Modular Labelled Sequent Calculi for Abstract Separation Logics

ZHÉ HÓU, Griffith University, Australia
RANALD CLOUSTON, Aarhus University, Denmark
RAJEEV GORÉ, The Australian National University, Australia
ALWEN TIU, The Australian National University, Australia

Abstract separation logics are a family of extensions of Hoare logic for reasoning about programs that manipulate resources such as memory locations. These logics are "abstract" because they are independent of any particular concrete resource model. Their assertion languages, called propositional abstract separation logics (PASLs), extend the logic of (Boolean) Bunched Implications (BBI) in various ways. In particular, these logics contain the connectives $*$ and $-\!\!*$, denoting the composition and extension of resources respectively.

This added expressive power comes at a price since the resulting logics are all undecidable. Given their wide applicability, even a semi-decision procedure for these logics is desirable. Although several PASLs and their relationships with BBI are discussed in the literature, the proof theory of, and automated reasoning for, these logics were open problems solved by the conference version of this paper, which developed a modular proof theory for various PASLs using cut-free labelled sequent calculi. This paper non-trivially improves upon this previous work by giving a general framework of calculi on which any new axiom in the logic satisfying a certain form corresponds to an inference rule in our framework, and the completeness proof is generalised to consider such axioms.

Our base calculus handles Calcagno et al.'s original logic of separation algebras by adding sound rules for partial-determinism and cancellativity, while preserving cut-elimination. We then show that many important properties in separation logic, such as indivisible unit, disjointness, splittability, and cross-split, can be expressed in our general axiom form. Thus our framework offers inference rules and completeness for these properties for free. Finally, we show how our calculi reduce to calculi with global label substitutions, enabling more efficient implementation.

CCS Concepts: • **Theory of computation → Separation logic**;

Additional Key Words and Phrases: Labelled sequent calculus, automated reasoning, abstract separation logics, counter-model construction, bunched implications

ACM Reference Format:
Zhé Hóu, Ranald Clouston, Rajeev Goré, and Alwen Tiu. 0000. Modular Labelled Sequent Calculi for Abstract Separation Logics. *ACM Trans. Comput. Logic* 0, 0, Article 00 (0000), 35 pages.
https://doi.org/0000001.0000001

1 INTRODUCTION

Reynolds's Separation logic (SL) [55] is an extension of Hoare logic for reasoning about programs that explicitly mutate memory. Its assertion logic, also called separation logic, extends the usual (additive) connectives for conjunction $\wedge$, disjunction $\vee$, implication $\rightarrow$, and the (additive) verum

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 0000 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
1529-3785/0000/0-ART00 $15.00
https://doi.org/0000001.0000001

constant $\top$, with the multiplicative connectives *separating conjunction* $*$, its unit $\top^*$ (denoted by *emp* in some literature), and *separating implication* $-\!\!*$, also called *magic wand*, from the logic of Bunched Implications (BI) [50]. Moreover, the assertion language introduces the *points-to* predicate $E \mapsto E'$ on expressions, along with the usual quantifiers and predicates of first-order logic with equality and arithmetic. The additive connectives may be either intuitionistic, as for BI, or classical, as for the logic of *Boolean* Bunched Implications (BBI). Classical additives are more expressive as they support reasoning about non-monotonic commands such as memory de-allocation, and assertions such as "the heap is empty" [36]. In this paper we consider classical additives only.

The concrete memory model for SL is given in terms of heaps, where a *heap* is a finite partial function from addresses to values. A heap satisfies $P * Q$ iff it can be partitioned into heaps satisfying P and Q respectively; it satisfies $\top^*$ iff it is empty; it satisfies $P -\!\!* Q$ iff any extension with a heap that satisfies P must then satisfy Q; and it satisfies $E \mapsto E'$ iff it is a singleton map sending the address specified by the expression E to the value specified by the expression E'. While the $\mapsto$ predicate refers to the *content* of heaps, the BI connectives refer only to their *structure*. Some basic spatial properties of heaps include the following:

Empty heap There is a unique empty heap ϵ;
Identity Combining heap h with the empty heap ϵ gives the original heap h;
Commutativity Combining heap h_1 with heap h_2 is the same as combining h_2 with h_1;
Associativity Combining heap h_1 with heap h_2 and then combining the result with heap h_3 is the same as combining heap h_1 with the combination of heaps h_2 and h_3.

These conditions define a *non-deterministic monoid*: giving algebraic models for BBI [24].

The idea of separation logic has proved fruitful for a range of memory (or, more generally, resource) models, some quite different from the original heap model. In this paper we will present examples drawn from [6, 7, 14, 20, 38, 52, 65], but this list is far from exhaustive. Each such model has its own notion of separation and sharing of resources, and hence may formally give rise to a new logic with respect to the BI connectives, let alone any special-purpose predicates which might be added to the logic. As new variations of separation logic are introduced, their relation to prior logics is seldom developed formally, and so new metatheory and tool support must be substantially reconstructed for each case. This has led to a subgenre of papers highlighting the need for organisation and generalisation across these logics [4, 14, 37, 53].

In this paper we take as a starting point *Abstract Separation Logic* [14], which is intended to generalise the logics of many concrete models. In particular we set quantifiers aside to work with Propositional Abstract Separation Logic (PASL). This logic is defined via the abstract semantics of partial cancellative monoids, or *separation algebras*, which are non-deterministic monoids restricted by:

Partial-determinism The combination of heap h_1 with heap h_2 is either undefined, or a unique heap;
Cancellativity If combining h_1 and h_2 gives h_3 and combining heap h_1 and h_4 also gives h_3, then $h_2 = h_4$.

Semantics in this style are reminiscent of the ternary algebraic semantics often used in connection with substructural logics [1], an observation we exploit in this paper. Separation algebras allow interpretation of $*$, $\top^*$ and $-\!\!*$, although the latter is not considered by [14]. The points-to ($\mapsto$) predicate is not a first class citizen of PASL; it may be introduced as a predicate only if an appropriate concrete separation algebra is fixed. PASL is appropriate to reasoning about the *structure* of memory, but not its *content*.

Precondition strengthening and postcondition weakening in Hoare-style logics require reasoning in the assertion logic, but proof search and structural proof theory for PASL have received little attention until recently. It is known that the added expressive power of the multiplicative connectives comes at a price, yielding a logic that is in general undecidable [12, 44]. Given the wide applicability of abstract separation logic, even a semi-decision procedure for PASL would assist program verification.

However the definition of separation algebras presented by [14] is not necessarily canonical. Most notably [20] suggested the following useful additional properties for spatial reasoning[1]:

Indivisible unit If combining heap h_1 with heap h_2 gives the empty heap, then h_1 and h_2 must themselves be the empty heap;

Disjointness If the result of combining heap h_1 with itself is defined, then h_1 must be the empty heap;

Splittability Every non-empty heap h_0 can be split into two non-empty heaps h_1 and h_2;

Cross-split If a heap can be split in two different ways, then there should be heaps that constitute the intersections of these splittings.

Conversely, following [26] some authors have further generalised separation algebras by dropping cancellativity; we will present concrete examples from [39, 63]. These extensions and restrictions of PASL point to a need to present *modular* proof theory and proof search techniques which allow the axiomatic properties of the abstract models to be adjusted according to the needs of a particular application.

This paper is an extended journal version of the conference paper [29]. In that paper, we solved the open problems of presenting sound and complete structural proof theory, and gave a semi-decision procedure, along with an efficient implementation, for Propositional Abstract Separation Logic. We further showed that our methods could encompass the axiomatic extensions of [20], and conversely that cancellativity and partial-determinism could be dropped, and so our proof theory was *modular* in the sense that it could be used for many neighbouring logics of PASL, including BBI. In this journal paper we make a major extension to the modularity of our approach by introducing a technique to synthesise proof rules from any spatial axiom in a certain format, general enough to encompass all axioms of [20]. The remainder of this introduction sketches the techniques used in this paper.

Because of the similarity between non-deterministic monoids, which provide semantics for BBI [24], and the separation algebras which provide semantics for PASL, it is natural to investigate whether techniques used successfully for BBI can be extended to PASL. This paper answers this question in the affirmative by extending the work on BBI of [34, 35]. In these papers, a sound and complete proof theory was provided for BBI in the style of *labelled sequent calculus* [49], a proof style for modal and substructural logics with Kripke-style frame semantics in which statements about the elements of the frame are explicitly included in the context of sequents. This allows relational properties of the semantics to be explicitly represented as proof rules, which allows labelled sequent calculi to encompass a wide variety of logics in a modular style – the addition or subtraction of semantic properties corresponds exactly to the addition or subtraction of the corresponding proof rules.

This paper builds on [35] by presenting a labelled sequent calculus for a sublogic BBI^{-} of BBI, which is of no intrinsic interest that we are aware of, but which does include all BI connectives.

[1]Dockins et al. [20] also suggested generalising separation algebras to have a *set* of units; it is an easy corollary of [13, Lemma 3.11] that single-unit and multiple-unit separation algebras satisfy the same set of formulae, and we do not pursue this generalisation in this paper.

We then show that it can be extended to a labelled sequent calculus for BBI, for PASL, and for various neighbouring logics, by extending it with instances of a *general structural rule* synthesised from axioms on the semantics. This is possible so long as the axiom is in a certain format, which is sufficiently general to encompass, for example, the spatial properties identified by [20]. We call an axiom in this format a *frame axiom*. We then show that our sequent calculi can be used for effective backward proof search, thereby providing semi-decision procedures for a variety of logics. Our implementation, Separata[2], is the first automated theorem prover for PASL and many of its neighbours. Separata differs from our previous implementation FVLS_{BBI} for BBI in two aspects: first, Separata can handle multiple abstract separation logics, including BBI, whereas FVLS_{BBI} is designed for BBI only; second, Separata is a semi-decision procedure, whereas FVLS_{BBI} adopts a heuristic proof search which is incomplete.

In this work, we are interested in proof search procedures that are complete. In this setting, sequent calculi are amenable to backward proof-search only if the cut rule is redundant. This result follows much as for the calculus LS_{BBI} of [35]. However completeness does not follow so easily; in [35] the completeness of LS_{BBI} was shown by mimicking derivations in the Hilbert axiomatisation of BBI. This avenue is no longer viable for PASL because partial-determinism and cancellativity are not axiomatisable in BBI [13]. That is, there can be no Hilbert calculus in the language of BBI which is sound and complete with respect to separation algebras. We instead prove the cut-free completeness of our labelled sequent calculi via a *counter-model construction* procedure which shows that if a formula is not cut-free derivable in our sequent calculus then it is falsifiable in some PASL-model.

The calculi of this paper differ in style from LS_{BBI} because, in [35], explicit *substitutions* are used in the proof rules, whereas in this paper these are replaced by explicit *equality* assertions. These are easier for us to manage with respect to proving the modular completeness of our family of calculi, but the presentation with substitutions is more amenable to implementation. We hence show how equivalent new calculi can be defined, with substitutions replacing equalities, and show how this allows a semi-decision procedure to be implemented. Experimental results show that our prover is usually faster than other provers for BBI when tested against the same benchmarks of BBI formulae.

This paper improves upon all aspects of the presentation of results from its conference predecessor [29], partly because of lesser limitations on space, but we here briefly summarise the more important differences between this paper and the earlier work:

- A new modular framework of calculi based on frame axioms and synthesised structural rules. The completeness of calculi in this framework can be obtained in one proof. In the previous work, each new calculus required a new proof;
- A new completeness proof by counter-model construction for a framework of calculi. This proof includes treatments for splittability and cross-split, which are not included in the previous work;
- A translation from the current calculi to previous calculi with global label substitutions;
- More comprehensive experiments with testing of randomly generated formulae;
- Many more examples of concrete separation algebras and their applications;
- Example derivations of various formulae; and a
- Discussion of applications of this work.

The remainder of this paper is structured as follows. Section 2 introduces Propositional Abstract Separation Logic via its separation algebra semantics, gives a number of concrete examples of these semantics, and defines the labelled sequent calculus for PASL. Fundamental results such as

[2]Available at http://users.cecs.anu.edu.au/~zhehou.

soundness and cut-elimination are also proved. Section 3 proves the completeness of our calculi framework by counter-model construction. Section 4 shows how our framework can encompass various neighbouring logics of PASL, based on models with different spatial properties, and discusses how these properties manifest in examples. Section 5 shows how to translate our calculi into a format that is more amenable to implementation, and presents some example derivations. Section 6 presents the implementation and experiments. Section 7 discusses applications and extensions of the calculi in this work. Finally, Section 8 discusses related work.

2 A LABELLED SEQUENT CALCULUS FOR PASL

In this section we define the *separation algebra* semantics of Calcagno et al. [14] for Propositional Abstract Separation Logic (PASL), present concrete examples of these semantics, and give the labelled sequent calculus $\mathrm{LS}_{\mathrm{PASL}}$ for this logic. Soundness and cut-elimination are then demonstrated for $\mathrm{LS}_{\mathrm{PASL}}$.

2.1 Propositional abstract separation logic

The formulae of PASL are defined inductively as follows, where p ranges over some set Var of propositional variables:

$$A ::= p \mid \top \mid \bot \mid \neg A \mid A \vee A \mid A \wedge A \mid A \to A \mid \top^* \mid A * A \mid A \mathrel{-\!\!*} A$$

PASL-formulae will be interpreted via the following semantics:

Definition 2.1. A *separation algebra*, or partial cancellative commutative monoid, is a triple $(H, \circ, \epsilon)$ where H is a non-empty set, $\circ$ is a partial binary function $H \times H \rightharpoonup H$ written infix, and $\epsilon \in H$, satisfying the following conditions, where '=' is interpreted as "either, both sides are undefined, or, both sides are defined and equal":

identity: $\forall h \in H.\, h \circ \epsilon = h$
commutativity: $\forall h_1, h_2 \in H.\, h_1 \circ h_2 = h_2 \circ h_1$
associativity: $\forall h_1, h_2, h_3 \in H.\, h_1 \circ (h_2 \circ h_3) = (h_1 \circ h_2) \circ h_3$
cancellativity: $\forall h_1, h_2, h_3, h_4 \in H.$ if $h_1 \circ h_2 = h_3$ and $h_1 \circ h_4 = h_3$ then $h_2 = h_4$

Note that the *partial-determinism* of the monoid is assumed since $\circ$ is a partial function: for any $h_1, h_2, h_3, h_4 \in H$, if $h_1 \circ h_2 = h_3$ and $h_1 \circ h_2 = h_4$ then $h_3 = h_4$.

Example 2.2. The paradigmatic example of a separation algebra is the set of *heaps* [55]: finite partial functions from an infinite set of *locations* to a set of *values*. Then $h_1 \circ h_2 = h_1 \cup h_2$ if h_1, h_2 have disjoint domains, and is undefined otherwise. ϵ is the empty function.

Example 2.3. A *partial commutative semigroup* [6], also known as a *permission algebra*[3] [14], is a set V equipped with an associative commutative partial binary operator $\star$, written infix. In other words, it is a separation algebra without the requirement to have a unit, or to be cancellative.

Fixing such a $(V, \star)$, for which we will give some example definitions shortly, and given an infinite set of locations Loc, we define two finite partial functions h_1, h_2 from Loc to V to be *compatible* iff for all l in the intersection of their domains, $h_1(l) \star h_2(l)$ is defined. We then define the binary operation $\circ$ on partial functions h_1, h_2 as undefined if they are not compatible. Where they are

[3]We prefer the former term, as many interesting examples have little to do with permissions, and the 'permissions algebra' terminology is not used consistently in the literature; compare [14, 62].

compatible, $(h_1 \circ h_2)(l)$ is defined as:

$$(h_1 \circ h_2)(l) = \begin{cases} h_1(l) \star h_2(l) & l \in dom(h_1) \cap dom(h_2) \\ h_1(l) & l \in dom(h_1) \setminus dom(h_2) \\ h_2(l) & l \in dom(h_2) \setminus dom(h_1) \\ \text{undefined} & l \notin dom(h_1) \cup dom(h_2) \end{cases}$$

Setting ϵ as the empty function, many examples of concrete separation algebras have this form, with the $\star$ operation, where defined, intuitively corresponding to some notion of *sharing* of resources. The following are some example definitions of such a construction:

- Heaps: let V be the set of values, and $\star$ be undefined everywhere.
- Fractional permissions [7]: let V be the set of pairs of values (denoted by v, w) and (real or rational) numbers (denoted by i, j) in the interval $(0, 1]$, and

$$(v, i) \star (w, j) = \begin{cases} (v, i + j) & v = w \text{ and } i + j \leq 1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

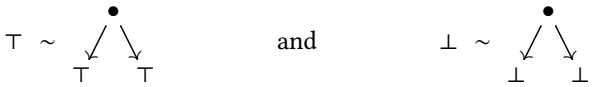
- Named permissions [52]: given a set $\mathbb{P}$ of *permission names*, let V be the set of pairs of values (denoted by v, w) and non-empty subsets (denoted by P, Q) of $\mathbb{P}$, and

$$(v, P) \star (w, Q) = \begin{cases} (v, P \cup Q) & v = w \text{ and } P \cap Q = \emptyset \\ \text{undefined} & \text{otherwise} \end{cases}$$

- Counting permissions [6]: let V be the set of pairs of values (denoted by v, w) and integers (denoted by i, j). Here 0 is interpreted as *total permission*, negative integers as *read permissions*, and positive integers as counters of the number of permissions taken. Let

$$(v, i) \star (w, j) = \begin{cases} (v, i + j) & v = w \text{ and } i < 0 \text{ and } j < 0 \\ (v, i + j) & v = w \text{ and } i + j \geq 0 \text{ and } (\, i < 0 \text{ or } j < 0\,) \\ \text{undefined} & \text{otherwise} \end{cases}$$

- Binary Tree Share Model [20]: Consider the set of finite non-empty binary trees whose leaves are labelled true ($\top$) or false ($\bot$), modulo the smallest congruence such that



  Let $\vee$ (resp. $\wedge$) be the pointwise disjunction (resp. conjunction) of representative trees of the same shape. Then let V be the pairs of values (denoted by v, w) and equivalence classes of trees (denoted by t, u) so defined, and with $\star$ defined as shown below, where $[\bot]$ is the equivalence class containing the tree whose only node contains $\bot$:

$$(v, t) \star (w, u) = \begin{cases} (v, t \vee u) & v = w \text{ and } t \wedge u = [\bot] \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Note that the construction above with partial commutative semigroups does not in general guarantee cancellativity of the separation algebra; for this we need to require further that $(V, \star)$ is cancellative and has no idempotent elements (satisfying $v \star v = v$). As we will see later, some interesting concrete models fail this requirement, and so we will generalise the results of the paper to drop cancellativity in Section 4.

$\mathcal{M}, h \Vdash p$	iff	$p \in Var$ and $h \in v(p)$	$\mathcal{M}, h \Vdash \top^*$	iff	$h = \epsilon$
$\mathcal{M}, h \Vdash A \wedge B$	iff	$\mathcal{M}, h \Vdash A$ and $\mathcal{M}, h \Vdash B$	$\mathcal{M}, h \Vdash \top$	iff	always
$\mathcal{M}, h \Vdash A \to B$	iff	$\mathcal{M}, h \nVdash A$ or $\mathcal{M}, h \Vdash B$	$\mathcal{M}, h \Vdash \bot$	iff	never
$\mathcal{M}, h \Vdash A \vee B$	iff	$\mathcal{M}, h \Vdash A$ or $\mathcal{M}, h \Vdash B$	$\mathcal{M}, h \Vdash \neg A$	iff	$\mathcal{M}, h \nVdash A$
$\mathcal{M}, h \Vdash A * B$	iff	$\exists h_1, h_2.(R(h_1, h_2, h)$ and $\mathcal{M}, h_1 \Vdash A$ and $\mathcal{M}, h_2 \Vdash B)$			
$\mathcal{M}, h \Vdash A \mathrel{-\!\!*} B$	iff	$\forall h_1, h_2.((R(h, h_1, h_2)$ and $\mathcal{M}, h_1 \Vdash A)$ implies $\mathcal{M}, h_2 \Vdash B)$			

Table 1. Semantics of PASL, where $\mathcal{M} = (H, R, \epsilon, v)$.

Example 2.4. Other concrete separation algebras resemble the construction of Example 2.3 without fitting it precisely:

- Finite set of locations: The concrete memory model of a 32-bit machine [38] has as its locations the set of integers $[0 \ldots 2^{32})$.
- Total functions: Markings of Petri nets [47] without capacity constraints are simply multisets. They may be considered as separation algebras [14] by taking *Loc* to be *Places* and $(V, \star)$ to be the set of natural numbers with addition, then considering the set of *total* functions $Places \to \mathbb{N}$, with $\circ$ defined as usual (hence, as multiset union), and ϵ as the constant 0 function. If there is a global capacity constraint κ then we let $i \star j$ be undefined if $i + j > \kappa$, and hence $\circ$ becomes undefined also in the usual way.
  Note that this example can only be made to exactly fit the construction of Example 2.3 if we restrict ourselves to *markings of infinite Petri nets with finite numbers of tokens*. In this case we would consider a place without tokens to have an undefined map, rather than map to 0, and set V to be the positive integers.
- Constraints on functions: The endpoint heaps of [65] are only those partial functions that are *dual*, *irreflexive* and *injective* (we refer to the citation for the definition of these properties). Similarly, if the places of a Petri net comes equipped with a capacity constraint function $\kappa : Places \to \mathbb{N}$, we consider only those functions compatible with those constraints.

The examples above, which we do not claim to be exhaustive, justify the study of the abstract properties shared by these concrete semantics. We hence now turn to the logic PASL, which has semantics in any separation algebra (Definition 2.1). In this paper we prefer to express PASL semantics in the style of *ternary relations*, which are standard in substructural logic [1] and in harmony with the most important work preceding this paper [35]. We give the ternary relations version of Definition 2.1, easily seen to be equivalent, as follows.

Definition 2.5. A *PASL Kripke relational frame* is a triple (H, R, ϵ), where H is a non-empty set of *worlds*, $R \subseteq H \times H \times H$, and $\epsilon \in H$, satisfying the following conditions for all h_1, h_2, h_3, h_4, h_5 in H:

identity: $R(h_1, \epsilon, h_2) \Leftrightarrow h_1 = h_2$
commutativity: $R(h_1, h_2, h_3) \Leftrightarrow R(h_2, h_1, h_3)$
associativity: $(R(h_1, h_5, h_4)\ \&\ R(h_2, h_3, h_5)) \Rightarrow \exists h_6.(R(h_6, h_3, h_4)\ \&\ R(h_1, h_2, h_6))$
cancellativity: $(R(h_1, h_2, h_3)\ \&\ R(h_1, h_4, h_3)) \Rightarrow h_2 = h_4$
partial-determinism: $(R(h_1, h_2, h_3)\ \&\ R(h_1, h_2, h_4)) \Rightarrow h_3 = h_4$.

A *PASL Kripke relational model* is a tuple (H, R, ϵ, v) of a PASL Kripke relational frame (H, R, ϵ) and a *valuation* function $v : Var \to \mathcal{P}(H)$ (where $\mathcal{P}(H)$ is the power set of H). The forcing relation $\Vdash$ between a model $\mathcal{M} = (H, R, \epsilon, v)$ and a formula is defined in Table 1, where we write $\mathcal{M}, h \nVdash A$ for the negation of $\mathcal{M}, h \Vdash A$. Given a model $\mathcal{M} = (H, R, \epsilon, v)$, a formula is *true at (world) h* iff $\mathcal{M}, h \Vdash A$. The formula A is *valid* iff it is true at all worlds of all models.

2.2 The labelled sequent calculus $\mathrm{LS}_{\mathrm{PASL}}$

Let $LVar$ be an infinite set of *label variables*, and let the set $\mathcal{L}$ of *labels* be $LVar \cup \{\epsilon\}$, where ϵ is a label constant not in $LVar$; here we overload the notation for the identity world in the semantics. Labels will be denoted by lower-case letters such as a, b, x, y, z. A *labelled formula* is a pair $a : A$ of a label a and formula A. As usual in a labelled sequent calculus, one needs to incorporate Kripke relations explicitly into the sequents. This is achieved via the syntactic notion of *relational atoms*, which have the form of either $a = b$ (*equality*), $a \neq b$ (*inequality*), or a *ternary relational atom* $(a, b \triangleright c)$ standing for $R(a, b, c)$, where a, b, c are labels. A *sequent* takes the form

$$\mathcal{G}; \Gamma \vdash \Delta$$

where $\mathcal{G}$ is a set of relational atoms, and Γ and Δ are *sets* of labelled formulae. We also use the symbol ";" inside $\mathcal{G}$, Γ and Δ to indicate set union: for example, $\Gamma; A$ is $\Gamma \cup \{A\}$. Given $\mathcal{G}$, we denote by $E(\mathcal{G})$ the set of equations occurring in $\mathcal{G}$.

We now abuse the sequent turnstile slightly to write $E \vdash s = t$, where E is a (possibly infinite) set of equations, to denote an *equality judgment* under the assumption E, defined inductively as follows:

$$\frac{(s = t) \in E}{E \vdash s = t} \qquad \frac{}{E \vdash s = s} \qquad \frac{E \vdash s = t}{E \vdash t = s} \qquad \frac{E \vdash s = t \qquad E \vdash t = u}{E \vdash s = u}$$

It is easy to see that $E \vdash s = t$ iff $E' \vdash s = t$ for a finite subset E' of E. Note that an equality judgement is not a sequent but abuses the sequent turnstile to keep track of equalities.

As we shall soon see, working within a labelled sequent calculus framework allows us to synthesise, in a generic way, proof rules that correspond to a variety of different properties of separation algebras, and their extensions. However we first must introduce a core logic, a sublogic of BBI (and hence, of PASL) which we call BBI^{-}, which consists only of identity, cut, logical rules and the structural rules NEq and EM. The proof system for this sublogic is presented in Figure 1. The structural rule EM is essentially a form of cut on equality predicates. The rules NEq and EM are admissible for BBI^{-} and many of its extensions, but will be needed for some extensions, such as the extension of PASL with *splittability*. Note that the equality judgment $E(\mathcal{G}) \vdash w = w'$ is not a premise requiring proof, but rather a condition for the rule id. Therefore the rules id, $\bot L$, $\top R$, $\top^* R$, NEq are *zero-premise* rules. In the rules $*R$ and $-\!\!* L$, the respective principal formulae $z : A * B$ and $y : A -\!\!* B$ also occur in the premises. This is to ensure that contraction is admissible, which is essential to obtain cut-elimination.

Given a relational frame (H, R, ϵ), a function $\rho : \mathcal{L} \to H$ from labels to worlds is a *label mapping* iff it satisfies $\rho(\epsilon) = \epsilon$, mapping the label constant ϵ to the identity world $\epsilon \in H$. Intuitively, a labelled formula $a : A$ means that formula A is true in world $\rho(a)$. Thus we define an *extended PASL Kripke relational model* $(H, R, \epsilon, v, \rho)$ as a model equipped with a label mapping.

Definition 2.6 (Sequent Falsifiability). A sequent $\mathcal{G}; \Gamma \vdash \Delta$ is *falsifiable in an extended model* $\mathcal{M} = (H, R, \epsilon, v, \rho)$ if for every $x : A \in \Gamma$, $(a, b \triangleright c) \in \mathcal{G}$, and for every $y : B \in \Delta$, we have each of $\mathcal{M}, \rho(x) \Vdash A$ and $R(\rho(a), \rho(b), \rho(c))$ and $\mathcal{M}, \rho(y) \nVdash B$. It is *falsifiable* if it is falsifiable in some extended model.

Synthesising structural rules from frame axioms. We now define extensions of the sublogic BBI^{-} via first-order axioms that correspond to various semantic conditions used to define PASL and its variations. For this work, we consider only axioms that are closed formulae of the following general axiom form where k, l, m, n, p are natural numbers:

$$\forall x_1, \ldots, x_m.(s_1 = t_1 \,\&\, \cdots \,\&\, s_p = t_p \,\&\, S_1 \,\&\, \cdots \,\&\, S_k \;\Rightarrow\; \exists y_1, \ldots, y_n.(T_1 \,\&\, \cdots \,\&\, T_l)) \tag{1}$$

Identity and Cut:

$$\frac{E(\mathcal{G}) \vdash w = w'}{\mathcal{G}; \Gamma; w : p \vdash w' : p; \Delta}\ id \qquad \frac{\mathcal{G}; \Gamma \vdash x : A; \Delta \qquad \mathcal{G}; \Gamma; x : A \vdash \Delta}{\mathcal{G}; \Gamma \vdash \Delta}\ cut$$

Logical Rules:

$$\frac{}{\mathcal{G}; \Gamma; w : \bot \vdash \Delta}\ \bot L \qquad \frac{\mathcal{G}; w = \epsilon; \Gamma \vdash \Delta}{\mathcal{G}; \Gamma; w : \top^* \vdash \Delta}\ \top^* L \qquad \frac{}{\mathcal{G}; \Gamma \vdash w : \top; \Delta}\ \top R \qquad \frac{E(\mathcal{G}) \vdash w = \epsilon}{\mathcal{G}; \Gamma \vdash w : \top^*; \Delta}\ \top^* R$$

$$\frac{\mathcal{G}; \Gamma; w : A; w : B \vdash \Delta}{\mathcal{G}; \Gamma; w : A \wedge B \vdash \Delta}\ \wedge L \qquad \frac{\mathcal{G}; \Gamma \vdash w : A; \Delta \qquad \mathcal{G}; \Gamma \vdash w : B; \Delta}{\mathcal{G}; \Gamma \vdash w : A \wedge B; \Delta}\ \wedge R$$

$$\frac{\mathcal{G}; \Gamma \vdash w : A; \Delta \qquad \mathcal{G}; \Gamma; w : B \vdash \Delta}{\mathcal{G}; \Gamma; w : A \to B \vdash \Delta}\ \to L \qquad \frac{\mathcal{G}; \Gamma; w : A \vdash w : B; \Delta}{\mathcal{G}; \Gamma \vdash w : A \to B; \Delta}\ \to R$$

$$\frac{\mathcal{G}; (x, y \triangleright z); \Gamma; x : A; y : B \vdash \Delta}{\mathcal{G}; \Gamma; z : A * B \vdash \Delta}\ *L \qquad \frac{\mathcal{G}; (x, z \triangleright y); \Gamma; x : A \vdash y : B; \Delta}{\mathcal{G}; \Gamma \vdash z : A \mathrel{-\!\!*} B; \Delta}\ \mathrel{-\!\!*} R$$

$$\frac{\mathcal{G}; (x, y \triangleright z'); \Gamma \vdash x : A; z : A * B; \Delta \qquad \mathcal{G}; (x, y \triangleright z'); \Gamma \vdash y : B; z : A * B; \Delta \qquad E(\mathcal{G}) \vdash z = z'}{\mathcal{G}; (x, y \triangleright z'); \Gamma \vdash z : A * B; \Delta}\ *R$$

$$\frac{\mathcal{G}; (x, y' \triangleright z); \Gamma; y : A \mathrel{-\!\!*} B \vdash x : A; \Delta \qquad \mathcal{G}; (x, y' \triangleright z); \Gamma; y : A \mathrel{-\!\!*} B; z : B \vdash \Delta \qquad E(\mathcal{G}) \vdash y = y'}{\mathcal{G}; (x, y' \triangleright z); \Gamma; y : A \mathrel{-\!\!*} B \vdash \Delta}\ \mathrel{-\!\!*} L$$

Structural Rules:

$$\frac{E(\mathcal{G}) \vdash u = v}{\mathcal{G}; u \neq v; \Gamma \vdash \Delta}\ NEq \qquad \frac{\mathcal{G}; x = y; \Gamma \vdash \Delta \qquad \mathcal{G}; x \neq y; \Gamma \vdash \Delta}{\mathcal{G}; \Gamma \vdash \Delta}\ EM$$

Side conditions:
In $*L$ and $\mathrel{-\!\!*} R$, the labels x and y do not occur in the conclusion.

Fig. 1. Inference rules for the core sublogic BBI$^-$.

Note that where k, l, or p are 0, we assume the empty conjunction is $\top$. We further require the following conditions:

- each S_i, for $1 \leq i \leq k$, is either a ternary relational atom or an inequality;
- each T_i, for $1 \leq i \leq l$, is a relational atom;
- every label variable in $\bigcup_{1 \leq i \leq k} S_i$ occurs only once;
- if S_i, for $1 \leq i \leq k$, is a ternary relational atom, then ϵ does not occur in S_i.

We call axioms of this form *frame axioms*. A frame axiom can be given semantics in terms of Kripke frames, following the standard classical first-order interpretation (see e.g., [23]). Recall that a first-order model is a pair (D, I) of a non-empty *domain* D and an *interpretation* function I that associates each constant in the first-order language to a member of D and every n-ary relation symbol to an n-ary relation over D. When interpreting first-order formulae with free variables, we additionally need to specify the valuation of the free variables, i.e., a mapping of the free variables to elements of D. The notion of truth of a first-order formula (under a model and a given valuation of its free variables) is standard and the reader is referred to, e.g., [23] for details.

Definition 2.7. A Kripke frame (H, R, ϵ) *satisfies* a frame axiom F iff F is true in the first order model (H, I), where I is the interpretation function that associates the symbol ϵ to the label constant ϵ in the set H, the predicate symbol $\triangleright$ to the relation R, and the equality symbol $=$ to the identity

relation over H. A Kripke frame satisfies a set $\mathcal{F}$ of frame axioms iff it satisfies every frame axiom in the set.

A frame axiom such as the one from Formula (1) induces the following *general structural rule*:

$$\frac{\mathcal{G}; S_1; \ldots; S_k; T_1; \ldots; T_l; \Gamma \vdash \Delta \qquad E(\mathcal{G}) \vdash s_1 = t_1 \qquad \cdots \qquad E(\mathcal{G}) \vdash s_p = t_p}{\mathcal{G}; S_1; \ldots; S_k; \Gamma \vdash \Delta}$$

where the existential condition in Equation 1 becomes a side-condition that the existentially quantified variables $y_1, \ldots, y_n$ must be fresh label variables not occurring in the conclusion of the rule.

Example 2.8. The semantic clauses in Definition 2.5 can be captured by the following frame axioms:

identity 1: $\forall h_1, h_2, h_3.(h_2 = \epsilon \;\&\; R(h_1, h_2, h_3)) \Rightarrow h_1 = h_3$
identity 2: $\forall h_1, h_2. h_1 = h_2 \Rightarrow R(h_1, \epsilon, h_2)$
commutativity: $\forall h_1, h_2, h_3. R(h_1, h_2, h_3) \Rightarrow R(h_2, h_1, h_3)$
associativity: $\forall h_1, h_2, h_3, h_4, h_5, h'_5.$
$(h_5 = h'_5 \;\&\; R(h_1, h_5, h_4) \;\&\; R(h_2, h_3, h'_5)) \Rightarrow \exists h_6.\ (R(h_6, h_3, h_4) \;\&\; R(h_1, h_2, h_6))$
cancellativity: $\forall h_1, h_2, h_3, h'_1, h'_3.$
$(h_1 = h'_1 \;\&\; h_3 = h'_3 \;\&\; R(h_1, h_2, h_3) \;\&\; R(h'_1, h_4, h'_3)) \Rightarrow h_2 = h_4$
partial-determinism: $\forall h_1, h'_1, h_2, h'_2, h_3, h_4.$
$(h_1 = h'_1 \;\&\; h_2 = h'_2 \;\&\; R(h_1, h_2, h_3) \;\&\; R(h'_1, h'_2, h_4)) \Rightarrow h_3 = h_4.$

These frame axioms are mostly a straightforward translation from the semantic clauses of Definition 2.5 into the syntactic form, replacing the relation R with the predicate symbol $\triangleright$. It is trivial to show that the Kripke frames defined in Definition 2.5 satisfy the frame axioms above. However, notice that the syntactic form of the frame axioms does not allow more than one occurrence of a variable in the left hand side of the implications. Thus, for each semantic clause of Definition 2.5, we need to identify each world that occurs multiple (say, n) times on the left hand side of the implications, make n distinct copies of that world, and add equalities relating them. If ϵ occurs in a ternary relational atom on the left hand side, we need to create a fresh (universally quantified) variable, e.g., w, and add that $w = \epsilon$.

Take the associativity axiom in Definition 2.5 as an example:

$$\forall h_1, h_2, h_3, h_4, h_5.(R(h_1, h_5, h_4) \,\&\, R(h_2, h_3, h_5) \;\Rightarrow\; \exists h_6.(R(h_6, h_3, h_4) \,\&\, R(h_1, h_2, h_6)))$$

The world h_5 occurs twice on the left hand side, so we make two copies of it: h_5 and h'_5. The corresponding axiom in frame axiom form is then:

$$\forall h_1, h_2, h_3, h_4, h_5.(h_5 = h'_5 \,\&\, (h_1, h_5 \triangleright h_4) \,\&\, (h_2, h_3 \triangleright h'_5) \;\Rightarrow\; \exists h_6.((h_6, h_3 \triangleright h_4) \,\&\, (h_1, h_2 \triangleright h_6)))$$

We may then synthesise the following structural rule for associativity:

$$\frac{\mathcal{G}; (u, y \triangleright x); (v, w \triangleright y'); (z, w \triangleright x); (u, v \triangleright z); \Gamma \vdash \Delta \qquad E(\mathcal{G}) \vdash y = y'}{\mathcal{G}; (u, y \triangleright x); (v, w \triangleright y'); \Gamma \vdash \Delta}\, A$$

with the "freshness" side-condition that z does not appear in the conclusion.

Note that this rule is applicable on $(u, y \triangleright x), (v, w \triangleright y)$, as $E(\mathcal{G}) \vdash y = y$ is trivial.

From the frame axioms in Example 2.8 above, we obtain the structural rules of Figure 2. The identity axiom, as it is a bi-implication, gives rise to two rules E and U. The commutativity axiom translates to rule *Com*, associativity to A, cancellativity to C and partial determinism to P. The proof system $\mathrm{LS}_{\mathrm{PASL}}$ is defined to be the rules of Figure 1 for the sublogic BBI^{-}, plus the synthesised structural rules of Figure 2.

$$\frac{\mathcal{G}; (x, y \triangleright z); x = z; \Gamma \vdash \Delta \qquad E(\mathcal{G}) \vdash y = \epsilon}{\mathcal{G}; (x, y \triangleright z); \Gamma \vdash \Delta}\ E \qquad \frac{\mathcal{G}; (x, \epsilon \triangleright y); \Gamma \vdash \Delta \qquad E(\mathcal{G}) \vdash x = y}{\mathcal{G}; \Gamma \vdash \Delta}\ U$$

$$\frac{\mathcal{G}; (x, y \triangleright z); (y, x \triangleright z); \Gamma \vdash \Delta}{\mathcal{G}; (x, y \triangleright z); \Gamma \vdash \Delta}\ Com$$

$$\frac{\mathcal{G}; (u, y \triangleright x); (v, w \triangleright y'); (z, w \triangleright x); (u, v \triangleright z); \Gamma \vdash \Delta \qquad E(\mathcal{G}) \vdash y = y'}{\mathcal{G}; (u, y \triangleright x); (v, w \triangleright y'); \Gamma \vdash \Delta}\ A$$

$$\frac{\mathcal{G}; (x, y \triangleright z); (x', w \triangleright z'); y = w; \Gamma \vdash \Delta \qquad E(\mathcal{G}) \vdash x = x' \qquad E(\mathcal{G}) \vdash z = z'}{\mathcal{G}; (x, y \triangleright z); (x', w \triangleright z'); \Gamma \vdash \Delta}\ C$$

$$\frac{\mathcal{G}; (w, x \triangleright y); (w', x' \triangleright z); y = z; \Gamma \vdash \Delta \qquad E(\mathcal{G}) \vdash w = w' \qquad E(\mathcal{G}) \vdash x = x'}{\mathcal{G}; (w, x \triangleright y); (w', x' \triangleright z); \Gamma \vdash \Delta}\ P$$

Side conditions:
In A, the label z does not occur in the conclusion.

Fig. 2. Structural rules synthesised from frame axioms. Together with Figure 1, these rules define the sequent calculus $\mathrm{LS}_{\mathrm{PASL}}$ for the logic PASL.

We remark here that it is not always obvious what the effect of each semantic property will be on the set of valid formulae; for example it was only recently discovered [45] that cancellativity does not affect validity in the presence of the other properties. This lends weight to the suggestion of [26] that cancellativity should be omitted from the definition of separation algebra; see also our examples of concrete separation algebras without cancellativity in Section 4.5. It is nonetheless harmless to include it in our rules, and may be useful for some extensions of PASL, as we discuss in Section 8.

It is easy to check that the following hold:

Theorem 2.9 (Soundness of the general structural rule). *Every synthesised instance of the general structural rule is sound with respect to the Kripke relational frames with the corresponding frame axiom.*

Corollary 2.10 (Soundness of $\mathrm{LS}_{\mathrm{PASL}}$). *For any formula A, and for an arbitrary label w, if the labelled sequent $\vdash w : A$ is derivable in $\mathrm{LS}_{\mathrm{PASL}}$ then A is valid.*

Note that the soundness of $\mathrm{LS}_{\mathrm{PASL}}$ has been formally verified via the interactive theorem prover Isabelle [31].

We will give the name LS_{G} to the general proof system that extends the rules of Figure 1 with any set of structural rules synthesised from frame axioms. A proof system consisting of the rules in Figure 1 plus a finite number of instances of the general structural rule is called an *instance* of LS_{G}.

2.3 Cut-elimination for the general proof system LS_{G}

In this section we see that the *cut* rule of Figure 1 is admissible in the general nested sequent calculus LS_{G}. Since cut-admissibility can be obtained indirectly from the cut-free completeness proof in the next section, we do not give full details here.

A *label substitution* is a mapping from label variables to labels. The *domain* of a substitution θ is the set $\{x \mid \theta(x) \neq x\}$. We restrict to substitutions with only finite domains. We use the notation $[a_1/x_1, \ldots, a_n/x_n]$ to denote a substitution mapping variables x_i to labels a_i. Application of a substitution θ to a term or a formula is written in a postfix notation, e.g., $F[a/x]$ denotes a

formula obtained by substituting a for every free occurrence of x in F. This notation generalises straightforwardly to applications of substitutions to (multi)sets of formulas, relational atoms and sequents.

We will first present a substitution lemma for instance systems of $\mathrm{LS_G}$. This requires the following lemma.

Lemma 2.11 (Substitution in equality judgments). *Given any set E of equality relational atoms, any labels x, y and z, and any label variable w, if $E \vdash x = y$, then $E[z/w] \vdash (x = y)[z/w]$, where every occurrence of w is replaced with z.*

In the substitution lemma below we use $ht(\Pi)$ to denote the height of the derivation Π.

Lemma 2.12 (Substitution for $\mathrm{LS_G}$). *In any instance system of $\mathrm{LS_G}$, if Π is a derivation for the sequent $\mathcal{G}; \Gamma \vdash \Delta$, then there is a derivation Π' of the sequent $\mathcal{G}[y/x]; \Gamma[y/x] \vdash \Delta[y/x]$ where every occurrence of label variable x is replaced by label y, such that $ht(\Pi') \leq ht(\Pi)$.*

Since $\mathrm{LS_G}$ does not involve explicit label substitutions in the rules anymore, the proof for the substitution lemma is actually simpler than the proof for $\mathrm{LS_{BBI}}$ [35], to which we refer interested readers.

The admissibility of weakening for any instance system of $\mathrm{LS_G}$ can be proved by a simple induction on the length of the derivation. The invertibility of the inference rules in $\mathrm{LS_G}$ can be proved in a similar way as for $\mathrm{LS_{BBI}}$ [35], which uses similar techniques as for *G3c* [49]. The proofs for the following lemmas are a straightforward adaptation of similar proofs from [35] so we omit details here.

Lemma 2.13 (Weakening admissibility of $\mathrm{LS_G}$). *If $\mathcal{G}; \Gamma \vdash \Delta$ is derivable in any instance system of $\mathrm{LS_G}$, then for any set $\mathcal{G}'$ of relational atoms, and any set Γ' and Δ' of labelled formulae, the sequent $\mathcal{G}; \mathcal{G}'; \Gamma; \Gamma' \vdash \Delta; \Delta'$ is derivable with the same height in that instance of $\mathrm{LS_G}$.*

Lemma 2.14 (Invertibility of rules in $\mathrm{LS_G}$). *In any instance system of $\mathrm{LS_G}$, if Π is a cut-free derivation of the conclusion of a rule, then there is a cut-free derivation for each premise, with height at most $ht(\Pi)$.*

Since the sequents in our definition consists of sets, the admissibility of contraction is trivial and we do not state it as a lemma here. The cut-elimination proof here is an adaptation of that of [35]. The proof in our case is simpler, as our cut rule does not split context, and our inference rules do not involve explicit label substitutions. We hence state the theorem here without proof:

Theorem 2.15 (Cut-elimination for $\mathrm{LS_G}$). *For any instance of $\mathrm{LS_G}$, if a formula is derivable in that instance, then it is derivable without using cut in that instance.*

3 COUNTER-MODEL CONSTRUCTION FOR $\mathrm{LS_G}$

We now give a counter-model construction procedure that works for all finite instances (systems with finite rules) of the general proof system $\mathrm{LS_G}$, and hence establishes their completeness.

As the counter-model construction involves infinite sets and sequents, we extend the definition of equality judgment:

Definition 3.1. Given a (possibly infinite) set $\mathcal{G}$ of relational atoms, the judgment $E(\mathcal{G}) \vdash x = y$ holds iff $E(\mathcal{G}_f) \vdash x = y$ holds for some finite $\mathcal{G}_f \subseteq \mathcal{G}$.

Given a set $\mathcal{G}$ of relational atoms, we define the relation $=_{\mathcal{G}}$ as follows: $a =_{\mathcal{G}} b$ iff $E(\mathcal{G}) \vdash a = b$. We next state a lemma which is an immediate result from our equality judgment rules and will be useful in our counter-model construction later:

Lemma 3.2. *Given a set $\mathcal{G}$ of relational atoms, the relation $=_{\mathcal{G}}$ is an equivalence relation on the set of labels.*

The equivalence relation $=_{\mathcal{G}}$ partitions $\mathcal{L}$ into equivalence classes $[a]_{\mathcal{G}}$ for each label $a \in \mathcal{L}$:

$$[a]_{\mathcal{G}} = \{a' \in \mathcal{L} \mid a =_{\mathcal{G}} a'\}.$$

The counter-model construction is essentially a procedure to saturate a sequent by applying all backward applicable rules repeatedly. The aim is to obtain an infinite saturated sequent from which a counter-model can be extracted. We first define a list of required conditions for such an infinite sequent which would allow the counter-model construction.

Definition 3.3 (General Hintikka sequent). A labelled sequent $\mathcal{G}; \Gamma \vdash \Delta$ is a *general Hintikka sequent* if it satisfies the following conditions for any formulae A, B and any labels $a, b, c, d, e, z \in \mathcal{L}$:

(1) It is not the case that $a : A \in \Gamma$, $b : A \in \Delta$ and $a =_{\mathcal{G}} b$.
(2) $a : \bot \notin \Gamma$ and $a : \top \notin \Delta$.
(3) If $a : \top^* \in \Gamma$ then $a =_{\mathcal{G}} \epsilon$.
(4) If $a : \top^* \in \Delta$ then $a \neq_{\mathcal{G}} \epsilon$.
(5) If $a : A \wedge B \in \Gamma$ then $a : A \in \Gamma$ and $a : B \in \Gamma$.
(6) If $a : A \wedge B \in \Delta$ then $a : A \in \Delta$ or $a : B \in \Delta$.
(7) If $a : A \to B \in \Gamma$ then $a : A \in \Delta$ or $a : B \in \Gamma$.
(8) If $a : A \to B \in \Delta$ then $a : A \in \Gamma$ and $a : B \in \Delta$.
(9) If $z : A * B \in \Gamma$ then $\exists x, y, z'$ s.t. $(x, y \triangleright z') \in \mathcal{G}$, $z =_{\mathcal{G}} z'$, $x : A \in \Gamma$ and $y : B \in \Gamma$.
(10) If $z : A * B \in \Delta$ then $\forall x, y, z'$ if $(x, y \triangleright z') \in \mathcal{G}$ and $z =_{\mathcal{G}} z'$ then $x : A \in \Delta$ or $y : B \in \Delta$.
(11) If $z : A \mathrel{-\!\!*} B \in \Gamma$ then $\forall x, y, z'$ if $(x, z' \triangleright y) \in \mathcal{G}$ and $z =_{\mathcal{G}} z'$, then $x : A \in \Delta$ or $y : B \in \Gamma$.
(12) If $z : A \mathrel{-\!\!*} B \in \Delta$ then $\exists x, y, z'$ s.t. $(x, z' \triangleright y) \in \mathcal{G}$, $z =_{\mathcal{G}} z'$, $x : A \in \Gamma$ and $y : B \in \Delta$.
(13) It is not the case that $a \neq b \in \mathcal{G}$ and $a =_{\mathcal{G}} b$.
(14) Either $a \neq b \in \mathcal{G}$ or $a =_{\mathcal{G}} b$.
(15) Given a frame axiom of the form

$$\forall x_1, \ldots, x_m.(s_1 = t_1 \,\&\, \cdots \,\&\, s_p = t_p \,\&\, S_1 \,\&\, \cdots \,\&\, S_k \;\Rightarrow\; \exists y_1, \ldots, y_n.(T_1 \,\&\, \cdots \,\&\, T_l))$$

for any labels $a_1, \cdots, a_m \in \mathcal{L}$ and substitution $\theta = [a_1/x_1, \cdots, a_m/x_m]$, if $\bigcup_{1 \leq i \leq k}\{S_i\theta\} \subseteq \mathcal{G}$ and $s_i\theta =_{\mathcal{G}} t_i\theta$ for $1 \leq i \leq p$, then there exist $b_1, \cdots, b_n \in \mathcal{L}$ and substitution $\sigma = [b_1/y_1, \ldots, b_n/y_n]$ such that $\bigcup_{1 \leq i \leq l}\{(T_i\theta)\sigma\} \subseteq \mathcal{G}$.

In condition 15, the variables $x_1, \cdots, x_m$ and $y_1, \cdots, y_n$ are schematic variables, i.e., symbols that belong to the metalanguage, and the substitutions θ and σ replace these schematic variables with labels. Since θ and σ have disjoint domains, we have that $(x_i\theta)\sigma = x_i\theta$ and $(y_i\theta)\sigma = y_i\sigma$. These will be useful in the proofs below.

We are often interested in some particular Hintikka sequents that correspond to certain frame axioms. Given a set $\mathcal{F}r$ of frame axioms, a *$\mathcal{F}r$-Hintikka sequent* is an instance of the general Hintikka sequent where condition 15 holds for each frame axiom in $\mathcal{F}r$. We say a Kripke frame satisfies $\mathcal{F}r$ when every frame axiom in $\mathcal{F}r$ is satisfied by the Kripke frame.

Next we show a parametric Hintikka lemma: a Hintikka sequent parameterised over a set of frame axioms gives a Kripke relational frame where the set of frame axioms are satisfied and the formulae in the right hand side of the sequent are false.

Lemma 3.4. *Given a set $\mathcal{F}r$ of frame axioms, every $\mathcal{F}r$-Hintikka sequent is falsifiable in some Kripke frame satisfying $\mathcal{F}r$.*

Proof. Let $\mathcal{G}; \Gamma \vdash \Delta$ be an arbitrary $\mathcal{F}r$-Hintikka sequent. We construct an extended model $\mathcal{M} = (H, R, \epsilon_{\mathcal{G}}, v, \rho)$ as follows:

- $H = \{[a]_{\mathcal{G}} \mid a \in \mathcal{L}\}$;
- $R([a]_{\mathcal{G}}, [b]_{\mathcal{G}}, [c]_{\mathcal{G}})$ iff $\exists a', b', c'$ s.t. $(a', b' \triangleright c') \in \mathcal{G}, a =_{\mathcal{G}} a', b =_{\mathcal{G}} b', c =_{\mathcal{G}} c'$;
- $\epsilon_{\mathcal{G}} = [\epsilon]_{\mathcal{G}}$;
- $v(p) = \{[a]_{\mathcal{G}} \mid a : p \in \Gamma\}$ for every $p \in Var$;
- $\rho(a) = [a]_{\mathcal{G}}$ for every $a \in \mathcal{L}$.

To reduce clutter, we shall drop the subscript $\mathcal{G}$ in $[a]_{\mathcal{G}}$.

We first show that the $\mathcal{F}r$-Hintikka sequent $\mathcal{G}; \Gamma \vdash \Delta$ gives rise to a Kripke relational frame that satisfies the frame axioms in $\mathcal{F}r$. Take an arbitrary frame axiom $F \in \mathcal{F}r$ of form

$$\forall x_1, \ldots, x_m.(s_1 = t_1 \,\&\, \cdots \,\&\, s_p = t_p \,\&\, S_1 \,\&\, \cdots \,\&\, S_k \;\Rightarrow\; \exists y_1, \ldots, y_n.(T_1 \,\&\, \cdots \,\&\, T_l))$$

We have to show that the frame axiom F above is true in the first-order model (H, I), where I is the interpretation function such that $\epsilon^I = [\epsilon]_{\mathcal{G}}$ and $\triangleright^I = R$. That is, for an arbitrary first-order valuation $\mu = \{x_1 \mapsto [a_1], \ldots, x_m \mapsto [a_m]\}$, we assume that $s_i = t_i$ for $1 \leq i \leq q$ and S_i holds for $1 \leq i \leq k$, in the first-order interpretation. We then show that μ can be extended with some valuation for the variables $y_1, \cdots, y_n$ such that under the new valuation, T_i holds $1 \leq i \leq l$ in the first-order interpretation.

We construct a series of substitutions as follows:

- We start with the substitution $\theta = [a_1/x_1, \cdots, a_n/x_n]$. We have that for each $1 \leq i \leq p$, $s_i\theta =_{\mathcal{G}} t_i\theta$ from the assumption that $s_i = t_i$ holds under μ.
- For each $1 \leq i \leq k$, if S_i has the form $x_u \neq x_v$ where $u, v \in \{1, \cdots, m\}$, then by assumption, $[a_u] \neq [a_v]$ holds. By condition (13) and (14) in Definition 3.3, either $x_u\theta =_{\mathcal{G}} x_v\theta$ or $x_u\theta \neq x_v\theta \in \mathcal{G}$. Since the former contradicts with the assumption, we have $x_u\theta \neq x_v\theta \in \mathcal{G}$.
- For each $1 \leq i \leq k$, if S_i has the form $(x_u, x_v \triangleright x_w)$ where $u, v, w \in \{1, \cdots, m\}$, then by assumption, $R([a_u], [a_v], [a_w])$ holds. This means that there are some a'_u, a'_v, a'_w such that $a_u =_{\mathcal{G}} a'_u, a_v =_{\mathcal{G}} a'_v, a_w =_{\mathcal{G}} a'_w$, and $(a'_u, a'_v \triangleright a'_w) \in \mathcal{G}$. We construct a new substitution θ' where θ' is the same as the previous substitution θ_0 except θ' replaces x_u, x_v, x_w respectively with a'_u, a'_v, a'_w. The following facts hold under the new substitution θ': (1) $S_i\theta' \in \mathcal{G}$. (2) For any $1 \leq q \leq p$, $s_q\theta_0 =_{\mathcal{G}} t_q\theta_0$ implies $s_q\theta' =_{\mathcal{G}} t_q\theta'$. If neither of s_q and t_q are one of x_u, x_v, x_w, then the equation is syntactically the same as before. Otherwise, if, for example, s_q is x_u, then it is replaced by a'_u under θ'. But since $a'_u =_{\mathcal{G}} a_u$, by transitivity of $=_{\mathcal{G}}$ (Lemma 3.2), we obtain the equivalence in the first-order interpretation. (3) Since we assume that each label variable only occurs once in $\bigcup_{1 \leq i \leq k} S_i$, and that the constant ϵ does not occur in ternary relations, for any $1 \leq q \leq k$ such that $q \neq i$, $S_q\theta'$ is syntactically equivalent to $S_q\theta_0$, therefore if $S_q\theta_0 \in \mathcal{G}$ then $S_q\theta' \in \mathcal{G}$.

Suppose we start with θ and iteratively construct a new substitution as the last case of the above, and obtain a final substitution θ''. It is easy to establish that for each $1 \leq i \leq p$, $s_i\theta'' =_{\mathcal{G}} t_i\theta''$, and for each $1 \leq i \leq k$, $S_i\theta'' \in \mathcal{G}$. Therefore by condition (15) of Definition 3.3, there exist $b_1, \cdots, b_n \in \mathcal{L}$ and a substitution $\sigma = [b_1/y_1, \cdots, b_n/y_n]$ such that $\bigcup_{1 \leq i \leq l}\{(T_i\theta'')\sigma\} \subseteq \mathcal{G}$. Note also that $x_i\theta =_{\mathcal{G}} x_i\theta''$ for each $1 \leq i \leq m$. Now we show that for each $1 \leq i \leq l$, T_i holds in the first-order model. This is done by extending the first-order valuation μ to μ', where μ' is the same as μ except μ' maps each y_i to $[b_i]$ for $1 \leq i \leq n$. We consider the following cases depending on the shape of T_i:

- T_i has the form $(w = w')$.
  - Suppose further that w is x_u and w' is x_v for some $u, v \in \{1, \cdots, m\}$. We need to show that $x_u = x_v$ under the valuation μ'. Since μ' only differs from μ in the mappings of $\{y_1, \cdots, y_n\}$, we only need to show that $[a_u] = [a_v]$. Since we have $(T_i\theta'')\sigma \in \mathcal{G}$, we

know that $(x_u\theta'')\sigma =_{\mathcal{G}} (x_v\theta'')\sigma$, which means $x_u\theta'' =_{\mathcal{G}} x_v\theta''$. By the construction of θ'', $x_u\theta'' =_{\mathcal{G}} x_u\theta = a_u$ and $x_v\theta'' =_{\mathcal{G}} x_v\theta = a_v$ Thus $[a_u] = [a_v]$ holds.
  - Suppose w is x_u for some $1 \leq u \leq m$ and w' is y_v for some $1 \leq v \leq n$. We show that $x_u = y_v$ under the valuation μ', which means $[a_u] = [b_v]$. Since $(T_i\theta'')\sigma \in \mathcal{G}$, we have $(x_u\theta'')\sigma =_{\mathcal{G}} (y_v\theta'')\sigma$, which equals $x_u\theta'' =_{\mathcal{G}} y_v\sigma$. Since $x_u\theta'' =_{\mathcal{G}} x_u\theta = a_u$, we have $[a_u] = [b_v]$.
  - If $w \in \{y_1, \cdots, y_n\}$ and $w' \in \{x_1, \cdots, x_m\}$, the case is symmetric to the above case.
  - Suppose w is y_u and w' is y_v for some $u, v \in \{1, \cdots, n\}$. We need to show that $y_u = y_v$ under the valuation μ', which means $[b_u] = [b_v]$. Since we have $(T_i\theta'')\sigma \in \mathcal{G}$, we know that $(y_u\theta'')\sigma =_{\mathcal{G}} (y_v\theta'')\sigma$, which means $y_u\sigma =_{\mathcal{G}} y_v\sigma$. Thus $[b_u] = [b_v]$ holds.
- T_i has the form $(w \neq w')$. This case is similar to the above case.
- T_i has the form $(w, w' \triangleright w'')$.
  - Suppose w is x_t, w' is x_u, w'' is x_v, for some $t, u, v \in \{1, \cdots, m\}$. We need to show that $R(x_t, x_u, x_v)$ holds under the valuation μ', which is equivalent to showing $R([a_t], [a_u], [a_v])$. We already have that $(T_i\theta'')\sigma \in \mathcal{G}$, that is, $(x_t\theta'', x_u\theta'' \triangleright x_v\theta'') \in \mathcal{G}$. By the construction of θ'', we have $x_t\theta'' =_{\mathcal{G}} x_t\theta = a_t$, $x_u\theta'' =_{\mathcal{G}} x_u\theta = a_u$, and $x_v\theta'' =_{\mathcal{G}} x_v\theta = a_v$. Thus the goal holds.
  - If any of w, w', w'' is in $\{y_1, \cdots, y_n\}$, say w is y_u for some $1 \leq u \leq n$, then $(y_u\theta'')\sigma = b_u$ and y_u is mapped to $[b_u]$ under μ'. These cases are similar to the above analysis.

This concludes the part of the proof which shows that the extended model $\mathcal{M}$ is indeed a model based on a Kripke relational frame satisfying $\mathcal{F}r$. We prove next that $\mathcal{G}; \Gamma \vdash \Delta$ is false in $\mathcal{M}$. We need to show the following where $\rho(m) = [m]$:

(1) If $(a, b \triangleright c) \in \mathcal{G}$ then $([a], [b] \triangleright_{\mathcal{G}} [c])$.
(2) If $m : A \in \Gamma$ then $\rho(m) \Vdash A$.
(3) If $m : A \in \Delta$ then $\rho(m) \nVdash A$.

Item (1) follows from the definition of $\triangleright_{\mathcal{G}}$. We prove (2) and (3) simultaneously by induction on the size of A.

Base cases: when A is an atomic proposition p.

- If $m : p \in \Gamma$ then $[m] \in v(p)$ by definition of v, so $[m] \Vdash p$.
- Suppose $m : p \in \Delta$, but $[m] \Vdash p$. Then $m' : p \in \Gamma$, for some m' s.t. $m' =_{\mathcal{G}} m$. This violates condition 1 in Definition 3.3. Thus $[m] \nVdash p$.

Inductive cases: when A is a compound formula we do a case analysis on the main connective of A.

- If $m : A \wedge B \in \Gamma$, by condition 5 in Definition 3.3, $m : A \in \Gamma$ and $m : B \in \Gamma$. By the induction hypothesis, $[m] \Vdash A$ and $[m] \Vdash B$, thus $[m] \Vdash A \wedge B$.
- If $m : A \wedge B \in \Delta$, by condition 6 in Definition 3.3, $m : A \in \Delta$ or $m : B \in \Delta$. By the induction hypothesis, $[m] \nVdash A$ or $[m] \nVdash B$, thus $[m] \nVdash A \wedge B$.
- If $m : A \to B \in \Gamma$, by condition 7 in Definition 3.3, $m : A \in \Delta$ or $m : B \in \Gamma$. By the induction hypothesis, $[m] \nVdash A$ or $[m] \Vdash B$, thus $[m] \Vdash A \to B$.
- If $m : A \to B \in \Delta$, by condition 8 in Definition 3.3, $m : A \in \Gamma$ and $m : B \in \Delta$. By the induction hypothesis, $[m] \Vdash A$ and $[m] \nVdash B$, thus $[m] \nVdash A \to B$.
- If $m : \top^* \in \Gamma$ then $[m] = [\epsilon]$ by condition 2 in Definition 3.3. Since $[\epsilon] \Vdash \top^*$, we obtain $[m] \Vdash \top^*$.
- If $m : \top^* \in \Delta$, by condition 3 in Definition 3.3, $[m] \neq [\epsilon]$ and then $[m] \nVdash \top^*$.

- If $m : A * B \in \Gamma$, by condition 9 in Definition 3.3, $\exists a, b, m'$ s.t. $(a, b \triangleright m') \in \mathcal{G}$ and $[m] = [m']$ and $a : A \in \Gamma$ and $b : B \in \Gamma$. By the induction hypothesis, $[a] \Vdash A$ and $[b] \Vdash B$. Thus $[a], [b] \triangleright_{\mathcal{G}} [m]$ holds and $[m] \Vdash A * B$.
- If $m : A * B \in \Delta$, by condition 10 in Definition 3.3, $\forall a, b, m'$ if $(a, b \triangleright m') \in \mathcal{G}$ and $[m] = [m']$, then $a : A \in \Delta$ or $b : B \in \Delta$. By the induction hypothesis, if such a, b exist, then $[a] \nVdash A$ or $[b] \nVdash B$. For any $[a], [b] \triangleright_{\mathcal{G}} [m]$, there must be some $(a', b' \triangleright m'') \in \mathcal{G}$ s.t. $[a] = [a']$, $[b] = [b']$, $[m] = [m'']$. Then $[a] \nVdash A$ or $[b] \nVdash B$ therefore $[m] \nVdash A * B$.
- If $m : A \mathrel{-\!\!*} B \in \Gamma$, by condition 11 in Definition 3.3, $\forall a, b, m'$ if $(a, m' \triangleright b) \in \mathcal{G}$ and $[m] = [m']$, then $a : A \in \Delta$ or $b : B \in \Gamma$. By the induction hypothesis, if such a, b exists, then $[a] \nVdash A$ or $[b] \Vdash B$. Consider any $[a], [m] \triangleright_{\mathcal{G}} [b]$. There must be some $(a', m'' \triangleright b') \in \mathcal{G}$ s.t. $[a] = [a']$, $[m''] = [m]$, and $[b] = [b']$. So $[a] \nVdash A$ or $[b] \Vdash B$, thus $[m] \Vdash A \mathrel{-\!\!*} B$.
- If $m : A \mathrel{-\!\!*} B \in \Delta$, by condition 12 in Definition 3.3, $\exists a, b, m'$ s.t. $(a, m' \triangleright b) \in \mathcal{G}$ and $[m] = [m']$ and $a : A \in \Gamma$ and $b : B \in \Delta$. By the induction hypothesis, $[a] \Vdash A$ and $[b] \nVdash B$ and $[a], [m] \triangleright_{\mathcal{G}} [b]$ holds, thus $[m] \nVdash A \mathrel{-\!\!*} B$.

This concludes the proof. □

To prove the completeness of an arbitrary finite instance of $\mathrm{LS_G}$, we have to show that any given unprovable sequent can be extended to a Hintikka sequent with the corresponding conditions. To do so we need a way to enumerate all possible backwards applicable rules in a fair way so that every rule will be chosen infinitely often. Traditionally, this is achieved via a fair enumeration strategy of every principal formula of every rule. Since our calculus may contain structural rules with no principal formulae, we need to include them in the enumeration strategy as well. For this purpose, we define a notion of *extended formulae*, given by the grammar:

$$ExF ::= F \mid \mathbb{EM} \mid \mathbb{GS}_1 \mid \cdots \mid \mathbb{GS}_q$$

where F is a formula, and $\mathbb{GS}_1, \cdots, \mathbb{GS}_q$ are "dummy" constant principal formulae corresponding to each structural rule respectively, and $\mathbb{EM}$ is for the rule EM.

Let q be the number of structural rules, synthesised from q frame axioms. In those frame axioms, let k_{max} be the largest number of relational atoms on the left hand side (i.e., Ss in the general axiom), and n_{max} be the largest number of existentially quantified variables (i.e., y's in the general axiom). A scheduler enumerates each combination of left or right of turnstile, a label, an extended formula and at most two relational atoms infinitely often.

Definition 3.5 (Scheduler ϕ). A *schedule* is a tuple $(O, m, ExF, \mathcal{R})$, where O is either 0 (left) or 1 (right), m is a label, ExF is an extended formula and $\mathcal{R}$ is a set of relational atoms such that $|\mathcal{R}| \leq k_{max}$. Let $\mathcal{S}$ denote the set of all schedules. A *scheduler* is a function from natural numbers $\mathcal{N}$ to $\mathcal{S}$. A scheduler ϕ is *fair* if for every schedule S, the set $\{i \mid \phi(i) = S\}$ is infinite.

Lemma 3.6. *There exists a fair scheduler.*

Proof. Our proof is similar to the proof of *fair strategy* of Larchey-Wendling [42]. To adapt their proof, we need to show that the set $\mathcal{S}$ is countable. This follows from the fact that $\mathcal{S}$ is a finite product of countable sets. □

From now on, we shall fix a fair scheduler, which we call ϕ. We assume that the set $\mathcal{L}$ of labels is totally ordered, and its elements can be enumerated as $a_0, a_1, a_2, \ldots$ where $a_0 = \epsilon$. This indexing is used to select fresh labels in our construction of Hintikka sequents.

We say the formula F is not cut-free provable in a finite instance of $\mathrm{LS_G}$ if for an arbitrary label $w \neq \epsilon$, the sequent $\vdash w : F$ is not cut-free derivable in that instance of $\mathrm{LS_G}$. Since we shall be concerned only with cut-free provability, in the following when we mention derivation, we mean cut-free derivation.

For a structural rule obtained from a frame axiom of the usual form (1) we say the structural rule is *backwards applicable* on a sequent $\mathcal{G};\Gamma \vdash \Delta$ iff there is a set $\mathcal{G}' \subseteq \mathcal{G}$ of non-equality relational atoms that matches the schema $S_1, \cdots, S_k$, and $E(\mathcal{G}) \vdash s_1 = t_1, \cdots, E(\mathcal{G}) \vdash s_p = t_p$ holds.

Definition 3.7. Let F be a formula which is not provable in an instance of $\mathrm{LS_G}$. We construct a series of finite sequents $\langle \mathcal{G}_i;\Gamma_i \vdash \Delta_i\rangle_{i\in\mathcal{N}}$ from F where $\mathcal{G}_1 = \Gamma_1 = \emptyset$ and $\Delta_1 = a_1 : F$.

Assuming that $\mathcal{G}_i;\Gamma_i \vdash \Delta_i$ has been defined, we define $\mathcal{G}_{i+1};\Gamma_{i+1} \vdash \Delta_{i+1}$ as follows. Suppose $\phi(i) = (O_i, m_i, ExF_i, \mathcal{R}_i)$. Recall that n_{max} is the maximum number of existentially quantified variables in the frame axioms, take α as $max(n_{max}, 2)$.

- Case $O_i = 0$, ExF_i is a PASL-formula C_i and $m_i : C_i \in \Gamma_i$:
  - $C_i = F_1 \wedge F_2$: then $\mathcal{G}_{i+1} = \mathcal{G}_i$, $\Gamma_{i+1} = \Gamma_i \cup \{m_i : F_1, m_i : F_2\}$, $\Delta_{i+1} = \Delta_i$.
  - $C_i = F_1 \rightarrow F_2$: if there is no derivation for $\mathcal{G}_i;\Gamma_i \vdash m_i : F_1;\Delta_i$ then $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i \cup \{m_i : F_1\}$. Otherwise $\Gamma_{i+1} = \Gamma_i \cup \{m_i : F_2\}$, $\Delta_{i+1} = \Delta_i$. In both cases, $\mathcal{G}_{i+1} = \mathcal{G}_i$.
  - $C_i = \top^*$: then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(m_i = \epsilon)\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.
  - $C_i = F_1 * F_2$: then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(a_{\alpha i}, a_{\alpha i+1} \triangleright m_i)\}$ and $\Gamma_{i+1} = \Gamma_i \cup \{a_{\alpha i} : F_1, a_{\alpha i+1} : F_2\}$, where $a_{\alpha i}, a_{\alpha i+1}$ are fresh labels, and $\Delta_{i+1} = \Delta_i$.
  - $C_i = F_1 \mathrel{-\!\!*} F_2$ and $\mathcal{R}_i = \{(x, m \triangleright y)\} \subseteq \mathcal{G}_i$ and $E(\mathcal{G}_i) \vdash (m = m_i)$: if $\mathcal{G}_i;\Gamma_i \vdash x : F_1;\Delta_i$ has no derivation, then $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i \cup \{x : F_1\}$. Otherwise $\Gamma_{i+1} = \Gamma_i \cup \{y : F_2\}$, $\Delta_{i+1} = \Delta_i$. In both cases, $\mathcal{G}_{i+1} = \mathcal{G}_i$.
- Case $O_i = 1$, ExF_i is a PASL-formula C_i, and $m_i : C_i \in \Delta$:
  - $C_i = F_1 \wedge F_2$: if there is no derivation for $\mathcal{G}_i;\Gamma_i \vdash m_i : F_1;\Delta_i$ then $\Delta_{i+1} = \Delta_i \cup \{m_i : F_1\}$. Otherwise $\Delta_{i+1} = \Delta_i \cup \{m_i : F_2\}$. In both cases, $\mathcal{G}_{i+1} = \mathcal{G}_i$ and $\Gamma_{i+1} = \Gamma_i$.
  - $C_i = F_1 \rightarrow F_2$: then $\Gamma_{i+1} = \Gamma \cup \{m_i : F_1\}$, $\Delta_{i+1} = \Delta_i \cup \{m_i : F_2\}$, and $\mathcal{G}_{i+1} = \mathcal{G}_i$.
  - $C_i = F_1 * F_2$ and $\mathcal{R}_i = \{(x, y \triangleright m)\} \subseteq \mathcal{G}_i$ and $E(\mathcal{G}_i) \vdash (m_i = m)$: if $\mathcal{G}_i;\Gamma_i \vdash x : F_1;\Delta_i$ has no derivation, then $\Delta_{i+1} = \Delta_i \cup \{x : F_1\}$. Otherwise $\Delta_{i+1} = \Delta_i \cup \{y : F_2\}$. In both cases, $\mathcal{G}_{i+1} = \mathcal{G}_i$ and $\Gamma_{i+1} = \Gamma_i$.
  - $C_i = F_1 \mathrel{-\!\!*} F_2$: then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(a_{\alpha i}, m_i \triangleright a_{\alpha i+1})\}$, $\Gamma_{i+1} = \Gamma_i \cup \{a_{\alpha i} : F_1\}$, where $a_{\alpha i}, a_{\alpha i+1}$ are fresh labels, and $\Delta_{i+1} = \Delta_i \cup \{a_{\alpha i+1} : F_2\}$.
- Case $ExF_i = \mathbb{EM}$, and $\mathcal{R}_i = \{(w, w' \triangleright w'')\}$ where $w, w' \in \{a_0, \cdots, a_{\alpha i+\alpha-1}\}$. If there is no derivation for $w = w';\mathcal{G}_i;\Gamma_i \vdash \Delta_i$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{w = w'\}$. Otherwise $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{w \neq w'\}$. In both cases, $\Gamma_{i+1} = \Gamma_i$ and $\Delta_{i+1} = \Delta_i$.
- Case $ExF_i \in \{\mathbb{GS}_1, \cdots, \mathbb{GS}_q\}$. Assume without loss of generality that ExF_i is $\mathbb{GS}_j$ for some $1 \leq j \leq q$ and $\mathbb{GS}_j$ represents a structural rule that corresponds to the frame axiom

  $$\forall x_1, \ldots, x_m.(s_1 = t_1 \,\&\, \cdots \,\&\, s_p = t_p \,\&\, S_1 \,\&\, \cdots \,\&\, S_k \;\Rightarrow\; \exists y_1, \ldots, y_n.(T_1 \,\&\, \cdots \,\&\, T_l))$$

  Suppose we can find some substitution θ such that $\bigcup_{1\leq i\leq k}\{S_i\theta\} = \mathcal{R}_i \subseteq \mathcal{G}_i$. Also suppose that for each $1 \leq u \leq p$, $E(\mathcal{G}_i) \vdash s_u\theta = t_u\theta$. We create n fresh labels $a_{\alpha i}, a_{\alpha i+1}, \cdots, a_{\alpha i+n-1}$ and a substitution $\sigma = [a_{\alpha i}/y_1, \cdots, a_{\alpha i+n-1}/y_n]$. Let $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(T_1\theta)\sigma, \cdots, (T_l\theta)\sigma\}$, $\Gamma_{i+1} = \Gamma_i$ and $\Delta_{i+1} = \Delta_i$. Note that finding such a θ is computable since S_i and $\mathcal{R}_i$ are given. This is a simple case of the ACUI unification problem [56].
- In all other cases, $\mathcal{G}_{i+1} = \mathcal{G}_i$, $\Gamma_{i+1} = \Gamma_i$ and $\Delta_{i+1} = \Delta_i$.

Intuitively, each tuple $(O_i, m_i, ExF_i, \mathcal{R}_i)$ corresponds to a potential (backwards) rule application. If the components of the rule application are in the current sequent, we apply the corresponding rule to these components to obtain the new premises. The indexing of labels guarantees that the choice of $a_{\alpha i}, \cdots, a_{\alpha i+\alpha-1}$ are always fresh for the sequent $\mathcal{G}_i;\Gamma_i \vdash \Delta_i$. The construction in Definition 3.7 non-trivially extends a similar construction of Hintikka Constrained Set of Statements due to [42], in addition to which we have to consider the cases for structural rules.

We say $\mathcal{G}';\Gamma' \vdash \Delta' \subseteq \mathcal{G};\Gamma \vdash \Delta$ iff $\mathcal{G}' \subseteq \mathcal{G}$, $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$. A labelled sequent $\mathcal{G};\Gamma \vdash \Delta$ is *finite* if $\mathcal{G}, \Gamma, \Delta$ are finite sets. Define $\mathcal{G}';\Gamma' \vdash \Delta' \subseteq_f \mathcal{G};\Gamma \vdash \Delta$ iff $\mathcal{G}';\Gamma' \vdash \Delta' \subseteq \mathcal{G};\Gamma \vdash \Delta$ and $\mathcal{G}';\Gamma' \vdash \Delta'$ is finite. If $\mathcal{G};\Gamma \vdash \Delta$ is a finite sequent, it is *non-provable* iff it does not have a derivation in the instance of $\mathrm{LS_G}$. A (possibly infinite) sequent $\mathcal{G};\Gamma \vdash \Delta$ is *finitely non-provable* iff every $\mathcal{G}';\Gamma' \vdash \Delta' \subseteq_f \mathcal{G};\Gamma \vdash \Delta$ is non-provable.

We write $\mathcal{L}_i$ for the set of labels occurring in the sequent $\mathcal{G}_i;\Gamma_i \vdash \Delta_i$. Thus $\mathcal{L}_1 = \{a_1\}$. The following lemma states some properties of the construction of the sequents $\mathcal{G}_i;\Gamma_i \vdash \Delta_i$.

Lemma 3.8. *For any $i \in \mathcal{N}$, the following properties hold:*

(1) *$\mathcal{G}_i;\Gamma_i \vdash \Delta_i$ is non-provable;*
(2) *$\mathcal{L}_i \subseteq \{a_0, a_1, \cdots, a_{\alpha i-1}\}$;*
(3) *$\mathcal{G}_i;\Gamma_i \vdash \Delta_i \subseteq \mathcal{G}_{i+1};\Gamma_{i+1} \vdash \Delta_{i+1}$.*

Proof. Item 1 is based on the fact that the inference rules preserve falsifiability upwards, and we always choose the branch with no derivation. To show item 2, we do an induction on i. Base case, $i = 1$, $\mathcal{L}_1 \subseteq \{a_0, a_1\}$ (recall that $a_0 = \epsilon$). Inductive cases: suppose item 2 holds for any $i \leq n$, for $n+1$, we consider four cases depending on which rule is applied on $\mathcal{G}_i;\Gamma_i \vdash \Delta_i$.

- If $*L$ is applied, then $\mathcal{L}_{i+1} = \mathcal{L}_i \cup \{a_{\alpha i}, a_{\alpha i+1}\} \subseteq \{a_1, \cdots, a_{\alpha i+\alpha-1}\}$ since $\alpha \leq 2$.
- If $-\!\!* R$ is applied, same as above.
- If a structural rule $\mathbb{GS}_j$ is applied, and it generates n' fresh labels. By construction, $n' \leq \alpha$, thus $\mathcal{L}_{i+1} = \mathcal{L}_i \cup \{a_{\alpha i}, a_{\alpha i+1}, \cdots, a_{\alpha i+n'-1}\} \subseteq \{a_1, \cdots, a_{\alpha i+\alpha-1}\}$.
- Otherwise, $\mathcal{L}_{i+1} = \mathcal{L}_i \subseteq \{a_1, \cdots, a_{2i+1}\}$.

Item 3 is obvious from the construction of $\mathcal{G}_{i+1};\Gamma_{i+1} \vdash \Delta_{i+1}$. □

Given the construction of the series of sequents we have just seen above, we define a notion of a *limit sequent*, as the union of every sequent in the series.

Definition 3.9 (Limit sequent). Let F be a formula unprovable in the instance of $\mathrm{LS_G}$. The *limit sequent for F* is the sequent $\mathcal{G}^\omega;\Gamma^\omega \vdash \Delta^\omega$ where $\mathcal{G}^\omega = \bigcup_{i\in\mathcal{N}} \mathcal{G}_i$, $\Gamma^\omega = \bigcup_{i\in\mathcal{N}} \Gamma_i$, and $\Delta^\omega = \bigcup_{i\in\mathcal{N}} \Delta_i$.

The following lemma shows that the limit sequent defined above is indeed an instance of the general Hintikka sequent, thus we can use it to extract a counter-model.

Lemma 3.10. *If F is a formula unprovable in the instance of $\mathrm{LS_G}$, then the limit labelled sequent for F is the instance of the general Hintikka sequent with the corresponding conditions.*

Proof. Let $\mathcal{G}^\omega;\Gamma^\omega \vdash \Delta^\omega$ be the limit sequent. First we show that $\mathcal{G}^\omega;\Gamma^\omega \vdash \Delta^\omega$ is finitely non-provable. Consider any $\mathcal{G};\Gamma \vdash \Delta \subseteq_f \mathcal{G}^\omega;\Gamma^\omega \vdash \Delta^\omega$, we show that $\mathcal{G};\Gamma \vdash \Delta$ has no derivation. Since $\mathcal{G}, \Gamma, \Delta$ are finite sets, there exists $i \in \mathcal{N}$ s.t. $\mathcal{G} \subseteq \mathcal{G}_i$, $\Gamma \subseteq \Gamma_i$, and $\Delta \subseteq \Delta_i$. Moreover, from Lemma 3.8 Item 1, $\mathcal{G}_i;\Gamma_i \vdash \Delta_i$ is not provable in $\mathrm{LS_G}$. Since weakening is admissible in $\mathrm{LS_G}$, $\mathcal{G};\Gamma \vdash \Delta \subseteq_f \mathcal{G}_i;\Gamma_i \vdash \Delta_i$ cannot be provable either. So condition 1, 2, 4, 13 in Definition 3.3 hold for the limit sequent, for otherwise we would be able to construct a provable finite labelled sequent from the limit sequent. We show the proofs that the other conditions in Definition 3.3 are also satisfied by the limit sequent. The following cases are numbered according to items in Definition 3.3.

(3) If $m : \top^* \in \Gamma^\omega$, then $m : \top^* \in \Gamma_i$, for some $i \in \mathcal{N}$, since each labelled formula from Γ^ω must appear somewhere in the sequence. Then there exists $j > i$ such that $\phi(j) = (0, m, \top^*, R)$ where this formula becomes principal. By construction, $(m = \epsilon) \in \mathcal{G}_{j+1} \subseteq \mathcal{G}^\omega$. So we deduce that $m =_{\mathcal{G}^\omega} \epsilon$.
(5) If $m : F_1 \wedge F_2 \in \Gamma^\omega$, then it is in some Γ_i, where $i \in \mathcal{N}$. Since ϕ select the formula infinitely often, there is $j > i$ such that $\phi(j) = (0, m, F_1 \wedge F_2, R)$. Then by construction $\{m : F_1, m : F_2\} \subseteq \Gamma_{j+1} \subseteq \Gamma^\omega$.

(6) If $m : F_1 \wedge F_2 \in \Delta^\omega$, then it is in some Δ_i, where $i \in \mathcal{N}$. Since ϕ select the formula infinitely often, there is $j > i$ such that $\phi(j) = (1, m, F_1 \wedge F_2, R)$. Then by construction $m : F_n \in \Delta_{j+1} \subseteq \Delta^\omega$, where $n \in \{1, 2\}$ and $\mathcal{G}_j; \Gamma_j \vdash m : F_n; \Delta_j$ does not have a derivation.
(7) If $m : F_1 \to F_2 \in \Gamma^\omega$, similar to case 3.
(8) If $m : F_1 \to F_2 \in \Delta^\omega$, similar to case 2.
(9) If $m : F_1 * F_2 \in \Gamma^\omega$, then it is in some Γ_i, where $i \in \mathcal{N}$. Then there exists $j > i$ such that $\phi(j) = (0, m, F_1 * F_2, R)$. By construction $\mathcal{G}_{j+1} = \mathcal{G}_j \cup \{(a_{2j}, a_{2j+1} \triangleright m)\} \subseteq \mathcal{G}^\omega$, and $\Gamma_{j+1} = \Gamma_j \cup \{a_{2j} : F_1, a_{2j+1} : F_2\} \subseteq \Gamma^\omega$.
(10) If $m : F_1 * F_2 \in \Delta^\omega$, then it is in some Δ_i, where $i \in \mathcal{N}$. For any $(x, y \triangleright m') \in \mathcal{G}^\omega$ such that $m =_{\mathcal{G}^\omega} m'$, there exists $j > i$ such that $(x, y \triangleright m') \in \mathcal{G}_j$ and $m =_{\mathcal{G}_j} m'$. Also, there exists $k > j$ such that $\phi(k) = (1, m, F_1 * F_2, \{(x, y \triangleright m')\})$ where the labelled formula becomes principal. Since $(x, y \triangleright m') \in \mathcal{G}_k$ and $m =_{\mathcal{G}_k} m'$, we have either $x : F_1 \in \Delta_{k+1} \subseteq \Delta^\omega$ or $y : F_2 \in \Delta_{k+1} \subseteq \Delta^\omega$.
(11) If $m : F_1 \mathrel{-\!\!*} F_2 \in \Gamma^\omega$, similar to case 9.
(12) If $m : F_1 \mathrel{-\!\!*} F_2 \in \Delta^\omega$, similar to case 10.
(14) For each pair $a_p, a_q \in \mathcal{L}$, assume without loss of generality that $p \geq q$. Then there is some natural number $j \geq q$ such that $\phi(j) = (O, m, \mathbb{EM}, \{(a_p, a_q \triangleright m')\})$. Then either (1) $\mathcal{G}_{j+1} = \mathcal{G}_j \cup \{a_p = a_q\}$, or (2) $\mathcal{G}_{j+1} = \mathcal{G}_j \cup \{a_p \neq a_q\}$, depending on which choice gives a finitely non-provable sequent $\mathcal{G}_{j+1}; \Gamma_{j+1} \vdash \Delta_{j+1}$. If (1) holds, then $a_p = a_q \in \mathcal{G}_{j+1} \subseteq \mathcal{G}^\omega$, and obviously $E(\mathcal{G}^w) \vdash a_p = a_q$, thus $a_p =_{\mathcal{G}^\omega} a_q$. If (2) holds, then $a_p \neq a_q \in \mathcal{G}_{j+1} \subseteq \mathcal{G}^\omega$.
(15) For an arbitrary instance $\mathbb{GS}_n$ of condition 15, assume without loss of generality that this instance is of the form

$$\forall x_1, \ldots, x_m.(s_1 = t_1 \,\&\, \cdots \,\&\, s_p = t_p \,\&\, S_1 \,\&\, \cdots \,\&\, S_k \;\Rightarrow\; \exists y_1, \ldots, y_n.(T_1 \,\&\, \cdots \,\&\, T_l))$$

Suppose there is a substitution θ such that for each $1 \leq i \leq k$, $S_i\theta \in \mathcal{G}^\omega$ and for each $1 \leq i \leq p$, $s_i =_{\mathcal{G}^\omega} t_i$. Then there must be some natural number j such that $\bigcup_{1 \leq i \leq k} S_i\theta \subseteq \mathcal{G}_j$ and for each $1 \leq i \leq p, E(\mathcal{G}_j) \vdash s_i = t_i$. There also exists $k \geq j$ such that $\phi(k) = (O, m, \mathbb{GS}_n, \mathcal{R})$ where $\bigcup_{1 \leq i \leq k} S_i\theta = \mathcal{R} \subseteq \mathcal{G}_k$. It is obvious that for each $1 \leq i \leq p, E(\mathcal{G}_k) \vdash s_i = t_i$. By construction, there is some substitution σ such that $\mathcal{G}_{k+1} = \bigcup_{1 \leq i \leq l}(T_i\theta)\sigma \cup \mathcal{G}_k$. Therefore the corresponding instance of condition (15) holds.

The above covers all the conditions in Definition 3.3. □

Finally we can state the completeness theorem: whenever a formula has no derivation in the instance of $\mathrm{LS_G}$, we can extract an infinite counter-model based on the limit sequent and the Kripke relational frame with the corresponding conditions.

Theorem 3.11. *Every formula F unprovable in the instance of $\mathrm{LS_G}$ is not valid in the Kripke relational models with the corresponding conditions.*

Proof. We construct a limit sequent $\mathcal{G}^\omega; \Gamma^\omega \vdash \Delta^\omega$ for F following Definition 3.9. Note that by the construction of the limit sequent, we have $a_1 : F \in \Delta^\omega$. By Lemma 3.10, this limit sequent is a Hintikka sequent, and therefore by Lemma 3.4, $\mathcal{G}^\omega; \Gamma^\omega \vdash \Delta^\omega$ is falsifiable. This means there exists a model $(\mathcal{F}, \nu, \rho)$ that satisfies $\mathcal{G}^\omega$ and Γ^ω and falsifies every element of Δ^ω, including $a_1 : F$, which means that F is false at world $\rho(a_1)$. Thus F is not valid. □

Corollary 3.12 (Completeness). *Every formula F unprovable in $\mathrm{LS_{PASL}}$ is not valid in PASL, or contrapositively, if F is PASL-valid then F is provable in $\mathrm{LS_{PASL}}$.*

Concrete model	Indivisible unit	Disjointness	Splittability	Cross-Split	Cancellativity
Heaps	✓	✓	×	✓	✓
Fractional permissions	✓	×	✓	✓	✓
Named permissions	✓	✓	×	✓	✓
Counting permissions	✓	×	×	✓	✓
Binary trees	✓	✓	✓	✓	✓
Heaps (finite locations)	✓	✓	×	✓	✓
Petri nets	✓	×	×	✓	✓
Petri nets with capacity 1	✓	✓	×	✓	✓
Endpoint heaps	✓	✓	×	✓	✓
Monotonic counter	✓	×	✓	✓	×
Logical heaps	✓	×	✓	✓	×

Table 2. Some concrete separation algebras and their abstract properties

4 EXTENSIONS OF PASL

In Section 2.2 we presented our proof theory as a *family* of sequent calculi LS_{G} parameterised by a choice of *frame axioms*. This section establishes the benefit of this modular approach by showing how the additional axioms for separation algebras proposed by [20], namely *indivisible unit*, *disjointness*, *splittability*, and *cross-split*, can be accommodated within our approach. We also look at the generalisation of the definition of separation algebra given by rejecting *cancellativity*. We further discuss which of these axioms manifest in the various examples of separation algebra surveyed in Section 2.1, along with two further non-cancellative examples, helping to establish how different abstract semantics correspond to different concrete semantics, as summarised in Table 2.

It is an immediate corollary of the soundness (Theorem 2.9) and completeness (Theorem 3.11) of the general proof system LS_{G} that we have sound and complete proof rules for any combination of these axioms.

By focusing on the properties emphasised by [20] we do not mean to imply that these constitute a canonical list of abstract properties worth considering. Indeed, an advantage of our modular setting is that it will allow investigations of other abstract properties. For example, the property below expresses that any element can be non-trivially extended:

$$\forall h_1.\exists h_2, h_3.\ h_2 \neq \epsilon \ \&\ R(h_1, h_2, h_3)$$

The property is satisfied by all models of Table 2 except heaps with finitely many locations [38], and Petri nets with finite capacity and enough tokens to fill all places. This axiom is in frame axiom form so we may synthesise the rule below with the side-condition that y and z may not appear in the conclusion:

$$\frac{\mathcal{G}; y \neq \epsilon; (x, y \triangleright z); \Gamma \vdash \Delta}{\mathcal{G}; \Gamma \vdash \Delta}$$

With this rule we may prove the formula $\neg(\neg\top^* \mathrel{-\!\!*} \bot)$ which would otherwise be unprovable.

4.1 Indivisible unit

The unit ϵ in a commutative monoid $(H, \circ, \epsilon)$ is *indivisible* iff the following holds for any $h_1, h_2 \in H$:

$$\text{if } h_1 \circ h_2 = \epsilon \text{ then } h_1 = \epsilon.$$

Relationally, this corresponds to the first-order condition:

$$\forall h_1, h_2 \in H.\ \text{if } R(h_1, h_2, \epsilon) \text{ then } h_1 = \epsilon.$$

This also implies that $h_2 = \epsilon$ whenever $h_1 \circ h_2 = \epsilon$. Indivisible unit can be axiomatised by the formula $\top^* \wedge (A * B) \rightarrow A$ [13].

To synthesise a proof rule for indivisible unit we first transform the above condition to the following form:

$$\forall h_1, h_2, h_0.\ h_0 = \epsilon\ \&\ R(h_1, h_2, h_0)\ \Rightarrow\ h_1 = \epsilon.$$

It is easy to check that this form satisfies the frame axiom conditions. The corresponding structural rule is:

$$\frac{\mathcal{G}; (x, y \triangleright z); x = \epsilon; \Gamma \vdash \Delta \qquad E(\mathcal{G}) \vdash z = \epsilon}{\mathcal{G}; (x, y \triangleright z); \Gamma \vdash \Delta}\ \text{IU}$$

We can also deduce that $E(\mathcal{G}) \vdash y = \epsilon$. The following is easy to confirm:

Proposition 4.1. *The formula* $\top^* \wedge (A * B) \rightarrow A$ *is provable in* $\text{LS}_{\text{PASL}} + IU$.

Example 4.2. It is trivial to confirm that all the concrete separation algebras surveyed in Section 2.1 satisfy the indivisible unit axiom; we are not aware of any separation algebras with applications to program verification that fail to do so.

4.2 Disjointness

The separating conjunction $*$ in Reynolds's separation logic requires that the two combined heaps have disjoint domains [55]. Without concrete semantics that give meaning to the "points-to" predicate $\mapsto$ we cannot express this notion of disjointness. However, abstract semantics do allow us to discuss a special case where we try to combine a non-empty heap with itself. In a separation algebra $(H, \circ, \epsilon)$, *disjointness* is defined by the following additional requirement:

$$\forall h_1, h_2 \in H.\ \text{if } h_1 \circ h_1 = h_2 \text{ then } h_1 = \epsilon.$$

The above can be expressed relationally:

$$\forall h_1, h_2 \in H.\ \text{if } R(h_1, h_1, h_2) \text{ then } h_1 = \epsilon.$$

To create a structural rule for it, we first need to convert the above condition into

$$\forall h_1, h_2, h_3.\ h_1 = h_3\ \&\ R(h_1, h_3, h_2)\ \Rightarrow\ h_1 = \epsilon.$$

Then we obtain the structural rule for disjointness as below.

$$\frac{\mathcal{G}; (x, y \triangleright z); x = \epsilon; \Gamma \vdash \Delta \qquad E(\mathcal{G}) \vdash x = y}{\mathcal{G}; (x, y \triangleright z); \Gamma \vdash \Delta}\ D$$

Note that $E(\mathcal{G}) \vdash z = \epsilon$ is a direct consequence.

Disjointness implies indivisible unit (but not vice versa), as shown by [20]. We can prove the axiom for indivisible unit by using $\text{LS}_{\text{PASL}} + D$:

Proposition 4.3. *The formula* $\top^* \wedge (A * B) \rightarrow A$ *is provable in* $\text{LS}_{\text{PASL}} + D$.

Example 4.4. In the cases of Example 2.3, where separation algebras are defined via a partial commutative semigroup $(V, \star)$, the disjointness property holds iff there exist no $v \in V$ such that $v \star v$ is defined. This is the case for heaps, named permissions, and the binary tree share model. On the other hand, disjointness fails to hold for fractional permissions (where $(v, i) \star (v, i)$ is defined so long as $i \leq 0.5$) and counting permissions (for example $(v, -1) \star (v, -1) = (v, -2)$).

Disjointness fails in general for markings of Petri nets, as a marking can be combined with itself by doubling its number of tokens at all places. However disjointness holds in the presence of a global capacity constraint $\kappa = 1$.

4.3 Splittability

The property of infinite splittability is useful when reasoning about the kinds of resource sharing that occur in divide-and-conquer style computations [20]. A separation algebra $(H, \circ, \epsilon)$ has *splittability* if

$$\forall h_0 \in H \setminus \{\epsilon\}, \exists h_1, h_2 \in H \setminus \{\epsilon\} \text{ such that } h_1 \circ h_2 = h_0.$$

Relationally, this corresponds to:

$$\forall h_0.\ h_0 \neq \epsilon \ \Rightarrow\ \exists h_1, h_2 . h_1 \neq \epsilon \ \&\ h_2 \neq \epsilon \ \&\ R(h_1, h_2, h_0).$$

Splittability can be axiomatised as the formula $\neg\top^* \rightarrow (\neg\top^* * \neg\top^*)$ [13]. We give the following structural rule for this property with the side-condition that x, y do not occur in the conclusion:

$$\frac{\mathcal{G}; (z \neq \epsilon); (x, y \triangleright z); x \neq \epsilon; y \neq \epsilon; \Gamma \vdash \Delta}{\mathcal{G}; (z \neq \epsilon); \Gamma \vdash \Delta}\ S$$

Proposition 4.5. *The axiom* $\neg\top^* \rightarrow (\neg\top^* * \neg\top^*)$ *for splittability is provable in* $\mathrm{LS}_{\mathrm{PASL}} + S$.

Example 4.6. In the case of separation algebras defined via a partial commutative semigroup $(V, \star)$, splittability holds iff all $v \in V$ are in the image of $\star$. This holds for fractional permissions, as each (v, i) is $(v, \frac{i}{2}) \star (v, \frac{i}{2})$. The binary tree share model also enjoys this property; see [20] for details.

On the other hand, splittability does not hold for heaps (for which the image of $\star$ is empty), for named permissions (singletons cannot be split), or for counting permissions (where $(v, -1)$ is not in the image of $\star$). Splittability also fails for Petri nets, as the marking assigning one token to one place, with all other places empty, cannot be split.

4.4 Cross-split

This more complicated property requires that if a heap can be split in two different ways, then there should be intersections of these splittings. Formally, in a separation algebra $(H, \circ, \epsilon)$, if $h_1 \circ h_2 = h_0$ and $h_3 \circ h_4 = h_0$, then there should be four elements $h_{13}, h_{14}, h_{23}, h_{24}$, informally representing the intersections $h_1 \cap h_3$, $h_1 \cap h_4$, $h_2 \cap h_3$ and $h_2 \cap h_4$ respectively, such that $h_{13} \circ h_{14} = h_1$, $h_{23} \circ h_{24} = h_2$, $h_{13} \circ h_{23} = h_3$, and $h_{14} \circ h_{24} = h_4$. The corresponding condition on Kripke relational frames is obvious. The following sound rule naturally captures cross-split, where $p, q, s, t, u, v, x, y, z$ are labels, and the labels p, q, s, t do not occur in the conclusion:

$$\frac{\mathcal{G}; (x, y \triangleright z); (u, v \triangleright z'); (p, q \triangleright x); (p, s \triangleright u); (s, t \triangleright y); (q, t \triangleright v); \Gamma \vdash \Delta \qquad E(\mathcal{G}) \vdash z = z'}{\mathcal{G}; (x, y \triangleright z); (u, v \triangleright z'); \Gamma \vdash \Delta}\ CS$$

Example 4.7. All examples of separation algebras presented in Section 2.1 satisfy *CS*; we are not aware of any separation algebras with applications to program verification that fail to do so. In the case of heaps, for example, the cross-splits are simply defined as intersections, but the situation becomes more complex in the case that sharing is possible, and in some cases the sub-splittings $h_{13}, h_{23}, \ldots$ need not be uniquely defined.

For example, take the case of counting permissions. Let $h = \{l \mapsto (v, 1)\}$ for some location l and value v. We will abuse notation by writing this as $h = 1$, as the identity of the location and value are not important here. Let (h_1, h_2, h_3, h_4) be $(-2, 3, -3, 4)$ respectively. Then the values of $(h_{13}, h_{14}, h_{23}, h_{24})$ may be $(-2, \text{undefined}, -1, 4)$, or $(\text{undefined}, -2, -3, 6)$, or $(-1, -1, 2, 5)$.

However, the definition of cross-split does not require uniqueness; it is sufficient to describe a method by which a valid cross-split may be defined. We consider each location $l \in dom(h)$ in turn. The most complex case has that l is in the domain of all of h_1, h_2, h_3, h_4, and that the two splittings are not identical on l. By definition at least one of $h_1(l), h_2(l)$ is negative, and similarly $h_3(l), h_4(l)$. Without loss of generality say $h_1(l)$ is the *strictly largest* negative number of the four.

Then we may set $(h_{13}(l), h_{14}(l), h_{23}(l), h_{24}(l))$ to be $(h_1(l), \text{undefined}, h_3(l) - h_1(l), h_4(l))$. Routine calculation confirms that this indeed defines a cross-split.

4.5 Separation algebras without cancellativity

We finally note that there has been interest in dropping the cancellativity requirement from the definition of separation algebra [26]. In our framework we need merely omit the C rule of Figure 2.

Example 4.8. The partial commutative semigroup construction of Example 2.3, as noted after that example, need not yield a cancellative structure. In particular, if there exists an idempotent element $v \star v = v$, then $\{l \mapsto v\} \circ \{l \mapsto v\} = \{l \mapsto v\} = \{l \mapsto v\} \circ \epsilon$, but clearly $\{l \mapsto v\} \neq \epsilon$. We give two examples:

- Monotonic Counters for Fictional Separation Logic [39]: fictional separation logic is a program verification framework in which every module is associated with its own notion of resource. We here note only the example of a monotonic counter, for which the partial commutative semigroup is the integers with a bottom element, with *max* as the operation. Clearly every element is idempotent.
- Logical Heaps for Relaxed Separation Logic [63]: we refer to *op. cit.* for the rather involved definition, and an example of an idempotent element.

Both of the examples above satisfy indivisible unit, splittability, and cross-split, but fail to satisfy disjointness.

5 FROM EQUALITY ATOMS TO SUBSTITUTIONS

Having equality atoms in the calculus is convenient when proving the completeness of the calculus. However labelled calculi with substitutions, cf. [35], are easier to implement. In particular, global substitution of labels reduces the number of labels in the sequent, and simplifies the structure of the sequent. This often leads to advantages in performance.

We begin by replacing the rules id, $\top^* L$, $\top^* R$, $*R$, $-\!\!* L$, which in the proofs rules of Figure 1 contained equality atoms, with versions without equality atoms, as shown in Figure 3. These are precisely the rules for BBI presented in [35]. We similarly transform the rules NEq and EM. Note the use of the explicit substitutions applied to the meta-variables $\mathcal{G}$, Γ, and Δ in the rules $\top^* L$ and EM.

If we are to move beyond the core sublogic BBI^- of Figure 1 to get the full logic of abstract separation algebras, we need a general method for converting general structural rules synthesised from frame axioms, to proof rules without equality atoms. Given a structural rule of the following form:

$$\frac{\mathcal{G}; S_1; \ldots; S_k; T_1; \ldots; T_l; \Gamma \vdash \Delta \qquad E(\mathcal{G}) \vdash s_1 = t_1 \qquad \cdots \qquad E(\mathcal{G}) \vdash s_p = t_p}{\mathcal{G}; \Gamma \vdash \Delta}$$

we use the following algorithm:

(1) Delete each judgment $E(\mathcal{G}) \vdash s_i = t_i$ and modify the proof rule as follows. If t_i is not ϵ we replace it with s_i everywhere it appears. Conversely if t_i is ϵ, but s_i is not, replace s_i with ϵ everywhere.
(2) For each S_i, if it is an equality $x = y$, we delete the equality and change the *premise only* of the proof rule as follows. If x is not ϵ we replace x everywhere in the premise by y, and apply the explicit substitution $[y/x]$ to the occurrences of $\mathcal{G}$, Γ, and Δ in the premises. We likewise produce another new proof rule in which y is replaced by x everywhere in the premise, provided y is not ϵ.

Identity:

$$\frac{}{\mathcal{G};\Gamma;w:p\vdash w:p;\Delta}\ id$$

Logical Rules:

$$\frac{}{\mathcal{G};\Gamma;w:\bot\vdash\Delta}\ \bot L \qquad \frac{\mathcal{G}[\epsilon/w];\Gamma[\epsilon/w]\vdash\Delta[\epsilon/w]}{\mathcal{G};\Gamma;w:\top^*\vdash\Delta}\ \top^* L \qquad \frac{}{\mathcal{G};\Gamma\vdash w:\top;\Delta}\ \top R \qquad \frac{}{\mathcal{G};\Gamma\vdash\epsilon:\top^*;\Delta}\ \top^* R$$

$$\frac{\mathcal{G};\Gamma;w:A;w:B\vdash\Delta}{\mathcal{G};\Gamma;w:A\wedge B\vdash\Delta}\ \wedge L \qquad \frac{\mathcal{G};\Gamma\vdash w:A;\Delta \qquad \mathcal{G};\Gamma\vdash w:B;\Delta}{\mathcal{G};\Gamma\vdash w:A\wedge B;\Delta}\ \wedge R$$

$$\frac{\mathcal{G};\Gamma\vdash w:A;\Delta \qquad \mathcal{G};\Gamma;w:B\vdash\Delta}{\mathcal{G};\Gamma;w:A\to B\vdash\Delta}\ \to L \qquad \frac{\mathcal{G};\Gamma;w:A\vdash w:B;\Delta}{\mathcal{G};\Gamma\vdash w:A\to B;\Delta}\ \to R$$

$$\frac{\mathcal{G};(x,y\triangleright z);\Gamma;x:A;y:B\vdash\Delta}{\mathcal{G};\Gamma;z:A*B\vdash\Delta}\ *L \qquad \frac{\mathcal{G};(x,z\triangleright y);\Gamma;x:A\vdash y:B;\Delta}{\mathcal{G};\Gamma\vdash z:A\mathrel{-\!\!*}B;\Delta}\ \mathrel{-\!\!*}R$$

$$\frac{\mathcal{G};(x,y\triangleright z);\Gamma\vdash x:A;z:A*B;\Delta \qquad \mathcal{G};(x,y\triangleright z);\Gamma\vdash y:B;z:A*B;\Delta}{\mathcal{G};(x,y\triangleright z);\Gamma\vdash z:A*B;\Delta}\ *R$$

$$\frac{\mathcal{G};(x,y\triangleright z);\Gamma;y:A\mathrel{-\!\!*}B\vdash x:A;\Delta \qquad \mathcal{G};(x,y\triangleright z);\Gamma;y:A\mathrel{-\!\!*}B;z:B\vdash\Delta}{\mathcal{G};(x,y\triangleright z);\Gamma;y:A\mathrel{-\!\!*}B\vdash\Delta}\ \mathrel{-\!\!*}L$$

Structural Rules:

$$\frac{}{\mathcal{G};w\neq w;\Gamma\vdash\Delta}\ NEq \qquad \frac{\mathcal{G}[x/y];\Gamma[x/y]\vdash\Delta[x/y] \qquad \mathcal{G};x\neq y;\Gamma\vdash\Delta}{\mathcal{G};\Gamma\vdash\Delta}\ EM$$

$$\frac{\mathcal{G}[x/z];(x,\epsilon\triangleright x);\Gamma[x/z]\vdash\Delta[x/z]}{\mathcal{G};(x,\epsilon\triangleright z);\Gamma\vdash\Delta}\ E_1 \qquad \frac{\mathcal{G}[z/x];(z,\epsilon\triangleright z);\Gamma[z/x]\vdash\Delta[z/x]}{\mathcal{G};(x,\epsilon\triangleright z);\Gamma\vdash\Delta}\ E_2$$

$$\frac{\mathcal{G};(x,\epsilon\triangleright x);\Gamma\vdash\Delta}{\mathcal{G};\Gamma\vdash\Delta}\ U \qquad \frac{(y,x\triangleright z);(x,y\triangleright z);\mathcal{G};\Gamma\vdash\Delta}{(x,y\triangleright z);\mathcal{G};\Gamma\vdash\Delta}\ Com$$

$$\frac{\mathcal{G};(u,y\triangleright x);(v,w\triangleright y);(z,w\triangleright x);(u,v\triangleright z);\Gamma\vdash\Delta}{\mathcal{G};(u,y\triangleright x);(v,w\triangleright y);\Gamma\vdash\Delta}\ A \qquad \frac{\mathcal{G}[y/w];(x,y\triangleright z);\Gamma[y/w]\vdash\Delta[y/w]}{\mathcal{G};(x,y\triangleright z);(x,w\triangleright z);\Gamma\vdash\Delta}\ C$$

$$\frac{\mathcal{G}[y/z];(w,x\triangleright y);\Gamma[y/z]\vdash\Delta[y/z]}{\mathcal{G};(w,x\triangleright y);(w,x\triangleright z);\Gamma\vdash\Delta}\ P \qquad \frac{\mathcal{G}[\epsilon/x];(\epsilon,\epsilon\triangleright z);\Gamma[\epsilon/x]\vdash\Delta[\epsilon/x]}{\mathcal{G};(x,x\triangleright z);\Gamma\vdash\Delta}\ D$$

Side conditions:
In $*L$ and $\mathrel{-\!\!*}R$, the labels x and y do not occur in the conclusion.
In the rule A, the label z does not occur in the conclusion.

Fig. 3. The labelled sequent calculus $\mathrm{LS}_{\mathrm{PASL}}{}' + D$ with explicit global label substitutions.

As an example, Figure 3 presents the equality-free rules for propositional abstract separation logic with disjointness. We refer to this logic as PASL_D and the proof system as $\mathrm{LS}_{\mathrm{PASL}}{}' + D$, using the prime to denote "equality free". We choose to present this particular logic because it will be used for the experiments of the next section. Note that the frame rule E obtained from Formula 1 in Section 2.2 is split into two rules E_1, E_2, following step (2) of the procedure, depending on the choice of substituting x for z in the premise, or vice versa. We do not need to split C and P into

two rules because we define our sequents via sets, rather than lists, and so the two resulting rules would be identical up to label renaming.

The soundness of the converted system is straightforward, since global substitution of labels naturally captures the equality of the two labels.

Given a sequent $\mathcal{G}; \Gamma \vdash \Delta$ in any instance calculus of $\mathrm{LS_G}$, we convert this sequent to a sequent in the equality-free version via a translation τ, which simply performs global substitutions to unify labels h_1, h_2, for each equality relational atom $h_1 = h_2$ in $\mathcal{G}$, with the restriction that we cannot substitute for ϵ. The corresponding equality-free sequent is written as $\tau(\mathcal{G}; \Gamma \vdash \Delta)$.

The following lemma is easy to show.

Lemma 5.1. *For any sequent $\mathcal{G}; \Gamma \vdash \Delta$, if $E(\mathcal{G}) \vdash h = h'$, then h and h' are unified in $\tau(\mathcal{G}; \Gamma \vdash \Delta)$.*

We then show that the conversion preserves completeness. This can be proved by a straightforward induction on the height of the derivation.

Theorem 5.2. *For any sequent $\mathcal{G}; \Gamma \vdash \Delta$, if it is derivable in an instance calculus LS_i of $\mathrm{LS_G}$, then $\tau(\mathcal{G}; \Gamma \vdash \Delta)$ is derivable in the corresponding equality-free calculus LS_i' given by the above conversion.*

Corollary 5.3. *For any PASL formula F, if it is derivable in an instance calculus LS_i, then it is also derivable in the corresponding equality-free calculus LS_i' given by the above conversion.*

It may seem odd that the equality judgements and their associated rules can just be deleted completely when converting an equality-based calculus into an equality-free calculus with global substitutions. Intuitively, remember that we are intending for these calculi to be used bottom-up: thus the global substitutions trivially "implement" the required equalities as backward proof-search proceeds upwards towards the leaves.

5.1 Example Derivations

We now present some example derivations of valid formulae using the calculus with label substitutions for the logic PASL + D. We shall use some inference rules for derived logical connectives $\neg$ and $\vee$, as shown below:

$$\frac{\mathcal{G}; \Gamma \vdash w : A; \Delta}{\mathcal{G}; \Gamma; w : \neg A \vdash \Delta}\ \neg L \qquad \frac{\mathcal{G}; \Gamma; w : A \vdash \Delta}{\mathcal{G}; \Gamma \vdash w : \neg A; \Delta}\ \neg R$$

$$\frac{\mathcal{G}; \Gamma; w : A \vdash \Delta \qquad \mathcal{G}; \Gamma; w : B \vdash \Delta}{\mathcal{G}; \Gamma; w : A \vee B \vdash \Delta}\ \vee L \qquad \frac{\mathcal{G}; \Gamma \vdash w : A; w : B; \Delta}{\mathcal{G}; \Gamma \vdash w : A \vee B; \Delta}\ \vee R$$

First let us consider the formula $\neg(\top^* \wedge A \wedge (B * \neg(C \mathrel{-\!\!*} (\top^* \to A))))$. This formula can be easily proved by the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(c, b \triangleright \epsilon); (a, b \triangleright \epsilon); \epsilon : A; a : B; c : C \vdash \epsilon : A}\ id}{(c, b \triangleright d); (a, b \triangleright \epsilon); \epsilon : A; a : B; c : C; d : \top^* \vdash d : A}\ \top^* L}{(c, b \triangleright d); (a, b \triangleright \epsilon); \epsilon : A; a : B; c : C \vdash d : \top^* \to A}\ \to R}{(a, b \triangleright \epsilon); \epsilon : A; a : B \vdash b : C \mathrel{-\!\!*} (\top^* \to A)}\ \mathrel{-\!\!*} R}{(a, b \triangleright \epsilon); \epsilon : A; a : B; b : \neg(C \mathrel{-\!\!*} (\top^* \to A)) \vdash}\ \neg L}{\epsilon : A; \epsilon : B * \neg(C \mathrel{-\!\!*} (\top^* \to A)) \vdash}\ *L}{w : \top^*; w : A; w : B * \neg(C \mathrel{-\!\!*} (\top^* \to A)) \vdash}\ \top^* L}{w : \top^* \wedge A \wedge (B * \neg(C \mathrel{-\!\!*} (\top^* \to A))) \vdash}\ \wedge L}{\vdash w : \neg(\top^* \wedge A \wedge (B * \neg(C \mathrel{-\!\!*} (\top^* \to A))))}\ \neg R$$

This formula is valid in non-deterministic BBI. However, we are not aware of any proof for this formula in other proof systems for non-deterministic BBI, such as the nested sequent calculus [51]

or the display calculus [9]. In particular, we ran this formula using the nested sequent calculus based prover [51] for a week on a CORE i7 2600 processor, without success.

The formula $(F * F) \to F$, where $F = \neg(\top \mathrel{-\!\!*} \neg\top^*)$, is not valid in non-deterministic BBI, but is valid in BBI plus partial-determinism. To prove this formula, we use the following derivation, where we write $\cdot^2$ for two consecutive applications of a rule and use semi-colon to denote sequencing of rule applications:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(w', w'' \triangleright \epsilon); (b', c' \triangleright w''); (b, c \triangleright w'); (b, c \triangleright a); \cdots}{(w', c' \triangleright w); (w, b' \triangleright \epsilon); \cdots}A}{(c', w' \triangleright w); (b, c \triangleright w'); (b', w \triangleright \epsilon); \cdots}E^2}{(b', w \triangleright \epsilon); (\epsilon, b \triangleright w); (c', c \triangleright \epsilon); \cdots}A}{(b, c \triangleright a); (b', b \triangleright \epsilon); (c', c \triangleright \epsilon); (\epsilon, \epsilon \triangleright \epsilon); \cdots}A}{(b, c \triangleright a); (b', b \triangleright \epsilon); (c', c \triangleright \epsilon); a : \top \mathrel{-\!\!*} \neg\top^*; b' : \top, c' : \top \vdash}U}{(b, c \triangleright a); (b', b \triangleright b''); (c', c \triangleright c''); a : \top \mathrel{-\!\!*} \neg\top^*; b' : \top, c' : \top; b'' : \top^*; c'' : \top^* \vdash}\top^*L^2}{(b, c \triangleright a); (b', b \triangleright b''); (c', c \triangleright c''); a : \top \mathrel{-\!\!*} \neg\top^*; b' : \top, c' : \top \vdash b'' : \neg\top^*; c'' : \neg\top^*}\neg R^2}{(b, c \triangleright a); a : \top \mathrel{-\!\!*} \neg\top^* \vdash b : \top \mathrel{-\!\!*} \neg\top^*; c : \top \mathrel{-\!\!*} \neg\top^*}\mathrel{-\!\!*} R^2}{(b, c \triangleright a); b : \neg(\top \mathrel{-\!\!*} \neg\top^*); c : \neg(\top \mathrel{-\!\!*} \neg\top^*) \vdash a : \neg(\top \mathrel{-\!\!*} \neg\top^*)}\neg L^2; \neg R}{a : F * F \vdash a : F}*L}{\vdash a : (F * F) \to F} \to R$$

Some sequents in the above derivation are too long, so we omit the part of a sequent that is not important in the rule application. The top sequent above the A rule instance contains only one non-atomic formula: $a : \top \mathrel{-\!\!*} \neg\top^*$ on the left hand side. The correct relational atom that is required to split $a : \top \mathrel{-\!\!*} \neg\top^*$ is $(w'', a \triangleright \epsilon)$. However, so far we have only obtained $(w'', w' \triangleright \epsilon)$. Although w' and a both have exactly the same children (b and c), the non-deterministic monoid allows the composition $b \circ c$ to yield multiple elements, or even ϵ. Thus we cannot conclude that $w' = a$ in non-deterministic BBI. This can be solved by applying the rule P to replace w' by a, then use E to obtain $(w'', a \triangleright \epsilon)$ on the left hand side of the sequent, then the derivation can go through:

$$\dfrac{\dfrac{\dfrac{}{(w'', a \triangleright \epsilon); \cdots ; \vdash \epsilon : \top^*}\top^*R}{(w'', a \triangleright \epsilon); \cdots ; \epsilon : \neg\top^* \vdash}\neg L \qquad \dfrac{}{(w'', a \triangleright \epsilon); \cdots \vdash w'' : \top}\top R}{(w'', a \triangleright \epsilon); \cdots ; a : \top \mathrel{-\!\!*} \neg\top^*; b' : \top, c' : \top \vdash}\mathrel{-\!\!*} L$$

The proof system $\mathrm{LS}_{\mathrm{PASL}}{}' + D$ does not have a rule for indivisible unit. We show that we can use the rule D for disjointness to prove the axiom $\top^* \wedge (A * B) \to A$ of indivisible unit. We highlight the principal relational atoms where they are not obvious.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\epsilon,\epsilon\triangleright\epsilon);\cdots;\epsilon:A;\epsilon:B\vdash\epsilon:A}\,id}}{(\epsilon,a_1\triangleright\epsilon);\cdots;a_1:A;\epsilon:B\vdash\epsilon:A}\,Eq_1}}{(a_1,w_2\triangleright w_1);(\epsilon,\epsilon\triangleright w_2);\boxed{(a_1,\epsilon\triangleright\epsilon)};\cdots;a_1:A;\epsilon:B\vdash\epsilon:A}\,E}}{(a_1,w_2\triangleright w_1);\boxed{(a_2,a_2\triangleright w_2)};(a_1,a_2\triangleright\epsilon);\cdots;a_1:A;a_2:B\vdash\epsilon:A}\,D}}{(a_1,w_1\triangleright\epsilon);\boxed{(\epsilon,a_2\triangleright w_1);(a_1,a_2\triangleright\epsilon)};\cdots;a_1:A;a_2:B\vdash\epsilon:A}\,A}}{(\epsilon,\epsilon\triangleright\epsilon);(a_1,a_2\triangleright\epsilon);a_1:A;a_2:B\vdash\epsilon:A}\,A}}{(a_1,a_2\triangleright\epsilon);a_1:A;a_2:B\vdash\epsilon:A}\,U}}{(a_1,a_2\triangleright a_0);a_0:\top^*;a_1:A;a_2:B\vdash a_0:A}\,\top^*L}}{;a_0:\top^*;a_0:A*B\vdash a_0:A}\,*L}}{;a_0:\top^*\wedge(A*B)\vdash a_0:A}\,\wedge L}}{;\vdash a_0:\top^*\wedge(A*B)\to A}\,\to R$$

6 IMPLEMENTATION AND EXPERIMENT

In this section we present an implementation, called Separata[4], of a semi-decision procedure for the logic PASL_D (propositional abstract separation logic with disjointness), and present experiments demonstrating its efficacy. We focus on PASL_D because, following Table 2, and recalling that disjointness implies indivisible unit, it captures much of the structure of the paradigmatic example of a separation algebra, namely heaps. Heaps also satisfy the cross-split property, but we are unaware of any theorems expressible in the language of abstract separation logic that require cross-split for proof (we are, however, aware of some such theorems when the points-to connective $\mapsto$ is introduced). We note also that Separata may handle various sublogics of PASL_D got by variously omitting partiality, cancellativity, or disjointness, or by weakening disjointness to indivisible unit.

Our implementation is based on the proof system $\mathrm{LS}_{\mathrm{PASL}}{}' + D$ of Figure 3, with three modifications. First, we modify the rule U to a rule U' that creates the identity relational atom $(x, \epsilon \triangleright x)$ only if x occurs in the conclusion. This does not reduce power:

Lemma 6.1. *If $\mathcal{G};\Gamma\vdash\Delta$ is derivable in $\mathrm{LS}_{\mathrm{PASL}}{}' + D$, then it is derivable in the proof system with U' replacing U.*

Second, the rules *NEq* and *EM* are omitted, as they are not necessary for completeness for PASL_D, nor its sublogics [28, Section 6.3].

Third, although disjointness implies indivisible unit, we find that including (the equality atom-free version of) the rule for indivisible unit can reduce the number of labels in sequents and hence lead to a smaller search space and better performance:

$$\dfrac{\mathcal{G}[\epsilon/x];(\epsilon,y\triangleright\epsilon);\Gamma[\epsilon/x]\vdash\Delta[\epsilon/x]}{\mathcal{G};(x,y\triangleright\epsilon);\Gamma\vdash\Delta}\,\mathrm{IU}$$

Our implementation is based on the following strategy when applying rules:

(1) Try to close the branch by rules *id*, $\bot L$, $\top^* R$, $\top^* R$.
(2) If (1) is not applicable, apply all possible E_1, E_2, P, C, IU, D rules to unify labels.
(3) If (1-2) are not applicable, apply invertible rules $\wedge L$, $\wedge R$, $\to L$, $\to R$, $*L$, $-\!\!* R$, $\top^* L$ in all possible ways.
(4) If (1-3) are not applicable, try $*R$ or $-\!\!* L$ by choosing existing relational atoms.

[4] Available at http://users.cecs.anu.edu.au/~zhehou.

	Formula	BBeye (opt)	FVLS_{BBI} (heuristic)	**Separata**
(1)	$(a \mathrel{-\!\!*} b) \wedge (\top * (\top^* \wedge a)) \to b$	0.076	0.002	0.002
(2)	$(\top^* \mathrel{-\!\!*} \neg(\neg a * \top^*)) \to a$	0.080	0.004	0.002
(3)	$\neg((a \mathrel{-\!\!*} \neg(a * b)) \wedge ((\neg a \mathrel{-\!\!*} \neg b) \wedge b))$	0.064	0.003	0.002
(4)	$\top^* \to ((a \mathrel{-\!\!*} (b \mathrel{-\!\!*} c)) \mathrel{-\!\!*} ((a * b) \mathrel{-\!\!*} c))$	0.060	0.003	0.002
(5)	$\top^* \to ((a * (b * c)) \mathrel{-\!\!*} ((a * b) * c))$	0.071	0.002	0.004
(6)	$\top^* \to ((a * ((b \mathrel{-\!\!*} e) * c)) \mathrel{-\!\!*} ((a * (b \mathrel{-\!\!*} e)) * c))$	0.107	0.004	0.008
(7)	$\neg((a \mathrel{-\!\!*} \neg(\neg(d \mathrel{-\!\!*} \neg(a * (c * b))) * a)) \wedge c * (d \wedge (a * b)))$	0.058	0.002	0.006
(8)	$\neg((c * (d * e)) \wedge B)$ where $B := ((a \mathrel{-\!\!*} \neg(\neg(b \mathrel{-\!\!*} \neg(d * (e * c))) * a)) * (b \wedge (a * \top)))$	0.047	0.002	0.013
(9)	$\neg(C * (d \wedge (a * (b * e))))$ where $C := ((a \mathrel{-\!\!*} \neg(\neg(d \mathrel{-\!\!*} \neg((c * e) * (b * a))) * a)) \wedge c)$	94.230	0.003	0.053
(10)	$(a * (b * (c * d))) \to (d * (c * (b * a)))$	0.030	0.004	0.002
(11)	$(a * (b * (c * d))) \to (d * (b * (c * a)))$	0.173	0.002	0.002
(12)	$(a * (b * (c * (d * e)))) \to (e * (d * (a * (b * c))))$	1.810	0.003	0.002
(13)	$(a * (b * (c * (d * e)))) \to (e * (b * (a * (c * d))))$	144.802	0.003	0.002
(14)	$\top^* \to (a * ((b \mathrel{-\!\!*} e) * (c * d)) \mathrel{-\!\!*} ((a * d) * (c * (b \mathrel{-\!\!*} e))))$	6.445	0.003	0.044
(15)	$\neg(\top^* \wedge (a \wedge (b * \neg(c \mathrel{-\!\!*} (\top^* \to a)))))$	timeout	0.003	0.003
(16)	$((D \to (E \mathrel{-\!\!*} (D * E))) \to (b \mathrel{-\!\!*} ((D \to (E \mathrel{-\!\!*} ((D * a) * a))) * b)))$, where $D := \top^* \to a$ and $E := a * a$	0.039	0.005	8.772
(17)	$((\top^* \to (a \mathrel{-\!\!*} (((a * (a \mathrel{-\!\!*} b)) * \neg b) \mathrel{-\!\!*} (a * (a * ((a \mathrel{-\!\!*} b) * \neg b)))))) \to ((((\top^* * a) * (a * ((a \mathrel{-\!\!*} b) * \neg b))) \to (((a * a) * (a \mathrel{-\!\!*} b)) * \neg b)) * \top^*))$	timeout	fail	49.584
(18)	$(F * F) \to F$, where $F := \neg(\top \mathrel{-\!\!*} \neg\top^*)$	invalid	invalid	0.004
(19)	$(\top^* \wedge (a * b)) \to a$	invalid	invalid	0.003

Table 3. Experiment 1 results.

(5) If (1-3) are not applicable, and (4) is not applicable because all combinations of $*R$, $\mathrel{-\!\!*} L$ formulae and relational atoms are already applied, apply structural rules on the set $\mathcal{G}_0$ of relational atoms in the sequent as follows.
   (a) Use *Com* to generate all commutative variants of existing relational atoms in $\mathcal{G}_0$, giving a set $\mathcal{G}_1$.
   (b) Apply A for each applicable pair in $\mathcal{G}_1$, generating a set $\mathcal{G}_2$. We do not apply A to a pair $(u, y \triangleright x)$; $(v, w \triangleright y)$ where for any z the pair $(z, w \triangleright x)$; $(u, v \triangleright x)$ already exists in the sequent.
   (c) Use U' to generate all identity relational atoms for each label in $\mathcal{G}_2$, giving $\mathcal{G}_3$.
(6) If (1-4) are not applicable, and (5) has been applied with result $\mathcal{G}_3 = \mathcal{G}_0$, then fail.

Step (2) is terminating, because each substitution eliminates a label, and we only have finitely many labels. It is also clear that step (5) is terminating.

In the implementation, we view Γ and Δ in a sequent $\mathcal{G}; \Gamma \vdash \Delta$ as lists, and each time a logical rule is applied, we place the subformulae at the front of the list. Thus our proof search has a "focusing flavour", that always tries to decompose the subformulae of a principal formula if possible. To guarantee completeness, each time we apply a $*R$ or $\mathrel{-\!\!*} L$ rule, the principal formula is moved to the end of the list in the corresponding premise, thus each principal formula for these non-deterministic rules is considered fairly, i.e., applied in turn.

We incorporate a number of optimisations in proof search. (1) Back-jumping [2] is used to collect the "unsatisfiable core" along each branch. When one premise of a binary rule has a derivation, we try to derive the other premise only when the unsatisfiable core is not included in that premise. (2) A search strategy discussed by Park et al. [51] is also adopted. For $*R$ and $\mathrel{-\!\!*} L$ applications, we forbid the search to consider applying the rule twice with the same pair of principal formula and principal

Axioms

$A \to (\top^* * A)$

$(\top^* * A) \to A$

$(A * B) \to (B * A)$

$(A * (B * C)) \to ((A * B) * C)$

Deduction Rules

$$\frac{\vdash A \qquad \vdash A \to B}{\vdash B}\ MP \qquad \frac{\vdash A \to C \qquad \vdash B \to D}{\vdash (A * B) \to (C * D)}\ *$$

$$\frac{\vdash A \to (B \mathrel{-\!\!*} C)}{\vdash (A * B) \to C}\ \mathrel{-\!\!*} 1 \qquad \frac{\vdash (A * B) \to C}{\vdash A \to (B \mathrel{-\!\!*} C)}\ \mathrel{-\!\!*} 2$$

Fig. 4. Some axioms and rules for the Hilbert system for BBI.

relational atom, since the effect is the same as contraction, which is admissible. (3) Previous work on theorem proving for BBI has shown that associativity of $*$ is a source of inefficiency in proof search [35, 51]. We borrow the idea of the heuristic method presented in [35] to quickly solve certain associativity instances. When we detect $z : A * B$ on the right hand side of a sequent, we try to search for possible worlds (labels) for the subformulae of A, B in the sequent, and construct a binary tree using these labels. For example, if we can find $x : A$ and $y : B$ in the sequent, we will take x, y as the children of z. When we can build such a binary tree of labels, the corresponding relational atoms given by the binary tree will be used (if they are in the sequent) as the prioritised ones when decomposing $z : A * B$ and its subformulae. Without a free-variable system, our handling of this heuristic method is just a special case of the original one, but this approach can speed up the search in certain cases.

The experiments in this paper are conducted on a Dell Optiplex 790 desktop with Intel CORE i7 2600 @ 3.4 GHz CPU and 8GB memory, running Ubuntu 13.04. The theorem provers are written in OCaml.

Experiment 1. We test our prover Separata for the logic PASL_D on the formulae listed in Table 3; the times displayed are in seconds and timeout is set as 1000s. Where the formulae are valid in BBI, we compare the results with two provers for BBI: the optimised implementation of BBeye [51] and the incomplete heuristic-based $\mathrm{FVLS}_{\mathrm{BBI}}$ [35]. We run BBeye in an iterative deepening way, and the time counted for BBeye is the total time it spends. Formulae (1-14) are used by Park et al. to test their prover BBeye for BBI [51]. We can see that for formulae (1-14) the performance of Separata is comparable with the heuristic based prover for $\mathrm{FVLS}_{\mathrm{BBI}}$. Both provers are generally faster than BBeye. Formula (15) is one that BBeye had trouble with [35], but Separata handles it trivially. However, there are cases where BBeye is faster than Separata, for example, formula (16) found from a set of tests on randomly generated BBI theorems. Formula (17) is a converse example where a randomly generated BBI theorem causes BBeye to time out and $\mathrm{FVLS}_{\mathrm{BBI}}$ with heuristics to terminate within the timeout but without finding a proof due to its incompleteness. Formula (18) is valid only when the monoid is partial, rather than merely non-deterministic [44], and formula (19) is the axiom of indivisible unit.

Experiment 2. Since PASL_D is not finitely axiomatisable, we cannot enumerate all theorems of this logic for testing purposes. In this experiment we instead use randomly generated BBI theorems, which are a subset of all PASL_D theorems, and so are provable in the calculus $\mathrm{LS}_{\mathrm{PASL}}{}' + D$. This test may not show the full power of our prover Separata, but it allows comparison with existing provers. We generate theorems via the Hilbert system for BBI [24], which consists of the axioms and rules of classical logic plus those shown in Figure 4.

We create random BBI theorems by first generating some random formulae (not necessarily theorems) of length n, and globally substituting these formulae at certain places in a randomly chosen BBI axiom schema. Then we use the deduction rules $\mathrel{-\!\!*} 1$ and $\mathrel{-\!\!*} 2$ in Figure 4 to mutate

Test	n	i	BBeye proved	BBeye avg. time	Separata proved	Separata avg. time
1	10	20	74%	0.27s	78%	0.19s
2	15	30	72%	4.63s	66%	2.40s
3	20	30	61%	8.88s	56%	8.76s
4	20	50	55%	12.39s	53%	0.88s
5	30	50	49%	11.81s	50%	6.26s
6	50	50	31%	11.82s	31%	3.79s

Table 4. Experiment 2 results.

the resultant formula at random places. The final theorem is obtained by repeating the mutation for i iterations.

The formulae generated from a fixed pair of n and i could have different lengths since much randomness is involved in the generation, but the average length grows as n increases. BBI axioms do not involve $-\!\!*$ at all, so the mutation step is vital to create $-\!\!*$ in the theorems. In general, a higher i value means that the theorems are more structurally different from the BBI axioms. The mutation step often makes formulae harder to prove.

We compare the performance of Separata with Park et al.'s BBI prover BBeye against our randomly generated theorem suites in Table 4. The "proved" column for each prover gives the percentage of successfully proved formulae within time out, and "avg. time" column is the average time used when a formula is proved. Timed out attempts are not counted in the average time. We set the time out as 200 seconds. Formulae in test suite 1 have 20 to 30 binary connectives on average, while these numbers are around 100 to 150 for the formulae in test suite 6. Each test suit contains 100 BBI theorems. The two provers have similar successful rates in these tests. Average time used on successful attempts increases for BBeye, but fluctuates for Separata. In fact, both provers spent less than 1 second on most successful attempts (even for test suite 6), but there were some "difficult" formulae that took them over 100 seconds to prove. Therefore, we only include average time because median time for both provers would be very small numbers that are difficult to compare. In general there are fewer formulae that are "difficult" for Separata than for BBeye. But in test suites 2,3,4, Separata proved fewer formulae in time. Lastly, we emphasise that Separata and BBeye are designed for different logics, so comparing their performance may not be fair; we do so only because there are no other provers to compete with.

7 APPLICATIONS

This section discusses applications that have been enabled by the calculi of this paper. In particular, we discuss how our calculi may be used in formal verification tasks, and how to extend them with the widely-used points-to predicate for concrete models.

We have formalised the proof system $\mathrm{LS}_{\mathrm{PASL}}{}' + D$ in Isabelle/HOL [31]. As an alternative to the OCaml implementation, we have developed Isabelle tactics based on the proof search techniques in this paper. The resultant proof method, also called Separata, combines certain strengths of our calculus and Isabelle's built-in proof methods. The Isabelle version of Separata serves as a tool for machine-assisted formal verification, and it is compliant with the existing separation algebra library [40]. We have also developed advanced tactics for spatial connectives $*$ and $-\!\!*$ to improve the performance and automation of the proof method [32]. As a result, our tactics can automatically prove some lemmas in the seL4 development [48] that originally required several lines of manual

proofs. Hóu et al. [32] also gives a case study to show the potential of our tactics by proving properties of the semantics of actions in local rely/guarantee reasoning [22].

Our unified framework for separation logic proof systems focuses on abstract semantics, and it does not have the points-to predicate $\mapsto$, which is essential to most real-life applications of separation logic. There are at least two directions to use our framework to solve this problem: (1) one can develop a proof system for separation logic with a concrete semantics, such as the heap model, by extending our labelled calculus with sound rules for the points-to predicate; or (2) one can develop a new abstract separation logic which extends this work with a "points-to"-like predicate that is defined in an abstract setting.

In separate work we have pursued the first direction to develop a proof system for separation logic with the heap model [30]. The new labelled sequent calculus extends $\mathrm{LS}_{\mathrm{PASL}}' + D$ with eight inference rules for the points-to predicate in the heap model. Since separation logic for the heap model is not recursively enumerable [8, 16], it is not possible to obtain a sound, complete, and finite proof system for such a logic. However, this incomplete proof system is still useful in the sense that it can reason about data structures such as linked lists and binary trees, and the eight rules for the points-to predicate are powerful enough to be complete for the *symbolic heap* fragment, which is widely used in program verification. Furthermore, this proof system can also prove many formulae that involve $-\!\!*$ and overlaid data structures, such as formulae in the form of $(A * B) \wedge (C * D)$ where A and C (similarly, B and D) represent overlaid resources. These cannot be expressed by symbolic heaps and cannot be handled by most other proof methods.

In the other direction, we have developed a first-order abstract separation logic which includes an "abstract points-to" predicate [33]. The advantage of this abstract logic is that it is recursively enumerable, and we have developed a sound and complete proof system for it. Moreover, the abstract points-to predicate can be equipped with various theories to approximate concrete semantics of different flavours, such as Reynolds's SL [55], Vafeiadis and Parkinson's SL [64], Lee et al.'s SL [46], and Thakur et al.'s SL [60]. Specifically, we can formulate the properties of points-to in Reynolds's SL as a set S of formulae (theories) in the logic. When we prove a formula F with such a points-to predicate, we derive $S \vdash h : F$ in our proof system where h is an arbitrary world. If we want to prove F in, e.g., Vafeiadis and Parkinson's SL, we merely need to modify the set S to some S' that captures properties of points-to in the corresponding semantics, and derive $S' \vdash h : F$. Again, it is impossible to completely capture the points-to predicate in a concrete model, but this abstract logic provides a language that is rich enough to capture most of the inference rules in our work on the heap model [30]. The benefit of simulating various semantics via theories is that we do not need to develop a new logic and prove important properties for each kind of semantics, but merely need to add or remove certain formulae from the set of theories.

8 RELATED WORK

There are many more automated tools, formalisations, and logical embeddings for separation logics than can reasonably be surveyed within the scope of this paper. Almost all are not directly comparable to this paper because (1) they focus on reasoning about the specification language (Hoare triples) and they deal with separation logic for some *concrete* semantics [3]; (2) they only consider a small subset of the assertion language such as symbolic heaps [10, 11, 57]; or (3) they focus on complexity and computability issues instead of automated reasoning [18, 19].

One exception to the above is Holfoot [61], a HOL mechanisation with support for automated reasoning about the 'shape' of SL specifications – exactly those aspects captured by abstract separation logic. However, unlike Separata, Holfoot does not support magic wand. This is a common restriction for automation of separation logic because magic wand is a source of undecidability [8]. Conversely, the mechanisations and embeddings that do incorporate magic wand tend to give little thought to

(semi-) decision procedures, for example, [59]. An exception to this are the tableaux of [25], but their methods have not been implemented, and we anticipate that such an implementation would not be trivial. Another partial exception to the trend to omit magic wand is SmallfootRG [15], which supports automation yet includes *septraction* [64], the De Morgan dual of magic wand. However SmallfootRG does not support additive negation nor implication, and so magic wand cannot be recovered; indeed in this setting septraction is mere syntactic sugar that can be eliminated. We also note the work [5, 58] in which program proofs involving magic wand can be automated, provided the code is annotated to assist the prover. Recently, Reynolds et al. [54] gave a decision procedure for quantifier-free heap model separation logic which contains all the logical connectives in this work. They have implemented an integrated subsolver in the DPLL-based SMT solver CVC4, and their experiment has shown promising results. In contrast to this paper, which considers various abstract algebraic semantics and a unified proof theory for different models, Reynolds et al.'s work is focused on a concrete heap model semantics with the points-to predicate.

Leaving out magic wand is not without cost, as the connective, while surely less useful than $*$, has applications. A non-exhaustive list follows: generating weakest preconditions via backwards reasoning [36]; specifying iterators [27, 41, 52]; reasoning about parallelism [21]; and various applications of septraction, such as the specification of iterators and buffers [17]. For a particularly deeply developed example, see the correctness proof for the Schorr-Waite Graph Marking Algorithm of [66], which involves non-trivial inferences involving magic wand (Lemmas 78 and 79). These examples provide ample motivation to build proof calculi and tool support that include magic wand. Undecidability, which in any case is pervasive in program proof, should not deter us from seeking practically useful automation.

This work builds upon earlier labelled sequent calculi for BBI [35] and PASL [29]. The extensions to previous work involve two main advances: first, a calculus framework that handles any separation algebra property expressible in the general axiom form; second, a new counter-model construction method that yields completeness proofs for any calculus in our framework. The future work section of the conference version of this paper [29] proposed some putative rules extending this work to deal with Reynolds's heap semantics. This proposal has since been developed in [30], in which we presented the first theorem prover to handle all the logical connectives in a separation logic with heap semantics. We believe that the theory and techniques discussed in this paper can be extended to many applications besides this model.

The link between BBI and separation logic is also emphasised as motivation by Park et al [51], whose BBI prover BBeye was used for comparisons in Section 6. Lee and Park [46] then extended [51], independently to our own work, to a labelled sequent calculus for Reynolds's heap model with the restriction that all values are addresses. Their paper [46] includes soundness and completeness theorems, but unfortunately, our investigations [30] showed that both claims are erroneous, and they have since been retracted by the authors.

Also related, but so far not implemented, are the tableaux for partial-deterministic (PD) BBI of Larchey-Wendling and Galmiche [42, 43]. In subsequent work, the authors showed that validity of partial-deterministic BBI-models coincides with validity of cancellative partial-deterministic BBI-models [45], which retrospectively shows that their earlier work on PD-BBI was actually complete for PASL, and that the rule C of our system is admissible. We nevertheless have presented C in our system, partly to emphasise that we can easily handle a range of structural properties in our system, and partly because evidence in separation logic with concrete semantics, and further language constructors such as $\mapsto$ that may express properties of the content of those semantics, show that the rule for cancellativity may not always be admissible. For example, see the discussion of [26]:

It is well-known that if $*$ is cancellative, then for a precise q in $\mathcal{P}(\Sigma)$ and any p_1, p_2 in $\mathcal{P}(\Sigma)$, we have $(p_1 \wedge p_2) * q = (p_1 * q) \wedge (p_2 * q)$

where a predicate is "precise" if it "unambiguously carves out an area of the heap". It is not likely that our rule for cancellativity will remain admissible in a semantics in which such considerations are important. It is not clear how Larchey-Wendling and Galmiche's tableaux method might be extended to handle concrete semantics without an explicit rule for cancellativity. Moreover, their latest result does not include any treatment for non-deterministic BBI, nor for properties such as splittability and cross-split. In contrast, the relative ease with which certain properties can be added or removed from labelled sequent calculi is an important benefit of our approach.

Finally we note that the counter-model construction of this paper was necessary to prove completeness because many of the properties we are interested in are not BBI-axiomatisable, as proved by Brotherston and Villard [13]; that paper goes on to give a sound and complete Hilbert axiomatisation of these properties by extending BBI with techniques from hybrid logic. Sequent calculus and proof search for this more powerful logic represents an interesting future direction for research.

Acknowledgements. We thank the anonymous reviewers of this paper for their helpful comments. Clouston was supported by a research grant (12386) from Villum Fonden. Tiu and Hóu received support from the National Research Foundation of Singapore under its National Cybersecurity R&D Program (Award No. NRF2014NCR-NCR001-30) and administered by the National Cybersecurity R&D Directorate.

REFERENCES

[1] Alan Ross Anderson and Nuel D. Belnap, Jr. 1976. *Entailment, Vol. 1: The Logic of Relevance and Necessity.* Princeton University Press.

[2] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider (Eds.). 2003. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press.

[3] Josh Berdine, Cristiano Calcagno, and Peter W. O'hearn. 2005. Smallfoot: Modular automatic assertion checking with separation logic. In *In International Symposium on Formal Methods for Components and Objects*. Springer, 115–137.

[4] Bodil Biering, Lars Birkedal, and Noah Torp-Smith. 2007. BI-Hyperdoctrines, Higher-Order Separation Logic, and Abstraction. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 29, 5 (2007), 24.

[5] Stefan Blom and Marieke Huisman. 2015. Witnessing the Elimination of Magic Wands. *International Journal on Software Tools for Technology Transfer* 17, 6 (2015), 757–781.

[6] Richard Bornat, Cristiano Calcagno, Peter W. O'Hearn, and Matthew Parkinson. 2005. Permission Accounting in Separation Logic, In ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL). *ACM SIGPLAN Notices* 40, 1, 259–270.

[7] John Boyland. 2003. Checking Interference with Fractional Permissions. In *Static Analysis*. Springer, 55–72.

[8] Rémi Brochenin, Stéphane Demri, and Etienne Lozes. 2012. On the Almighty Wand. *Information and Computation* 211 (2012), 106–137.

[9] James Brotherston. 2010. A Unified Display Proof Theory for Bunched Logic. *Electr. Notes Theor. Comput. Sci.* 265 (2010), 197–211.

[10] James Brotherston, Carsten Fuhs, Juan A. Navarro Pérez, and Nikos Gorogiannis. 2014. A Decision Procedure for Satisfiability in Separation Logic with Inductive Predicates. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (CSL-LICS '14)*. Article 25, 25:1–25:10 pages.

[11] James Brotherston, Nikos Gorogiannis, Max Kanovich, and Reuben Rowe. 2016. Model Checking for Symbolic-heap Separation Logic with Inductive Predicates. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '16)*. 84–96.

[12] James Brotherston and Max Kanovich. 2014. Undecidability of Propositional Separation Logic and its Neighbours. *Journal of the ACM (JACM)* 61, 2 (2014).

[13] James Brotherston and Jules Villard. 2014. Parametric Completeness for Separation Theories, In ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). *ACM SIGPLAN Notices* 49, 1, 453–464.

[14] Cristiano Calcagno, Peter W. O'Hearn, and Hongseok Yang. 2007. Local Action and Abstract Separation Logic. In *IEEE Symposium on Logic in Computer Science (LICS)*. 366–378.

[15] Cristiano Calcagno, Matthew Parkinson, and Viktor Vafeiadis. 2007. Modular Safety Checking for Fine-Grained Concurrency, In Static Analysis. *Lecture Notes in Computer Science* 4634, 233–248.

[16] Cristiano Calcagno, Hongseok Yang, and Peter W. O'Hearn. 2001. *Computability and Complexity Results for a Spatial Assertion Language for Data Structures.* Springer Berlin Heidelberg, Berlin, Heidelberg, 108–119.

[17] Renato Cherini and Javier O. Blanco. 2009. Local Reasoning for Abstraction and Sharing. In *ACM Symposium on Applied Computing (SAC)*. 552–557.

[18] Stéphane Demri and Morgan Deters. 2016. Expressive Completeness of Separation Logic with Two Variables and No Separating Conjunction. *ACM Trans. Comput. Logic* 17, 2, Article 12 (Jan. 2016), 12:1–12:44 pages.

[19] Stéphane Demri, Didier Galmiche, Dominique Larchey-Wendling, and Daniel Méry. 2017. Separation Logic with One Quantified Variable. *Theory Comput. Syst.* 61, 2 (2017), 371–461.

[20] Robert Dockins, Aquinas Hobor, and Andrew W. Appel. 2009. A Fresh Look at Separation Algebras and Share Accounting, In Programming Languages and Systems (APLAS). *Lecture Notes in Computer Science* 5904, 161–177.

[21] Mike Dodds, Suresh Jagannathan, and Matthew J. Parkinson. 2011. Modular Reasoning for Deterministic Parallelism, In ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). *ACM SIGPLAN Notices* 46, 1, 259–270.

[22] Xinyu Feng. 2009. Local Rely-guarantee Reasoning. In *POPL '09*. ACM, 315–327.

[23] Melvin Fitting. 1996. *First-order Logic and Automated Theorem Proving (2nd Ed.).* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[24] Didier Galmiche and Dominique Larchey-Wendling. 2006. Expressivity Properties of Boolean BI through Relational Models, In Foundations of Software Technology and Theoretical Computer Science (FST TCS). *Lecture Notes in Computer Science* 4337, 357–368.

[25] Didier Galmiche and Daniel Méry. 2007. Tableaux and Resource Graphs for Separation Logic. *Journal of Logic and Computation* 20, 1 (2007), 189–231.

[26] Alexey Gotsman, Josh Berdine, and Byron Cook. 2011. Precision and the Conjunction Rule in Concurrent Separation Logic. *Electronic Notes in Theoretical Computer Science* 276 (2011), 171–190.

[27] Christian Haack and Clément Hurlin. 2009. Resource Usage Protocols for Iterators. *Journal of Object Technology* 8, 4 (2009), 55–83.

[28] Zhé Hóu. 2015. *Labelled Sequent Calculi and Automated Reasoning for Assertions in Separation Logic.* Ph.D. Dissertation. Australian National University.

[29] Zhé Hóu, Ranald Clouston, Rajeev Goré, and Alwen Tiu. 2014. Proof Search for Propositional Abstract Separation Logics via Labelled Sequents, In ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). *ACM SIGPLAN Notices*, 465–476.

[30] Zhé Hóu, Rajeev Goré, and Alwen Tiu. 2015. Automated Theorem Proving for Assertions in Separation Logic with all Connectives, In Automated Deduction (CADE). *Lecture Notes in Computer Science* 9195, 501–516.

[31] Zhé Hóu, David Sanan, Alwen Tiu, Rajeev Goré, and Ranald Clouston. 2016. Separata: Isabelle tactics for Separation Algebra. *Archive of Formal Proofs* (Nov. 2016). http://isa-afp.org/entries/Separata.shtml, Formal proof development.

[32] Zhé Hóu, David Sanán, Alwen Tiu, and Yang Liu. 2017. Proof Tactics for Assertions in Separation Logic. In *To appear in Interactive Theorem Proving, 2017, Proceedings.*

[33] Zhé Hóu and Alwen Tiu. 2016. Completeness for a First-Order Abstract Separation Logic. In *Programming Languages and Systems - 14th Asian Symposium, APLAS 2016, Hanoi, Vietnam, November 21-23, 2016, Proceedings.* 444–463.

[34] Zhé Hóu, Alwen Tiu, and Rajeev Goré. 2013. A Labelled Sequent Calculus for BBI: Proof Theory and Proof Search, In Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX). *Lecture Notes in Computer Science*, 172–187.

[35] Zhé Hóu, Alwen Tiu, and Rajeev Goré. 2015. A Labelled Sequent Calculus for BBI: Proof Theory and Proof Search. *Journal of Logic and Computation* (2015).

[36] Samin S. Ishtiaq and Peter W. O'Hearn. 2001. BI as an Assertion Language for Mutable Data Structures, In ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). *ACM SIGPLAN Notices* 36, 3, 14–26.

[37] Jonas B. Jensen. 2013. *Enabling Concise and Modular Specifications in Separation Logic. Ph.D. dissertation.* IT University of Copenhagen, Chapter 7 (Techniques for Model Construction in Separation Logic).

[38] Jonas B. Jensen, Nick Benton, and Andrew Kennedy. 2013. High-Level Separation Logic for Low-Level Code, In ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). *ACM SIGPLAN Notices* 48, 1, 301–314.

[39] Jonas B. Jensen and Lars Birkedal. 2012. Fictional Separation Logic, In Programming Languages and Systems (ESOP). *Lecture Notes in Computer Science* 7211, 377–396.

[40] Gerwin Klein, Rafal Kolanski, and Andrew Boyton. 2012. Separation Algebra. *Archive of Formal Proofs* (May 2012). http://isa-afp.org/entries/Separation_Algebra.shtml, Formal proof development.

[41] Neelakantan R. Krishnaswami. 2006. Reasoning about Iterators with Separation Logic. In *Conference on Specification and Verification of Component-Based Systems (SAVCBS)*. ACM, 83–86.

[42] Dominique Larchey-Wendling. 2016. The formal strong completeness of partial monoidal Boolean BI. *J. Log. Comput.* 26, 2 (2016), 605–640.

[43] Dominique Larchey-Wendling and Didier Galmiche. 2009. Exploring the relation between intuitionistic BI and Boolean BI: An unexpected embedding. *Mathematical Structures in Computer Science* 19, 3 (2009), 435–500.

[44] Dominique Larchey-Wendling and Didier Galmiche. 2010. The Undecidability of Boolean BI through Phase Semantics. In *IEEE Symposium on Logic in Computer Science (LICS)*. 140–149.

[45] Dominique Larchey-Wendling and Didier Galmiche. 2014. Looking at Separation Algebras with Boolean BI-eyes, In IFIP TC 1/WG 2.2 International Conference on Theoretical Computer Science (TCS). *Lecture Notes in Computer Science* 8705.

[46] Wonyeol Lee and Sungwoo Park. 2014. A Proof System for Separation Logic with Magic Wand, In ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). *ACM SIGPLAN Notices* 49, 1, 477–490.

[47] Tadao Murata. 1989. Petri nets: Properties, analysis and applications. *Proc. IEEE* 77, 4 (1989), 541–580.

[48] Toby Murray, Daniel Matichuk, Matthew Brassil, Peter Gammie, Timothy Bourke, Sean Seefried, Corey Lewis, Xin Gao, and Gerwin Klein. 2013. seL4: from General Purpose to a Proof of Information Flow Enforcement. In *SP'13*. 415–429.

[49] Sara Negri and Jan von Plato. 2001. *Structural Proof Theory*. Cambridge University Press.

[50] Peter W. O'Hearn and David J. Pym. 1999. The Logic of Bunched Implications. *Bulletin of Symbolic Logic* 5, 2 (1999), 215–244.

[51] Jonghyun Park, Jeongbong Seo, and Sungwoo Park. 2013. A Theorem Prover for Boolean BI, In ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). *ACM SIGPLAN Notices* 48, 1, 219–232.

[52] Matthew Parkinson. 2005. *Local Reasoning for Java*. Ph.D. Dissertation. Cambridge.

[53] Matthew Parkinson. 2010. The Next 700 Separation Logics. In *Verified Software: Theories, Tools, Experiments (VSTTE)*. Vol. 6217. 169–182.

[54] Andrew Reynolds, Radu Iosif, Cristina Serban, and Tim King. 2016. *A Decision Procedure for Separation Logic in SMT*. Springer International Publishing, Cham, 244–261.

[55] John C. Reynolds. 2002. Separation logic: A logic for shared mutable data structures. In *IEEE Symposium on Logic in Computer Science (LICS)*. 55–74.

[56] Alan Robinson and Andrei Voronkov (Eds.). 2001. *Handbook of Automated Reasoning*. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands.

[57] Reuben N. S. Rowe and James Brotherston. 2017. Automatic Cyclic Termination Proofs for Recursive Procedures in Separation Logic. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs (CPP 2017)*. 53–65.

[58] Malte Schwerhoff and Alexander J. Summers. 2015. Lightweight Support for Magic Wands in an Automatic Verifier, In 29th European Conference on Object-Oriented Programming (ECOOP 2015). *Leibniz International Proceedings in Informatics (LIPIcs)* 37, 614–638.

[59] Élodie-Jane Sims. 2007. *Pointer Analysis and Separation Logic*. Ph.D. Dissertation. Kansas State University.

[60] Aditya Thakur, Jason Breck, and Thomas Reps. 2014. Satisfiability Modulo Abstraction for Separation Logic with Linked Lists. In *Proceedings of the 2014 International SPIN Symposium on Model Checking of Software (SPIN 2014)*. 58–67.

[61] Thomas Tuerk. 2009. A Formalisation of Smallfoot in HOL, In Theorem Proving in Higher Order Logics (TPHOLs). *Lecture Notes in Computer Science* 5674, 469–484.

[62] Viktor Vafeiadis. 2007. *Modular Fine-Grained Concurrency Verification*. Ph.D. Dissertation. PhD thesis, University of Cambridge.

[63] Viktor Vafeiadis and Chinmay Narayan. 2013. Relaxed Separation Logic: A Program Logic for C11 Concurrency, In ACM SIGPLAN international conference on Object Oriented Programming Systems Languages & Applications (OOPSLA). *ACM SIGPLAN Notices* 48, 10, 867–884.

[64] Viktor Vafeiadis and Matthew Parkinson. 2007. A Marriage of Rely/Guarantee and Separation Logic, In International Conference on Concurrency Theory (CONCUR). *Lecture Notes in Computer Science* 4703, 256–271.

[65] Jules Villard, Étienne Lozes, and Cristiano Calcagno. 2009. Proving Copyless Message Passing. In *Programming Languages and Systems (ESOP)*. Vol. 5904. 194–209.

[66] Hongseok Yang. 2001. *Local Reasoning for Stateful Programs*. Ph.D. Dissertation. Illinois at Urbana-Champaign.