

Proof Search for Propositional Abstract Separation Logics via Labelled Sequents

Zhé Hóu¹, Ranald Clouston¹, Rajeev Goré¹, and Alwen Tiu^{1,2}

¹ Research School of Computer Science, The Australian National University

² School of Computer Engineering, Nanyang Technological University

Abstract. Abstract separation logics are a family of extensions of Hoare logic for reasoning about programs that mutate memory. These logics are “abstract” because they are independent of any particular concrete memory model. Their assertion languages, called propositional abstract separation logics, extend the logic of (Boolean) Bunched Implications (BBI) in various ways.

We develop a modular proof theory for various propositional abstract separation logics using cut-free labelled sequent calculi. We first extend the cut-free labelled sequent calculus for BBI of Hóu et al to handle Calcagno et al’s original logic of separation algebras by adding sound rules for partial-determinism and cancellativity, while preserving cut-elimination. We prove the completeness of our calculus via a sound intermediate calculus that enables us to construct counter-models from the failure to find a proof. We then capture other propositional abstract separation logics by adding sound rules for indivisible unit and disjointness, while maintaining completeness and cut-elimination. We present a theorem prover based on our labelled calculus for these logics.

1 Introduction

Separation logic (SL) [29] is an extension of Hoare logic for reasoning about programs that explicitly mutate memory. This is achieved via an assertion language that, along with the usual (additive) connectives and predicates for first-order logic with arithmetic, has the multiplicative connectives *separating conjunction* $*$, its unit \top^* , and *separating implication*, or *magic wand*, \multimap , from the logic of Bunched Implications (BI) [26], as well as the *points-to* predicate \mapsto . 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 deallocation, and assertions such as “the heap is empty” [17]. In this paper we consider classical additives only.

The assertion language of SL must provide a notion of inference to support precondition strengthening and postcondition weakening, yet little such proof theory exists, despite its link with the proof-theoretically motivated BI. Instead, inference must proceed via reasoning directly about the concrete semantics of *heaps*, or finite partial functions 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 $P \multimap Q$ 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' . Such concrete semantics are appropriate for proving the correctness of a specific program in a specific environment, but mean that if a different notion of memory (or more generally, resource) is required then a new logic is also required.

Calcagno et al’s Abstract Separation Logic (ASL) [7] introduced the abstract semantics of partial cancellative monoids, or *separation algebras*, to unify notions of resources for heaps, heaps with permissions, Petri nets, and other examples. These semantics allow interpretation of $*$, \top^* and \multimap , although the latter is not considered by Calcagno et al. However \mapsto has no meaning in separation algebras in general, and is therefore not a first class citizen of ASL; it may be introduced as a predicate only if an appropriate concrete separation algebra is fixed. Calcagno et al do not consider proof theory for their assertion language, whose propositional fragment we call Propositional Abstract Separation Logic (PASL), but separation algebras are a restriction of *non-deterministic monoids*, which are known to give sound and complete semantics for BBI [13]. In this sense PASL is a refinement of BBI, differing only by the addition of the semantic properties of *partial-determinism* and *cancellativity*.

This link between BBI and PASL semantics raises the question of whether existing proof theory for BBI can be extended to give a sound and cut-free complete proof system for PASL; we answer this question in

the affirmative by extending the labelled sequent calculus LS_{BBI} of Hóu et al [16] by adding explicit rules for partial-determinism and cancellativity. The completeness of LS_{BBI} was demonstrated via the Hilbert axiomatisation of BBI, but this avenue is not open to us as partial-determinism and cancellativity are not axiomatisable in BBI [6]; instead completeness follows via a *counter-model construction* procedure. A novelty of our counter-model construction is that it can be modularly extended to handle extensions and sublogics of PASL.

We have also implemented proof search using our calculus (although no decision procedure for PASL is possible [5]). To our knowledge this is the first proof to be presented of the cut-free completeness of a calculus for PASL³, and our implementation is the first automated theorem prover for PASL.

Just as we have a family of separation logics, ranging across different concrete semantics, we now also have a family of abstract separation logics for different abstract semantics. These abstract semantics are often expressed as extensions of the usual notion of separation algebra; most notably Dockins et al [11] suggested the additional properties of positivity (here called *indivisible unit*), *disjointness*, *cross-split*, and *splittability*⁴. Conversely, the abstract semantics for Fictional Separation Logic [18] generalise separation algebras by dropping cancellativity. Hence there is demand for a modular approach to proof theory and proof search for propositional abstract separation logics. Labelled sequent calculi, with their explicitly semantics-based rules, provide good support for this modularity, as rules for the various properties can be added and removed as required. We investigate which properties can be combined without sacrificing our cut-free completeness result.

While we work with abstract models of separation logics, the reasoning principles behind our proof-theoretic methods should be applicable to concrete models also, so we investigate as further work how concrete predicates such as \mapsto might be integrated into our approach. Proof search strategies that come out of our proof-theoretic analysis could also potentially be applied to guide proof search in various encodings of separation logics [1,30,24] in proof assistants, e.g., they can guide the constructions of proof tactics needed to automate the reasoning tasks in those embeddings.

The remainder of this paper is structured as follows. Section 2 introduces propositional abstract separation logic based on separation algebra semantics, and gives the labelled sequent calculus for this logic. Fundamental results such as soundness and cut-elimination are also shown. Section 3 proves the completeness of the labelled calculus by counter-model construction. Section 4 discusses extensions of the labelled calculus with desirable properties in separation theory. Implementation and experiment are shown in Section 5, followed by Section 6 where a preliminary future work on how concrete predicates such as \mapsto might be integrated into our approach is outlined. Finally, Section 7 discusses related work.

2 The labelled sequent calculus for PASL

In this section we define the *separation algebra* semantics of Calcagno et al [7] for Propositional Abstract Separation Logic (PASL), and present the labelled sequent calculus LS_{PASL} for this logic, extending the calculus LS_{BBI} for BBI of Hóu et al [16] with partial-determinism and cancellativity. Soundness and cut-elimination are then demonstrated for LS_{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:

$$A ::= p \mid \top \mid \perp \mid \neg A \mid A \vee A \mid A \wedge A \mid A \rightarrow A \mid \top^* \mid A * A \mid A \multimap A$$

PASL-formulae are interpreted with respect to the following semantics:

³ Larchey-Wendling [20] claims that the tableaux for BBI with partial-determinism in [21] can be extended to cover cancellativity, but the “rather involved” proof has not appeared yet.

⁴ Dockins et al [11] also suggest generalising separation algebras to have a *set* of units; it is an easy corollary of [6, Lem. 3.11] that single-unit and multiple-unit separation algebras satisfy the same set of formulae.

Definition 2.1 A separation algebra, or *partial cancellative commutative monoid*, is a triple (H, \circ, ϵ) where H is a non-empty set, \circ is a partial binary function $H \times H \rightarrow H$ written infix, and $\epsilon \in H$, satisfying the following conditions, where ‘=’ is interpreted as ‘both sides undefined, or both sides 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 *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 $h_3 = h_4$. The paradigmatic example of a separation algebra is the set of *heaps*; here \circ is the combination of two heaps with disjoint domain, and ϵ is the empty heap.

In this paper we prefer to express PASL semantics in the style of *ternary relations*, to maintain consistency with the earlier work of Hóu et al on BBI [16]; it is easy to see that the definition below is a trivial notational variant of Def. 2.1.

Definition 2.2 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, ν) of a PASL Kripke relational frame (H, R, ϵ) and a *valuation* function $\nu : Var \rightarrow \mathcal{P}(H)$ (where $\mathcal{P}(H)$ is the power set of H). The forcing relation \Vdash between a model $\mathcal{M} = (H, R, \epsilon, \nu)$ and a formula is defined in Table 1, where we write $\mathcal{M}, h \not\Vdash A$ for the negation of $\mathcal{M}, h \Vdash A$. Given a model $\mathcal{M} = (H, R, \epsilon, \nu)$, 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.

$\mathcal{M}, h \Vdash p$	iff $p \in Var$ and $h \in \nu(p)$	$\mathcal{M}, h \Vdash \top^*$	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
$\mathcal{M}, h \Vdash A \rightarrow B$	iff $\mathcal{M}, h \not\Vdash A$ or $\mathcal{M}, h \Vdash B$	$\mathcal{M}, h \Vdash \perp$	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 $\mathcal{M}, h \not\Vdash A$
$\mathcal{M}, h \Vdash A * B$	iff $\exists h_1, h_2. (R(h_1, h_2, h)$ and $\mathcal{M}, h_1 \Vdash A$ and $\mathcal{M}, h_2 \Vdash B)$		
$\mathcal{M}, h \Vdash A \multimap B$	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 $\mathcal{M} = (H, R, \epsilon, \nu)$.

2.2 The labelled sequent calculus LS_{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 $(a, b \triangleright c)$, 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 multisets of labelled formulae. Then, $\Gamma; A$ is the multiset union of Γ and $\{A\}$.

As the interpretation of the logical connectives of PASL are the same as those for BBI, we may obtain a labelled sequent calculus for PASL, called LS_{PASL} , by adding the rules P (partial-determinism) and C (cancellativity) to LS_{BBI} [16]. The rules for LS_{PASL} are presented in Fig. 1, where p is a propositional variable, A, B are formulae, and $w, x, y, z \in \mathcal{L}$. Note that some rules use *label substitutions*. We write $\mathcal{G}[y/x]$ (resp. $\Gamma[y/x]$) for the (multi)set of relational atoms (resp. labelled formulae) for which the label variable x has been uniformly replaced by the label y . In each rule, the formula (resp. relational atom) shown explicitly

Identity and Cut:

$$\frac{}{\mathcal{G}; \Gamma; w : p \vdash w : p; \Delta} \text{id} \qquad \frac{\mathcal{G}; \Gamma \vdash x : A; \Delta \quad \mathcal{G}'; \Gamma'; x : A \vdash \Delta'}{\mathcal{G}; \mathcal{G}'; \Gamma; \Gamma' \vdash \Delta; \Delta'} \text{cut}$$

Logical Rules:

$$\begin{array}{c} \frac{}{\mathcal{G}; \Gamma; w : \perp \vdash \Delta} \perp L \qquad \frac{(\epsilon, w \triangleright \epsilon); \mathcal{G}; \Gamma \vdash \Delta}{\mathcal{G}; \Gamma; w : \top^* \vdash \Delta} \top^* L \qquad \frac{}{\mathcal{G}; \Gamma \vdash w : \top; \Delta} \top R \qquad \frac{}{\mathcal{G}; \Gamma \vdash \epsilon : \top^*; \Delta} \top^* R \\ \\ \frac{\mathcal{G}; \Gamma; w : A; w : B \vdash \Delta}{\mathcal{G}; \Gamma; w : A \wedge B \vdash \Delta} \wedge L \qquad \frac{\mathcal{G}; \Gamma \vdash w : A; \Delta \quad \mathcal{G}; \Gamma \vdash w : B; \Delta}{\mathcal{G}; \Gamma \vdash w : A \wedge B; \Delta} \wedge R \\ \\ \frac{\mathcal{G}; \Gamma \vdash w : A; \Delta \quad \mathcal{G}; \Gamma; w : B \vdash \Delta}{\mathcal{G}; \Gamma; w : A \rightarrow B \vdash \Delta} \rightarrow L \qquad \frac{\mathcal{G}; \Gamma; w : A \vdash w : B; \Delta}{\mathcal{G}; \Gamma \vdash w : A \rightarrow B; \Delta} \rightarrow R \\ \\ \frac{(x, y \triangleright z); \mathcal{G}; \Gamma; x : A; y : B \vdash \Delta}{\mathcal{G}; \Gamma; z : A * B \vdash \Delta} *L \qquad \frac{(x, z \triangleright y); \mathcal{G}; \Gamma; x : A \vdash y : B; \Delta}{\mathcal{G}; \Gamma \vdash z : A * B; \Delta} *R \\ \\ \frac{(x, y \triangleright z); \mathcal{G}; \Gamma \vdash x : A; z : A * B; \Delta \quad (x, y \triangleright z); \mathcal{G}; \Gamma \vdash y : B; z : A * B; \Delta}{(x, y \triangleright z); \mathcal{G}; \Gamma \vdash z : A * B; \Delta} *R \\ \\ \frac{(x, y \triangleright z); \mathcal{G}; \Gamma; y : A * B \vdash x : A; \Delta \quad (x, y \triangleright z); \mathcal{G}; \Gamma; y : A * B; z : B \vdash \Delta}{(x, y \triangleright z); \mathcal{G}; \Gamma; y : A * B \vdash \Delta} *L \end{array}$$

Structural Rules:

$$\begin{array}{c} \frac{(y, x \triangleright z); (x, y \triangleright z); \mathcal{G}; \Gamma \vdash \Delta}{(x, y \triangleright z); \mathcal{G}; \Gamma \vdash \Delta} E \qquad \frac{(u, w \triangleright z); (y, v \triangleright w); (x, y \triangleright z); (u, v \triangleright x); \mathcal{G}; \Gamma \vdash \Delta}{(x, y \triangleright z); (u, v \triangleright x); \mathcal{G}; \Gamma \vdash \Delta} A \\ \\ \frac{(x, \epsilon \triangleright x); \mathcal{G}; \Gamma \vdash \Delta}{\mathcal{G}; \Gamma \vdash \Delta} U \qquad \frac{(x, w \triangleright x); (y, y \triangleright w); (x, y \triangleright x); \mathcal{G}; \Gamma \vdash \Delta}{(x, y \triangleright x); \mathcal{G}; \Gamma \vdash \Delta} A_C \\ \\ \frac{(\epsilon, w' \triangleright w'); \mathcal{G}[w'/w]; \Gamma[w'/w] \vdash \Delta[w'/w]}{(\epsilon, w \triangleright w'); \mathcal{G}; \Gamma \vdash \Delta} E_{q1} \qquad \frac{(\epsilon, w' \triangleright w'); \mathcal{G}[w'/w]; \Gamma[w'/w] \vdash \Delta[w'/w]}{(\epsilon, w' \triangleright w); \mathcal{G}; \Gamma \vdash \Delta} E_{q2} \\ \\ \frac{(x, y \triangleright z); \mathcal{G}[z/w]; \Gamma[z/w] \vdash \Delta[z/w]}{(x, y \triangleright z); (x, y \triangleright w); \mathcal{G}; \Gamma \vdash \Delta} P \qquad \frac{(x, y \triangleright z); \mathcal{G}[y/w]; \Gamma[y/w] \vdash \Delta[y/w]}{(x, y \triangleright z); (x, w \triangleright z); \mathcal{G}; \Gamma \vdash \Delta} C \end{array}$$

Side conditions:

Only label variables (not ϵ) may be substituted for.

In $*L$ and $*R$, the labels x and y do not occur in the conclusion.

In the rules A, A_C , the label w does not occur in the conclusion.

Fig. 1. The labelled sequent calculus LS_{PASL} for Propositional Abstract Separation Logic.

in the conclusion is called the *principal* formula (resp. relational atom). A rule with no premise is called a *zero-premise* rule. Note that the $\rightarrow L$ rule is the classical implication left rule.

A function $\rho : \mathcal{L} \rightarrow H$ from labels to worlds is a *label mapping* iff it satisfies $\rho(\epsilon) = \epsilon$, mapping the label constant ϵ to the identity world of 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* $(H, R, \epsilon, \nu, \rho)$ as a model equipped with a label mapping.

Definition 2.3 (Sequent Falsifiability) *A sequent $\mathcal{G}; \Gamma \vdash \Delta$ is falsifiable in an extended model $\mathcal{M} = (H, R, \epsilon, \nu, \rho)$ if for every $x : A \in \Gamma$, $(a, b \triangleright c) \in \mathcal{G}$, and for every $y : B \in \Delta$, we have $(\mathcal{M}, \rho(x) \Vdash A)$, $R(\rho(a), \rho(b), \rho(c))$ and $(\mathcal{M}, \rho(y) \not\Vdash B)$. It is falsifiable if it is falsifiable in some extended model.*

To show that a formula A is *valid* in LS_{PASL} , we prove $\vdash w : A$ for an arbitrary label w . An example derivation is given in Fig. 2.

3.1 The intermediate system $ILSPASL$

We introduce an intermediate system where rules with substitutions (Eq_1, Eq_2, P, C) are isolated into an equivalence entailment \vdash_E , so that the resultant calculus does not involve substitutions.

Let r be an instance of a structural rule in which the substitution used is θ : this is the identity substitution except when r is Eq_1, Eq_2, P or C . We can view r (upwards) as a function that takes a set of relational atoms (in the conclusion of the rule) and outputs another set (in the premise). We write $r(\mathcal{G}, \theta)$ for the output relational atoms of an instance of r with substitution θ and with conclusion containing \mathcal{G} . Let σ be a sequence of instances of structural rules $[r_1(\mathcal{G}_1, \theta_1); \dots; r_n(\mathcal{G}_n, \theta_n)]$. Given a set of relational atoms \mathcal{G} , the result of the (backward) application of σ to \mathcal{G} , denoted by $\mathcal{S}(\mathcal{G}, \sigma)$, is defined as below, where \wedge is used for sequence concatenation:

$$\mathcal{S}(\mathcal{G}, \sigma) = \begin{cases} \mathcal{G} & \text{if } \sigma = [] \\ \mathcal{S}(\mathcal{G}\theta \cup r(\mathcal{G}', \theta), \sigma') & \text{if } \mathcal{G}' \subseteq \mathcal{G} \text{ and} \\ & \sigma = [r(\mathcal{G}', \theta)] \wedge \sigma' \\ \text{undefined} & \text{otherwise} \end{cases}$$

Given $\sigma = [r_1(\mathcal{G}_1, \theta_1); \dots; r_n(\mathcal{G}_n, \theta_n)]$, let $\text{subst}(\sigma)$ be the composite substitution $\theta_1 \circ \dots \circ \theta_n$, where $t(\theta_1 \circ \theta_2)$ means $(t\theta_1)\theta_2$. We write $s \equiv t$ to mean that s and t are syntactically equal.

Definition 3.1 (Equivalence entailment) *Let \mathcal{G} be a set of relational atoms. The entailment $\mathcal{G} \vdash_E (a = b)$ holds iff there exists a sequence σ of Eq_1, Eq_2, P, C applications s.t. $\mathcal{S}(\mathcal{G}, \sigma)$ is defined, and $a\theta \equiv b\theta$, where $\theta = \text{subst}(\sigma)$.*

Since substitution is no longer in the calculus, some inference rules that involve matching two equal labels need to be changed. We define the intermediate system $ILSPASL$ as $LSPASL$ minus $\{Eq_1, Eq_2, P, C\}$, with certain rules changed following Fig. 3. Note that the equivalence entailment \vdash_E is not a premise, but rather a condition of the rules.

$$\begin{array}{c} \frac{\mathcal{G} \vdash_E (w_1 = w_2)}{\mathcal{G}; \Gamma; w_1 : p \vdash w_2 : p; \Delta} \textit{id} \qquad \frac{\mathcal{G} \vdash_E (w = \epsilon)}{\mathcal{G}; \Gamma \vdash w : \top^*; \Delta} \top^*R \\ \\ \frac{(x, w \triangleright x'); (y, y \triangleright w); (x, y \triangleright x'); \mathcal{G}; \Gamma \vdash \Delta}{(x, y \triangleright x'); \mathcal{G}; \Gamma \vdash \Delta} A_C \\ \\ \frac{(u, w \triangleright z); (y, v \triangleright w); (x, y \triangleright z); (u, v \triangleright x'); \mathcal{G}; \Gamma \vdash \Delta}{(x, y \triangleright z); (u, v \triangleright x'); \mathcal{G}; \Gamma \vdash \Delta} A \\ \\ \frac{(x, y \triangleright w'); \mathcal{G}; \Gamma \vdash x : A; w : A * B; \Delta \quad (x, y \triangleright w'); \mathcal{G}; \Gamma \vdash y : B; w : A * B; \Delta}{(x, y \triangleright w'); \mathcal{G}; \Gamma \vdash w : A * B; \Delta} *R \\ \\ \frac{(x, w' \triangleright z); \mathcal{G}; \Gamma; w : A -* B \vdash x : A; \Delta \quad (x, w' \triangleright z); \mathcal{G}; \Gamma; w : A -* B; z : B \vdash \Delta}{(x, w' \triangleright z); \mathcal{G}; \Gamma; w : A -* B \vdash \Delta} -*L \end{array}$$

Side conditions:

$$\begin{array}{ll} \text{In } A, A_C, \text{ the label } w \text{ does not occur in the conclusion.} & \\ \text{In } A_C, (x, y \triangleright x'); \mathcal{G} \vdash_E (x = x') & \text{In } A, (x, y \triangleright z); (u, v \triangleright x'); \mathcal{G} \vdash_E (x = x') \\ \text{In } *R, (x, y \triangleright w'); \mathcal{G} \vdash_E (w = w') & \text{In } -*L, (x, w' \triangleright z); \mathcal{G} \vdash_E (w = w') \end{array}$$

Fig. 3. Changed rules in the intermediate system $ILSPASL$.

Given a set of relational atoms \mathcal{G} , we define the relation $=_{\mathcal{G}}$ as follows: $a =_{\mathcal{G}} b$ iff $\mathcal{G} \vdash_E (a = b)$. We show next that $=_{\mathcal{G}}$ is in fact an equivalence relation. This equivalence relation will be useful in our counter-model construction later.

Lemma 3.1 *Let \mathcal{G} be a set of relational atoms, if $\mathcal{G} \vdash_E (a = b)$ by applying σ_1 and $\mathcal{G} \vdash_E (c = d)$ by applying σ_2 , then $\exists \sigma_3$ s.t. $\mathcal{S}(\mathcal{G}, \sigma_1) \vdash_E (c\theta = d\theta)$ by σ_3 , where $\theta = \text{subst}(\sigma_1)$.*

Proof. Note that $\mathcal{S}(\mathcal{G}, \sigma_1) = \mathcal{G}\theta$. So essentially we need to show that if $\mathcal{G} \vdash_E (c = d)$, then $\mathcal{G}\theta \vdash_E (c\theta = d\theta)$. This is a consequence of the substitution Lemma 2.2. \square

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

The intermediate system $ILSP_{ASL}$ is equivalent to LS_{PASL} , i.e., every sequent provable in $ILSP_{ASL}$ is also provable in LS_{PASL} , and vice versa. This connection is easy to make, as is shown by Hóu et al. [16]. Properties such as contraction admissibility, closure under substitution etc. also hold for $ILSP_{ASL}$.

Lemma 3.3 *The intermediate labelled calculus $ILSP_{ASL}$ is equivalent to LS_{PASL} .*

3.2 Counter-model construction

We now give a counter-model construction procedure for $ILSP_{ASL}$ which, by Lemma 3.3, applies to LS_{PASL} as well. In the construction, we assume that labelled sequents such as $\mathcal{G}; \Gamma \vdash \Delta$ are built from **sets** $\mathcal{G}, \Gamma, \Delta$ rather than **multisets**. This is harmless since contraction is admissible in LS_{PASL} (and thus also in $ILSP_{ASL}$). Detailed proofs in this section can be found in Appendix A.3.

As the counter-model construction involves infinite sets and sequents, we extend the definition of \vdash_E appropriately as below.

Definition 3.2 *A (possibly infinite) set \mathcal{G} of relational atoms satisfies $\mathcal{G} \vdash_E (x = y)$ iff $\mathcal{G}_f \vdash_E (x = y)$ for some finite $\mathcal{G}_f \subseteq \mathcal{G}$.*

Given a set of relational atoms \mathcal{G} , 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 procedure is essentially a procedure to saturate a sequent by applying all 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 desired properties of such an infinite sequent which would allow the counter-model construction. This is given in the following definition.

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

1. *It is not the case that $a : A \in \Gamma$, $b : A \in \Delta$ and $a =_{\mathcal{G}} b$.*
2. *If $a : A \wedge B \in \Gamma$ then $a : A \in \Gamma$ and $a : B \in \Gamma$.*
3. *If $a : A \wedge B \in \Delta$ then $a : A \in \Delta$ or $a : B \in \Delta$.*
4. *If $a : A \rightarrow B \in \Gamma$ then $a : A \in \Delta$ or $a : B \in \Gamma$.*
5. *If $a : A \rightarrow B \in \Delta$ then $a : A \in \Gamma$ and $a : B \in \Delta$.*
6. *If $a : \top^* \in \Gamma$ then $a =_{\mathcal{G}} \epsilon$.*
7. *If $a : \top^* \in \Delta$ then $a \neq_{\mathcal{G}} \epsilon$.*
8. *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 $y : B \in \Gamma$.*
9. *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 $y : B \in \Delta$.*
10. *If $z : A * B \in \Gamma$ then $\forall x, y, z'$ if $(x, z' \triangleright y) \in \mathcal{G}$ and $z =_{\mathcal{G}} z'$, then $x : A \in \Delta$ or $y : B \in \Gamma$.*
11. *If $z : A * B \in \Delta$ then $\exists x, y, z'$ s.t. $(x, z' \triangleright y) \in \mathcal{G}$, $z =_{\mathcal{G}} z'$, $x : A \in \Gamma$ and $y : B \in \Delta$.*
12. *For any label $m \in \mathcal{L}$, $(m, \epsilon \triangleright m) \in \mathcal{G}$.*
13. *If $(a, b \triangleright c) \in \mathcal{G}$ then $(b, a \triangleright c) \in \mathcal{G}$.*
14. *If $(a, b \triangleright c) \in \mathcal{G}$ and $(d, e \triangleright a') \in \mathcal{G}$ and $a =_{\mathcal{G}} a'$, then $\exists f, f'$ s.t. $(d, f \triangleright c) \in \mathcal{G}$, $(b, e \triangleright f') \in \mathcal{G}$ and $f =_{\mathcal{G}} f'$.*
15. *$a : \perp \notin \Gamma$ and $a : \top \notin \Delta$.*

The next lemma shows that a Hintikka sequent gives a PASL Kripke relational frame which is a (counter-)model of the formulae in the sequent.

Lemma 3.4 *Every Hintikka sequent is falsifiable.*

To prove the completeness of $ILSPASL$, we have to show that any given unprovable sequent can be extended to a Hintikka sequent. To do so we need a way to enumerate all possible 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 contains structural rules with no principal formulas, 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:

$$ExF ::= F \mid \mathbb{U} \mid \mathbb{E} \mid \mathbb{A} \mid \mathbb{A}_C$$

where F is a formula, and $\mathbb{U}, \mathbb{E}, \mathbb{A}, \mathbb{A}_C$ are constants that are used as “dummy” principal formulae for the structural rules U, E, A , and A_C , respectively. 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 3.4 (Scheduler ϕ) A schedule is a tuple (O, m, ExF, R) , where O is either 0 (left) or 1 (right), m is a label, ExF is an extended formula and R is a set of relational atoms such that $|R| \leq 2$. 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 3.5 There exists a fair scheduler.

Proof. Our proof is similar to the proof of *fair strategy* of Larchey-Wendling [20]. 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. \square

From now on, we shall fix a fair scheduler, which we call ϕ . We assume that the set of labels \mathcal{L} is totally ordered, and its elements can be enumerated as a_0, a_1, a_2, \dots 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 $ILSPASL$ if the sequent $\vdash w : F$ is not cut-free derivable in $ILSPASL$ for any label $w \neq \epsilon$. Since we shall be concerned only with cut-free provability, in the following when we mention derivation, we mean cut-free derivation.

Definition 3.5 Let F be a formula which is not provable in $ILSPASL$. 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 $\phi(i) = (O_i, m_i, ExF_i, R_i)$.

- If $O_i = 0$, ExF_i is a PASL formula C_i and $m_i : C_i \in \Gamma_i$:
 - If $C_i = F_1 \wedge F_2$, then $\mathcal{G}_{i+1} = \mathcal{G}_i$, $\Gamma_{i+1} = \Gamma_i \cup \{m_i : F_1, m_i : F_2\}$, $\Delta_{i+1} = \Delta_i$.
 - If $C_i = F_1 \rightarrow F_2$. 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, $\mathcal{G}_{i+1} = \mathcal{G}_i$.
 - If $C_i = \top^*$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(\epsilon, m_i \triangleright \epsilon)\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.
 - If $C_i = F_1 * F_2$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(a_{2i}, a_{2i+1} \triangleright m_i)\}$, $\Gamma_{i+1} = \Gamma_i \cup \{a_{2i} : F_1, a_{2i+1} : F_2\}$, $\Delta_{i+1} = \Delta_i$.
 - If $C_i = F_1 \multimap F_2$ and $R_i = \{(x, m \triangleright y)\} \subseteq \mathcal{G}_i$ and $\mathcal{G}_i \vdash_E (m = m_i)$. 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, $\mathcal{G}_{i+1} = \mathcal{G}_i$.
- If $O_i = 1$, ExF_i is a PASL formula C_i , and $m_i : C_i \in \Delta$:
 - If $C_i = F_1 \wedge F_2$. 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 $\Gamma_{i+1} = \Gamma_i$.
 - If $C_i = F_1 \rightarrow F_2$, 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 $R_i = \{(x, y \triangleright m)\} \subseteq \mathcal{G}_i$ and $\mathcal{G}_i \vdash_E (m_i = m)$. 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 $\Gamma_{i+1} = \Gamma_i$.
 - If $C_i = F_1 \multimap F_2$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(a_{2i}, m_i \triangleright a_{2i+1})\}$, $\Gamma_{i+1} = \Gamma_i \cup \{a_{2i} : F_1\}$, and $\Delta_{i+1} = \Delta_i \cup \{a_{2i+1} : F_2\}$.
- If $ExF_i \in \{\mathbb{U}, \mathbb{E}, \mathbb{A}, \mathbb{A}_C\}$, we proceed as follows:
 - If $ExF_i = \mathbb{U}$, $R_i = \{(a_n, \epsilon \triangleright a_n)\}$, where $n \leq 2i + 1$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(a_n, \epsilon \triangleright a_n)\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.
 - If $ExF_i = \mathbb{E}$, $R_i = \{(x, y \triangleright z)\} \subseteq \mathcal{G}_i$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(y, x \triangleright z)\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.

- If $ExF_i = \mathbb{A}$, $R_i = \{(x, y \triangleright z); (u, v \triangleright x')\} \subseteq \mathcal{G}_i$ and $\mathcal{G}_i \vdash_E (x = x')$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(u, a_{2i} \triangleright z), (y, v \triangleright a_{2i})\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.
 - If $ExF_i = \mathbb{A}_C$, $R_i = \{(x, y \triangleright x')\} \subseteq \mathcal{G}_i$, and $\mathcal{G}_i \vdash_E (x = x')$ then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(x, a_{2i} \triangleright x), (y, y \triangleright a_{2i})\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.
- 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, R_i) corresponds to a potential rule application. If the components of the rule application are in the current sequent, we apply the corresponding rule to these components. The indexing of labels guarantees that the choice of a_{2i} and a_{2i+1} are always fresh for the sequent $\mathcal{G}_i; \Gamma_i \vdash \Delta_i$. The construction in Def. 3.5 non-trivially extends a similar construction of Hintikka CSS due to Larchey-Wendling [20], 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 *consistent* iff it does not have a derivation in $ILSPASL$. A (possibly infinite) sequent $\mathcal{G}; \Gamma \vdash \Delta$ is *finitely-consistent* iff every $\mathcal{G}'; \Gamma' \vdash \Delta' \subseteq_f \mathcal{G}; \Gamma \vdash \Delta$ is consistent.

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 $\mathcal{G}_i; \Gamma_i \vdash \Delta_i$, e.g., the labels a_{2i}, a_{2i+1} are always fresh for $\mathcal{G}_i; \Gamma_i \vdash \Delta_i$. This can be proved by a simple induction on i .

Lemma 3.6 *For any $i \in \mathcal{N}$, the following properties hold:*

1. $\mathcal{G}_i; \Gamma_i \vdash \Delta_i$ has no derivation
2. $\mathcal{L}_i \subseteq \{a_0, a_1, \dots, a_{2i-1}\}$
3. $\mathcal{G}_i; \Gamma_i \vdash \Delta_i \subseteq_f \mathcal{G}_{i+1}; \Gamma_{i+1} \vdash \Delta_{i+1}$

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 3.6 (Limit sequent) *Let F be a formula unprovable in $ILSPASL$. The limit sequent for F is the sequent $\mathcal{G}^\omega; \Gamma^\omega \vdash \Delta^\omega$ where $\mathcal{G}^\omega = \bigcup_{i \in \mathcal{N}} \mathcal{G}_i$ and $\Gamma^\omega = \bigcup_{i \in \mathcal{N}} \Gamma_i$ and $\Delta^\omega = \bigcup_{i \in \mathcal{N}} \Delta_i$ and where $\mathcal{G}_i; \Gamma_i \vdash \Delta_i$ is as defined in Def.3.5.*

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

Lemma 3.7 *If F is a formula unprovable in $ILSPASL$, then the limit labelled sequent for F is a Hintikka sequent.*

Finally we can state completeness: whenever a formula has no derivation in $ILSPASL$, we can extract an infinite counter-model based on the limit sequent and the Kripke relational frame.

Theorem 3.8 (Completeness) *Every formula F unprovable in $ILSPASL$ is not valid (in PASL relational Kripke models).*

Proof. We construct a limit sequent $\mathcal{G}^\omega; \Gamma^\omega \vdash \Delta^\omega$ for F following Def. 3.6. Note that by the construction of the limit sequent, we have $a_1 : F \in \Delta^\omega$. By Lemma 3.7, 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}, ν, ρ) that satisfies \mathcal{G}^ω 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. \square

Corollary 3.9 *If formula F is unprovable in LS_{PASL} then F is not valid (in PASL relational Kripke models).*

Proof. By Lemma 3.3 and Theorem 3.8. \square

4 Extensions of PASL

We now consider some extensions of PASL obtained by imposing additional properties on the semantics, as suggested by Dockins et al [11]. We show that sound rules for *indivisible unit* and the stronger property of *disjointness* can be added to our labelled sequent calculus without jeopardising our completeness proof, but that the more exotic properties of *splittability* and *cross-split* are not fully compatible with our current framework, they require non-trivial changes to the proofs in previous sections. See Appendix A.4 for the proofs in this section.

Indivisible unit. The unit ϵ in a commutative monoid (H, \circ, ϵ) is indivisible iff the following holds for any $h_1, h_2 \in H$: if $h_1 \circ h_2 = \epsilon$ then $h_1 = \epsilon$. Relationally, this corresponds to the first-order condition: $\forall h_1, h_2 \in H. \text{ if } R(h_1, h_2, \epsilon) \text{ then } h_1 = \epsilon$. Note that this also means that $h_2 = \epsilon$ whenever $h_1 \circ h_2 = \epsilon$. Most memory models in the literature obey indivisible unit [5], so this property seems appropriate for reasoning about concrete applications of separation logic. Indivisible unit can be axiomatised by the formula [6]: $\top^* \wedge (A * B) \rightarrow A$. We use the following sound rule to capture this property:

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

Note that we can then instantiate the label y to ϵ by applying Eq_1 upwards. Recall that the sequent calculus LS_{BBI} [16] is just the sequent calculus LS_{PASL} minus the rules C and P .

Proposition 4.1 *The formula $\top^* \wedge (A * B) \rightarrow A$ is provable in $LS_{BBI} + IU$.*

Theorem 4.2 *$LS_{PASL} + IU$ is sound and cut-free complete with respect to the class of PASL Kripke relational frames (and separation algebras) with indivisible unit.*

Disjointness. The separating conjunction $*$ in separation logic requires that the two combined heaps have disjoint domains [29]. In a separation algebra (H, \circ, ϵ) , disjointness is defined by the following additional requirement: $\forall h_1, h_2 \in H. \text{ if } h_1 \circ h_1 = h_2 \text{ then } h_1 = \epsilon$. Relationally: $\forall h_1, h_2 \in H. \text{ if } R(h_1, h_1, h_2) \text{ then } h_1 = \epsilon$. This condition is captured by the following rule, where x, y are labels.

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

Disjointness implies indivisible unit (but not vice versa), as shown by Dockins et al. [11]. We can prove the axiom for indivisible unit by using $LS_{BBI} + D$.

Proposition 4.3 *The formula $\top^* \wedge (A * B) \rightarrow A$ is provable in $LS_{BBI} + D$.*

Theorem 4.4 *$LS_{PASL} + D$ is sound and cut-free complete with respect to the class of PASL Kripke relational frames (and separation algebras) with disjointness.*

Proof. Similar to Theorem 4.2. □

Splittability and cross-split. The property of infinite splittability is sometimes useful when reasoning about the kinds of resource sharing that occur in divide-and-conquer style computations [11]. A separation algebra (H, \circ, ϵ) has splittability if for every $h_0 \in H \setminus \{\epsilon\}$, there are $h_1, h_2 \in H \setminus \{\epsilon\}$ such that $h_1 \circ h_2 = h_0$. Relationally, if $h_0 \neq \epsilon$ then there exist $h_1 \neq \epsilon, h_2 \neq \epsilon$ s.t. $R(h_1, h_2, h_0)$. This property can be axiomatised as the formula $\neg \top^* \rightarrow (\neg \top^* * \neg \top^*)$ [6]. We give the following rules for splittability:

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

$$\frac{(w \neq \epsilon); \mathcal{G}; \Gamma \vdash \Delta \quad (\epsilon, w \triangleright \epsilon); \mathcal{G}; \Gamma \vdash \Delta}{\mathcal{G}; \Gamma \vdash \Delta} \text{EM}$$

We add a new type of structure, namely *inequality*, to our calculus. The inequality structures are grouped with relational atoms in \mathcal{G} . The rule S directly encodes the semantics of splittability. We then need another rule $\neq L$ to conclude that $(w \neq w)$, for any label w , cannot be valid. Finally, the rule EM , named as the law of excluded middle for $(w = \epsilon) \vee (w \neq \epsilon)$, is essentially a cut on $w : \top^*$.

The advantage of formulating the above as structural rules is that these rules do not require extra arguments in the cut-elimination proof.

Cross-split is a rather complicated property. It specifies 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, ϵ) , 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:

$$\frac{(p, q \triangleright x); (p, s \triangleright u); (s, t \triangleright y); (q, t \triangleright v); (x, y \triangleright z); (u, v \triangleright z); \mathcal{G}; \Gamma \vdash \Delta}{(x, y \triangleright z); (u, v \triangleright z); \mathcal{G}; \Gamma \vdash \Delta} \text{cs}$$

The labels p, q, s, t do not occur in the conclusion

However, to ensure contraction admissibility, we need the following special case for this rule where the two principal relational atoms are the same.

$$\frac{(p, q \triangleright x); (p, s \triangleright x); (s, t \triangleright y); (q, t \triangleright y); (x, y \triangleright z); \mathcal{G}; \Gamma \vdash \Delta}{(x, y \triangleright z); \mathcal{G}; \Gamma \vdash \Delta} \text{cs}_C$$

The labels p, q, s, t do not occur in the conclusion

We note that Reynolds' heap model [29] falsifies splittability, as heaps are finite objects that only non-trivially split finitely often. On the other hand cross-split is true in the heap model; however we are not aware of any formulae whose proof requires this property.

Proposition 4.5 *The rules $S, \neq L, EM, CS, CS_C$ are sound.*

Proposition 4.6 *The axiom $\neg\top^* \rightarrow (\neg\top^* * \neg\top^*)$ for splittability is provable in $LS_{BBI} + \{S, \neq L, EM\}$.*

It is easy to check that the rules $S, \neq L, EM, CS, CS_C$ do not break cut-elimination. Since the completeness proofs for splittability and cross-split both involve modifications of our previous counter-model construction method, we present them together in Appendix A.4.

Theorem 4.7 *$LS_{PASL} + \{S, \neq L, EM, CS, CS_C\}$ is sound and cut-free complete with respect to the class of PASL Kripke relational frames (and separation algebras) with splittability and cross-split.*

Following previous work [11,6], we refer to the set of additional properties: cancellativity, partial-determinism, indivisible unit, disjointness, splittability, and cross-split (we assume single unit) as *separation theory*. And the set of rules $C, P, IU, D, S, \neq L, EM, CS, CS_C$ is called LS_{ST} . By a proof that appropriately incorporates the treatments for these properties in this section, we obtain the following result:

Theorem 4.8 *$LS_{BBI} + LS_{ST}$ is sound and cut-free complete with respect to the class of BBI Kripke relational frames with separation theory.*

Subsystems of $LS_{BBI} + LS_{ST}$ are also easy to obtain, we leave them for the reader to verify. We give some examples of subsystems in the next section.

5 Example subsystems of $LS_{BBI} + LS_{ST}$

We now consider various labelled calculi obtained by extending LS_{BBI} with one or more structural rules that correspond to partial-determinism (P), cancellativity (C), indivisible unit (IU), and disjointness (D). Most of the results in this section either directly follow from the proofs in previous sections, or are easy adaptations. As those conditions for monoids are often given in a modular way, e.g., in [11,6], it is not surprising that our structural rules can also be added modularly to LS_{BBI} , since they just simulate those conditions directly and individually in the labelled sequent calculus.

Calculi without cancellativity. Some notions of separation logic omit cancellativity [18], so dropping the rule C in LS_{PASL} gives an interesting system. The proofs in Sec. 3 still work if we just insist on a partial commutative monoid, and drop C in \vdash_E .

Theorem 5.1 *The labelled sequent calculus $LS_{BBI} + P$ is sound and cut-free complete with respect to the partial commutative monoidal semantics for BBI.*

As a result, it is easy to obtain the following sound and complete labelled calculi for the corresponding semantics: $LS_{BBI} + P + IU$ and $LS_{BBI} + P + D$. The proofs are similar to that for Theorem ??.

Calculi without partial-determinism. Similar to above, dropping partial-determinism gives another sound and complete labelled calculus $LS_{BBI} + C$, although we are not aware of any concrete models in separation logic that employ this framework.

Theorem 5.2 *The labelled sequent calculus $LS_{BBI} + C$ is sound and cut-free complete with respect to the cancellative commutative monoidal semantics for BBI.*

Again, using a similar argument as in Theorem. ??, we can obtain sound and complete labelled calculi $LS_{BBI} + C + IU$ and $LS_{BBI} + C + D$.

Calculi without partial-determinism and cancellativity. The labelled calculus $LS_{BBI} + IU$ is sound and complete by Prop. 4.1, and cut-elimination holds.

Theorem 5.3 *The labelled sequent calculus $LS_{BBI} + IU$ is sound and cut-free complete with respect to the commutative monoidal semantics for BBI with indivisible unit.*

To prove the completeness of the calculus $LS_{BBI} + D$, we need to go through the counter-model construction proof, since disjointness is not axiomatisable. It is easy to check that the proofs in Section 3 do not break when we define \vdash_E by using Eq_1, Eq_2, D only, and the Hintikka sequent then gives the BBI Kripke relational frame that obeys disjointness. The other proofs remain the same.

Theorem 5.4 *The labelled sequent calculus $LS_{BBI} + D$ is sound and cut-free complete with respect to the commutative monoidal semantics for BBI with disjointness.*

To summarise, our approach offers a sound and cut-free calculus for the extension of BBI with every combination of the properties P, C, IU, D . The case where none of the properties hold, i.e. regular BBI, have already been solved [16,27]. Omitting the cases covered by the implication of IU by D , this provides us with the following eleven labelled calculi:

$$\begin{array}{lll}
LS_{BBI} + IU & LS_{BBI} + C & LS_{BBI} + D \\
LS_{BBI} + P & LS_{PASL}(= LS_{BBI} + P + C) & \\
LS_{BBI} + P + IU & LS_{BBI} + C + IU & LS_{PASL} + IU \\
LS_{BBI} + P + D & LS_{BBI} + C + D & LS_{PASL} + D
\end{array}$$

The subsystems containing splittability or cross-split can be obtained by incorporating the treatments in the proof for Thm. 4.7. As there are too many combinations, we do not show them here.

6 Implementation and experiment

We discuss here an implementation of the proof system $LS_{PASL} + D$. It turns out that the A_C rule is admissible in this system; in fact it is admissible in the subsystem $LS_{BBI} + C$, as shown next. So we do not implement the A_C rule. See Appendix A.5 for detailed proofs in this section.

Proposition 6.1 *The A_C rule is admissible in $LS_{BBI} + C$.*

On the other hand, we restrict the rule U to create the identity relational atom $(w, \epsilon \triangleright w)$ only if w occurs in the conclusion (denoted as U' below). This does not reduce the power of LS_{PASL} , as will be shown next.

Lemma 6.2 *If $\mathcal{G}; \Gamma \vdash \Delta$ is derivable in LS_{PASL} , then it is derivable in $LS_{PASL} - U + U'$.*

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

1. Try to close the branch by rules $id, \perp L, \top^* R, \top^* R$.
2. If (1) not applicable, apply all possible Eq_1, Eq_2, P, C, IU, D rules to unify labels ⁵.
3. If (1-2) not applicable, apply invertible rules $\wedge L, \wedge R, \rightarrow L, \rightarrow R, *L, -* R, \top^* L$ in all possible ways.
4. If (1-3) not applicable, try $*R$ or $-* L$ by choosing existing relational atoms.
5. If none of the existing relational atoms are applicable, or all combinations of $*R, -* L$ formulae and relational atoms are already applied in (4), apply structural rules on the set \mathcal{G}_0 of relational atoms in the sequent as follows.
 - (a) Use E 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 .
 - (c) Use U' to generate all identity relational atoms for each label in \mathcal{G}_2 , giving the set \mathcal{G}_3 .
6. If none of above is applicable, fail.

Step (2) is terminating, because each substitution eliminates a label, and we only have finitely many labels. Step (5) is not applicable when $\mathcal{G}_3 = \mathcal{G}_0$. It is also clear that step (5) is terminating. Note that we forbid applications of the rule A to the pair $\{(x, y \triangleright z), (u, v \triangleright x)\}$ of relational atoms when $\{(u, w \triangleright z), (y, v \triangleright w)\}$, for some label w , (or any commutative variants of this pair, e.g., $\{(w, u \triangleright z); (v, y \triangleright w)\}$) is already in the sequent. This is because the created relational atoms in such an A application can be unified to existing ones by using rules P, C .

We view Γ, Δ in a sequent $\mathcal{G}; \Gamma \vdash \Delta$ as lists, and each time a logical rule is applied, we place the subformulae in 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 $-* L$ rule, the principal formula is moved to the end of the list, so that each principal formula for non-determinism rules $*R, -* L$ is considered fairly, i.e., applied in turn.

We incorporate a number of optimisations in the 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 [27] is also adopted. For $*R$ and $-* L$ applications, we forbid the search to consider applying the rule twice with the same pair of principal formula and principal 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 [27,16]. We borrow the idea of the heuristic method presented in [16] 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. Of course, 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.

We test our prover Separata for $LS_{PASL} + D$ on the formulae listed in Table 2; the times displayed are in seconds. We compare the results with provers for BBI, BBeye [27] and the incomplete heuristic-based $FVLS_{BBI}$ [16], when the formula is valid in BBI. 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 [27]. We can see that for formulae (1-14) the performance of Separata is comparable with the heuristic based prover for $FVLS_{BBI}$. Both provers are generally faster than BBeye. Formula (15) is one that BBeye had trouble with [16], but Separata handles it trivially. However, there are cases where BBeye is faster than Separata. We found the example formula (16) from a set of testings on randomly generated BBI theorems. Formula (17) is a converse example where a randomly generated BBI theorem causes BBeye to time out and $FVLS_{BBI}$ with heuristics to terminate within the timeout but without finding a proof due

⁵ Although IU is admissible, we keep it because it simplifies proof search.

	Formula	BBeye (opt)	FVLS _{BBI} (heuristic)	Separata
(1)	$(a \multimap b) \wedge (\top * (\top^* \wedge a)) \rightarrow b$	0.076	0.002	0.002
(2)	$(\top^* \multimap \neg(\neg a * \top^*)) \rightarrow a$	0.080	0.004	0.002
(3)	$\neg((a \multimap \neg(a * b)) \wedge ((\neg a \multimap \neg b) \wedge b))$	0.064	0.003	0.002
(4)	$\top^* \rightarrow ((a \multimap (b \multimap c)) \multimap ((a * b) \multimap c))$	0.060	0.003	0.002
(5)	$\top^* \rightarrow ((a * (b * c)) \multimap ((a * b) * c))$	0.071	0.002	0.004
(6)	$\top^* \rightarrow ((a * ((b \multimap e) * c)) \multimap ((a * (b \multimap e)) * c))$	0.107	0.004	0.008
(7)	$\neg((a \multimap \neg(\neg(d \multimap \neg(a * (c * b))) * a)) \wedge c * (d \wedge (a * b)))$	0.058	0.002	0.006
(8)	$\neg((c * (d * e)) \wedge B)$ where $B := ((a \multimap \neg(\neg(b \multimap \neg(d * (e * c))) * a)) * (b \wedge (a * \top)))$	0.047	0.002	0.013
(9)	$\neg(C * (d \wedge (a * (b * e))))$ where $C := ((a \multimap \neg(\neg(d \multimap \neg((c * e) * (b * a)))) * a) \wedge c)$	94.230	0.003	0.053
(10)	$(a * (b * (c * d))) \rightarrow (d * (c * (b * a)))$	0.030	0.004	0.002
(11)	$(a * (b * (c * d))) \rightarrow (d * (b * (c * a)))$	0.173	0.002	0.002
(12)	$(a * (b * (c * (d * e)))) \rightarrow (e * (d * (a * (b * c))))$	1.810	0.003	0.002
(13)	$(a * (b * (c * (d * e)))) \rightarrow (e * (b * (a * (c * d))))$	144.802	0.003	0.002
(14)	$\top^* \rightarrow (a * ((b \multimap e) * (c * d)) \multimap ((a * d) * (c * (b \multimap e))))$	6.445	0.003	0.044
(15)	$\neg(\top^* \wedge (a \wedge (b * \neg(c \multimap (\top^* \rightarrow a))))$	timeout(1000s)	0.003	0.003
(16)	$((D \rightarrow (E \multimap (D * E))) \rightarrow (b \multimap ((D \rightarrow (E \multimap ((D * a) * a))) * b)))$, where $D := \top^* \rightarrow a$ and $E := a * a$	0.039	0.005	8.772
(17)	$((\top^* \rightarrow (a \multimap (((a * (a \multimap b)) * \neg b) \multimap (a * (a * ((a \multimap b) * \neg b)))))) \rightarrow$ $((((\top^* * a) * (a * ((a \multimap b) * \neg b))) \rightarrow (((a * a) * (a \multimap b)) * \neg b)) * \top^*))$	timeout(1000s)	fail	49.584
(18)	$(F * F) \rightarrow F$, where $F := \neg(\top \multimap \neg \top^*)$	invalid	invalid	0.004
(19)	$(\top^* \wedge (a * b)) \rightarrow a$	invalid	invalid	0.003

Table 2. Experimental results from the prover Separata.

to its incompleteness. Formula (18) is valid only when the monoid is partial [22], and formula (19) is the axiom of indivisible unit. Some interesting cases for disjointness will be shown later. We do not investigate the details in the performances between these provers because they are for different logics. We leave further optimisations for Separata as future work.

7 Future work

In this paper we have focused on *propositional* inference, but the assertion language of separation logic is generally taken to include first-order logic, usually extended with arithmetic, or at least equality. More importantly, this language is interpreted only with respect to some *concrete* semantics, the most well-known of which is the original heap model of Reynolds [29]. We refer readers to that paper for a more careful description of this model; for the purposes of this section we will remark that *values* range across the integers, and *addresses* across some specified subset of the integers; that *heaps* are finite partial functions from addresses to values; and that *expressions* are built up from variables (evaluated with respect to some store), values, and the usual arithmetic operations.

The advantage of this model is that it supports the interpretation of the *points-to* predicate \mapsto , which allows direct reference to the contents of the heap: $E \mapsto E'$ is satisfied by a heap iff it is a singleton map sending the address specified by the expression E to the value specified by the expression E' .

The question for future research is whether our labelled sequent calculus and implementation could be extended to reason about such concrete predicates; this section presents preliminary work in this direction. While the full power of pointer arithmetic is an important subject for future work, for the purpose of this work we set arithmetic aside and let expressions range across store variables e, e_1, e_2, \dots only, as is done for example by Berdine et al [3]. The rules for quantifiers are straightforward, e.g.:

$$\frac{\mathcal{G}; \Gamma; h : F[e/x] \vdash \Delta}{\mathcal{G}; \Gamma; h : \exists x. F \vdash \Delta} \exists L \quad \frac{\mathcal{G}; \Gamma \vdash h : F[e/x]; \Delta}{\mathcal{G}; \Gamma \vdash h : \exists x. F; \Delta} \exists R$$

where e does not appear free in the conclusion of $\exists L$.

	Formula	Separata+
(1)	$((e_1 \mapsto e_2) * (e_1 \mapsto e_2)) \rightarrow \perp$	0.004
(2)	$((e_1 \mapsto e_2) * (e_3 \mapsto e_4)) \wedge ((e_1 \mapsto e_2) * (e_5 \mapsto e_6)) \rightarrow ((e_3 \mapsto e_6) * \top)$	0.002
(3)	$(\exists x_3 x_2 x_1. ((x_3 \mapsto x_2) * (x_1 \mapsto e)) \wedge (x_2 = x_1)) \rightarrow (\exists x_4 x_5. ((x_4 \mapsto x_5) * (x_5 \mapsto e)))$	0.001
(4)	$\neg((e_1 \mapsto e_2) * \neg(e_3 \mapsto e_4)) \rightarrow ((e_1 = e_3) \wedge ((e_2 = e_4) \wedge \top^*))$	0.004
(5)	$\neg(((e_1 \mapsto p) * (e_2 \mapsto q)) * \neg(e_3 \mapsto r)) \rightarrow \neg(((e_1 \mapsto p) * \neg(\neg((e_2 \mapsto q) * \neg(e_3 \mapsto r))))))$	0.002
(6)	$\neg((e_1 \mapsto p) * \neg(e_2 \mapsto q)) \rightarrow \neg((e_1 \mapsto p) * \neg((e_2 \mapsto q) \wedge ((e_1 \mapsto p) * \top)))$	0.003

Table 3. Experimental results from the prover Separata+.

Equality between variables simply requires that they are assigned by the store to the same value, giving rise to the rules

$$\frac{\mathcal{G}; \Gamma[e_2/e_1] \vdash \Delta[e_2/e_1]}{\mathcal{G}; \Gamma; h : e_1 = e_2 \vdash \Delta} =_L \quad \frac{}{\mathcal{G}; \Gamma \vdash h : e = e; \Delta} =_R$$

Points-to poses a more complex problem as it involves direct interaction with the contents of heaps; Fig. 4 presents putative labelled sequent rules for this predicate. The semantics of $e_1 \mapsto e_2$ first require that the heap be a singleton, which is a spatial property that can be captured by abstract semantics: a ‘singleton’ world is not equal to the identity world ϵ , and cannot be split into two non- ϵ worlds. This motivates rules $\mapsto L_1$ and $\mapsto L_2$. The rules $\mapsto L_3$ and $\mapsto L_4$ address the content of heaps: $\mapsto L_3$ says that two heaps with the same address (value of e_1) must be the same heap, and $\mapsto L_4$ says that a singleton heap makes a unique assignment.

$$\frac{}{\mathcal{G}; \Gamma; \epsilon : e_1 \mapsto e_2 \vdash \Delta} \mapsto_{L_1} \quad \frac{(\epsilon, h_0 \triangleright h_0); \mathcal{G}[\epsilon/h_1][h_0/h_2]; \Gamma[\epsilon/h_1][h_0/h_2]; h_0 : e_1 \mapsto e_2 \vdash \Delta[\epsilon/h_1][h_0/h_2]}{(h_0, \epsilon \triangleright h_0); \mathcal{G}[\epsilon/h_2][h_0/h_1]; \Gamma[\epsilon/h_2][h_0/h_1]; h_0 : e_1 \mapsto e_2 \vdash \Delta[\epsilon/h_2][h_0/h_1]} \mapsto_{L_2}$$

$$\frac{}{(h_1, h_2 \triangleright h_0); \mathcal{G}; \Gamma; h_0 : e_1 \mapsto e_2 \vdash \Delta}$$

$$\frac{\mathcal{G}[h/h']; \Gamma[h/h']; h : e_1 \mapsto e_2; h : e_1 \mapsto e_3 \vdash \Delta[h/h']}{\mathcal{G}; \Gamma; h : e_1 \mapsto e_2; h' : e_1 \mapsto e_3 \vdash \Delta} \mapsto_{L_3} \quad \frac{\mathcal{G}; \Gamma[e_1/e_3][e_2/e_4]; h : e_1 \mapsto e_2 \vdash \Delta[e_1/e_3][e_2/e_4]}{\mathcal{G}; \Gamma; h : e_1 \mapsto e_2; h : e_3 \mapsto e_4 \vdash \Delta} \mapsto_{L_4}$$

Fig. 4. Some rules for the predicate \mapsto in separation logic.

Our implementation of the calculus defined by adding these rules to $LS_{PASL} + D$ is not complete w.r.t. Reynolds’ semantics: for example it is unable to prove the formula below, which is based on a property for septraction due to Vafeiadis and Parkinson [32], and is valid in the heap model:

$$\top^* \rightarrow \neg((e_1 \mapsto e_2) * \neg(e_1 \mapsto e_2)) \quad (1)$$

This formula essentially asserts that a heap satisfying $e_1 \mapsto e_2$ is possible to construct, but our prover does not support explicit heap construction. Nevertheless this incomplete calculus does support strikingly elegant proofs of non-trivial separation logic inferences, such as the DISJOINT axiom of Parkinson [28, Cha. 5.3]:

$$\frac{\frac{\frac{}{(\epsilon, \epsilon \triangleright a_0); \epsilon : (e_1 \mapsto e_2) \vdash a_0 : \perp} \mapsto_{L_1}}{(a_1, a_1 \triangleright a_0); a_1 : (e_1 \mapsto e_2) \vdash a_0 : \perp} D}}{(a_1, a_2 \triangleright a_0); a_1 : (e_1 \mapsto e_2); a_2 : (e_1 \mapsto e_2) \vdash a_0 : \perp} \mapsto_{L_3}} \frac{}{; a_0 : (e_1 \mapsto e_2) * (e_1 \mapsto e_2) \vdash a_0 : \perp} *_L} \frac{}{; \vdash a_0 : ((e_1 \mapsto e_2) * (e_1 \mapsto e_2)) \rightarrow \perp} \rightarrow_R$$

Our experimental prover Separata+, extending $LS_{PASL} + D$ with the rules of this section, has proved a number of tested SL formulae very rapidly; see Table 3 for some examples. Formulae (1-3) are taken from Galniche and Méry [14]; in particular, the first is the DISJOINT axiom proved above. Formulae (4-6) are taken from Vafeiadis and Parkinson’s study of magic wand’s De Morgan dual *septraction*, $\neg(A * \neg B)$ [32]. These results present encouraging evidence that the work of this paper may form the basis of practical theorem proving for the assertion language of separation logic.

Dealing with splittability and cross-split in our labelled calculi is one of our next goals. We are also interested in extending the techniques of this paper to concrete semantics other than Reynolds’ heap models, such as those surveyed by Calcagno et al [7] and Jensen and Birkedal [18].

8 Related work

There are many more automated tools, formalisations, and logical embeddings for separation logic than can reasonably be surveyed within the scope of this conference paper. Almost all are not directly comparable to this paper because they deal with separation logic for some *concrete* semantics.

One exception to this concrete approach is Holfoot [31], 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 when automating any notion of SL, because \multimap is a source of undecidability [4]. Conversely, the mechanisations and embeddings that do incorporate magic wand tend to give little thought to (semi-) decision procedures. An important exception to this is the tableaux of Galmiche and Méry [14], which are designed for the decidable fragment of the assertion language of concrete separation logic with \multimap identified by Calcagno et al [9], but may also be extendable to the full assertion language. These methods have not been implemented, and given the difficulty of the development we expect that practical implementation would be non-trivial. Another partial exception to the trend to omit \multimap is SmallfootRG [8], which supports automation yet includes *septraction* [32], the De Morgan dual of \multimap . However SmallfootRG does not support additive negation nor implication, and so \multimap cannot be recovered; indeed in this setting *septraction* is mere ‘syntactic sugar’ that can be eliminated.

The denigration of magic wand is not without cost, as the connective, while surely less useful than $*$, has found application. A non-exhaustive list follows: generating weakest preconditions via backwards reasoning [17]; specifying iterators [28,19,15]; reasoning about parallelism [12]; and various applications of *septraction*, such as the specification of iterators and buffers [10]. For a particularly deeply developed example, see the correctness proof for the Schorr-Waite Graph Marking Algorithm of Yang [33], which involves non-trivial inferences involving \multimap (Lems. 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.

Our work builds upon the labelled sequent calculi for BBI of Hóu et al [16]. Their prover $FVLS_{BBI}$ implements a free-variable calculus for BBI but is incomplete. Our extensions to Hóu et al involves two main advances: first, a counter-model construction necessary to prove completeness; second, our prover deals with labelled sequents directly and (given certain fairness assumptions) is a complete semi-decision procedure for PASL and its variants. The link between BBI and SL is also emphasised as motivation by Park et al [27], whose BBI prover BBeye was used for comparisons in Sec. 6. This work was recently refined by Lee and Park [23], in work independent to our own, to a labelled sequent calculus for Reynolds’ heap model. Their calculus, like ours, cannot prove the formula (1)⁶ and so is not complete for these semantics. Also related, but so far not implemented, are the tableaux for partial-deterministic BBI of Larchey-Wendling and Galmiche [21,20], which, as mentioned in the introduction to this paper, are claimed to be extendable with cancellativity to attain PASL via a “rather involved” proof. 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; this advantage comes from structural rules which directly capture the conditions on Kripke relational frames, and handle the equality of worlds by explicit global substitutions.

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 [6]; 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 in this setting is another promising future direction.

References

1. Andrew W. Appel. Tactics for separation logic. Unpublished, 2006.

⁶ Confirmed by private communications with authors.

2. Franz Baader. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
3. Josh Berdine, Cristiano Calcagno, and Peter W O'Hearn. Symbolic execution with separation logic. In *APLAS*, volume 3780 of *LNCS*, pages 52–68, 2005.
4. Rmi Brochenin, Stéphane Demria, and Etienne Lozes. On the almighty wand. *Inform. and Comput.*, 211:106–137, 2012.
5. James Brotherston and Max Kanovich. Undecidability of propositional separation logic and its neighbours. In *LICS*, pages 130–139. IEEE, 2010.
6. James Brotherston and Jules Villard. Parametric completeness for separation theories. Technical Report RN/13/11, UCL, 2013.
7. Cristiano Calcagno, Peter W. O'Hearn, and Hongseok Yang. Local action and abstract separation logic. In *LICS*, pages 366–378. IEEE, 2007.
8. Cristiano Calcagno, Matthew Parkinson, and Viktor Vafeiadis. Modular safety checking for fine-grained concurrency. In *SAS*, volume 4634 of *LNCS*, 2007.
9. Cristiano Calcagno, Hongseok Yang, and Peter W. O'Hearn. Computability and complexity results for a spatial assertion language for data structures. In *FSTTCS*, volume 2245 of *LNCS*, pages 108–119, 2001.
10. Renato Cherini and Javier O. Blanco. Local reasoning for abstraction and sharing. In *SAC*, pages 552–557. ACM, 2009.
11. Robert Dockins, Aquinas Hobor, and Andrew W. Appel. A fresh look at separation algebras and share accounting. In *APLAS*, volume 5904 of *LNCS*, pages 161–177, 2009.
12. Mike Dodds, Suresh Jagannathan, and Matthew J. Parkinson. Modular reasoning for deterministic parallelism. In *POPL*, pages 259–270. ACM, 2011.
13. Didier Galmiche and Dominique Larchey-Wendling. Expressivity properties of boolean BI through relational models. In *FSTTCS*, volume 4337 of *LNCS*, pages 357–368, 2006.
14. Didier Galmiche and Daniel Méry. Tableaux and resource graphs for separation logic. *J. Logic Comput.*, 20(1):189–231, 2007.
15. C. Haack and C. Hurlin. Resource usage protocols for iterators. *J. Object Tech.*, 8(4):55–83, 2009.
16. Zhe Hóu, Alwen Tiu, and Rajeev Goré. A labelled sequent calculus for BBI: Proof theory and proof search. In *Tableaux*, LNCS, 2013. page 172-187; extended version at arXiv:1302.4783.
17. S. Ishtiaq and Peter W. O'Hearn. BI as an assertion language for mutable data structures. In *POPL*, pages 14–26. ACM, 2001.
18. Jonas Braband Jensen and Lars Birkedal. Fictional separation logic. In *ESOP*, volume 7211 of *LNCS*, pages 377–396, 2012.
19. Neelakantan R. Krishnaswami. Reasoning about iterators with separation logic. In *SAVCBS*, pages 83–86. ACM, 2006.
20. Dominique Larchey-Wendling. The formal strong completeness of partial monoidal boolean BI. To appear in *J. Logic. Comput.*, 2013.
21. Dominique Larchey-Wendling and Didier Galmiche. Exploring the relation between intuitionistic BI and boolean BI: An unexpected embedding. *Math. Structures Comput. Sci.*, 19(3):435–500, 2009.
22. Dominique Larchey-Wendling and Didier Galmiche. The undecidability of boolean BI through phase semantics. In *LICS*, pages 140–149. IEEE, 2010.
23. Wonyeol Lee and Sungwoo Park. A proof system for separation logic with magic wand. Technical Report CSE-2013-7, POSTECH, 2013.
24. Andrew McCreight. Practical tactics for separation logic. In *TPHOLs*, volume 5674 of *Lecture Notes in Computer Science*, pages 343–358. Springer, 2009.
25. Sara Negri and Jan von Plato. *Structural Proof Theory*. Cambridge University Press, 2001.
26. Peter W. O'Hearn and David J. Pym. The logic of bunched implications. *Bull. Symbolic Logic*, 5(2):215–244, 1999.
27. Jonghyun Park, Jeongbong Seo, and Sungwoo Park. A theorem prover for boolean BI. In *POPL*, pages 219–232. ACM, 2013.
28. Matthew Parkinson. *Local Reasoning for Java*. PhD thesis, Cambridge, 2005.
29. John C Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS*, pages 55–74. IEEE, 2002.
30. Harvey Tuch, Gerwin Klein, and Michael Norrish. Types, bytes, and separation logic. In *POPL*, pages 97–108. ACM, 2007.
31. Thomas Tuerk. A formalisation of smallfoot in HOL. In *TPHOLs*, volume 5674 of *LNCS*, 2009.
32. Viktor Vafeiadis and Matthew Parkinson. A marriage of rely/guarantee and separation logic. In *CONCUR*, volume 4703 of *LNCS*, pages 256–271, 2007.
33. Hongseok Yang. *Local Reasoning for Stateful Programs*. PhD thesis, Illinois at Urbana-Champaign, 2001.

A Appendix

This section provides the details of the proofs in this paper. When presenting derivations, we use double line to indicate that the premise and the conclusion are equivalent; and use dotted line to indicate the use of lemmas, theorems, etc..

A.1 Proofs in Section 2.3

Proof of Lemma 2.2.

Proof. By induction on $ht(\Pi)$, we do a case analysis on the last rule applied in the derivation. There are three sub-cases: (1) neither x nor y is the label of the principal formula, (2) y is the label of the principal formula, and (3) x is the label of the principal formula. Most of the rules can be proved as in LS_{BBI} , here we only illustrate the new rules P and C . Apparently they all fall into the first sub-case, as they are structural rules and there is no principal formula for them.

If the last rule in Π is P , which generally runs as below,

$$\frac{\mathcal{G}[c/d]; (a, b \triangleright c); \Gamma[c/d] \vdash \Delta[c/d]}{\mathcal{G}; (a, b \triangleright c); (a, b \triangleright d); \Gamma \vdash \Delta} P$$

we further distinguish three cases: (1) $x \neq d$ and $x \neq c$; (2) $x = d$; and (3) $x = c$.

1. If $x \neq d$ and $x \neq c$, we need to consider three sub-cases:

- (a) If $y \neq d$ and $y \neq c$, then the two substitutions $[y/x]$ and $[c/d]$ do not interfere with each other, thus we can use the induction hypothesis to substitute $[y/x]$ and reorder the substitutions to obtain the desired derivation.

$$\frac{\frac{\mathcal{G}[c/d][y/x]; (a, b \triangleright c)[y/x]; \Gamma[c/d][y/x] \vdash \Delta[c/d][y/x]}{\mathcal{G}[y/x][c/d]; (a, b \triangleright c)[y/x]; \Gamma[y/x][c/d] \vdash \Delta[y/x][c/d]} \Pi'}{\mathcal{G}[y/x]; (a, b \triangleright c)[y/x]; (a, b \triangleright d)[y/x]; \Gamma[y/x] \vdash \Delta[y/x]} P$$

- (b) If $y = d$ we first use the induction hypothesis, substituting $[c/x]$, then obtain the following derivation:

$$\frac{\frac{\frac{\mathcal{G}[c/d][c/x]; (a, b \triangleright c)[c/x]; \Gamma[c/d][c/x] \vdash \Delta[c/d][c/x]}{\mathcal{G}[c/x][c/d]; (a, b \triangleright c)[c/x]; \Gamma[c/x][c/d] \vdash \Delta[c/x][c/d]} \Pi'}{\mathcal{G}[d/x][c/d]; (a, b \triangleright c)[d/x]; \Gamma[d/x][c/d] \vdash \Delta[d/x][c/d]} P}{\mathcal{G}[d/x]; (a, b \triangleright c)[d/x]; (a, b \triangleright d)[d/x]; \Gamma[d/x] \vdash \Delta[d/x]} P$$

- (c) If $y = c$, the proof is similar to above, without the second last step.

2. If $x = d$, we consider three sub-cases:

- (a) If $y \neq c$ and $y \neq \epsilon$, we use the induction hypothesis to substitute $[c/y]$, and obtain the following derivation:

$$\frac{\frac{\frac{\mathcal{G}[c/d][c/y]; (a, b \triangleright c)[c/y]; \Gamma[c/d][c/y] \vdash \Delta[c/d][c/y]}{\mathcal{G}[y/d][c/y]; (a, b \triangleright c)[c/y]; \Gamma[y/d][c/y] \vdash \Delta[y/d][c/y]} \Pi'}{\mathcal{G}[y/d]; (a, b \triangleright c); (a, b \triangleright y); \Gamma[y/d] \vdash \Delta[y/d]} P}{\mathcal{G}[\epsilon/d]; (a, b \triangleright c); (a, b \triangleright \epsilon); \Gamma[\epsilon/d] \vdash \Delta[\epsilon/d]} P$$

- (b) If $y \neq c$ but $y = \epsilon$, then we use induction hypothesis to substitute $[\epsilon/c]$, and obtain the following derivation:

$$\frac{\frac{\frac{\mathcal{G}[c/d][\epsilon/c]; (a, b \triangleright \epsilon); \Gamma[c/d][\epsilon/c] \vdash \Delta[c/d][\epsilon/c]}{\mathcal{G}[\epsilon/d][\epsilon/c]; (a, b \triangleright \epsilon); \Gamma[\epsilon/d][\epsilon/c] \vdash \Delta[\epsilon/d][\epsilon/c]} \Pi'}{\mathcal{G}[\epsilon/d]; (a, b \triangleright c); (a, b \triangleright \epsilon); \Gamma[\epsilon/d] \vdash \Delta[\epsilon/d]} P}{\mathcal{G}[\epsilon/d]; (a, b \triangleright c); (a, b \triangleright \epsilon); \Gamma[\epsilon/d] \vdash \Delta[\epsilon/d]} P$$

- (c) If $y = c$, then the case is reduced to admissibility of weakening on relational atoms.

3. If $x = c$, the cases are similar to those for $x = d$.

If the last rule in Π is C , the proof is analogous to the proof for P . □

Proof of Lemma 2.4.

Proof. The rules P, C themselves are trivially invertible, since the inverted versions can be proved by using Lemma 2.2. The invertibility of all other rules except for \top^*L can be proved similarly as in LS_{BBI} . Here we show the proof for \top^*L in LS_{PASL} . We do an induction on the height of the derivation. Base case is the same as the proof for LS_{BBI} . For the inductive case, we illustrate the cases where the last rule in the derivation is P or C . Assume w.l.o.g. that the principal formula for the rule \top^*L is $x : \top^*$.

1. If the last rule is P , which runs as below.

$$\frac{\mathcal{G}[c/d]; (a, b \triangleright c); \Gamma[c/d]; x : \top^*[c/d] \vdash \Delta[c/d]}{\mathcal{G}; (a, b \triangleright c); (a, b \triangleright d); \Gamma; x : \top^* \vdash \Delta} P$$

we distinguish three sub-cases:

- (a) If $x \neq d$ and $x \neq c$, then the substitutions $[\epsilon/x]$ and $[c/d]$ are independent, thus we can use the induction hypothesis and applied the rule \top^*L (meanwhile switch the order of substitutions) to obtain the desired derivation.
- (b) If $x = d$, the original derivation is as follows.

$$\frac{\Pi}{\mathcal{G}[c/d]; (a, b \triangleright c); c : \top^*; \Gamma[c/d] \vdash \Delta[c/d]} P$$

$$\frac{\mathcal{G}; (a, b \triangleright c); (a, b \triangleright d); d : \top^*; \Gamma \vdash \Delta}{\mathcal{G}[c/d]; (a, b \triangleright c); (a, b \triangleright d); d : \top^*; \Gamma \vdash \Delta} P$$

We apply the induction hypothesis on the premise, then apply P to obtain the following derivation:

$$\frac{\frac{\frac{\mathcal{G}[c/d][\epsilon/c]; (a, b \triangleright \epsilon); \Gamma[c/d][\epsilon/c] \vdash \Delta[c/d][\epsilon/c]}{\mathcal{G}[\epsilon/d][\epsilon/c]; (a, b \triangleright \epsilon); \Gamma[\epsilon/d][\epsilon/c] \vdash \Delta[\epsilon/d][\epsilon/c]} P}{\mathcal{G}[\epsilon/d]; (a, b \triangleright c); (a, b \triangleright \epsilon); \Gamma[\epsilon/d] \vdash \Delta[\epsilon/d]} P}{\mathcal{G}[c/d][\epsilon/c]; (a, b \triangleright \epsilon); \Gamma[c/d][\epsilon/c] \vdash \Delta[c/d][\epsilon/c]} P$$

- (c) If $x = c$, the case is similar.

2. The case where the last rule is C is similar to above.

□

A.2 Proofs in Section 3.1

Proof of Lemma 3.2.

Proof. We show that \vdash_E satisfies the following conditions:

Reflexivity: for any label a that occurs in \mathcal{G} , we have $\mathcal{G} \vdash_E (a = a)$ by applying an empty sequence of Eq_1, Eq_2, P, C rules.

Symmetry: if $\mathcal{G} \vdash_E (x = y)$, via a sequence σ of Eq_1, Eq_2, P, C applications. Let $\theta = \text{subst}(\sigma)$, then by definition $x\theta \equiv y\theta$ in $\mathcal{G}\theta$. Thus $y\theta \equiv x\theta$, and we obtain that $\mathcal{G} \vdash_E (y = x)$.

Transitivity: if $\mathcal{G} \vdash_E (x = y)$ and $\mathcal{G} \vdash_E (y = z)$, then by Lemma 3.1 we obtain a sequence σ of Eq_1, Eq_2, P, C applications, and let $\theta = \text{subst}(\sigma)$, then $x\theta \equiv y\theta \equiv z\theta$. Thus $\mathcal{G} \vdash_E (x = z)$. □

A.3 Proofs in Section 3.2

Proof of Lemma 3.4.

Proof. Let $\mathcal{G}; \Gamma \vdash \Delta$ be a Hintikka sequent. We construct an extended model $\mathcal{M} = (H, \triangleright_{\mathcal{G}}, \epsilon_{\mathcal{G}}, \nu, \rho)$ as follows:

- $H = \{[a]_{\mathcal{G}} \mid a \in \mathcal{L}\}$
- $\triangleright_{\mathcal{G}}([a]_{\mathcal{G}}, [b]_{\mathcal{G}}, [c]_{\mathcal{G}})$ iff $\exists a', b', c'. (a', b' \triangleright c') \in \mathcal{G}, a =_{\mathcal{G}} a', b =_{\mathcal{G}} b', c =_{\mathcal{G}} c'$
- $\epsilon_{\mathcal{G}} = [\epsilon]_{\mathcal{G}}$
- $\nu(p) = \{[a]_{\mathcal{G}} \mid a : p \in \Gamma\}$ for every $p \in \text{Var}$
- $\rho(a) = [a]_{\mathcal{G}}$ for every $a \in \mathcal{L}$

To reduce clutter, we shall drop the subscript \mathcal{G} in $[a]_{\mathcal{G}}$ and write $[a], [b] \triangleright_{\mathcal{G}} [c]$ instead of $\triangleright_{\mathcal{G}}([a], [b], [c])$. We first show that $\mathcal{F} = (H, \triangleright_{\mathcal{G}}, \epsilon_{\mathcal{G}})$ is a PASL Kripke relational frame.

identity: for each $[a] \in H$, by definition, there must be a label $a' \in \mathcal{L}$ such that $[a] = [a']$. It follows from condition 12 in Def. 3.3 that $(a', \epsilon \triangleright a') \in \mathcal{G}$, thus $[a], [\epsilon] \triangleright_{\mathcal{G}} [a]$ holds.

commutativity: if $[a], [b] \triangleright_{\mathcal{G}} [c]$ holds, there must be some $(a', b' \triangleright c') \in \mathcal{G}$ s.t. $[a] = [a'], [b] = [b'], [c] = [c']$. Then by condition 13 in Def. 3.3, $(b', a' \triangleright c') \in \mathcal{G}$, therefore $[b], [a] \triangleright_{\mathcal{G}} [c]$ holds.

associativity: if $[a], [b] \triangleright_{\mathcal{G}} [c]$ and $[d], [e] \triangleright_{\mathcal{G}} [a]$ holds, then there exist some $(a', b' \triangleright c') \in \mathcal{G}$ and $(d', e' \triangleright a'') \in \mathcal{G}$ s.t. $[a] = [a'] = [a''], [b] = [b'], [c] = [c'], [d] = [d'], [e] = [e']$. Then by condition 14 in Def. 3.3, there also exist labels f, f' s.t. $(d', f \triangleright c') \in \mathcal{G}$ and $(b', e' \triangleright f') \in \mathcal{G}$ and $[f] = [f']$. Thus we can find $[f]$ s.t. $[d], [f] \triangleright_{\mathcal{G}} [c]$ and $[b], [e] \triangleright_{\mathcal{G}} [f]$ hold.

Partial-determinism: If $[a], [b] \triangleright_{\mathcal{G}} [c]$ and $[a], [b] \triangleright_{\mathcal{G}} [d]$ hold, then there exists some $(a', b' \triangleright c') \in \mathcal{G}$ and $(a'', b'' \triangleright d') \in \mathcal{G}$ s.t. $[a] = [a'] = [a''], [b] = [b'] = [b''], [c] = [c'], [d] = [d']$. Then by Lemma 3.1, $\mathcal{G} \vdash_E (c' = d')$ by using rule P to unify c' and d' , thus we obtain that $[c] = [c'] = [d] = [d']$.

Cancellativity: if $[a], [b] \triangleright_{\mathcal{G}} [c]$ and $[a], [d] \triangleright_{\mathcal{G}} [c]$ hold, then we can find some $(a', b' \triangleright c') \in \mathcal{G}$ and $(a'', d' \triangleright c'') \in \mathcal{G}$ s.t. $[a] = [a'] = [a''], [c] = [c'] = [c''], [b] = [b'], [d] = [d']$. Then by Lemma 3.1, $\mathcal{G} \vdash_E (b' = c')$ by using C to unify b' and c' , thus we obtain that $[b] = [b'] = [c] = [c']$.

So \mathcal{M} is indeed a model based on a PASL Kripke relational frame. We prove next that $\mathcal{G}; \Gamma \vdash \Delta$ is falsifiable 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 $\mathcal{M}, \rho(m) \Vdash A$.
- (3) If $m : A \in \Delta$ then $\mathcal{M}, \rho(m) \not\Vdash A$.

Item (1) follows from the definition of $\triangleright_{\mathcal{G}}$. We prove (2) and (3) simultaneously by induction on the size of A . In the following, to simplify presentation, we omit the \mathcal{M} from the forcing relation.

Base cases: when A is an atomic proposition p .

- If $m : p \in \Gamma$ then $[m] \in \nu(p)$ by definition of ν , 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 Def. 3.3. Thus $[m] \not\Vdash p$.

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 2 in Def. 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 3 in Def. 3.3, $m : A \in \Delta$ or $m : B \in \Delta$. By the induction hypothesis, $[m] \not\Vdash A$ or $[m] \not\Vdash B$, thus $[m] \not\Vdash A \wedge B$.
- If $m : A \rightarrow B \in \Gamma$, by condition 4 in Def. 3.3, $m : A \in \Delta$ or $m : B \in \Gamma$. By the induction hypothesis, $[m] \not\Vdash A$ or $[m] \Vdash B$, thus $[m] \Vdash A \rightarrow B$.
- If $m : A \rightarrow B \in \Delta$, by condition 5 in Def. 3.3, $m : A \in \Gamma$ and $m : B \in \Delta$. By the induction hypothesis, $[m] \Vdash A$ and $[m] \not\Vdash B$, thus $[m] \not\Vdash A \rightarrow B$.
- If $m : \top^* \in \Gamma$ then $[m] = [\epsilon]$ by condition 6 in Def. 3.3. Since $[\epsilon] \Vdash \top^*$, we obtain $[m] \Vdash \top^*$.
- If $m : \top^* \in \Delta$, by condition 7 in Def. 3.3, $[m] \neq [\epsilon]$ and then $[m] \not\Vdash \top^*$.
- If $m : A * B \in \Gamma$, by condition 8 in Def. 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 9 in Def. 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 $[a] \not\Vdash A$ or $[b] \not\Vdash B$. 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 $[a] \not\Vdash A$ or $[b] \not\Vdash B$ therefore $[m] \not\Vdash A * B$.
- If $m : A * B \in \Gamma$, by condition 10 in Def. 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 $[a] \not\Vdash A$ 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 $[a] \not\Vdash A$ or $[b] \Vdash B$, thus $[m] \Vdash A * B$.
- If $m : A * B \in \Delta$, by condition 11 in Def. 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 $[b] \not\Vdash B$ and $[a], [m] \triangleright_{\mathcal{G}} [b]$ holds, thus $[m] \not\Vdash A * B$.

□

Proof of Lemma 3.6.

Proof. Item 1 is based on the fact that the inference rules preserves 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 five cases depending on which rule is applied on $\mathcal{G}_i; \Gamma_i \vdash \Delta_i$.

1. If $*L$ is applied, then $\mathcal{L}_{i+1} = \mathcal{L}_i \cup \{a_{2i}, a_{2i+1}\} \subseteq \{a_1, \dots, a_{2i+1}\}$.
2. If $-*R$ is applied, same as above.
3. If U is applied, which generates $(a_n, \epsilon \triangleright a_n)$, then $n \leq 2i + 1$, thus $\mathcal{L}_{i+1} = \mathcal{L}_i \cup \{a_n\} \subseteq \{a_1, \dots, a_{2i+1}\}$.
4. If A is applied, the fresh label in the premise is a_{2i} . Thus $\mathcal{L}_{i+1} = \mathcal{L}_i \cup \{a_{2i}\} \subseteq \{a_1, \dots, a_{2i+1}\}$.
5. Otherwise, $\mathcal{L}_{i+1} = \mathcal{L}_i \subseteq \{a_1, \dots, a_{2i+1}\}$.

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

Proof of Lemma 3.7.

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-consistent. 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, $\mathcal{G}_i; \Gamma_i \vdash \Delta_i$ is not provable in $ILSPASL$. Since weakening is admissible in $ILSPASL$, $\mathcal{G}; \Gamma \vdash \Delta \subseteq_f \mathcal{G}_i; \Gamma_i \vdash \Delta_i$ cannot be provable either. So condition 1, 7, and 15 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.

2. 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 $\{m : F_1, m : F_2\} \subseteq \Gamma_{j+1} \subseteq \Gamma^\omega$.
3. 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.
4. If $m : F_1 \rightarrow F_2 \in \Gamma^\omega$, similar to case 3.
5. If $m : F_1 \rightarrow F_2 \in \Delta^\omega$, similar to case 2.
6. 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 $(\epsilon, m \triangleright \epsilon) \in \mathcal{G}_{j+1} \subseteq \mathcal{G}^\omega$. Then $\mathcal{G}^\omega \vdash_E (m = \epsilon)$ because $\mathcal{G}_{j+1} \vdash_E (m = \epsilon)$. So $m =_{\mathcal{G}^\omega} \epsilon$.
8. 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 $\Gamma_{j+1} = \Gamma_j \cup \{a_{2j} : F_1, a_{2j+1} : F_2\} \subseteq \Gamma^\omega$.
9. 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 $\mathcal{G}^\omega \vdash_E (m = m')$, there exists $j > i$ such that $(x, y \triangleright m') \in \mathcal{G}_j$ and $\mathcal{G}_j \vdash_E (m = m')$. 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 $\mathcal{G}_k \vdash (m = m')$, we have either $x : F_1 \in \Delta_{k+1} \subseteq \Delta^\omega$ or $y : F_2 \in \Delta_{k+1} \subseteq \Delta^\omega$.
10. If $m : F_1 \multimap F_2 \in \Gamma^\omega$, similar to case 8.
11. If $m : F_1 \multimap F_2 \in \Delta^\omega$, similar to case 9.
12. For each $a_n \in \mathcal{L}$, there is a $j \geq n$ such that $\phi(j) = (0, m, \mathbb{U}, \{(a_n, \epsilon \triangleright a_n)\})$ where U is applied to a_n . Then $\mathcal{G}_{j+1} = \mathcal{G}_j \cup \{(a_n, \epsilon \triangleright a_n)\} \subseteq \mathcal{G}^\omega$, because $n \leq 2j + 1$.
13. If $(x, y \triangleright z) \in \mathcal{G}^\omega$, then it is in some \mathcal{G}_i , where $i \in \mathcal{N}$. Then there is a $j > i$ such that $\phi(j) = (0, m, \mathbb{E}, \{(x, y \triangleright z)\})$ where E is applied. Then $\mathcal{G}_{j+1} = \mathcal{G}_j \cup \{(y, x \triangleright z)\} \subseteq \mathcal{G}^\omega$.

A special case where $x = b$ can be shown similarly.

2. If $x = a$, again, we consider two sub-cases:

- (a) If $y \neq \epsilon$, we use the induction hypothesis to substitute $[\epsilon/y]$, and then use IU to obtain the derivation.

$$\frac{\frac{\mathcal{G}[\epsilon/a][\epsilon/y]; (\epsilon, b \triangleright \epsilon); \Gamma[\epsilon/a][\epsilon/y] \vdash \Delta[\epsilon/a][\epsilon/y]}{\mathcal{G}[y/a][\epsilon/y]; (\epsilon, b \triangleright \epsilon); \Gamma[y/a][\epsilon/y] \vdash \Delta[y/a][\epsilon/y]} \Pi'}{\mathcal{G}[y/a]; (y, b \triangleright \epsilon); \Gamma[y/a] \vdash \Delta[y/a]} IU$$

- (b) If $y = \epsilon$, then the substitution gives $\mathcal{G}[\epsilon/a]; (\epsilon, b \triangleright \epsilon); \Gamma[\epsilon/a] \vdash \Delta[\epsilon/a]$, which is known to be derivable by using Π .

Invertibility: The rule IU is trivially invertible, as can be proved by using the substitution lemma. We show here (by induction on the height of the derivation) the case for the rule \top^*L , where the last rule in the derivation is IU . The other rules can be proved similarly as in [16]. The last rule IU runs as below.

$$\frac{\mathcal{G}[\epsilon/a]; (\epsilon, b \triangleright \epsilon); \Gamma[\epsilon/a]; x : \top^*[\epsilon/a] \vdash \Delta[\epsilon/a]}{\mathcal{G}; (a, b \triangleright \epsilon); \Gamma; x : \top^* \vdash \Delta} IU$$

we consider three sub-cases: (1) if $x \neq a$ and $x \neq b$, then we can safely apply the induction hypothesis on the premise, switch the order of substitutions, and apply IU to obtain $\mathcal{G}[\epsilon/x]; (a, b \triangleright \epsilon); \Gamma[\epsilon/x] \vdash \Delta[\epsilon/x]$. (2) If $x = a$, then the premise of the IU application is what we need to derive (with a dummy $\epsilon : \top^*$ on the left hand side of the sequent). (3) If $x = b$, then $\mathcal{G}[\epsilon/b]; (a, \epsilon \triangleright \epsilon); \Gamma[\epsilon/b] \vdash \Delta[\epsilon/b]$ can be derived by applying E and Eq_1 backwards then use the induction hypothesis to obtain the derivation. The details are the same as the derivation shown in 1(a) of the proof for substitution.

Completeness can be proved via the same counter-model construction for LS_{PASL} (Corollary 3.9). That is, we first define an intermediate calculus $ILS_{PASL} + IU$ that is equivalent to $LS_{PASL} + IU$, and do counter-model construction in $ILS_{PASL} + IU$. Since the IU rule involves substitution, the rule will be localised into the entailment relation \vdash_E , so the definition of \vdash_E in Definition 3.1 is modified to include IU in addition to Eq_1, Eq_2, P and C . Thus the rules of $ILS_{PASL} + IU$ are exactly the same as ILS_{PASL} , and the only change is in the definition of \vdash_E . The equivalence between $LS_{PASL} + IU$ and $ILS_{PASL} + IU$ can be proved as in Lemma 3.3.

Then we only need to show that a Hintikka sequent yields a Kripke relational frame that corresponds to a separation algebra with indivisible unit. In particular, no additional clauses are needed in the definition of Hintikka sequent since it is parametric on the entailment relation \vdash_E .

For a Hintikka sequent $\mathcal{G}; \Gamma \vdash \Delta$, suppose $(H, \triangleright_{\mathcal{G}}, [\epsilon])$ is the PASL Kripke relational frame generated by \mathcal{G} . Given any $[a], [b] \triangleright_{\mathcal{G}} [\epsilon]$, we can find a $(a', b' \triangleright c') \in \mathcal{G}$ such that $[a] = [a']$, $[b] = [b']$, $[\epsilon] = [c']$. Also, we can use the rule IU to derive $\mathcal{G} \vdash_E (a' = \epsilon)$. Thus by Lemma 3.1, we obtain $[a] = [a'] = [\epsilon]$. So the structure $(H, \triangleright_{\mathcal{G}}, [\epsilon])$ generated by \mathcal{G} is indeed a PASL Kripke relational frame that obeys indivisible unit.

The saturation with logical rules and structural rules $\mathbb{E}, \mathbb{U}, \mathbb{A}, \mathbb{A}_C$ is then the same as in Section 3. \square

Proof of Proposition 4.3.

Proof. we highlight the principal relational atoms where they are not obvious.

Note that \vdash_E is a side condition instead of a premise.

The system $ILSP_{ASL2}$ is equivalent to $LS_{PASL} + \{S, \neq, L, EM, CS, CS_C\}$, which is straightforward to show. Thus in what follows we give a counter-model construction procedure for $ILSP_{ASL2}$, then obtain the completeness result for both systems. As $ILSP_{ASL2}$ is just an extension of $ILSP_{ASL}$, we only give the additional definitions and proofs, and the parts where modifications are made.

Definition A.1 (Hintikka sequent) *A labelled sequent $\mathcal{G}; \Gamma \vdash \Delta$ is a Hintikka sequent if it satisfies the conditions in Definition 3.3 and the following, for any formulae A, B and any labels a, a', b, c, d, e, z, z' :*

16. For any label $m \in \mathcal{L}$, either $(m \neq \epsilon) \in \mathcal{G}$ or $m =_G \epsilon$.
17. If $(z \neq \epsilon) \in \mathcal{G}$, then $\exists x, y$, s.t. $(x, y \triangleright z) \in \mathcal{G}$, and $(x \neq \epsilon) \in \mathcal{G}$, and $(y \neq \epsilon) \in \mathcal{G}$.
18. It is not the case that $(a \neq a') \in \mathcal{G}$ and $a =_G a'$.
19. If $(a, b \triangleright z) \in \mathcal{G}$ and $(c, d \triangleright z') \in \mathcal{G}$ and $z =_G z'$, then $\exists ac, bc, ad, bd$ s.t. $(ad, ac \triangleright a) \in \mathcal{G}$, $(ad, bd \triangleright d) \in \mathcal{G}$, $(ac, bc \triangleright c) \in \mathcal{G}$, and $(bc, bd \triangleright b) \in \mathcal{G}$.

We extend the proof for Lemma 3.4 for the additional items in the above definition. That is, we show that every Hintikka sequent is satisfiable by additionally showing that the constructed model $(H, \triangleright_G, \epsilon_G, \nu, \rho)$ from the Hintikka sequent satisfies splittability and cross-split.

Proof. The following belongs to the first part of the proof for Lemma 3.4.

Splittability: for each $[a] \in H$, there is some $a' \in \mathcal{L}$ s.t. $[a] = [a']$. By condition 17 in Definition A.1, either (1) $(a' \neq \epsilon) \in \mathcal{G}$ or (2) $a' =_G \epsilon$. If (1) holds, by condition 18 in Definition A.1, there exist x, y s.t. $(x, y \triangleright a') \in \mathcal{G}$, $(x \neq \epsilon) \in \mathcal{G}$, and $(y \neq \epsilon) \in \mathcal{G}$ hold. Thus we can find $[x], [y]$ s.t. $[x], [y] \triangleright_G [a]$ holds and $[x] \neq [\epsilon]$, $[y] \neq [\epsilon]$. If (2) holds, splittability trivially holds.

Cross-split: if $[a], [b] \triangleright_G [z]$ and $[c], [d] \triangleright_G [z]$ hold, then we can find some $(a', b' \triangleright z') \in \mathcal{G}$ and $(c', d' \triangleright z'') \in \mathcal{G}$ s.t. $[a] = [a']$, $[b] = [b']$, $[c] = [c']$, $[d] = [d']$, and $[z] = [z'] = [z'']$. This implies that $z' =_G z''$. Then by condition 20 in Definition A.1, there are ad, ac, bc, bd , s.t. $(ad, ac \triangleright a') \in \mathcal{G}$, $(ad, bd \triangleright d') \in \mathcal{G}$, $(ac, bc \triangleright c') \in \mathcal{G}$, and $(bc, bd \triangleright b') \in \mathcal{G}$. Therefore we can find $[ad], [ac], [bc], [bd]$, s.t. $[ad], [ac] \triangleright_G [a]$, $[ad], [bd] \triangleright_G [d]$, $[ac], [bc] \triangleright_G [c]$, and $[bc], [bd] \triangleright_G [b]$. □

To deal with the new rules we added, we now define the *extended formulae* as

$$ExF ::= F \mid \mathbf{U} \mid \mathbf{E} \mid \mathbf{A} \mid \mathbf{A}_C \mid \mathbf{S} \mid \mathbf{EM} \mid \mathbf{CS} \mid \mathbf{CS}_C$$

where F is a BBI formula, and the others are constants. We also need to redefine the schedule to handle the inequality structure. Thus Definition 3.4 is now modified as below.

Definition A.2 (Scheduler ϕ) *A schedule is a tuple (O, m, ExF, R, I) , where O is either 0 (left) or 1 (right), m is a label, ExF is an extended formula, R is a set of relational atoms such that $|R| \leq 2$, and I is a singleton inequality. Let \mathcal{S} denote the set of all schedules. A scheduler is a function from the set of natural numbers \mathcal{N} to \mathcal{S} . A scheduler ϕ is fair if for every schedule $S \in \mathcal{S}$, the set $\{i \mid \phi(i) = S\}$ is infinite.*

It follows from the same reason that there exists a fair scheduler. The new component I in the scheduler is ignored in all the cases in Definition 3.5. However, we have to slightly extend the definition to accommodate splittability and cross-split. The original definition is rewritten as follows.

Definition A.3 *Let F be a formula which is not provable in $ILSP_{ASL2}$. We construct a series of finite sequents $\{\mathcal{G}_i; \Gamma_i \vdash \Delta_i\}_{i \in \mathcal{N}}$ 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 $\phi(i) = (O_i, m_i, ExF_i, R_i, I_i)$.

- If $O_i = 0$, ExF_i is a PASL formula C_i and $m_i : C_i \in \Gamma_i$:
 - If $C_i = F_1 \wedge F_2$, same as original def..
 - If $C_i = F_1 \rightarrow F_2$, same as original def..
 - If $C_i = \top^*$, same as original def..
 - If $C_i = F_1 * F_2$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(a_{4i}, a_{4i+1} \triangleright m_i)\}$, $\Gamma_{i+1} = \Gamma_i \cup \{a_{4i} : F_1, a_{4i+1} : F_2\}$, $\Delta_{i+1} = \Delta_i$.

- If $C_i = F_1 \multimap F_2$ and $R_i = \{(x, m \triangleright y)\} \subseteq \mathcal{G}_i$ and $\mathcal{G}_i \vdash_E (m = m_i)$, same as original def..
- If $O_i = 1$, ExF_i is a PASL formula C_i , and $m_i : C_i \in \Delta$:
 - If $C_i = F_1 \wedge F_2$, same as original def..
 - If $C_i = F_1 \rightarrow F_2$, same as original def..
 - $C_i = F_1 * F_2$ and $R_i = \{(x, y \triangleright m)\} \subseteq \mathcal{G}_i$ and $\mathcal{G}_i \vdash_E (m_i = m)$, same as original def..
 - If $C_i = F_1 \multimap F_2$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(a_{4i}, m_i \triangleright a_{4i+1})\}$, $\Gamma_{i+1} = \Gamma_i \cup \{a_{4i} : F_1\}$, and $\Delta_{i+1} = \Delta_i \cup \{a_{4i+1} : F_2\}$.
- If $ExF_i \in \{\mathbb{U}, \mathbb{E}, \mathbb{A}, \mathbb{A}_C, \mathbb{S}, \mathbb{EM}, \mathbb{CS}, \mathbb{CS}_C\}$, we proceed as follows:
 - If $ExF_i = \mathbb{U}$, $R_i = \{(a_n, \epsilon \triangleright a_n)\}$, where $n \leq 4i + 3$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(a_n, \epsilon \triangleright a_n)\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.
 - If $ExF_i = \mathbb{E}$, same as original def..
 - If $ExF_i = \mathbb{A}$, $R_i = \{(x, y \triangleright z); (u, v \triangleright x')\} \subseteq \mathcal{G}_i$ and $\mathcal{G}_i \vdash_E (x = x')$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(u, a_{4i} \triangleright z), (y, v \triangleright a_{4i})\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.
 - If $ExF_i = \mathbb{A}_C$, $R_i = \{(x, y \triangleright x')\} \subseteq \mathcal{G}_i$, and $\mathcal{G}_i \vdash_E (x = x')$ then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(x, a_{4i} \triangleright x), (y, y \triangleright a_{4i})\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.
 - If $ExF_i = \mathbb{S}$ and $I_i = \{(w \neq \epsilon)\} \subseteq \mathcal{G}_i$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(a_{4i}, a_{4i+1} \triangleright w), (a_{4i} \neq \epsilon), (a_{4i+1} \neq \epsilon)\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.
 - If $ExF_i = \mathbb{EM}$ and $R_i = \{(\epsilon, a_n \triangleright \epsilon)\}$, where $n \leq 4i + 3$. If there is no derivation for $(\epsilon, a_n \triangleright \epsilon)$; $\mathcal{G}_i; \Gamma_i \vdash \Delta_i$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(\epsilon, a_n \triangleright \epsilon)\}$, otherwise $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{a_n \neq \epsilon\}$. In both cases, $\Gamma_{i+1} = \Gamma_i$ and $\Delta_{i+1} = \Delta_i$.
 - If $ExF_i = \mathbb{CS}$, $R_i = \{(x, y \triangleright z), (u, v \triangleright z')\} \subseteq \mathcal{G}_i$, and $\mathcal{G}_i \vdash_E (z = z')$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(a_{4i}, a_{4i+1} \triangleright x), (a_{4i}, a_{4i+2} \triangleright u), (a_{4i+2}, a_{4i+3} \triangleright y), (a_{4i+1}, a_{4i+3} \triangleright v)\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.
 - If $ExF_i = \mathbb{CS}_C$ and $R_i = \{(x, y \triangleright z)\} \subseteq \mathcal{G}_i$, then $\mathcal{G}_{i+1} = \mathcal{G}_i \cup \{(a_{4i}, a_{4i+1} \triangleright x), (a_{4i}, a_{4i+2} \triangleright x), (a_{4i+2}, a_{4i+3} \triangleright y), (a_{4i+1}, a_{4i+3} \triangleright y)\}$, $\Gamma_{i+1} = \Gamma_i$, $\Delta_{i+1} = \Delta_i$.
- In all other cases, $\mathcal{G}_{i+1} = \mathcal{G}_i$, $\Gamma_{i+1} = \Gamma_i$ and $\Delta_{i+1} = \Delta_i$.

Lemma 3.6 is easy to show for the new definitions, the second item in the lemma should now be stated as $\mathcal{L} \subseteq \{a_0, a_1, \dots, a_{4i-1}\}$. We are ready to prove Lemma 3.7 for the new conditions in the Hintikka sequent.

Proof. Condition 18 holds because the limit sequent is finitely-consistent. We show the cases for conditions 16, 17, 19 as follows.

16. For each $a_n \in \mathcal{L}$, there is some natural number $j \geq n$ s.t. $\phi(j) = (O, m, \mathbb{EM}, \{(\epsilon, a_n \triangleright \epsilon)\}, I)$, where EM is applied to a_n . Then either (1) $\mathcal{G}_{j+1} = \mathcal{G}_j \cup \{(\epsilon, a_n \triangleright \epsilon)\}$ or (2) $\mathcal{G}_{j+1} = \mathcal{G}_j \cup \{a_n \neq \epsilon\}$, depending on which choice gives a finitely-consistent sequent $\mathcal{G}_{j+1}; \Gamma_{j+1} \vdash \Delta_{j+1}$. If (1) holds, then $(\epsilon, a_n \triangleright \epsilon) \in \mathcal{G}_{j+1} \subseteq \mathcal{G}^\omega$, and $\mathcal{G}^\omega \vdash_E (a_n = \epsilon)$ by an E_{q1} application, giving $a_n =_{\mathcal{G}^\omega} \epsilon$. If (2) holds, then $(a_n \neq \epsilon) \in \mathcal{G}_{j+1} \subseteq \mathcal{G}^\omega$.
17. If $(z \neq \epsilon) \in \mathcal{G}^\omega$, then $(z \neq \epsilon) \in \mathcal{G}_i$, for some $i \in \mathcal{N}$. Then there exists $j > i$ s.t. $\phi(j) = (O, m, \mathbb{S}, R, \{(z \neq \epsilon)\})$. Then $\mathcal{G}_{j+1} = \mathcal{G}_j \cup \{(a_{4j}, a_{4j+1} \triangleright z), (a_{4j} \neq \epsilon), (a_{4j+1} \neq \epsilon)\} \subseteq \mathcal{G}^\omega$.
19. If $(x, y \triangleright z) \in \mathcal{G}^\omega$ and $(u, v \triangleright z') \in \mathcal{G}^\omega$ and $z =_{\mathcal{G}^\omega} z'$. There must be some $i \in \mathcal{N}$ s.t. $\{(x, y \triangleright z), (u, v \triangleright z')\} \subseteq \mathcal{G}_i$ and $\mathcal{G}_i \vdash_E (z = z')$. Suppose $(x, y \triangleright z)$ and $(u, v \triangleright z')$ are distinct, then there exists $j > i$ s.t. $\phi(j) = (O, m, \mathbb{CS}, \{(x, y \triangleright z), (u, v \triangleright z')\}, I)$, and $\{(x, y \triangleright z), (u, v \triangleright z')\} \subseteq \mathcal{G}_j$, $\mathcal{G}_j \vdash_E (z = z')$ hold. By construction, $\mathcal{G}_{j+1} = \mathcal{G}_j \cup \{(a_{4j}, a_{4j+1} \triangleright x), (a_{4j}, a_{4j+2} \triangleright u), (a_{4j+2}, a_{4j+3} \triangleright y), (a_{4j+1}, a_{4j+3} \triangleright v)\} \subseteq \mathcal{G}^\omega$. If $(x, y \triangleright z)$ and $(u, v \triangleright z')$ are the same, a similar argument can be applied by using \mathbb{CS}_C .

The other cases are similar to the original proof. Note that the subscript of new labels needs to be adjusted accordingly. \square

Therefore the limit sequent in the new definition is indeed an Hintikka sequent, and we can extract an infinite counter-model from it. The completeness result follows.

A.5 Proofs for Section 6

Proof of Proposition 6.1.

Proof. We show that every derivation in $LS_{BBI} + C$ can be transformed into one with no applications of A_C . It is sufficient to show that we can eliminate a single application of A_C ; then we can eliminate all A_C in a derivation successively starting from the topmost applications in that derivation. So suppose we have a derivation in $LS_{BBI} + C$ of the form:

$$\frac{\Pi}{(x, w \triangleright x); (y, y \triangleright w); (x, y \triangleright x); \mathcal{G}; \Gamma \vdash \Delta} \text{ }_{A_C}$$

where w is a new label not in the root sequent. This is transformed into the following derivation:

$$\frac{\frac{\frac{\Pi'}{(x, \epsilon \triangleright x); (\epsilon, \epsilon \triangleright \epsilon); \mathcal{G}[\epsilon/y]; \Gamma[\epsilon/y] \vdash \Delta[\epsilon/y]}{(x, \epsilon \triangleright x); \mathcal{G}[\epsilon/y]; \Gamma[\epsilon/y] \vdash \Delta[\epsilon/y]} \text{ }_C}{(x, \epsilon \triangleright x); (x, y \triangleright x); \mathcal{G}; \Gamma \vdash \Delta} \text{ }_U}{(x, y \triangleright x); \mathcal{G}; \Gamma \vdash \Delta} \text{ }_U$$

where Π' is obtained by applying the substitutions $[\epsilon/y]$ and $[\epsilon/w]$ to Π (by using Lemma 2.2). Note that since w does not occur in the root sequent, $\mathcal{G}[\epsilon/y][\epsilon/w] = \mathcal{G}[\epsilon/y]$, $\Gamma[\epsilon/y][\epsilon/w] = \Gamma[\epsilon/y]$ and $\Delta[\epsilon/y][\epsilon/w] = \Delta[\epsilon/y]$. These substitutions do not introduce new instances of A_C . \square

Proof of Lemma 6.2.

Proof. The original U rule can be separated into two cases: (1) U' , with the restriction as described above and (2) U'' , where the created relational atom $(x, \epsilon \triangleright x)$ satisfies that x does not occur in the conclusion. We show that case (2) is admissible, leaving case (1) complete.

A simple induction on the height n of the derivation for $\mathcal{G}; \Gamma \vdash \Delta$. Suppose U'' is the last rule in the derivation, with a premise:

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

where x is a fresh label. Assume that the conclusion is not an empty sequent, there must be some label w that occurs in conclusion. By Lemma 2.2, replacing x by w , we obtain that

$$(w, \epsilon \triangleright w); \mathcal{G}; \Gamma \vdash \Delta$$

is derivable in $n - 1$ steps. By the induction hypothesis, there is a U'' -free derivation of this sequent, which leads to the derivation of $\mathcal{G}; \Gamma \vdash \Delta$ by applying the restricted rule U' . \square