False Failure:
Creating Failure Models for Separation Logic

Callum Bannister[1,2] and Peter Höfner[1,2]

[1] Data61, CSIRO, Sydney, Australia
[2] Comput. Sci. and Engineering, University of New South Wales, Sydney, Australia

Abstract. Separation logic, an extension of Floyd-Hoare logic, finds countless applications in areas of program verification, but does not allow forward reasoning in the setting of total or generalised correctness. To support forward reasoning, separation logic needs to be equiped with a failure element. We present several ways on how to add such an element. We show that none of the 'obvious' extensions preserve all the algebraic properties desired. We develop more complicated models, satisfying the desired properties, and discuss their use for forward reasoning.

1 Introduction

Some of the most prominent methods for formal reasoning about the correctness of programs are Floyd-Hoare logic [13,14], Dijkstra's weakest-precondition calculus [10], and strongest postconditions [13]. The usefulness and importance of these approaches are undeniable and they have been used in the area of formal verification countless times.

However, a shortcoming of techniques based on Hoare logic is that they lack expressiveness for shared mutable data structures, such as structures where updatable fields can be referenced from more than one point. To overcome this deficiency, Reynolds, O'Hearn and others developed separation logic [27,30]. It extends Hoare logic by separating conjunctions, and adds assertions to express separation between memory regions, which allows for local reasoning by splitting memory into two halves: the part the program interacts with, and the part which remains untouched, called the *frame*. Later, O'Hearn extended this language to concurrent programs that work on shared mutable data structures [26].

It is known that sets of assertions, in combination with separating conjunction form a quantale [8]. Quantales [24,31], sometimes called standard Kleene algebras [6], or the more general concept of semirings have been used to derive algebraic characterisations for Hoare logic [19,22] and the wp-calculus of Dijkstra [23]. In the quantale of assertions all operations of separation logic, such as separating conjunction and separating implication are related via Galois connections and dualities [8,1]. Many useful theorems about separation logic follow 'for free' from the underlying algebraic theory.

Separation logic has been used for reasoning in weakest-precondition style [10], which proceeds backwards from a given postcondition and a given program by determining the weakest precondition [1]. It has also been used in the setting of forward reasoning, using strongest postconditions. However, in the latter setting only partial correctness can be considered, as separation logic does not contain formulae corresponding to the *failed* execution of programs. Both forward and backward reasoning in separation logic heavily rely on the Galois connections and dualities mentioned above.

In this paper we discuss how to extend the assertion quantale of separation logic to handle failed executions, bearing the application of forward reasoning in mind. The created models should maintain as much of the algebraic structure as possible, which would allow us to reuse knowledge from the original separation logic and from the algebraic meta-theory. We show that all *simple* models fail to satisfy all algebraic properties we hope for: some lose associativity of separating conjunction, while others maintain associativity but do not relate the operators via Galois connections. As our search for a good model is systematic, we conclude that there cannot be a simple, but powerful model for separation logic, which features failure and has 'nice' algebraic properties at the same time. Our final models, inspired by the construction of integers from natural numbers, achieve the desired properties. The cost is splitting separating implication from its dual. Although this looks like an acceptable trade-off, we conclude that the new model is not suitable for forward reasoning either as it leads to undesirable behaviour.

2 Algebraic Separation Logic

Assertions are a crucial ingredient in separation logic. They describe states consisting of a store and a heap, roughly corresponding to the state of local variables and dynamically-allocated objects. They are used as predicates to describe the contents of heaps and stores, and as pre- or postconditions of programs, as in Hoare logic. The semantics of the assertion language is given by the relation $s, h \models p$ of *satisfaction*.[3] Informally, $s, h \models p$ holds if the state (s, h) satisfies the assertion p. A *state* (s, h) contains a *store* $s : V \rightharpoonup \mathit{Values}$ – a partial function from the set V of all variables into a set of *Values*[4] – and a heap $h : \mathit{Addr} \rightharpoonup \mathit{Values}$, which maps an arbitrary set of (heap) addresses to values. An assertion p is called *valid* (or *pure*) iff p holds in every state and p is *satisfiable* if there exists a state (s, h) that satisfies p.

We denote the set of all heaps by *Heaps*, and the set of all states by *States*.

The semantics of assertions is defined inductively as follows (e.g. [30]).

$$\begin{array}{lll}
s,h \models b & \Leftrightarrow_{df} & b^s = \mathsf{true} \\
s,h \models \neg p & \Leftrightarrow_{df} & s,h \not\models p \\
s,h \models p \vee q & \Leftrightarrow_{df} & s,h \models p \quad \text{or} \quad s,h \models q \\
s,h \models \forall v.\ p & \Leftrightarrow_{df} & \forall x \in \mathit{Values} : \{(v,x)\} \mid s,h \models p \\
s,h \models \mathsf{emp} & \Leftrightarrow_{df} & h = \emptyset \\
s,h \models e_1 \mapsto e_2 & \Leftrightarrow_{df} & h = \{(\, e_1^s\,,\, e_2^s\,)\} \\
s,h \models p * q & \Leftrightarrow_{df} & \exists h_1,h_2 \in \mathit{Heaps}.\ \mathsf{dom}(h_1) \cap \mathsf{dom}(h_2) = \emptyset \text{ and} \\
& & \qquad h = h_1 \cup h_2 \text{ and } s,h_1 \models p \text{ and } s,h_2 \models q
\end{array}$$

[3] We introduce its syntax implicitly; see e.g. [30] for an explicit definition.

[4] Often one assumes $\mathit{Values} = \mathbb{Z}$.

Here, b is a Boolean, and e_1 and e_2 are arbitrary expressions; p, q are assertions. The semantics e^s of an expression e with regards to a store s is straightforward. The domain of a relation modelling a partial function R is defined by $\mathsf{dom}(R) =_{df} \{x : \exists y.\ (x,y) \in R\}$, and the update function $|$ is defined by $f \,|\, g =_{df} f \cup \{(x,y) : (x,y) \in g \wedge x \notin \mathsf{dom}(f)\}$.

The first four clauses do not make any assumptions about the heap and are well known. The fifth clause defines the assertion emp, which ensures that the heap h is empty and does not contain any addressable cell. The assertion $e_1 \mapsto e_2$ characterises the heap of a state to contain exactly one cell at address e_1^s and value e_2^s. More complex heaps are built using *separating conjunction* $*$; it is a connective that ensures properties on disjoint regions of the underlying heap.

To create a denotational model one lifts the satisfaction-based semantics to a set-based one:

$$[\![p]\!] =_{df} \{(s,h) : s,h \models p\} \ .$$

In particular, $[\![\mathsf{false}]\!] = \emptyset$. This definition offers a set-based semantics [8].

$$\begin{array}{ll}
[\![\neg p]\!] & = \{(s,h) : s,h \not\models p\} = \overline{[\![p]\!]} \\
[\![p \vee q]\!] & = [\![p]\!] \cup [\![q]\!] \\
[\![\forall v.\ p]\!] & = \bigcap_{x \in Values} \{(s,h) : ((v,x) \,|\, s,h) \in [\![p]\!]\} \\
[\![\mathsf{emp}]\!] & = \{(s,h) : h = \emptyset\} \\
[\![e_1 \mapsto e_2]\!] & = \{(s,h) : h = \{(e_1^s, e_2^s)\}\} \\
[\![p * q]\!] & = [\![p]\!] \mathbin{\cup\!\!\!\!*} [\![q]\!] \ , \\
 & \quad \text{with } P \mathbin{\cup\!\!\!\!*} Q =_{df} \{(s, h \cup h') : (s,h) \in P \wedge (s,h') \in Q \\
 & \qquad\qquad\qquad\qquad \wedge\ \mathsf{dom}(h) \cap \mathsf{dom}(h') = \emptyset\}
\end{array}$$

Here, $\overline{}$ denotes set complementation with respect to the carrier set *States*. It has been shown that set union in combination with set complementation and lifted separating conjunction forms a useful algebraic structure.

A *quantale* [24,31] is a structure $(S, \leq, 0, \cdot, 1)$ where $(S, \leq)$ is a complete lattice and $(S, \cdot, 1)$ is a monoid where multiplication is completely disjunctive, i.e.,

$$a \cdot (\bigsqcup T) = \bigsqcup\{a \cdot x : x \in T\} \quad \text{and} \quad (\bigsqcup T) \cdot a = \bigsqcup\{x \cdot a : x \in T\} \ ,$$

for $T \subseteq S$ and $\bigsqcup$ denoting the supremum operator. The least element is 0.

The supremum of two elements a, b is denoted by $a \sqcup b$, and relates to the order by $a \sqcup b = b \Leftrightarrow a \leq b$. The definition implies that $\cdot$ is a *full annihilator* (strict), i.e., that $0 \cdot a = 0$ and $a \cdot 0 = 0$ for all $a \in S$. The notion of a quantale is equivalent to that of a *standard Kleene algebra* [6].

A quantale is *commutative* if $a \cdot b = b \cdot a$, for all $a, b \in S$; it is called *Boolean* if the underlying lattice is distributive and complemented, hence a Boolean algebra.

Classical examples are the algebra of Booleans $\mathtt{BA} = (\mathbb{B}, \Rightarrow, \mathsf{false}, \wedge, \mathsf{true})$, and binary relations $\mathtt{REL}_X = (\mathcal{P}(X \times X), \subseteq, \emptyset, ;, \mathrm{I})$, where ; denotes sequential composition, and $\mathrm{I} = \{(x,x) : x \in X\}$ is the identity relation.

We now relate sets of assertions of separation logic to algebra.

Theorem 1. [8] The structure $\mathtt{AS} =_{df} (\mathcal{P}(States), \subseteq, [\![\mathsf{false}]\!], \mathbin{\cup\!\!\!\!*}, [\![\mathsf{emp}]\!])$ of separation logic assertions is a commutative and Boolean quantale with $P \sqcup Q = P \cup Q$.

In its early days separation logic was based on intuitionistic logic [29]. Hence the underlying algebra, called BI algebra, was based on Heyting algebras, rather than Boolean algebras [28]. Ishtiaq and O'Hearn expanded BI algebras to capture the Boolean nature of contemporary separation logic [17]. Their approach, called Boolean BI algebra, does not require an underlying order. A more detailed comparison between these approaches is given in [7]. As we require an ordering for our verification technique – see below – we work in the setting of quantales.

Separation logic features three additional operators: separating implication (e.g. [30]), septraction [32], and separating coimplication [1].

$$\begin{aligned}
s,h \models p \mathrel{-\!\!*} q \;&\Leftrightarrow_{df}\; \forall h_1 \in \mathit{Heaps} : (\mathsf{dom}(h_1) \cap \mathsf{dom}(h) = \emptyset \text{ and } s,h_1 \models p)\\
&\qquad \text{implies } s,h_1 \cup h \models q\\
s,h \models p \mathrel{-\!\circledast} q \;&\Leftrightarrow_{df}\; \exists h_1 \in \mathit{Heaps} : \mathsf{dom}(h_1) \cap \mathsf{dom}(h) = \emptyset \text{ and}\\
&\qquad s,h_1 \models p \text{ and } s,h \cup h_1 \models q\\
s,h \models p \rightsquigarrow\!\!* \, q \;&\Leftrightarrow_{df}\; \forall h_1,h_2 \in \mathit{Heaps} : (\mathsf{dom}(h_1) \cap \mathsf{dom}(h_2) = \emptyset \text{ and}\\
&\qquad h = h_1 \cup h_2 \text{ and } s,h_1 \models p) \text{ implies } s,h_2 \models q
\end{aligned}$$

A state (s,h) satisfies the *separating implication* $p \mathrel{-\!\!*} q$ if h ensures that whenever it is extended with a disjoint heap h_1 satisfying p then the combined heap $h \cup h_1$ satisfies q. *Septraction* ($-\!\circledast$) denotes an existential version of separating implication, which quantifies over all subheaps h_1; it expresses that the heap can be extended with a state satisfying p, so that the extended state satisfies q. *Separating coimplication* ($\rightsquigarrow\!\!*$) states that whenever there is a subheap h_1 satisfying p then the remaining heap satisfies q. It is straightforward to lift these operations to the algebra AS.

Algebraically these operators are related. In a quantale, the *left residual* $a\backslash b$ and the *right residual* a/b exist [2] and are defined by the Galois connections

$$x \le a\backslash b \Leftrightarrow_{df} a \cdot x \le b \quad \text{and} \quad x \le a/b \Leftrightarrow_{df} x \cdot b \le a\ .$$

The element $a\backslash b$ is a pseudo-inverse of multiplication and the greatest solution of the inequality $a \cdot x \le b$. Hence $a\backslash b$ can also be defined as $\bigsqcup\{x : a \cdot x \le b\}$. In case the underlying quantale is commutative both residuals coincide, i.e., $a\backslash b = b/a$. In a Boolean quantale, the *left detachment* $a\rfloor b$ and the *right detachment* $a\lfloor b$ are defined based on residuals.

$$a\rfloor b =_{df} \overline{a\backslash\overline{b}} \quad \text{and} \quad a\lfloor b =_{df} \overline{\overline{a}/b}$$

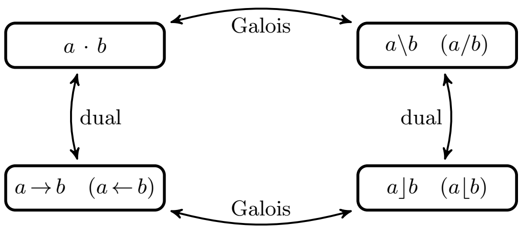
In BA residuals coincide with implication, and detachments with conjunction. In $\mathtt{REL}_X$, $R\backslash S = \overline{R^{\smile};\overline{S}}$ and $R\rfloor S = R^{\smile};S$, where $^{\smile}$ denotes the converse of a relation.

Theorem 2. [8] In the algebra of assertions AS, separating implication is the residual of separating conjunction; septraction coincides with the detachment.

$$[\![\, p \mathrel{-\!\!*} q \,]\!] = [\![\, p \,]\!]\backslash[\![\, q \,]\!] = [\![\, q \,]\!]/[\![\, p \,]\!] \quad \text{and} \quad [\![\, p \mathrel{-\!\circledast} q \,]\!] = [\![\, p \,]\!]\rfloor[\![\, q \,]\!] = [\![\, q \,]\!]\lfloor[\![\, p \,]\!]$$

The algebraic theory of quantales is well established, and implies plenty of properties for separation logic. Examples are monotonicity properties of the operators, modus ponens, as well as the following exchange law.

$$\begin{gathered}
a \cdot \overline{b} \le \overline{c} \;\Leftrightarrow\; a\rfloor c \le b\\
[\![\, p \,]\!] \uplus \overline{[\![\, q \,]\!]} \subseteq \overline{[\![\, r \,]\!]} \;\Leftrightarrow\; [\![\, p \mathrel{-\!\circledast} r \,]\!] \subseteq [\![\, q \,]\!]\\
(p * \neg q \Rightarrow \neg r) \;\Leftrightarrow\; (p \mathrel{-\!\circledast} r \Rightarrow q)
\end{gathered}$$



$$\begin{tikzcd}
\boxed{a \cdot b} \arrow[rr, leftrightarrow, bend left, "\text{Galois}"] \arrow[d, leftrightarrow, bend right, "\text{dual}"] & & \boxed{a\backslash b \quad (a/b)} \arrow[d, leftrightarrow, bend left, "\text{dual}"'] \\
\boxed{a \rightarrow b \quad (a \leftarrow b)} \arrow[rr, leftrightarrow, bend right, "\text{Galois}"'] & & \boxed{a \rfloor b \quad (a \lfloor b)}
\end{tikzcd}$$

Fig. 1. Relationship between operators of quantales

The first equivalence shows the law in the general setting of Boolean quantales, the second one in AS, and the last one the corresponding law in separation logic. It is standard that in algebraic settings the order coincides with implication. In this paper we use these representations interchangeably, depending on the current situation. Many more properties about quantales can be found in the literature; many useful properties about residuals and detachments are summarised in [21].

The exchange law implies another Galois connection based on detachments:

$$b \le a \rightarrow c \quad \Leftrightarrow \quad a \rfloor b \le c \ ,$$

where $a \rightarrow c \ =_{df} \ \overline{a \cdot \overline{c}}$. As for residuals and detachments, a symmetric operator exists: $a \leftarrow c \ =_{df} \ \overline{\overline{a} \cdot c}$. In sum, any Boolean quantale features four operators (and their symmetric ones), which are related via dualities and Galois connections; Figure 1 summarises the situation.

Theorem 3. [1] In AS, separating coimplication is upper adjoint of septraction.

$$[\![\, p \leadsto\!\!\ast\ q \,]\!] = [\![\, p \,]\!] \rightarrow [\![\, q \,]\!] = \overline{[\![\, p \,]\!] \circledast \overline{[\![\, q \,]\!]}} \quad \text{and} \quad [\![\, q \,]\!] \subseteq [\![\, p \leadsto\!\!\ast\ r \,]\!] \Leftrightarrow [\![\, p \multimap\!\!\!\circledast\ q \,]\!] \subseteq [\![\, r \,]\!]$$

3 Forward and Backward Reasoning in Separation Logic

The Galois connections are extremely useful when reasoning forwards and backwards in separation logic [1]. Since we aim at an extension of forward reasoning, including the possibility of failure, we briefly explain the method in this section.

Forward reasoning [13] proceeds forwards from a given precondition P and a given program m by calculating the *strongest postcondition* $\mathtt{sp}(P, m)$ such that $\{P\}\, m\, \{\mathtt{sp}(P, m)\}$ is a valid Hoare triple. *Backward reasoning* [10] proceeds the other way round and determines the *weakest precondition* $\mathtt{wp}(m, Q)$, for a given program m and postcondition Q.

Although backward reasoning is more common when it comes to verification efforts, there are several applications where forward reasoning is more convenient, such as for programs where the precondition is 'trivial', and the postcondition either too complex or unknown. Both reasoning techniques are well established for Hoare logic, and calculational reasoning is feasible. For example, the strongest postcondition for $\mathtt{sp}(P, m_1; m_2)$ for a sequential program equals $\mathtt{sp}(\mathtt{sp}(P, m_1), m_2)$.

Using separation logic in the context of forward and backward reasoning yield the problem commonly known as *frame calculation*. In separation logic any given specification $\{P\}\, m\, \{Q\}$, can be extended to $\{P * R\}\, m\, \{Q * R\}$, where R is a frame – a region of the memory that remains untouched during execution of m. Forward reasoning starts with a given precondition X, which then needs to be split into the actual precondition P and a frame R. That means for given X and P one has to find a frame R such that $X = P * R$. In large projects, frame calculations are usually challenging since X can be arbitrarily complex.

For forward reasoning the Galois connection between septraction and separating coimplication comes to aid.

$$\forall R.\, \{P \rightsquigarrow\!\!\!* \, R\}\, m\, \{Q * R\} \;\Leftrightarrow\; \forall X.\, \{X\}\, m\, \{Q * (P \mathbin{-\!\circledast} X)\}$$

The right hand side has the advantage that it works for *arbitrary* preconditions X and does not require an explicit calculation of the frame. Intuitively, it states that the postcondition is given by X, pulling out the precondition P and replacing it by Q. We note that septraction plays a crucial role here.

Specifications $\{P \, * \, R\}\, m\, \{Q \, * \, R\}$ can almost always be rewritten into $\{P \rightsquigarrow\!\!\!* R\}\, m\, \{Q \, * \, R\}$, especially if P is precise – preciseness ensures the existence of a unique subheap, i.e., $p \mapsto v$. A similar equivalence that can be used for backward reasoning exists, and is based on the Galois connection between $*$ and $-\!\!*$. Both techniques have been implemented in the theorem prover Isabelle/HOL [25] and are ready to be used for verification tasks. [1]

4 Simple Models for Separation Logic

While backward reasoning works for both partial and total correctness, forward reasoning is more useful in the setting of partial correctness, where termination has to be proved separately. In total correctness, the existence of a postcondition implies termination, and therefore, it is not guaranteed that forward reasoning can proceed from a given precondition, limiting its application. We target *general correctness* [18], where the postcondition can handle both termination and nontermination. This is in contrast to partial correctness, where failure coincides with false, and total correctness where failure is unrepresentable.

We extend the assertion language by some notion of failure. We explore different ways to add failure to separation logic in this and the next section.

Ideally we can develop a failure model maintaining the algebraic relationships between the operators depicted in Figure 1. In this section we present a series of models, all lacking at least one algebraic property. As our development is systematic, we conclude that no simple failure model exists.

A Single Failure Element

We start by extending our model by a single element $\bot$ indicating failure. Within separation logic that means we are looking at the set $\mathcal{P}(\mathit{States}) \cup \{\bot\}$; or in the abstract algebraic setting at $S \cup \{\bot\}$ as underlying set.

Two choices need to be made: extend multiplication, and integrate $\bot$ into the lattice of elements.

Model I: *Near-annihilation and largest element.* It seems reasonable to assume that failure cannot be erased by any non-zero element (in AS by any assertion different to $[\![\mathsf{false}]\!]$), i.e., $a \cdot \bot = \bot \cdot a = \bot$, for all $a \in (S \cup \{\bot\}) - \{0\}$. The interaction between $\bot$ and 0 ($[\![\mathsf{false}]\!]$ in AS) needs to be decided independently. Let us assume that zero cancels out failure: $0 \cdot \bot = \bot \cdot 0 = 0$. Moreover, let $\bot$ be the largest element in the underlying lattice: $a \leq \bot$, for all a.

If there are elements that cancel each other, i.e., $\exists a, b \in S.\, a \cdot b = 0$, we have

$$(a \cdot b) \cdot \bot = 0 \cdot \bot = 0 \quad \text{and} \quad a \cdot (b \cdot \bot) = a \cdot \bot = \bot\ .$$

Therefore either multiplication is not associative, and hence the underlying multiplicative structure $(S \cup \{\bot\}, \cdot, 1)$ is not even a monoid, or the algebra collapses: $\bot = 0 \leq a \leq \bot$. Separation logic features cancellative elements such as $(p \mapsto v) * (p \mapsto v) = \mathsf{false}$, where $p \mapsto v$ characterises a pointer p pointing to value v. Hence separating conjunction cannot be associative if a failure element with the above properties is present. ▲

Loosing associativity is not a priori bad (e.g. [11,9]), and we can still use models without associativity for reasoning in separation logic. We discuss this in Section 6. However, many useful properties are lost. For example the law $(a \cdot b) \rfloor c = a \rfloor (b \rfloor c)$, which holds in commutative quantales, does not hold without associativity. In separation logic this translates to $(p * q) \mathrel{-\!\circledast} r = p \mathrel{-\!\circledast} (q \mathrel{-\!\circledast} r)$ and states that a heap can be 'removed' by removing subheaps consecutively.

Model II: *Near-annihilation and least element.* When using the same set up as in Model I, but forcing $\bot$ to be the least element, i.e., $\bot \leq a$, the problem with associativity stays, but the algebra does not collapse any longer when associativity is enforced – one does not have $\bot \leq a \leq \bot$. However, we get $\bot = 0$ when assuming associativity and the existence of elements that cancel each other. ▲

Model III: *Full annihilation and largest element.* Since near-annihilation of failure ($\bot \cdot a = a \cdot \bot = \bot$ for $a \neq 0$) yields problems with associativity, we now focus on situations where failure is a (full) annihilator: $a \cdot \bot = \bot \cdot a = \bot$, for all $a \in S \cup \{\bot\}$. As a consequence, 0 does not fully annihilate any longer, as we only have $a \cdot 0 = 0 \cdot a = 0$ for $a \neq \bot$. For this model we assume that $\bot$ is the largest element. While the resulting algebra is still a complete lattice[5], it does not form a quantale as multiplication is only positively distributive, i.e.,

$$T \neq \emptyset \ \Rightarrow\ (a \cdot (\bigsqcup T) = \bigsqcup\{a \cdot x : x \in T\} \ \text{ and } \ (\bigsqcup T) \cdot a = \bigsqcup\{x \cdot a : x \in T\})\ .$$

For $T = \emptyset$ we have $\bot \cdot \bigsqcup \emptyset = \bot \cdot 0 = \bot$ and $\bigsqcup\{a \cdot x : x \in \emptyset\} = \bigsqcup \emptyset = 0$. While one could still define residuals as suprema, the Galois connections between multiplication and residuals do not hold any longer. Since these connections are crucial for backward reasoning (see Section 3), this model is of no use for us. ▲

Model IV: *Full annihilation and least element.* Our final model featuring a single failure element defines failure as least element of the lattice and as full annihilator. By straightforward calculations it can be shown that this algebra forms indeed a quantale. In particular, multiplication is associative and completely

[5] We omit straightforward proofs; most of them are available online (see Section 6).

distributive. Therefore this model is the first one which features an associative multiplication and establishes Galois connections. ▲

However, it is impossible to extend a Boolean quantale $(S, \leq, 0, \cdot, 1)$ to a Boolean quantale $(S \cup \{\bot\}, \sqsubseteq, \bot, \circ, 1)$ in this setting, if $\bot \notin S$: as Boolean algebras have size 2^n (for $n \in \mathbb{N}$) and $2^n + 1$ is not a power of 2, no complementation can be defined on the extended structure.

While this model is still a fine failure model for separation logic (without complementation), which can probably be used in many circumstances, it is not suitable for forward reasoning. As we want to use it in combination with reasoning in Hoare logic, we have to maintain fundamental aspects of this logic such, as the weakening rule

$$\frac{\{P\}\, m\, \{Q\} \qquad Q \leq Q'}{\{P\}\, m\, \{Q'\}} \ .$$

As mentioned above, $\leq$ coincides with logical implication and hence $Q \leq Q'$ describes the fact that the precondition Q' is weaker than Q. This rule in combination with the fact that $\bot$ is the least element yields false conclusions. Assume the program $\texttt{set\_ptr}\ p\ v$, which assigns value v to pointer p. If the heap does not contain the pointer, for example when the heap is empty, then the program fails. We would like to have the Hoare triple $\{\mathsf{emp}\}\, \texttt{set\_ptr}\ p\ v\, \{\bot\}$ to be valid. Using the weakening rule we can replace the postcondition by any other, as $\bot$ is the least element. Using the weakening rule conclude that $\{\mathsf{emp}\}\, \texttt{set\_ptr}\ p\ v\, \{q \mapsto 7\}$ or $\{\mathsf{emp}\}\, \texttt{set\_ptr}\ p\ v\, \{\mathsf{true}\}$ hold. Both are invalid Hoare triples under general correctness.

Therefore, in a setting where separation logic is used for forward or backward reasoning the element representing false ($\emptyset = [\![\mathsf{false}]\!]$ in $\mathtt{AS}$) needs to be the least element of any order to be used with Hoare logic.

The above four models conclude our failure models for separation logic featuring a single failure element; other models are not useful as it is not realistic that non-zero elements (proper heaps in $\mathtt{AS}$) cancel out failure. It also does not seem plausible to have the failure element sitting at some place in the lattice that is not at the bottom or the top.

Sets of Failure and Non-Failure Elements

We have shown that a failure element cannot be the least element w.r.t. the order underlying Hoare logic. Moreover, failure should be an annihilator in case elements exist that cancel each other out – otherwise associativity is lost.

Since a single failure yields severe shortcomings we now look at models based on subsets of $\mathit{States}' = \mathit{States} \cup \{\bot\}$. The intuition behind these models is that failure does not forget about the heap setting, but combines failure with possible heaps. It can be seen as introducing a flag indicating whether a calculation has failed or not; since there is only one flag, any calculation that *may* fail will have the failure flag set (non-failure executions may exist). This setting allows more flexibility when it comes to defining multiplication.

For the following two models we use the following separating conjunction.

$$P *_1 Q \ =_{df}\ ((P - \{\bot\}) \uplus (Q - \{\bot\})) \ \cup\ ((P \cup Q) \cap \{\bot\})$$

The first part calculates 'classical' separating conjunction on the non-failure parts of P and Q, and the second part adds the failure element in case either P or Q contains a failure. Using distributivity of $\cap$ over $\cup$, and introducing the shorthands $P_\perp$ for $P \cap \{\perp\}$ and $P^{\text{-}\perp}$ for $P - \{\perp\}$, the equation becomes

$$P *_1 Q \;=\; (P^{\text{-}\perp} \mathbin{\cup\!\!\!\!\ast} Q^{\text{-}\perp}) \;\cup\; P_\perp \;\cup\; Q_\perp \;.$$

It is easy to see that $*_1$ is associative and commutative. Moreover, it is straightforward to prove that the set $\{\perp\}$ is an annihilator w.r.t. $*_1$, i.e., $P *_1 \{\perp\} = \{\perp\}$.

Model V: *Full annihilation with subset order.* One of the first models that comes into mind when creating a failure model for separation logic is the structure $(\mathcal{P}(\mathit{States} \cup \{\perp\}), \subseteq, \emptyset, *_1, [\![\mathsf{emp}]\!])$. Clearly, $(\mathcal{P}(\mathit{States} \cup \{\perp\}), \subseteq, \emptyset)$ is a complete, distributed and complemented lattice. Moreover, $(\mathcal{P}(\mathit{States} \cup \{\perp\}), *_1, [\![\mathsf{emp}]\!])$ is a monoid. Similar to Model III this algebra is only positively distributive. As before, residuals can be defined via the supremum operator, but do not establish the desired Galois connections. ▲

Model VI: *Full annihilation with more sophisticated order.* To turn Model V into a quantale we define a more sophisticated order.

$$\begin{array}{rl} P \sqsubseteq Q \;\Leftrightarrow_{df} & P^{\text{-}\perp} \subseteq Q^{\text{-}\perp} \;\wedge\; Q_\perp \subseteq P_\perp \\ \Leftrightarrow & P^{\text{-}\perp} \subseteq Q^{\text{-}\perp} \;\wedge\; (\perp \in Q \Rightarrow \perp \in P) \end{array}$$

This order, illustrated in Figure 2, is the subset-order on non-failure elements. It classifies a set containing $\perp$ worse than the same set without failure.

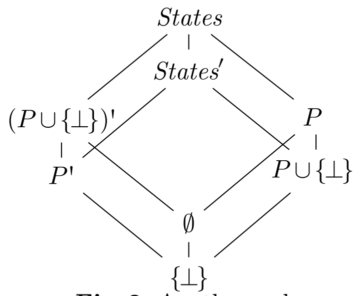


Fig. 2. Another order

The structure $(\mathcal{P}(\mathit{States}'), \sqsubseteq, \{\perp\})$ is a complete lattice, the supremum operator coincides with $P \mathbin{[\,]} Q =_{df} (P^{\text{-}\perp} \cup Q^{\text{-}\perp}) \cup (Q_\perp \cap P_\perp)$, which is associative and commutative. The structure $(\mathcal{P}(\mathit{States}'), \sqsubseteq, \{\perp\}, *_1, [\![\mathsf{emp}]\!])$ forms a Boolean commutative quantale when using set-theoretic complementation over States', denoted by $'$. In particular $(P \cup \{\perp\})' = \overline{P}$, with $\overline{}$ is complementation of States.

Since the largest element does not contain the failure element, this model suffers from the same problems as Model IV, and cannot be used in combination with forward reasoning. However, it is a decent extension of algebraic separation logic, and features all the desired algebraic properties. In particular, we can define the other three operators via Galois connections and duals. As this is an extension of separation logic, we use the symbols from separation logic rather than their algebraic counterparts (e.g. $-\!\!\circledast$ rather than $\rfloor$).

$$\begin{array}{rcl} P \mathrel{-\!\!\!*}_1 Q &=& (P^{\text{-}\perp} \mathrel{-\!\!\!*} Q^{\text{-}\perp}) \cup ((P')_\perp \cap Q_\perp) \\ P \mathrel{\leadsto\!\!\!*}_1 Q &=& (P^{\text{-}\perp} \mathrel{\leadsto\!\!\!*} Q^{\text{-}\perp}) \cup ((P')_\perp \;\cap\; Q_\perp) \\ P \mathrel{-\!\!\circledast}_1 Q &=& (P^{\text{-}\perp} \mathrel{-\!\!\circledast} Q^{\text{-}\perp}) \cup P_\perp \cup Q_\perp \end{array}$$

All these operations have the advantage that they are built based on the original operations of separation logic. In particular, the new and the original definitions behave identical on the non-failure elements (heaps). The second component defines the effect if either P or Q contain the error element. ▲

The above model, as well as all forthcoming ones, can be lifted to a more abstract level, based on a quantale $\mathcal{S} = (\mathcal{P}(S), \subseteq, \emptyset, \cdot, 1)$. As our motivation stems from separation logic, we stick with models based on `AS`.

Although the model cannot be used for forward reasoning, it is useful to show another important shortcoming: septraction as defined in Model VI cannot yield failure: $P \mathbin{-\circledast_1} Q$ contains an error only if either P or Q contains a failure element.

Consider the program `delete_ptr` p. It clears the allocated memory pointed to by p, and should fail if p does not exist. Since we would like to work with Hoare triples of the form

$$\{X\}\,\texttt{delete\_ptr}\ p\,\{p \mapsto \_ \mathbin{-\circledast_?} X\}$$

we require that $-\circledast_?$ can fail, even in the setting of non-failed X, where $-\circledast_?$ is an extension of $-\circledast$. More concretely, the modified septraction should imply

$$p \mapsto \_ \mathbin{-\circledast_?} \mathsf{emp} = \bot \ ,$$

written as a formula of separation logic. A possible solution would be to modify $*_1$ or $-\circledast_1$ by generating a failure whenever separation logic would lead to false.

$$P *_2 Q = \mathtt{F}(P^{\text{-}\bot} * Q^{\text{-}\bot}) \cup P_\bot \cup Q_\bot \ ,$$
$$P \mathbin{-\circledast_2} Q = \mathtt{F}(P^{\text{-}\bot} \mathbin{-\circledast} Q^{\text{-}\bot}) \cup P_\bot \cup Q_\bot \ ,$$

where function $\mathtt{F}$ is defined as $\mathtt{F}(X) =_{df} \{\bot\}$ if $X = \emptyset$ and $\mathtt{F}(X) =_{df} X$ otherwise. Both $*_2$ and $-\circledast_2$ can be used to create failure models. However, the only useful orderings we found for these algebras are either the subset-order or orders closely related to $\sqsubseteq$ of Model VI. As a consequence either multiplication is only positively distributive, or the set $\{\bot\}$ is the least element. As discussed, either setting is not ideal for forward reasoning.

As a side remark it is worth mentioning that in the commutative Boolean quantale $(\mathcal{P}(States'), \sqsubseteq, \{\bot\}, *_2, [\![\mathsf{emp}]\!])$ the operator $-\circledast_2$ is *not* the detachment, despite the symmetric definition.

Failure Flags for Every Single Heap

We turn to the finest sets of models, featuring a failure flag for every heap. In separation logic a heap would be of type $(Addr \rightharpoonup Values) \times \mathbb{B}$, where the Boolean flag indicates (non)failure: true indicates non-failure, and false failure.

In the set-theoretic setting where we consider sets of states, we model failure flags by pairs of sets. The first set lists all heaps stemming from non-failure calculations, and the second one lists all failed heaps: an element of the form $(P, \emptyset)$ indicates non-failure calculations.

Since we distinguish failed from non-failed heaps we do not need a separate failure element. Moreover, since the algebra is now built on $\mathcal{P}(States) \times \mathcal{P}(States)$, we can use standard algebraic constructions to create further models.

Model VII: *Cross-Product with component-wise multiplication.* We consider the structure $(\mathcal{P}(States) \times \mathcal{P}(States), \sqsubseteq, (\emptyset, \emptyset), \circ, ([\![\mathsf{emp}]\!], [\![\mathsf{emp}]\!]))$, where $\sqsubseteq$ and $\circ$ are the component-wise lifted order $\subseteq$ and multiplication $\uplus$, e.g.

$$(P_1, P_2) \circ (Q_1, Q_2) =_{df} (P_1 \uplus Q_1, P_2 \uplus Q_2) \ .$$

Clearly, this model forms a Boolean commutative quantale, which is a proper extension of separation logic. However, since failed and non-failed heaps are kept separate, the derived operation of septraction (detachment) cannot produce failure heaps out of non-failure ones; a similar behaviour as Model VI. Using the same example as above, adapted to the model at hand, we get

$$([\![p \mapsto \_]\!], \emptyset) \mathbin{-\!\circledast_3} ([\![\mathsf{emp}]\!], \emptyset) = (\emptyset, \emptyset) \ ,$$

where $-\!\circledast_3$ is the component-wise lifted septraction. ▲

In general, there are multiple product constructions to create new algebras. The most common constructions, such as Model VII, form quantales again, which can be proved in an algebraic setting, e.g. by using general results from universal algebra.

However, all of these constructions keep the components more or less separate. Hence these models, although decent failure extensions, suffer the same shortcoming as Model VII. Some of those models, e.g. one that is identical to Model VII except that the order is given by $\subseteq \times \supseteq$, yield problems already discussed earlier – here, the problem occurring in reasoning in Hoare-logic style.

5 More Sophisticated Models

Since simple extensions do not work we consider two sophisticated models in this section. Instead of considering flags, we now split every heap into a set of actual configuration, and one of 'failed' configurations. Formally a heap is of type $Heaps_2 =_{df} (Addr \rightharpoonup Values) \times (Addr \rightharpoonup Values)$. The extended heap $(p \mapsto v, q \mapsto \_)$ states that a pointer p exists, pointing to value v, and that a pointer q is required, but does not exist. This idea is inspired by the construction of integers from natural numbers (e.g. [5]). Both models build on this idea of 'negative' heaps; the base set is $States_2 =_{df} \mathcal{P}(Stores \times Heaps_2)$. Instead of adding elements and operations on top of `AS` we change the underlying logic of `AS`.

As before the forthcoming models can be lifted to an abstract level, based on a quantale $(\mathcal{P}(S), \subseteq, \emptyset, \cdot, 1)$, but we stay within the setting of separation logic.

We aim at a failure model that allows non-failure elements to create failure, when used with septraction. Hence we have to develop operators that combine both parts of the pair of the underlying set of elements.

We define a new unary operator that eliminates those parts of a heap that are present and missing at the same time; again this is similar to the construction on top of natural numbers. Intuitively such heaps cannot exist:

$$\begin{aligned} {}^* : Heaps_2 &\rightarrow Heaps_2 \\ (h_1, h_2)^* &=_{df} (h_1|_{\overline{\mathsf{dom}(h_2)}}, h_2|_{\overline{\mathsf{dom}(h_1)}}) \ , \end{aligned}$$

where the heaps are restricted to those parts that do not occur in both. This operator is lifted to States_2 by $P^* =_{df} \{(s, \mathtt{h}^*) : (s, \mathtt{h}) \in P\}$, where $\mathtt{h} \in \mathit{Heaps}_2$ is an extended heap.

Model VIII: *Septraction based on heap reduction.* The first model of this section modifies $\mathtt{AS}$ to take heap reduction * into account: in detail we consider the structure $(\mathcal{P}(\mathit{States}_2), \subseteq, \emptyset, \uplus_1, [\![\mathsf{emp}']\!])$, where $[\![\mathsf{emp}']\!] =_{df} \{(s, \emptyset, \emptyset)\}$ and $\uplus_1$ is an adapted version of $\uplus$:

$$\begin{aligned} P \uplus_1 Q =_{df} \{(s, h_1 \cup h_1', h_2 \cup h_2') : (s, h_1, h_2) \in P^* \wedge (s, h_1', h_2') \in Q^* \\ \wedge\ \mathsf{dom}(h_1) \cap \mathsf{dom}(h_1') = \mathsf{dom}(h_2) \cap \mathsf{dom}(h_2') = \emptyset\}\ . \end{aligned}$$

The structure forms a commutative quantale, with $(\mathcal{P}(\mathit{States}_2), \subseteq, \emptyset)$ being a complete and distributive lattice. Therefore the residuals of $\uplus_1$ exist, denoted by $-\!\!*_4$. We further define a new complementation operator $\sim$ by $\sim P =_{df} \overline{P^*}$. The idea of this negation is based on the negation of non-classical *relevance logics* (also called *relevant logics*) [20,12], where * is related to the logic's 'Routley Star'. We introduce separating coimplication and septraction as

$$P \leadsto\!\!*_4 Q =_{df} \sim(P *_4 \sim Q) \quad \text{and} \quad P -\!\!\circledast_4 Q =_{df} \sim(P -\!\!*_4 \sim Q)\ .$$

Lifting the satisfaction-based semantics to a set-based one by $[\![p]\!] =_{df} \{(s, h_1, h_2) : s, h_1, h_2 \models p\}$, and using the shorthand $[\![h_1, h_2]\!]$ for $\{(s, h_1, h_2)\}$, we can derive useful laws such as

$$[\![p \mapsto v, \emptyset]\!] -\!\!\circledast_4 [\![\emptyset, \emptyset]\!] = [\![\emptyset, p \mapsto v]\!] \quad \text{and} \quad [\![p \mapsto v, \emptyset]\!] -\!\!\circledast_4 [\![p \mapsto v, \emptyset]\!] \supseteq [\![\mathsf{emp}']\!]\ .$$

Both properties are desired and fit the intuition of septraction involving failure. The first property states that pulling out a resource $(p \mapsto v)$ from the empty heap yields a negative heap; the second property states that one can pull out a resource if it exists. However, we can derive inequalities such as $[\![q \mapsto \_, q \mapsto \_]\!] \subseteq [\![p \mapsto v, \emptyset]\!] -\!\!\circledast_4 [\![p \mapsto v, \emptyset]\!]$, which seems to be weird: why should a statement that deletes a pointer p from a heap that only consists of p talk about another pointer q? Applying the operator * would remove such heap elements, but the current model has another severe disadvantage: $\leadsto\!\!*_4$ and $-\!\!\circledast_4$ do not form a Galois connection. This is not a surprise because $\sim$ is not a proper complement, e.g. $\sim(\sim P) \neq P$. ▲

Using standard set complementation turns $(\mathcal{P}(\mathit{States}_2), \subseteq, \emptyset)$ into a complemented lattice and the above structure into a Boolean quantale. However, it would not solve any of the problems mentioned before as it is similar to the original algebra $\mathtt{AS}$, featuring two instead of one heap; but not interpreting the second heap as a negative one.

A similar model with the same problems as Model VIII is based on failure flags for every single heap (see Section 4): it considers the structure

$$(\mathcal{P}(\mathit{States}) \times \mathcal{P}(\mathit{States}), \subseteq \times \subseteq, (\emptyset, \emptyset), \uplus_2, ([\![\mathsf{emp}]\!], [\![\mathsf{emp}]\!]))\ ,$$

with $(P_1, P_2) \uplus_2 (Q_1, Q_2) =_{df} (P_1 \uplus P_2, Q_1 \uplus Q_2)$, and heap restriction operator $(P_1, P_2)^* =_{df} (P_1 - P_2, P_2 - P_1)$.

Model IX: *Septraction based on heap reduction under adapted heap addition.* As in Model VIII we consider the structure $(\mathcal{P}(\mathit{States}_2), \subseteq, \emptyset, \uplus_2, [\![\mathsf{emp}']\!])$, but this time we define multiplication as

$$P \uplus_2 Q \ =_{df}\ \{(s, \mathtt{h}^* \cup \mathtt{h}'^*) : (s, \mathtt{h}) \in P \wedge (s, \mathtt{h}') \in Q \wedge \mathsf{dom}(\mathtt{h}) \cap \mathsf{dom}(\mathtt{h}') = \emptyset\}\ ,$$

where $\cup$, $\cap$ and dom are defined component-wise on Heaps_2. As we changed the underlying heap addition, which now uses heap reduction *, we are not required to use a non-standard complement any longer. It is straightforward to show that this structure is indeed a commutative Boolean quantale; the proof is similar to the one of AS. As a consequence, residuals, detachments and duals exist; their characterisation is similar to the one of AS, presented in Section 2. At the level of septraction and separating coimplication they read as follows

$$\begin{aligned} s, \mathtt{h} \models p \multimap\!\circledast_5\, q \ \Leftrightarrow\ & \exists \mathtt{h}_1 \in \mathit{Heaps}_2 : \mathsf{dom}(\mathtt{h}_1^*)\ \cap\ \mathsf{dom}(\mathtt{h}^*) = \emptyset \text{ and} \\ & s, \mathtt{h}_1 \models p \text{ and } s, \mathtt{h} \cup \mathtt{h}_1 \models q \\ s, \mathtt{h} \models p \leadsto\!\!\ast_5\, q \ \Leftrightarrow\ & \forall \mathtt{h}_1, \mathtt{h}_2 \in \mathit{Heaps} : (\mathsf{dom}(\mathtt{h}_1^*) \cap \mathsf{dom}(\mathtt{h}_2^*) = \emptyset \text{ and} \\ & \mathtt{h} = \mathtt{h}_1 \cup \mathtt{h}_2 \text{ and } s, \mathtt{h}_1 \models p) \text{ implies } s, \mathtt{h}_2 \models q\ . \end{aligned}$$

A big advantage of this model is that all the definitions are 'identical' to the ones of AS; even the logical formulas are identical. Moreover, it satisfies many useful properties such as

$$[\![p \mapsto v, \emptyset]\!] \multimap\!\circledast_5 [\![p \mapsto v, \emptyset]\!] \ =\ [\![\mathsf{emp}']\!] \quad \text{and} \quad [\![p \mapsto v, \emptyset]\!] \multimap\!\circledast_5 [\![\mathsf{emp}']\!] \ \subseteq\ [\![\emptyset, p \mapsto v]\!]$$

The first equation states that a subheap that exists can actually be pulled out; the second that pulling out a non-existing heap does indeed yield a failure – represented by its appearance in the second component.

As Galois connections are available we can use this model for forward reasoning (see Section 3). The connectives are related to Bi-Intuitionistic Boolean Bunched Implications (BiBBI) [4,3], which is a fragment of linear logic. Although the theories are slightly different, the proof theory developed for BiBBI should be usable for our model.

The model comes very close to the one we are looking for. However, it does not satisfy all the properties we desire. For example, we have

$$[\![p \mapsto v, \emptyset]\!] \multimap\!\circledast_5 [\![\emptyset, p \mapsto v]\!] \ =\ \emptyset\ =\ [\![\mathsf{false}]\!]\ .$$

This law states that pulling out a heap, which does not exist and which is already part of the negative heap, leads to the empty heap in both components. That means a non-existent heap cannot be pulled out twice. ▲

Clearly, the equation mentioned at the end of Model IX is not the intended behaviour we would expect from a model that can be used for forward reasoning in separation logic. The reason is that we use separating conjunction in both parts of the extended heap, on the 'positive' heap as well on the 'negative' heap. We suspect that relaxing the 'negative' heap structure could help. Rather than using a partial function $\mathit{Addr} \rightharpoonup \mathit{Values}$ one could use multisets. By doing so, we hope to achieve the following behaviour

$$(p \mapsto v, \emptyset) \multimap\!\circledast_? (\emptyset, p \mapsto v) \ =\ (\emptyset, \{p \mapsto v, p \mapsto v\})\ ,$$

written in separation-logical style and stating that the resource $p \mapsto v$ is already missing twice. When using a multiset as negative heap, however, it is not possible to use set-theoretic complementation as negation. Part of our future work is to figure out all the details, e.g. how to integrate the heap reduction operator *.

6 Discussion and Conclusion

In this paper we have reported on our quest of extending separation logic in a way that it can be used for forward reasoning in non-partial contexts; in particular in the setting of total and general correctness. To support forward reasoning there, separation logic needs to be equipped with failure element(s). The problem with creating such a model is that we want it to be natural and intuitive when it comes to septraction, but at the same time we want to maintain the algebraic properties of the original separation logic.

In this paper we have focussed on the algebraic structure of separation logic, namely on quantales. We have developed a series of models that could potentially be used as failure models. It turns out that all extensions that we thought of being useful either loose algebraic properties such as dualities, Galois connections and/or associativity; or we cannot use the developed models for forward reasoning as the weakening rule of Hoare logic would not be valid any longer.

Although these models could be seen as negative results, we have decided to publish all the models nevertheless: (a) as we have shown that simple models cannot be used for extending separation logic, we want to share the experience to prevent other researchers from creating 'simple' models; (b) we want to give a justification why the models we are looking at at the moment are complicated; and (c) we suspect that the developed models can also be used in other settings, such as Concurrent Kleene Algebra [15] or hybrid system analysis [16].

To present as many models as possible, we have decided to skip all the proofs as most of them are lengthy and boring, but not hard. Some of the proofs such as the proof that Model VI forms a commutative Boolean quantale have been mechanised in the interactive theorem prover Isabelle/HOL; these proofs can be found at `http://hoefner-online.de/ramics18/`.

We have mentioned in Section 3 that backward and forward reasoning for separation logic – the latter only for partial correctness – is supported by Isabelle/HOL. Although we have not found the perfect model yet, we have integrated some of the presented models in our Isabelle framework. In particular Model VIII, Model IX, as well as a similar (non-associative) one, which we have not listed. By doing so, we have figured out that the loss of associativity is not too bad. Non-associativity makes automation slightly more complicated, but certainly not impossible, as associativity still holds when no non-failure elements are involved. When performing forward reasoning, a single failed conjunct already indicates a problem of a program; hence a quick search for a failure element over a derived formula indicates failure. A link to our Isabelle framework can also be found at the above-mentioned webpage.

For future work we obviously will continue our search for an ideal model for our framework. If we prove that none exist, we will have to decide on which tradeoff suits our setting the most.

Acknowledgement. We are grateful to Gerwin Klein and Bernhard Möller for fruitful discussions and inspiring ideas. We also thank the anonymous referees for their valuable feedback.

References

1. Bannister, C., Höfner, P., Klein, G.: Backwards and forwards with separation logic. In: Avigad, J., Mahboubi, A. (eds.) Interactive Theorem Proving (ITP'18). Lecture Notes in Computer Science, vol. 10895, pp. 68–87. Springer (2018)
2. Birkhoff, G.: Lattice Theory, Colloquium Publications, vol. XXV. Annals of Mathematics Studies, 3rd edn. (1967)
3. Brotherston, J., Calcagno, C.: Classical BI: its semantics and proof theory. Logical Methods in Computer Science 6(3) (2010)
4. Brotherston, J., Villard, J.: Sub-classical boolean bunched logics and the meaning of par. In: Kreutzer, S. (ed.) Computer Science Logic (CSL '15). Leibniz International Proceedings in Informatics (LIPIcs), vol. 41, pp. 325–342. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2015)
5. Campbell, H.E.: The Structure of Arithmetic. Appleton-Century-Crofts (1970)
6. Conway, J.H.: Regular Algebra and Finite Machines. Chapman & Hall (1971)
7. Dang, H.H.: Algebraic Calculi for Separation Logic. Ph.D. thesis, University of Augsburg, Germany (2014)
8. Dang, H.H., Höfner, P., Möller, B.: Algebraic separation logic. Journal of Logic and Algebraic Programming 80(6), 221–247 (2011)
9. Desharnais, J., Möller, B.: Non-associative Kleene algebra and temporal logics. In: Höfner, P., Pous, D., Struth, G. (eds.) Relational and Algebraic Methods in Computer Science (RAMiCS'17). Lecture Notes in Computer Science, vol. 10226, pp. 93–108. Springer (2017)
10. Dijkstra, E.W.: A Discipline of Programming. Prentice Hall (1976)
11. Dongol, B., Hayes, I.J., Struth, G.: Relational convolution, generalised modalities and incidence algebras. arXiv: abs/1702.04603 (2017)
12. Dunn, J.: Star and perp. Philosophical Perspectives 7, 331–357 (1993)
13. Floyd, R.W.: Assigning meanings to programs. Mathematical aspects of computer science 19, 19–32 (1967)
14. Hoare, C.A.R.: An axiomatic basis for computer programming. Communications of the ACM 12, 576–580 (1969)
15. Hoare, T., Möller, B., Struth, G., Wehrman, I.: Concurrent Kleene algebra and its foundations. Journal of Logic and Algebraic Programming 80(6), 266–296 (2011)
16. Höfner, P., , Möller, B.: An algebra of hybrid systems. Journal of Logic and Algebraic Programming 78, 74–97 (2009)
17. Ishtiaq, S.S., O'Hearn, P.W.: BI as an assertion language for mutable data structures. SIGPLAN Notices 36, 14–26 (2001)
18. Jacobs, D., Gries, D.: General correctness: A unification of partial and total correctness. Acta Inf. 22(1), 67–83 (1985)
19. Kozen, D.: On Hoare logic and Kleene algebra with tests. ACM Transactions on Computational Logic 1(1), 60–76 (2000)

20. Mares, E.: Relevance logic. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, spring 2014 edn. (2014)
21. Möller, B.: Residuals and detachments. Tech. Rep. 2005-20, Institut für Informatik, Universität Augsburg (2005)
22. Möller, B., Struth, G.: Algebras of modal operators and partial correctness. Theoretical Computer Science 351(2), 221–239 (2006)
23. Möller, B., Struth, G.: WP is WLP. In: MacCaull, W., Winter, M., Düntsch, I. (eds.) Relational Methods in Computer Science (RelMiCS '06). Lecture Notes in Computer Science, vol. 3929, pp. 200–211. Springer (2006)
24. Mulvey, C.: &. Rendiconti del Circolo Matematico di Palermo 12(2), 99–104 (1986)
25. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL — A Proof Assistant for Higher-Order Logic, Lecture Notes in Computer Science, vol. 2283. Springer (2002)
26. O'Hearn, P.: Resources, concurrency, and local reasoning. Theoretical Computer Science 375, 271–307 (2007)
27. O'Hearn, P.W., Reynolds, J.C., Yang, H.: Local reasoning about programs that alter data structures. In: Fribourg, L. (ed.) Computer Science Logic (CSL '01). Lecture Notes in Computer Science, vol. 2142, pp. 1–19. Springer (2001)
28. Pym, D.: The Semantics and Proof Theory of the Logic of Bunched Implications. Kluwer Academic Publishers (2002)
29. Reynolds, J.C.: Intuitionistic reasoning about shared mutable data structure. In: Davies, J., Roscoe, B., Woodcock, J. (eds.) Millennial Perspectives in Computer Science. pp. 303–321. Palgrave (2000)
30. Reynolds, J.C.: An introduction to separation logic. In: Broy, M., Sitou, W., Hoare, T. (eds.) Engineering Methods and Tools for Software Safety and Security, NATO Science for Peace and Security Series - D: Information and Communication Security, vol. 22, pp. 285–310. IOS Press (2009)
31. Rosenthal, K.: Quantales and their applications. Pitman Research Notes in Mathematics Series 234 (1990)
32. Vafeiadis, V., Parkinson, M.: A marriage of rely/guarantee and separation logic. In: Caires, L., Vasconcelos, V.T. (eds.) Conference on Concurrency Theory (CONCUR '07). Lecture Notes in Computer Science, vol. 4703, pp. 256–271. Springer (2007)