattice with least element 0, then so is $(L^S, +, \sqcap, \mathbb{0})$. Completeness and infinite distributivity laws between infima and suprema lift from L to L^S.*

The final lifting result in this section deals with the multiplicative structure as well. This requires S to be a partial semigroup instead of a set.

THEOREM 3.4. *Let $(S, \cdot)$ be a partial semigroup. If $(Q, \leq, \cdot)$ is a (distributive) quantale, then so is $(Q^S, \leq, \cdot)$. In addition, commutativity in Q lifts to Q^S if S is commutative; unitality in Q lifts to Q^S if S is a generalised partial monoid.*

PROOF. Since Q is a quantale, all suprema and infima exist; in particular those needed for convolutions.

The lifting to complete (distributive) lattices is covered by Lemma 3.3. It therefore remains to check the multiplicative monoid laws, distributivity of multiplication and annihilation. For left distributivity, for instance,

$$(f \cdot \sum_{i \in I} g_i)\,x = \sum_{x = y \cdot z} (f\,y \cdot \sum_{i \in I} g_i\,z) = \sum_{i \in I} \sum_{x = y \cdot z} (f\,y \cdot g_i\,z) = \sum_{i \in I} (f \cdot g_i)\,x.$$

The proof of right distributivity is opposition dual.

Left distributivity ensures associativity, the proof of which lifts as with rational power series (Proposition 3.1). The restriction to partial semigroups is insignificant because in $x = y \cdot z$, it is required that $(y, z) \in D$. The same holds for unitality proofs.

Commutativity lifts from S and Q as follows:

$$(f \cdot g)\,x = \sum_{x = y \cdot z} f\,y \cdot g\,z = \sum_{x = z \cdot y} g\,z \cdot f\,y = (g \cdot f)\,x.$$

For the right unit law,

$$(f \cdot \mathbb{1})\,x = \sum_{x = y \cdot z} f\,y \cdot \mathbb{1}\,z = \sum_{x = x \cdot e, e \in I} f\,x \cdot \mathbb{1}\,e = f\,x \cdot 1 = f\,x.$$

The proof of the left unit law is opposition dual. □

The distributivity laws on Q^S imply the annihilation laws $\mathbb{0} \cdot f = \mathbb{0}$ and $f \cdot \mathbb{0} = \mathbb{0}$ for all $f : S \to Q$. When only finite sums are needed, Q can be assumed to be a semiring or dioid instead of a quantale. The following corollary to Theorem 3.4 provides an example.

COROLLARY 3.5. *Let $(S, \cdot)$ be a finite partial semigroup. If $(Q, +, \cdot, 0)$ is a semiring, then so is $(Q^S, +, \cdot, \mathbb{0})$. In addition, idempotency of $+$ in Q lifts to Q^S; commutativity of multiplication $\cdot$ in S and Q lifts to Q^S; unitality in Q lifts to Q^S if S is a generalised partial monoid.*

As another specialisation, Proposition 3.1 is recovered easily when S is the free monoid over a given alphabet and Q a semiring or dioid.

Corollary 3.5 indicates the two different kinds of liftings used in the constructions of Q^S. That of additive properties is pointwise; it depends only on properties of quantale Q. That of multiplicative properties, by contrast, uses convolution; it depends on properties of quantale Q as well as semigroup S.

A construction similar to the one in Theorem 3.4 is well known in category theory [Day 1970]: the functors Set^C form a (symmetric) monoidal category whenever C is a (symmetric) monoidal category. The tensor on this functor category is known as *Day convolution*; it is defined in terms of a coend. The precise relationship to our lifting result and its consideration in the setting of enriched categories is beyond the scope of this article.

4. POWER SERIES INTO THE BOOLEAN QUANTALE

In many applications, the target quantale Q is formed by the booleans $\mathbb{B}$. Power series are then of type $S \to \mathbb{B}$ and can be interpreted as characteristic functions or predicates. In fact, $\mathbb{B}^S$ is isomorphic to the power set 2^S of S as well as the set of all predicates over S, identifying predicates with their extensions.

Theorem 3.4 then specialises to the powerset lifting of a partial semigroup or monoid S. For each $x \in S$, the boolean value $f\,x$ expresses whether or not x is in the set corresponding to f. Powerset liftings have been studied widely in mathematics since the late nineteenth century (cf. [Goldblatt 1989; Brink 1993]). They are ubiquitous in program semantics, for instance as power domains (cf. [Abramsky and Jung 1994]).

COROLLARY 4.1. *Let S be a partial (commutative) semigroup. Then $\mathbb{B}^S$ forms a (commutative) distributive quantale where $\mathbb{B}^S \cong 2^S$, $\leq$ corresponds to $\subseteq$ and convolution, for all $X, Y \subseteq S$, to the complex product*

$$X \cdot Y = \{x \cdot y \in S \mid x \in X \land y \in Y\}.$$

If S has generalised units in I, then $\mathbb{B}^S$ has unit $\mathbb{1} = I$.

Various instances of Corollary 4.1 are discussed in Sections 5 and 6. Jónsson and Tarski [1952] have shown that the powerset lifting of a generalised Brand groupoid yields a complete atomistic relation algebra, of which Corollary 4.1 is a special case. Even more generally, they have studied powerset liftings of relational structures to boolean algebras with operators [1951], which foreshadows the construction of predicate transformer semantics from relational ones.

The power quantale $\mathbb{B}^S$ carries a natural logical structure with elements of $\mathbb{B}^S$ corresponding to predicates, suprema to existential quantification, infima to universal quantification and the lattice order to implication. In particular, $+$ corresponds to disjunction and $\sqcap$ to conjunction.

More interesting is the logical interpretation of convolution

$$(f \cdot g)\, x = \sum_{x=y\cdot z} f\, y \cdot g\, z$$

in the power quantale $\mathbb{B}^S$. The expression $x = y \cdot z$ denotes the decomposition or separation of the semigroup element x into parts y and z. The composition $f\, y \cdot g\, z = f\, y \sqcap g\, z$ in $\mathbb{B}$ models the conjunction of predicate f applied to y with predicate g applied to z. Finally, the supremum $\sum$ models the existential quantification over these conjunctions with respect to all possible decompositions of x.

The commutative case of Corollary 4.1 is immediately relevant to separation logic. In this context, the partial commutative semigroup $(S, *)$ is called a *resource semigroup* [Calcagno et al. 2007]; it provides an algebraic abstraction of the heap. Its powerset lifting $\mathbb{B}^S$ captures the algebra of resource predicates that form the assertions of an extended Hoare logic — the assertion quantale of separation logic. In this assertion quantale, separating conjunction is precisely convolution: the product $x = y * z$ on the resource semigroup S separates the resource x into y and z and the product $f\, y \sqcap g\, z$ in $\mathbb{B}$ conjoins $f\, y$ and $g\, z$. The concrete case of the heap is considered in more detail in Example 6.2.

It seems interesting to characterise the properties that can be lifted from algebras S and Q to Q^S. For $Q = \mathbb{B}$ it is well known that an equation $s = t$ can be lifted if and only if each variable in that equation occurs only once both in s and t or else s and t are identical (cf. the discussion in Brink [1993]). It follows, for instance, that idempotency of multiplication in S does not lift to $\mathbb{B}^S$ and therefore not to Q^S in general.

In sum, the power series approach yields a simple algebraic view on a lifting to function spaces in which convolution into the booleans allows various interpretations, including that of a complex product, that of separating conjunction — commutative or non-commutative — and that of separating conjunction as a complex product according to Corollary 4.1. In the commutative setting it gives a simple account of the category-theoretical approach to O'Hearn and Pym's logic of bunched implication [1999], where separating conjunction is Day convolution [Day 1970] and proofs require the Yoneda lemma. This bridges the gap between the logic of bunched implications and the more elementary algebraic approach outlined by Calcagno et al. [2007]. The relative simplicity of power series is evidenced in the implementation of a lightweight program construction and verification tool for separation logic in Isabelle/HOL [Dongol et al. 2015]. Reasoning with monoidal categories in such interactive theorem provers would be much more complicated and probably infeasible.

5. NON-COMMUTATIVE EXAMPLES

After the conceptual development of the previous sections, we now discuss a series of examples that underpin the universality and relevance of convolution in computing. All of them can be obtained as instances of Theorem 3.4 after setting up partial semigroups or monoids appropriately. For all these structures, the lifting to the function space is then generic and automatic. The booleans form a particularly interesting target quantale. This section considers only examples with a non-commutative notion of convolution; for commutative examples see Section 6. More details on the constructions outlined can be found in a report [Dongol et al. 2014a]. First we revisit the formal languages example in the quantale setting.

Example 5.1 (*Formal Languages*). Let $(X^*, \cdot, \varepsilon)$ be the free monoid generated by the finite alphabet X with ε denoting the empty word. Let Q form a distributive unital quantale. Then by Theorem 3.4, Q^{X^*} forms a distributive unital quantale as well. More precisely, because the suprema in convolutions are always finite, one obtains the unital dioid $(Q^{X^*}, +, \cdot, \mathbb{0}, \mathbb{1})$ by lifting from a dioid $(Q, +, \cdot, 0, 1)$. This is the well known rational power series dioid of formal language theory. For $Q = \mathbb{B}$ one obtains, by Corollary 4.1, the quantale $\mathbb{B}^{X^*}$ of formal languages over X. □

Our next set of examples is based on a product construction at the semigroup level.

Example 5.2 (*Matrices*). Square matrices are functions $f : A \times A \to Q$ from an index set $A{\times}A$ into a suitable coefficient algebra Q. (Non-square matrices are discussed in Section 8.) Consider the generalised partial monoid $(A \times A, D, I, \cdot)$ over set A with $D = \{(p,q) \in (A \times A) \times (A \times A) \mid \pi_2\, p = \pi_1\, q\}$, where π_1 and π_2 are the projections to the first and second coordinate of an ordered pair, $I = \{(i,i) \mid i \in A\}$ and composition $p \cdot q = (\pi_1\, p, \pi_2\, q)$. Then matrix addition $(f + g)\,(i,j) = f\,(i,j) + g\,(i,j)$ corresponds to a pointwise lifting to $Q^{A\times A}$; matrix multiplication $(f \cdot g)\,(i,j) = \sum_{k\in A} f\,(i,k) \cdot g\,(k,j)$ is a convolution on that function space. This requires suitable restrictions to guarantee the existence of sums. The zero and unit matrices are defined by $\mathbb{1}\,(i,j) = \delta_{ij}$ (the Kronecker delta) and $\mathbb{0}\,(i,j) = 0$.

Theorem 3.4 then shows that quantales are closed under matrix formation. It can easily be adapted to showing that square matrices of finite dimension over a semiring form a semiring or that matrices over a dioid form a dioid. □

Example 5.3 (*Binary Relations, Probabilistic Relations, Automata*).

(a) The quantale of binary relations is formed by boolean matrices, which have type $A \times A \to \mathbb{B}$.

(b) The quantales of probabilistic or fuzzy relations are formed by matrices of type $A \times A \to [0,1]$ with suitable weight algebras defined on the unit interval.

(c) An extension of Corollary 3.5 shows that any Kleene algebra K is closed under formation of matrices of type $A \times A \to K$. The necessary treatment of the Kleene star is beyond the scope of this article. Transition relations of finite automata over state space V and alphabet X are finite matrices of type $V \times V \to \mathsf{Rex}(X)$ into the Kleene algebra $\mathsf{Rex}(X)$ of regular expressions over X. It is well known that regular languages need not be closed under general unions, hence do not form quantales. The approach generalises to probabilistic or weighted automata. See [Dongol et al. 2014b] for details. □

Finally, we study examples based on fusion products.

Example 5.4 (*Interval Functions*). Let $(P, \leq)$ be a linear order and I_P the set of closed intervals over P — the empty interval is open by definition. We impose a generalised partial monoid structure on I_P by defining $D = \{(x,y) \in I_P{\times}I_P \mid \max x = \min y\}$, $I = \{[a,a] \mid a \in P\}$ as the set of all point intervals and the *fusion product* $x \cdot y = x \cup y$. It follows that the set Q^{I_P} of all interval functions into a distributive quantale Q forms a distributive quantale. Like matrices, the unit interval function is $\mathbb{1}\,[a,b] = \delta_{ab}$. The quantale of interval functions then becomes unital.

Interval predicates are interval functions of type $I_P \to \mathbb{B}$. Convolution

$$(f \cdot g)\, x = \sum_{x = y\cdot z} f\, y \sqcap g\, z$$

of interval predicates f and g is known as the *chop* operation [Moszkowski 2000]: the predicate $f \cdot g$ holds over the interval $[a,c]$ if this interval can be split into intervals $[a,b]$ and $[b,c]$ such that $f\,[a,b]$ and $g\,[b,c]$ both hold. □

Interval predicates typically involve modal operators. For instance, the interval predicate $(\Diamond p)\, x$ means that property p holds at some point in x, whereas $(\Box p)\, x$ means that p holds at all points in x [Moszkowski 2000]. Consequently, $(\Box p) \cdot (\Box \neg p)$ is always false, since p and $\neg p$ cannot hold at the fusion point. Such effects can be avoided by basing the construction of the interval functions quantale on compositions of arbitrary bounded intervals $[a, b]$, $(a, b]$, $[a, b)$ or (a, b) [Dongol et al. 2014a]. Alternatively, the duration calculus uses an interval predicate that holds almost everywhere on an interval [Zhou and Hansen 2004]. Others have defined 'jump conditions' leaving the possibility of both f and $\neg f$ holding at the fusion point open [Höfner and Möller 2009]. The model of trajectories by Höfner and Möller [2009] can be encoded as convolution in a straightforward manner, and because their theory encapsulates hybrid automata [Henzinger 1996], so does our algebraic treatment in this paper.

Interval functions have also been studied in terms of *incidence algebras* formed by real-valued functions of two variables ranging over locally finite posets, which are posets in which all closed intervals are finite [Rota 1964]. Incidence algebras form associative algebras over the real field with convolution as multiplication. Rota attributes the idea of interval functions to Dedekind and Bell in the late nineteenth century.

Example 5.5 (*Traces, Paths, Words*).

(a) Let V be a finite set of state symbols and X a finite set of transition symbols. A *trace* over V and X [Eilenberg 1974] is a word in $T = V \cdot (X \cdot V)^*$, i.e., states and symbols alternate, starting and finishing with a state. A generalised partial monoid is obtained with $D = \{(\tau_1, \tau_2) \in T \times T \mid \mathrm{last}\,\tau_1 = \mathrm{first}\,\tau_2\}$, $I = V$ and for any state s fusion product $\tau_1\ s \cdot s\ \tau_2 = \tau_1\ s\ \tau_2$. Thus the set Q^T of trace functions into the (distributive) quantale Q forms a (distributive) quantale. If Q is unital, then Q^T is unital with $\mathbb{1}$ defined in the standard way. Setting $Q = \mathbb{B}$ yields the well-known quantale of sets of traces with unit V.

(b) When $|X| = 1$, traces are isomorphic to sets of paths, which are finite sequences of state symbols, and the constructions of the quantale Q^{V^+} of path functions over the generalised partial monoid V^+ of non-empty paths follows the trace case.

(c) When $|V| = 1$, traces are isomorphic to words, trace fusion becomes the total operation of word concatenation and the element of V yields the empty word, which is a unit of composition. The trace function quantale then specialises once more to the formal language quantale. □

The fusion-based constructions generalise from intervals, traces, paths and so forth to equivalence classes of labelled bounded linear orders in which partial orders with the same labelling and order structure are identified — c.f. (bounded) *linear pomsets* [Gischer 1988; Grabowski 1981]. Relation D then holds between two such pomsets p_1 and p_2 if the label of $\mathrm{lub}\,p_1$ is equal to the label of $\mathrm{glb}\,p_2$. The fusion product $p_1 \cdot p_2$ makes every element in p_1 precede every element p_2 with its lower bound removed. Trace, path and word functions are instances obtained with appropriate labelling disciplines. The formal construction of such equivalence classes is known from the theory of pomsets and leads beyond the scope of this article. A unification of the product and fusion based construction can possibly be obtained within category theory.

The examples in this section show that the generic lifting construction in Theorem 3.4 allows a uniform treatment of various mathematical objects, including relations, formal languages, matrices and sets of intervals. In each case, a (partial) composition on the underlying objects needs to be defined, e.g., on words, ordered pairs, traces, paths or intervals. Lifting to the function space is then generic.

6. COMMUTATIVE EXAMPLES

This section provides instances of Theorem 3.4 and Corollary 4.1 for the commutative case. As discussed in Section 4, this typically arises when the composition of the underlying semigroup $(S, *)$ is used to split resources in a spatial fashion, which is in contrast to the previous section where $f \cdot g$ meant that there was a dependency between f and g, which often carries a temporal meaning. One can sometimes think of convolution instantiated to such a spatial separation in terms of parallelism or concurrency.

In particular we instantiate Theorem 3.4 to four kinds of resource monoids based on multisets under multiset addition, sets under disjoint union, partial functions under union and vectors. Notions of separating conjunction as convolution arise in all these examples in a natural way. In the disjoint union and vector examples, the relationship between convolution, separation and concurrency becomes more apparent. Previously, this observation of separating conjunction as a notion of concurrency with a strongly spatial meaning has been one of the motivations for concurrent separation logic [O'Hearn 2004] and concurrent Kleene algebra [Hoare et al. 2011b].

Example 6.1 (*Separating Conjunction on Multisets*). The free commutative monoid $(X^{(*)}, *, 0)$ generated by the alphabet X is isomorphic to the set of all multisets over X with $*$ being multiset addition $\uplus$. By Theorem 3.4, $Q^{X^{(*)}}$ forms a commutative quantale if Q does; distributivity and unitality lift as usual.

Convolution $(f * g)\, x$ separates the multiset or resource x in all possible ways and then applies the functions f and g in parallel to the result, depending on the interpretation of multiplication in Q. For $Q = \mathbb{B}$, the function space $\mathbb{B}^{X^{(*)}}$ forms the resource predicate quantale over multisets. Convolution is separating conjunction based on multiset addition as a separator: $(f * g)\, x = \sum_{x = y \uplus z} f\, y \sqcap g\, z$. □

In many contexts, multisets form a paradigmatic data type for resources.

Example 6.2 (*Heaplets, Sets, Vectors*).

(a) Let $(S, D, *, 0)$ be the partial commutative monoid of partial functions (*heaplets*), $\eta : A \rightharpoonup B$ where $D = \{(\eta_1, \eta_2) \in S \times S \mid dom(\eta_1) \cap dom(\eta_2) = \emptyset\}$, $\eta_1 * \eta_2 = \eta_1 \cup \eta_2$, and $0 : A \rightharpoonup B$ is the empty heaplet. By Theorem 3.4, Q^S forms a commutative distributive unital quantale whenever Q does. In particular, $\mathbb{B}^S$ forms an algebra of heap assertions with convolution as separating conjunction over the heap.

(b) The free commutative idempotent monoid generated by the alphabet X is isomorphic to 2^X under union and the empty set. Defining disjoint union based on $D = \{(x, y) \mid x \cap y = \emptyset\}$ turns this set into a partial commutative monoid. The quantale lifting is then similar to the heaplet case.

(c) Consider n-dimensional vectors as finite functions of type $[1, n] \to A$ and define the support of a vector v as $supp(v) = \{i \in dom(v) \mid v(i) \neq 0\}$. In this case $D = \{(v, w) \mid supp(v) \cap supp(w) = \emptyset\}$ and $v * w = v + w$ define a partial commutative monoid with the all zero vector as unit. The quantale lifting is as before. □

Beyond these instances of separating conjunction, the resource semigroups considered in this section allow for an interpretation of convolution $f * g$ applied to a resource x that is naturally concurrent: programs f and g are executed in parallel on the resources y and z allocated by decomposing x. Their outputs after execution are then composed by $f\, y \cdot g\, z$. In the case of a powerset lifting, this composition may yield a global resource. The sum over all these compositions yields the possible behaviours of the parallel execution $f * g$ on x in terms of their outputs.

Further notions of resource monoid and lifting to assertion algebras have been studied in the Views framework [Dinsdale-Young et al. 2013]. Their generic soundness results for Hoare logics seem to arise as instances of the constructions in this article.

Quantales are automatically residuated with respect to their monoidal operations. Order-theoretically, these are upper adjoints of the functions $\lambda y.\ x \cdot y$ and $\lambda y.\ y \cdot x$. The two residuations coincide in the commutative case. In separation logic, this residuation is known as the magic wand operation. More generally, all convolutions studied in this article have upper adjoints, and hence it is also possible to characterise adjoints for the examples in Section 5, providing operations analogous to the magic wand in non-commutative settings.

7. POWER SERIES OVER PARTIAL QUANTALES

This section generalises Theorem 3.4 to situations in which the target algebra Q is a partial quantale in the sense that its multiplication is partial. Now, partiality of composition shows up not only in the decomposition $x = y \cdot z$, but also in the product $f\,y \cdot g\,z$ in convolutions. By definition of suprema and infima, only elements that are defined contribute, and in particular the supremum of the empty set (which does not contain any defined elements) is equal to 0. This means that all suprema are defined.

It turns out that the quantale structure of the target algebra is preserved in the function space, including partiality.

As an example we consider linear transformations of certain vectors implemented by matrices, in which vectors that are separated can be transformed in parallel fashion by matrices which can be separated into non-zero blocks along the diagonal. This is a particular manifestation of the correspondence between separation and concurrency in the context of convolution.

PROPOSITION 7.1. *Let $(S, \cdot)$ be a partial semigroup. If $(Q, \leq, \cdot)$ is a (distributive) partial quantale, then so is $(Q^S, \leq, \cdot)$. In addition, commutativity lifts from S and Q to Q^S and unitality lifts if S is a generalised partial monoid.*

The proof follows the lines of Theorem 3.4.

Example 7.2 (*Linear Transformations of Vectors*). Consider again the partial semigroup $(S, *)$ on n-dimensional vectors from Example 6.2. We assume that addition is idempotent in the coefficient algebra. It is easy to check that S actually forms a partial commutative dioid under multiplication $*$ and standard vector addition. Distributivity $x * (y + z) = (x * y) + (x * z)$ follows immediately from the definition.

Proposition 7.1 then implies as a special case that the functions of type $S \to S$ form a commutative dioid; they form a trioid with the other multiplication being function composition. The sum in the convolution is obviously finite since there are only finitely many ways of splitting a vector of finite dimension. In addition, the functions f and g in a convolution are not only applied to separate parts y and z of vector x, but they must map to separate parts $f\,y$ and $g\,z$ of the resulting vector as well.

It is easy to check that the unit with respect to $*$ on S^S is defined as $\mathbb{1}\,x = \delta_{x0}$.

For further illustration consider the linear transformations on n-dimensional vectors given by multiplying n-dimensional vectors with an $n \times n$ matrix. To obtain idempotent addition we consider vectors and matrices over a dioid or Kleene algebra, as in Example 5.3(c). The matrices in this example correspond to the transition relations of finite automata, whereas the vectors store partial languages starting from some initial state [Conway 1971]. As a simple example of a term contributing to a convolution consider

$$\begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix} \begin{pmatrix} x \\ 0 \end{pmatrix} * \begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix} \begin{pmatrix} 0 \\ y \end{pmatrix} = \begin{pmatrix} a_1 x \\ c_1 x \end{pmatrix} * \begin{pmatrix} b_2 y \\ d_2 y \end{pmatrix},$$

which is not defined, whereas

$$\begin{pmatrix} a_1 & b_1 \\ 0 & d_1 \end{pmatrix} \begin{pmatrix} x \\ 0 \end{pmatrix} * \begin{pmatrix} a_2 & 0 \\ c_2 & d_2 \end{pmatrix} \begin{pmatrix} 0 \\ y \end{pmatrix} = \begin{pmatrix} a_1 x \\ 0 \end{pmatrix} * \begin{pmatrix} 0 \\ d_2 y \end{pmatrix} = \begin{pmatrix} a_1 x \\ d_2 y \end{pmatrix}.$$

This shows that matrices contributing to convolutions must essentially be of the form

$$\begin{pmatrix} M_1 & M_3 \\ \mathbb{0} & M_2, \end{pmatrix} \qquad \text{or} \qquad \begin{pmatrix} M_1 & \mathbb{0} \\ M_3 & M_2, \end{pmatrix}.$$

The concurrent flavour of $*$ is particularly apparent when matrices consist of two non-trivial blocks along the diagonal modulo (synchronised) permutations of rows and columns. That is, they are of the form

$$\begin{pmatrix} M_1 & \mathbb{0} \\ \mathbb{0} & M_2, \end{pmatrix},$$

where $\mathbb{0}$ represents zero matrices of appropriate dimension. Each pair of vectors resulting from a decomposition can be rearranged such that the first vector consists of an upper block of non-zero coefficients and a lower block of zeros, whereas the second vector consists of an upper zero and a lower non-zero block, and such that the two non-zero blocks do not overlap. One must be able to decompose matrices and vectors of the linear transformation into the same blocks to make convolutions non-trivial.

The transformations implemented by the above block matrix on rearranged vectors, and more generally all linear transformations, can clearly be executed independently or in parallel by the matrices M_1 and M_2 on parts of a vector if the convolution is non-trivial. In this sense the convolution $*$ on linear transformations over suitable vectors is a notion of concurrent composition. □

8. PARTIAL POWER SERIES

Another form of partiality is where the function, $f : S \nrightarrow Q$, representing the power series is partial. The definition of convolution $f \otimes g$ can be revised to handle this case by requiring that, if $x = y \circ z$, then both $y \in dom(f)$ and $z \in dom(g)$ for that decomposition of x to contribute to the sum.

$$(f \otimes g)x = \sum_{\substack{y \in dom(f) \\ z \in dom(g) \\ x = y \circ z}} f\, y \cdot g\, z$$

Example 8.1 (*Sparse matrices*). Sparse matrices can be represented as partial functions, $f : A \times A \nrightarrow Q$, from index set $A \times A$ into a suitable coefficient algebra Q. The generalised partial monoid $(A \times A, D, I, \cdot)$ is the same as for Example 5.2. A non-square matrix $f \in A_1 \times A_2 \to Q$, for $A_1, A_2 \subseteq A$, can be viewed as a partial function $f \in A \times A \nrightarrow Q$, where $dom(f) = A_1 \times A_2$, and hence this example covers non-square matrices as well. □

9. POWER SERIES OVER BI-SEMIGROUPS

Theorem 3.4 shows that the quantale structure Q is preserved at the level of the function space Q^S provided that S is a partial semigroup. This can easily be adapted from partial semigroups S to partial n-semigroups and n-quantales with n operations of composition which may or may not be commutative. Here we restrict our attention to bi-semigroups and bi-quantales and we discuss two examples.

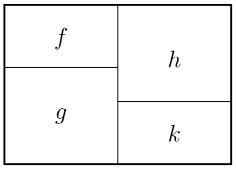
PROPOSITION 9.1. *Let $(S, \circ, \bullet)$ be a partial bi-semigroup (bi-monoid). If $(Q, \leq, \circ, \bullet)$ is a (distributive unital) bi-quantale, then so is $(Q^S, \leq, \circ, \bullet)$.*

It is obvious that properties such as commutativity and unitality lift as before. This setting yields two convolutions

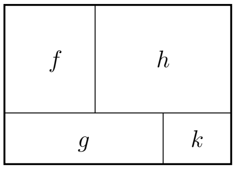
$$(f \circ g)\, x = \sum_{x = y \circ z} f\, y \circ g\, z \qquad \text{and} \qquad (f \bullet g)\, x = \sum_{x = y \bullet z} f\, y \bullet g\, z.$$

In the case of predicates these be visualised as a horizontal and a vertical composition, as the following diagrams show.

$(f \bullet g) \circ (h \bullet k)$



$(f \circ h) \bullet (g \circ k)$



The visualisation is made more concrete by the following example.

Example 9.2 (*Functions over Two-Dimensional Intervals*). Closed two-dimensional intervals over a linear order can be defined in a straightforward way. For one-dimensional intervals x and y, we write $x \times y$ for the two dimensional interval $\{(a, b) \mid a \in x \wedge b \in y\}$. We obtain a partial bi-semigroup by defining

$$D_\circ = \{(x_1 \times y_1, x_2 \times y_2) \mid \max x_1 = \min x_2 \wedge y_1 = y_2\},$$
$$D_\bullet = \{(x_1 \times y_1, x_2 \times y_2) \mid x_1 = x_2 \wedge \max y_1 = \min y_2\}$$

as well as horizontal and vertical composition

$$(x_1 \times y_1) \circ (x_2 \times y_2) = (x_1 \cup x_2) \times (y_1 \cup y_2) = (x_1 \times y_1) \bullet (x_2 \times y_2).$$

Whenever the target algebra forms a bi-quantale, Proposition 9.1 applies and the function space forms a bi-quantale as well. In particular, horizontal and vertical convolution can be written as

$$(f \circ g)\,(x \times y) = \sum_{x = x_1 \cdot x_2} f\,(x_1 \times y) \circ g\,(x_2 \times y),$$
$$(f \bullet g)\,(x \times y) = \sum_{y = y_1 \cdot y_2} f\,(x \times y_1) \bullet g\,(x \times y_2)$$

where $x_1 \cdot x_2$ and $y_1 \cdot y_2$ denote interval fusion. This generalises easily to n-dimensional intervals with n convolutions that may or may not be commutative. □

The impact of convolution as a concurrent composition is apparent in the next example.

Example 9.3 (*Series-Parallel Pomset Languages*). Let $(S, \cdot, *, 1)$ be a bi-monoid with non-commutative composition $\cdot$, commutative composition $*$ and shared unit 1. Furthermore, let $(Q, \leq, \cdot, *, 1)$ be a bi-quantale with non-commutative composition $\cdot$, commutative composition $*$ and shared unit 1. Then Q^S forms a bi-quantale according to Proposition 9.1 with a non-commutative convolution given by $\cdot$ and a commutative convolution given by $*$. For $\mathbb{B}^S$ and S being freely generated by a finite alphabet X, we obtain the *series-parallel pomset languages* or *partial word languages* over X, which have been studied by Grabowski [1981], Gischer [1988] and others. They form a standard model of true concurrency. □

10. POWER SERIES FROM MULTIPLE PARTIAL SEMIGROUPS

This section considers two separate partial semigroups or monoids $(S_1, \circ)$ and $(S_2, \bullet)$, which can be reduced easily to the bi-semigroup case[1], and also generalised to n semigroups. Quite often, S_2 is assumed to be commutative.

[1] We are grateful to an anonymous reviewer for suggesting this simplification.

PROPOSITION 10.1. *Let $(S_1, \circ)$ and $(S_2, \bullet)$ be partial semigroups. If $(Q, \leq, \circ, \bullet)$ is a (distributive) bi-quantale, then so is $(Q^{S_1 \times S_2}, \leq, \circ, \bullet)$. It is unital whenever Q is unital and S_1 and S_2 are generalised partial monoids. A convolution on $Q^{S_1 \times S_2}$ is commutative if the corresponding semigroup and quantale operations are commutative.*

PROOF. Let $D_\circ$ be the domain relation for $\circ$ and $D_\bullet$ that for $\bullet$. We define a bi-monoid on $S = S_1 \times S_2$ with

$$D'_\circ = \{((x_1, x_2), (y_1, y_2)) \in S \times S \mid (x_1, y_1) \in D_\circ \wedge x_2 = y_2\},$$
$$D'_\bullet = \{((x_1, x_2), (y_1, y_2)) \in S \times S \mid x_1 = y_1 \wedge (x_2, y_2) \in D_\bullet\}$$

providing the domains of the two compositions defined by

$$(x_1, x_2) \circ (y_1, y_2) = (x_1 \circ y_1, x_2) \qquad \text{and} \qquad (x_1, x_2) \bullet (y_1, y_2) = (x_1, x_2 \bullet y_2).$$

Our result for partial semigroups then follows from Proposition 9.1 □

COROLLARY 10.2. *Let $(S_1, \circ)$ be a partial semigroup and S_2 a set. If $(Q, \leq, \circ)$ is a (distributive) quantale, then so is $(Q^{S_1 \times S_2}, \leq, \circ)$. It is unital whenever Q is unital and S_1 is a generalised partial monoid. Convolution on $Q^{S_1 \times S_2}$ is commutative if the compositions on S_1 and Q are both commutative.*

As usual, properties such as commutativity and unitality can be lifted from the underlying structures. In addition, the construction generalises to more than two partial semigroups.

It is well known that the following function spaces are isomorphic: $(C^A)^B \cong (C^B)^A \cong C^{A \times B} \cong C^{B \times A}$. We move freely between them and use curried as well as uncurried functions in examples.

First we present two examples in which a spatial or a temporal separation interacts with a set, as in Corollary 10.2.

Example 10.3 (*Separating Conjunction over Store-Heap Pairs*). Applications of separation logic are based on program states that are pairs (s, h) of a store s and a heap h. The store is modelled as a set; more concretely as a function from program variables to values. The heap is modelled as a resource monoid; more concretely as an object similar to the one in Example 6.2(a). With Corollary 10.2, separating conjunction on states can be defined in a simple natural way as a convolution

$$(f * g)\,(s, h) = \sum_{h = h_1 * h_2} f\,(s, h_1) \sqcap g\,(s, h_2)$$

that splits the heap and leaves the store unchanged; the function space forms once more an assertion quantale for separation logic. Dongol et al. [2015] have implemented and applied this construction in a verification tool for separation logic.

In addition, the store can be split using disjoint union, $s = s_1 \oplus s_2$, e.g., to distinguish between local and global variables [Back and von Wright 1994]. In this case, $(S, \oplus)$ forms a partial semigroup as well and a second convolution or separating conjunction $f \oplus g$ that splits the store and leaves the heap unchanged can be defined. The resulting assertion bi-quantale is now described by Proposition 10.1. □

Example 10.4 (*Stream Interval Functions*). Let $(S_1, \cdot)$ be the partial semigroup $(I_P, \cdot)$ of closed intervals I_P under interval fusion as in Example 5.4 and let S_2 be the set of all functions of type $P \to A$ for an arbitrary set A. It follows from Corollary 10.2 that $Q^{I_P \times A^P}$ forms a distributive quantale whenever Q is a distributive quantale. Generalised units can be lifted to a unit of $Q^{I_P \times A^P}$ along the lines of Example 5.4.

As a typical interpretation, fix $P = \mathbb{R}$ with the standard order as a model of time. Functions $f : \mathbb{R} \to A$ can then used to model the temporal behaviour or trajectories of some system — for instance as solutions of differential equations. In that case, $F\ x\ f$ evaluates the behaviour of system f in the interval x, similar to a higher-order function in functional programming. We call F a *stream interval function*. The convolution

$$(F \cdot G)\, x\, f = \sum_{x=y\cdot z} (F\, y\, f) \cdot (G\, z\, f)$$

splits the interval x into all possible prefix/suffix pairs y and z, applies F to the behaviour of f on interval y and G to the behaviour of f on interval z and then combines these results. There are different ways in which the application of stream interval functions can be realised. Moreover, the situation generalises to arbitrary finitely bounded intervals without fusion.

Stream interval predicates, where $Q = \mathbb{B}$, have been studied by Dongol et al. [2014a] in the context of real-time action systems, but not in the power series setting. The composition in the sum becomes the conjunction $(F\, y\, f) \sqcap (G\, z\, f)$ and convolution reduces to chop.

A predicate F could, for instance, test the values of a function f over an interval x — at all points of x, at some points of x, at almost all points of x, at no points of x and so on. It could, for instance, test, whether the trajectory of system f evolves within given boundaries, that is a flight path is within a given corridor or that a train moves according to a given time schedule.

More concretely, let $P = A = \mathbb{R}$ and let $f\ t = t^3$. Let

$$F\ x\ f = \forall t \in x.\ f\ t \geq 0 \qquad \text{and} \qquad G\ x\ f = \forall t \in x.\ f\ t \leq 0.$$

Then $F\ [0, 10]\ f = 1$ and $G\ [-7, -1]\ f = 1$, but $F\ [-2, -1]\ f\ =\ 0$ and $G\ [-7, 1]\ f = 0$.

Moreover, for instance,

$$(F \cdot G)\, [-1, 1]\, f = \sum_{\tau \in [-1,1]} (\forall t \in [-1, \tau].\ t^3 \geq 0) \sqcap (\forall t \in [\tau, 1].\ t^3 \leq 0) = 0,$$

because it is impossible to split the interval $[-1, 1]$ in such a way that t^3 is first positive and then negative. However,

$$(G \cdot F)\, [-1, 1]\, f = \sum_{\tau \in [-1,1]} (\forall t \in [-1, \tau].\ t^3 \leq 0) \sqcap (\forall t \in [\tau, 1].\ t^3 \geq 0) = 1,$$

because $\forall t \in [-1, \tau].\ t^3 \leq 0$ and $\forall t \in [\tau, 1].\ t^3 \geq 0$ hold for $\tau = 0$, which splits the interval in such a way that t^3 is first negative and then positive. □

Next we present two interval-based models with a temporal and a spatial or concurrent convolution.

Example 10.5 (*Vector Stream Interval Functions*). Let f from the previous example now be a vector or product of functions f_i such that $f : P \to A^n$, or more concretely $f : \mathbb{R} \to A^n$. One can then split $f\, t$ with respect to the commutative operation $*$ on A^n. For functions $f, g : P \to A^n$ we define $(f * g)\, t = f\, t * g\, t$ by pointwise lifting. This turns $(S_2, *) = ((A^n)^P, *)$ into a partial commutative semigroup, whereas $(S_1, \cdot)$ is again the partial semigroup $(I_P, \cdot)$. By Proposition 10.1, $Q^{S_1 \times S_2}$ forms a distributive bi-quantale with commutative convolution $*$ whenever Q does.

The stream interval predicates again yield an interesting special case. Now a vector of functions, for instance the solution to a system of differential equations, is applied to arguments ranging over an interval and the stream interval predicates evaluate the temporal behaviour modelled by this vector of functions on the interval.

The convolutions $(F \cdot G)\,x\,f$, which is based on splitting $x = y \cdot z$, and $(F * G)\,x\,f$, which is based on splitting $f = g * h$, can be visualised as vertical and horizontal splits, respectively.

The vertical convolution evaluates the evolution of the system modelled by f over consecutive intervals y and z, using F to evaluate f within y and then G to evaluate f within z. In the context of interval logics this corresponds to a chop operation by which the predicates F and G make statements about the evolution of f during consecutive periods of time. The horizontal one evaluates the parallel evolution of the subsystems g and h of f over the full interval x, using F for evaluating g over x and separately G for evaluating h over x. This adds an algebraic notion of parallelism or concurrency to interval calculi by which F and G make statements about the evolution of subsystems g and h over a period of time. □

Example 10.6 (*Store-Heap Pairs over Intervals*). The previous examples can be combined into a three-dimensional one. Consider, for instance, the store and heap as functions of time and suppose we can split the heap with respect to separating conjunction as well as the store with respect to disjoint union. For a store/heap pair, $(s, h)\,t = (s\,t, h\,t)$. Now assume stream interval functions $F\,s\,h\,x$ that evaluate the evolution of (s, h) within a given interval x. This yields three possible convolutions. The first is separating conjunction on the store, splitting s and leaving h and x unchanged; the second is separating conjunction on the heap, splitting h and leaving s and x unchanged; and the third is temporal chop, splitting interval x and leaving s and h unchanged. The function space with these three convolutions forms a tri-quantale with three monoidal operations as a generalisation of Proposition 10.1. Generalising from predicates to functions is straightforward when the target algebra Q is a tri-quantale. □

11. POWER SERIES OVER RESTRICTED SEMIGROUPS

This section adapts the power series approach to a case which is appropriate, for instance, for languages with finite and infinite words and for intervals which may be semi-infinite in the sense that they have no upper bounds. Such approaches are, for instance, relevant for total correctness reasoning, where termination cannot be assumed or for reactive (concurrent) systems, which may not terminate. We model these cases abstractly with special kinds of partial semigroups.

A *left-restricted semigroup* (*lr-semigroup*) is a structure $(S_1, S_2, D, \cdot)$ where $S_1 \cap S_2 = \emptyset$ and $(S_1 \cup S_2, D, \cdot)$ is a partial semigroup with $D = \{(x, y) \mid x \in S_1 \wedge y \in S_1 \cup S_2\}$ and where $x \cdot y \in S_1$ if and only if $x, y \in S_1$.

These conditions imply that $x \cdot y \in S_2$ if and only if $x \in S_1$ and $y \in S_2$. Henceforth we write $S = S_1 \cup S_2$.

In that case, for $f, g : S \to Q$, we define an extended convolution as

$$(f \cdot g)\,x = \sum_{x = y \cdot z} (f\,y) \cdot (g\,z) + \begin{cases} f\,x, & \text{if } x \in S_2, \\ 0, & \text{if } x \in S_1. \end{cases}$$

LEMMA 11.1. *Let $(S, \cdot)$ be a lr-semigroup. If Q is a (distributive) quantale, then Q^S is a (distributive) quantale with $\mathbb{0} : S \to Q$ not necessarily a right annihilator and left distributivity holding only for non-empty suprema.*

PROOF. We need to verify the laws involving convolution with our new multiplication. It suffices to consider the cases where $x \in S_2$; the others are covered by Theorem 3.4. For left distributivity we calculate, for $I \neq \emptyset$,

$$\begin{aligned}(f \cdot \sum_{i \in I} g_i)\, x &= f\, x + \sum_{x = y \cdot z} (f\, y \cdot \sum_{i \in I} (g_i\, z)) \\ &= (\sum_{i \in I} f\, x) + \sum_{i \in I} \sum_{x = y \cdot z} (f\, y \cdot g_i\, z) \\ &= \sum_{i \in I} (f\, x + \sum_{x = y \cdot z} (f\, y \cdot g_i\, z)) \\ &= (\sum_{i \in I} (f \cdot g_i))\, x.\end{aligned}$$

For $I = \emptyset$, however $(f \cdot \mathbb{0})\, x = f\, x$ if $x \in S_2$, hence in this case left distributivity fails.

For right distributivity, which is no longer opposition dual, we calculate for $x \in S_2$,

$$\begin{aligned}((\sum_{i \in I} f_i) \cdot g)\, x &= (\sum_{i \in I} f_i\, x) + \sum_{x = y \cdot z} ((\sum_{i \in I} f_i\, y) \cdot g\, z) \\ &= (\sum_{i \in I} f_i\, x) + \sum_{i \in I} \sum_{x = y \cdot z} (f_i\, y \cdot g\, z) \\ &= \sum_{i \in I} (f_i\, x + \sum_{x = y \cdot z} (f_i\, y \cdot g\, z)) \\ &= (\sum_{i \in I} (f_i \cdot g))\, x.\end{aligned}$$

Left annihilation is as usual a special case of right distributivity. We calculate for $x \in S_2$,

$$(\mathbb{0} \cdot f)\, x = \mathbb{0}\, x + \sum_{x = y \cdot z} \mathbb{0}\, y \cdot f\, z = 0 + 0 = 0.$$

Finally, for associativity, we calculate for $x \in S_2$,

$$\begin{aligned}(f \cdot (g \cdot h))\, x &= f\, x + \sum_{x = y \cdot z} (f\, y \cdot (g\, z + \sum_{z = u \cdot v} (g\, u \cdot h\, v))) \\ &= f\, x + (\sum_{x = y \cdot z} f\, y \cdot g\, z) + \sum_{x = y \cdot z} (f\, y \cdot (\sum_{z = u \cdot v} g\, u \cdot h\, v)) \\ &= (f \cdot g)\, x + \sum_{x = y \cdot u \cdot v} (f\, y \cdot g\, u \cdot h\, v) \\ &= (f \cdot g)\, x + \sum_{x = w \cdot v} ((\sum_{w = y \cdot u} f\, y \cdot g\, u) \cdot h\, v) \\ &= (f \cdot g)\, x + \sum_{x = w \cdot v} ((f \cdot g)\, w \cdot h\, v) \\ &= ((f \cdot g) \cdot h)\, x.\end{aligned}$$

The second last step uses the fact that $w \in S_1$ because $w \cdot v = x$ is defined. □

A treatment of right-restricted semigroups is dual and allows us to model, for instance, semi-infinite intervals without lower bounds. The properties of right-restricted semigroups are also dual, i.e., left annihilation fails.

Example 11.2 (*Formal Languages with Infinite Words*). Let X be a finite alphabet. Let X^* denote the set of finite words X and X^ω the set of infinite words over X. Let

$X^\infty = X^* \cup X^\omega$. Then $X^* \cap X^\omega = \emptyset$ by definition. Every language $L \subseteq X^\infty$ may contain finite as well as infinite words. It is natural to disallow the concatenation of an infinite word with another word, hence X^∞ is endowed with an lr-monoidal structure. Writing $\mathsf{fin}\, L$ for the finite and $\mathsf{inf}\, L$ for the infinite words in L, the product of $L_1, L_2 \subseteq X^\infty$ is commonly defined as

$$L_1 \cdot L_2 = \{vw \mid v \in \mathsf{fin}\, L_1 \wedge w \in L_2\} \cup \mathsf{inf}\ L_1.$$

This is captured by the above extended convolution with $Y = \mathbb{B}$. It then follows from Lemma 11.1 that X^∞ forms a distributive quantale in which $L \cdot \emptyset = \emptyset$ need not hold and left distributivity holds only for non-empty suprema. □

Example 11.3 (*Functions and Predicates over Unbounded Intervals*). Let $(P, \leq)$ be a linear order that is unbounded on the right. Let I_P^b stand for the set of all non-empty closed intervals over P and let I_P^u denote the set of all unbounded intervals $[a) = \{b \mid b \geq a\}$. Then $I_P = I_P^b \cup I_P^u$ and $I_P^b \cap I_P^u = \emptyset$ and fusion product of intervals is defined $D = \{(x, y) \in I_P^b \times I_P \mid \max x = \min y\}$ and $x \cdot y = x \cup y$. This turns I_P into an lr-semigroup and Lemma 11.1 implies that Q^{I_P} forms a distributive quantale in which $\mathbb{0}$ is not necessarily a right annihilator. □

Models with finite/infinite paths and traces can be built in a similar fashion; an example of closed and open intervals without fusion can be obtained along the same lines. Examples of bi-quantales based on stream functions over unbounded intervals with a notion of separating conjunction can be obtained in a straightforward way.

12. INTERCHANGE LAWS

Algebras in which a spatial or concurrent separation operation interacts with a temporal or sequential one have been studied, for instance, in the context of concurrent Kleene algebra by Hoare et al. [2011b]. In addition to the trioid or bi-quantale laws, these algebras provide interesting interaction laws between the two compositions.

More concretely, the following *interchange laws* hold in concurrent Kleene algebras:

$$x \cdot y \leq x * y, \tag{1}$$
$$(x * y) \cdot z \leq x * (y \cdot z), \tag{2}$$
$$x \cdot (y * z) \leq (x \cdot y) * z, \tag{3}$$
$$(w * x) \cdot (y * z) \leq (w \cdot y) * (x \cdot z). \tag{4}$$

These hold, for instance, in shuffle languages and certain classes of partially ordered multisets [Gischer 1988]. It has been shown that the interchange law (2) is equivalent to a separation logic style frame rule in a certain encoding of Hoare logic [Hoare et al. 2011a]. Interchange law (4), in turn, is equivalent to a concurrency rule for Hoare logic, which is similar to those considered by Hoare [1972; 1975], in Owicki and Gries' logic [Owicki and Gries 1976] or in concurrent separation logic [O'Hearn 2004]. This relationship is considered further in Section 13.

The close relationship between power series and separation logic and the similarity between two-dimensional power series and concurrent Kleene algebras make it worth considering the interchange laws in this setting. Two-dimensional power series are more general than standard concurrent Kleene algebra because the units of sequential and concurrent composition may be different. Here, we obtain negative results, refuting the interchange laws of concurrent Kleene algebra above.

PROPOSITION 12.1. *There are vector stream interval predicates $F, G, H, K : S_1 \to S_2 \to \mathbb{B}$ such that*

(a) $F \cdot G \not\leq F * G$,

(b) $(F * G) \cdot H \not\leq F * (G \cdot H)$,
(c) $F \cdot (G * H) \not\leq (F \cdot G) * H$,
(d) $(F * G) \cdot (H * K) \not\leq (F \cdot H) * (G \cdot K)$.

PROOF. Note that $\leq$ can be interpreted as $\Rightarrow$ for stream interval predicates, and recall the following two facts. First, parallel composition of predicates is separating conjunction when f is a vector of functions. Second, the two multiplicative unit predicates are different: that of $\cdot$ holds on all unit intervals $[a, a]$; that of $*$ holds on the all zero vector of dimension n. In the case of a joint unit, the interchange law (4) would imply the others. Here they require separate consideration.

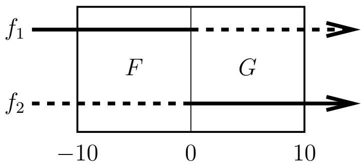
(a) To refute $F \cdot G \leq F * G$, let $x = [-10, 10]$, $f = (f_1, f_2)$ with

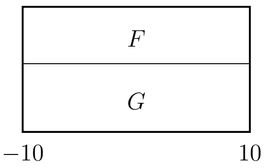
$$f_1\, t = \begin{cases} 1, & t \leq 0, \\ 0, & t > 0, \end{cases} \qquad f_2\, t = \begin{cases} 0, & t \leq 0, \\ 1, & t > 0, \end{cases}$$

and

$$F\, x\, f = \forall t \in x.\ f_1\, t = 1, \qquad G\, x\, f = \forall t \in x.\ f_2\, t = 1.$$

Then $(F \cdot G)\, x\, f = 1$, splitting interval x at $t = 0$, whereas $(F * G)\, x\, f = 0$ since neither F nor G holds on the entire interval x. This may be visualised using the diagrams below, where dashed lines represent that the corresponding function has value 0, and solid lines represent a value 1. For the right diagram, there is no possible way for the vectors f_1 and f_2 to go through F and G.

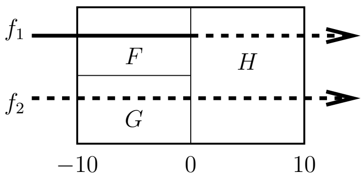


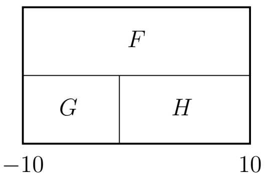


(b) To refute $(F * G) \cdot H \leq F * (G \cdot H)$, let $x = [-10, 10]$, f_1 as in (a) and $f_2 = \lambda t.\ 0$, where

$$\begin{aligned} F\, x\, f &= \forall t \in x.\ f_1\, t = 1, \\ G\, x\, f &= \forall t \in x.\ f_2\, t = 0, \\ H\, x\, f &= \forall t \in x.\ f_1\, t = 0 \vee f_2\, t = 0. \end{aligned}$$

This makes the left hand side 1 and the right hand side 0. This is visualised by the diagram below — f_1 may not go through F.





(c) $H \cdot (G * F) \leq (H \cdot G) * F$ can be refuted by function

$$f_1'\, t = \begin{cases} 0, & t \leq 0, \\ 1, & t > 0, \end{cases}$$

and f_2 as in (b), exploiting opposition duality between the two interchange laws and realising that f_1' is the "time reverse" of f_1.

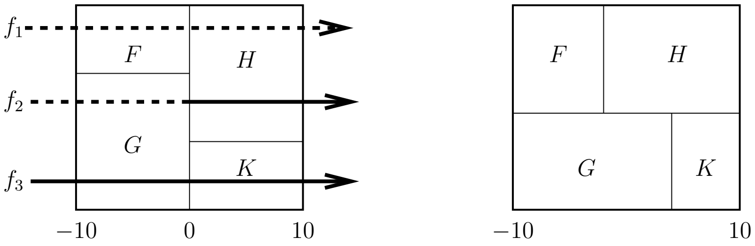
(d) To refute $(F * G) \cdot (H * K) \leq (F \cdot H) * (G \cdot K)$, consider $f = (f_1, f_2, f_3)$ where

$$f_1\,t = 0, \qquad f_2\,t = \begin{cases} 0, & t \leq 0, \\ 1, & t > 0, \end{cases} \qquad f_3\,t = 1$$

and

$$\begin{aligned} F\,f\,x &= \forall t \in x.\ f_1\,t = 0, \\ G\,f\,x &= \forall t \in x.\ f_2\,t < f_3\,t, \\ H\,f\,x &= \forall t \in x.\ f_1\,t < f_2\,t, \\ K\,f\,x &= \forall t \in x.\ f_3\,t = 1. \end{aligned}$$

For $x = [-10, 10]$, the diagram on the left below shows that $(F * G) \cdot (H * K)$ holds. However, in the diagram for $(F \cdot H) * (G \cdot K)$ on the right, there is no possible combination of horizontal and vertical splits that satisfy f. In particular, f_1 must go through F and f_3 must go through K. Function f_2 must either go through F and H, or through G and K, however, neither alternative evaluates to true.



□

The consideration of additional algebraic restrictions, which would allow the derivation of interchange laws, is left for future work. It is known that all interchange laws hold in the pomset model (Example 9.3), where the unit of sequential and concurrent composition is shared. Other promising directions are the consideration of locality assumptions, as in separation logic [O'Hearn 2004; Calcagno et al. 2007], which are briefly explained in the following section, or the addition of dependency relations [Hoare et al. 2011b] in the definition of the semigroup operations.

13. EXTENSIONS AND APPLICATIONS

This section outlines a number of extensions and applications of the power series framework. First we discuss how algebras of predicate transformers over boolean or quantale-based assertion algebras can be constructed in this setting. Second, we sketch how propositional Hoare logics, that is, Hoare logic without assignment axioms, can be obtained in a generic fashion. In particular, one can derive a propositional Hoare logic with the frame rule of separation logic. Finally, we outline some ramifications for deriving concurrency inference rules and comment on directions for future work with power series as semantics for linear logics and other substructural logics.

Predicate Transformer Quantales. The powerset lifting discussed in Section 3 suggests that state and predicate transformers can be modelled as power series as well. A state transformer $f_R\,a = \{b \mid (a, b) \in R\}$ of type $f_R : A \to 2^B$ is often associated with a relation $R \subseteq A \times B$. State transformers are turned into predicate transformers $\hat{f}_R : 2^B \to 2^A$ by the lifting $\hat{f}_R\,Y = \{x \mid f_R\,x \subseteq Y\}$.

State transformers in $(2^B)^A$ and the predicate transformers in $(2^A)^{2^B}$ form complete distributive lattices [Back and von Wright 1999]. This follows by applying Lemma 3.3

twice. First, $2^B \cong \mathbb{B}^B$ forms a complete distributive lattice because $\mathbb{B}$ forms a complete distributive lattice. The same argument applies to 2^A. By a second lifting, $(2^B)^A$ and $(2^A)^{2^B}$ form again complete distributive lattices. A quantale-like structure can be imposed by considering the monoidal operation of function composition; a full quantale is obtained by imposing continuity $f\,(\sum_{i\in I} X_i) = \sum_{i\in I}(f\,X_i)$ on all predicate transformers. Predicate transformers arise more abstractly as endofunctions over boolean algebras, similar to the boolean algebras with operators of Jónsson and Tarski [1951].

The approach can be adapted to predicate transformers over assertion quantales, as they occur in separation logic [Dongol et al. 2015]. In this setting, both the construction of assertion algebras from resource semigroups and of predicate transformer algebras over assertion quantales can be obtained from power series. Once more, the monoidal operation of the predicate transformer algebra is function composition, but not convolution. The whole approach has been implemented in Isabelle/HOL.

Iteration. Since quantales are complete lattices, least and greatest fixpoints of isotone functions exist. Moreover, due to their infinite distributivity laws, functions such as $\lambda\alpha.\ x+\alpha$, $\lambda\alpha.\ x\cdot\alpha$ or $\lambda\alpha.\ \alpha\cdot x$ are continuous, hence least fixpoints of combinations of these functions can be obtained by iteration from 0 to the first limit ordinal.

More specifically, the function $\varphi = \lambda\alpha.\ 1{+}x{\cdot}\alpha$ has the least fixpoint $x^* = \sum_{i\in\mathbb{N}} \varphi^i(0) = \sum_{i\in\mathbb{N}} x^i$. This notion of finite iteration is needed for deriving a while rule for a finite loop in a partial correctness setting. In a total correctness setting, a notion of possibly infinite iteration is preferable, which corresponds to the greatest fixpoint of φ. Infinite iteration is also useful for lr-monoids Section 11, for example, when reasoning about reactive systems, and Hoare rules for these can be developed.

Propositional Hoare logics. Equipped with the star in the quantale setting one can obtain propositional Hoare logics in two different ways.

First, following Hoare et al. [2011b] by adapting a previous approach by Tarlecki [1985], one can define validity of a Hoare triple as $\{x\}y\{z\} \Leftrightarrow x\cdot y \le z$ in an arbitrary quantale. The standard inference rules of Hoare logic except the assignment rule are then derivable. In our setting this yields a propositional Hoare logic for each example constructed. Instantiation to the binary relations quantale reproduces, for instance, Tarlecki's original soundness result. Other instances yield, in a generic way, Hoare logics over computationally meaningful semantics based on finite words (traces in the sense of concurrency theory), paths in graphs (sequences of events in concurrency theory), paths in the sense of automata theory, or pomsets. We also obtain generic propositional Hoare logics for reasoning about interval and stream interval predicates in algebraic variants of interval logics.

Second, validity of a Hoare triple can be defined as $\{p\}x\{q\} \Leftrightarrow p \le \hat{f}_x\,q$, using predicate transformers as usual. The rules of propositional Hoare logic can then be derived on the subspace of monotonic predicate transformers. In addition, the frame rule $\{p\}x\{q\} \;\Rightarrow\; \{p*r\}x\{q*r\}$ of separation logic is derivable on the subspace of local monotonic predicate transformers, i.e. monotonic predicate transformers satisfying $(f\,p)*q \;\le\; f\,(p*q)$. These transformers form distributive pre-quantales, in which the distributivity law $f\cdot\sum g_i = \sum(f\cdot g_i)$ is weakened to the monotonicity law $f \le g \Rightarrow h{\cdot}f \le h{\cdot}g$. The approach has been implemented in Isabelle/HOL and expanded into a verification tool for separation logic [Dongol et al. 2015].

Concurrency. For applications involving concurrency, such as the vector stream interval functions in Example 10.5, additional rules are desirable. In concurrent Kleene algebra, Owicki-Gries-style concurrency rules and frame rules in the style of separation logic can be derived. The same derivation, however, is ruled out in the general bi-quantale context, because the concurrency rule obtained would be equivalent to interchange law (4) and the frame rule to interchange law (2), both of which have been refuted in Proposition 12.1. Finding conditions under which such concurrency rules can be derived, is an important avenue of future work. In the case of pomset languages, for instance, the interchange law requires down-closure of languages, which roughly means adding pomsets with less parallelism. Intuitively, interchange law (4) holds in that case because $\leq$ is interpreted as set inclusion and the left-hand side of (4) admits less concurrency than its right-hand side.

Substructural and Linear Logics. O'Hearn and Pym's category-theoretic construction of the assertion quantale of separation logic is an instance of similar previous constructions of phase-space models of (intuitionistic) linear logic, which can be obtained by lifting certain symmetric monoidal categories to symmetric monoidal closed functor categories (cf. [Ambler 1991]). From an algebraic point of view, this amounts to a powerset lifting of certain generalised partial commutative monoids to quantales, which can be modelled by convolution.

Similar constructions for other substructural logics such as relevance logics are well known [Urquhart 1972; Meyer and Routley 1973; Allwein and Dunn 1993], but usually these have not been presented with convolution. In these constructions, our equational condition $x = y \cdot z$ on the partial monoid S in convolutions must be generalised to a ternary relation $R(x, y, z)$ for Kripke frames involving S. Extensions of our approach in these directions, both from an algebraic and a category-theoretic point of view, would further underpin its universality.

14. CONCLUSION

The aim of this article is to demonstrate that convolution is a versatile and interesting construction in mathematics and computer science. Used in the context of power series and integrated into lifting results, it yields a powerful tool for setting up various mathematical structures and computational models and calculi endowed with generic algebraic properties.

Beyond the language models known from formal language theory, these include assertion quantales of separation logic (which can be lifted from an underlying resource monoid), assertion quantales of interval logics (which can be lifted from an underlying semigroup of intervals) and stream interval functions (which have applications in the analysis of dynamic and real-time systems). For all these examples, the power series approach provides a simple new approach. For the latter two, new kinds of concurrency operations are provided.

In addition, the modelling framework based on power series can be combined with verification approaches by deriving, in generic fashion, propositional Hoare logics for virtually all examples considered. In particular, state and predicate transformers, which can be used for constructing these logics, arise as instances of power series.

This article focuses mainly on the proof of concept of the relevance of convolution. Many of the modelling examples and verification approaches featured require further investigation. This includes in particular the derivation of more comprehensive sets of Hoare-style inference rules for concurrency verification and interval temporal logics, the construction of algebraic counterparts of temporal and dynamic logics [Höfner et al. 2006] and more detailed case studies with interval and stream interval algebras, and with concurrent systems with infinite behaviours.

For all these case studies, the formalisation of the power series approach and the implementation of modelling tools plays an important role. In fact, the basic lifting lemma and a detailed predicate transformer approach based on power series have already been formalised within the Isabelle/HOL proof assistant [Nipkow et al. 2002].

The development of a power series based verification tool for separation logic has been the first step in the tool chain [Dongol et al. 2015].

ACKNOWLEDGMENT

The authors are indebted to anonymous referees for numerous suggestions that have helped us to improve this paper. We would like to thank Alasdair Armstrong, Kirill Bogdanov, John Derrick, Victor Gomes and Lindsay Groves for discussions of earlier versions.

REFERENCES

S. Abramsky and A. Jung. 1994. Domain Theory. In *Handbook of Logic in Computer Science*, S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum (Eds.). Vol. III. Oxford University Press.

J. Allwein and J. M. Dunn. 1993. Kripke Models for Linear Logic. *The Journal of Symbolic Logic* 58, 2 (1993), 514–545.

S. Ambler. 1991. *First Order Linear Logic in Symmetric Monoidal Closed Categories*. Ph.D. Dissertation. University of Edinburgh.

R.-J. Back and J. von Wright. 1994. Trace Refinement of Action Systems. In *CONCUR (LNCS)*, B. Jonsson and J. Parrow (Eds.), Vol. 836. Springer, 367–384.

R.-J. Back and J. von Wright. 1999. *Refinement Calculus - A Systematic Introduction*. Springer.

V. Bergelson, A. Blass, and N. Hindman. 1994. Partition Theorems for Spaces of Variable Words. In *Proc. London Math. Soc. 68*. 449–476.

J. Berstel and C. Reutenauer. 1984. *Les Séries Rationnelles et Leurs Langagues*. Masson.

C. Brink. 1993. Power Structures. *Algebra Universalis* 30 (1993), 177–216.

C. Calcagno, P. W. O'Hearn, and H. Yang. 2007. Local Action and Abstract Separation Logic. In *LICS*. IEEE Computer Society, 366–378.

J. H. Conway. 1971. *Regular Algebra and Finite Machines*. Chapman and Hall.

B. Day. 1970. On Closed Categories of Functors. In *Reports of the Midwest Category Seminar IV (Lecture Notes in Mathematics)*, Vol. 137. Springer, 1–38.

T. Dinsdale-Young, L. Birkedal, P. Gardner, M. J. Parkinson, and H. Yang. 2013. Views: Compositional Reasoning for Concurrent Programs. In *POPL*, R. Giacobazzi and R. Cousot (Eds.). ACM, 287–300.

B. Dongol, V. B. F. Gomes, and G. Struth. 2015. A Program Construction and Verification Tool for Separation Logic. In *MPC (LNCS)*, R. Hinze and J. Voigtländer (Eds.), Vol. 9129. Springer, 137–158.

B. Dongol, I. J. Hayes, and J. Derrick. 2014a. Deriving Real-Time Action Systems with Multiple Time Bands using Algebraic Reasoning. *Sci. Comput. Program.* 85 (2014), 137–165.

B. Dongol, I. J. Hayes, and G. Struth. 2014b. Convolution, Separation and Concurrency. *CoRR* abs/1410.4235 (2014).

M. Droste, W. Kuich, and H. Vogler (Eds.). 2009. *Handbook of Weighted Automata*. Springer.

S. Eilenberg. 1974. *Automata, Languages and Machines*. Vol. A. Academic Press.

J. L. Gischer. 1988. The Equational Theory of Pomsets. *Theoretical Computer Science* 61 (1988), 199–224.

R. Goldblatt. 1989. Varieties of Complex Algebras. *Annals of Pure and Applied Logic* 44 (1989), 173–242.

J. Grabowski. 1981. On Partial Languages. *Fundamentae Informaticae* 4 (1981), 427–498.

T. A. Henzinger. 1996. The Theory of Hybrid Automata. In *LICS*. IEEE Computer Society, 278–292.

C.A.R. Hoare. 1972. Towards a Theory of Parallel Programming. In *Operating System Techniques*. Academic Press, 61–71.

C.A.R. Hoare. 1975. Parallel Programming: An Axiomatic Approach. *Computer Languages* 1, 2 (1975), 151–160.

C. A. R. Hoare, A. Hussain, B. Möller, P. W. O'Hearn, R. Lerchedahl Petersen, and G. Struth. 2011a. On Locality and the Exchange Law for Concurrent Processes. In *CONCUR (LNCS)*, J.-P. Katoen and B. König (Eds.), Vol. 6901. Springer, 250–264.

T. Hoare, B. Möller, G. Struth, and I. Wehrman. 2011b. Concurrent Kleene Algebra and its Foundations. *J. Log. Algebr. Program.* 80, 6 (2011), 266–296.

P. Höfner and B. Möller. 2009. An Algebra of Hybrid Systems. *J. Log. Algebr. Program.* 78, 2 (2009), 74–97.

P. Höfner, B. Möller, and G. Struth. 2006. Quantales and Temporal Logics. In *AMAST (LNCS)*, M. Johnson and V. Vene (Eds.), Vol. 4019. Springer, 263–277.

S. S. Ishtiaq and P. W. O'Hearn. 2001. BI as an Assertion Language for Mutable Data Structures. In *POPL*, C. Hankin and D. Schmidt (Eds.). ACM, 14–26.

B. Jónsson and A. Tarski. 1951. Boolean Algebras with Operators, Part I. *Americal Journal of Mathematics* 73, 4 (1951), 207–215.

B. Jónsson and A. Tarski. 1952. Boolean Algebras with Operators, Part II. *American Journal of Mathematics* 74, 1 (1952), 127–162.

G. M. Kelly. 1982. Basic Concepts of Enriched Category Theory. *LMS Lecture Notes Series* 64 (1982).

R. K. Meyer and R. Routley. 1973. Classical Relevant Logic (I). *Studia Logica* 32 (1973), 51–66.

B. C. Moszkowski. 2000. A Complete Axiomatization of Interval Temporal Logic with Infinite Time. In *LICS*. IEEE Computer Society, 241–252.

T. Nipkow, L. C. Paulson, and M. Wenzel. 2002. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. LNCS, Vol. 2283. Springer.

P. W. O'Hearn. 2004. Resources, Concurrency and Local Reasoning. In *CONCUR (LNCS)*, P. Gardner and N. Yoshida (Eds.), Vol. 3170. Springer, 49–67.

P. W. O'Hearn and D. J. Pym. 1999. The Logic of Bunched Implications. *Bulletin of Symbolic Logic* 5, 2 (1999), 215–244.

S. S. Owicki and D. Gries. 1976. Verifying Properties of Parallel Programs: An Axiomatic Approach. *Commun. ACM* 19, 5 (1976), 279–285.

J. C. Reynolds. 2002. Separation Logic: A Logic for Shared Mutable Data Structures. In *LICS*. IEEE Computer Society, 55–74.

G.-C. Rota. 1964. On the Foundations of Combinatorial Theory I. Theory of Möbius Functions. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 2, 4 (1964), 340–368.

A. Tarlecki. 1985. A Language of Specified Programs. *Science of Computer Programming* 5 (1985), 59–81.

A. Urquhart. 1972. Semantics for Relevant Logics. *The Journal of Symbolic Logic* 37, 1 (1972), 159–169.

C. Zhou and M. R. Hansen. 2004. *Duration Calculus - A Formal Approach to Real-Time Systems*. Springer.