Proof Search for Propositional Abstract Separation Logics via Labelled Sequent

Zhé Hóu Ranald Clouston Rajeev Goré
Research School of Computer Science
The Australian National University
{zhe.hou,ranald.clouston,rajeev.gore}@anu.edu.au

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 disjunctivity, while maintaining completeness and cut-elimination. We present a theorem prover based on our labelled calculus for these logics.

Categories and Subject Descriptors F.3.1 [Specifying and Verifying and Reasoning about Programs]: Logics of programs

General Terms Languages, Theory, Verification

Keywords Abstract separation logics, automated reasoning, labelled sequents, counter-model construction, bunched implications

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 ⊤∗, and separating implication, or magic wand, →∗, from the logic of Bunched Implications (BI) [26], as well as the points-to predicate →. 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 \( P \land Q \) iff it can be partitioned into heaps satisfying \( P \) and \( Q \) respectively; it satisfies \( P \lor Q \) iff it is empty; it satisfies \( P \land Q \) iff any extension with a heap that satisfies \( P \) must then satisfy \( Q \); and it satisfies \( E \rightarrow E' \) 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) [8] 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 \( \rightarrow \), although the latter is not considered by Calcagno et al. However \( \rightarrow \) 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 PASL\(^1\), 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 splitability\(^2\). 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 $\rightarrow$ 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, 24, 30] in proof assistants, e.g. they can guide the constructions of proof tactics needed to automate the reasoning tasks in those embeddings.

Acknowledgment. This work is partly supported by the Australian Research Council Discovery Grant DP110130173.

2. The labelled sequent calculus for PASL

In this section we define the separation algebra semantics of Calcagno et al [8] for Propositional Abstract Separation Logic (PASL), and present the labelled sequent calculus $LSP_{PASL}$ for this logic, adapting the calculus $LS_{SBFI}$ for BBI of H\o u et al [16]. Soundness and cut-elimination are then demonstrated for $LSP_{PASL}$.

2.1 Propositional abstract separation logic

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

$A ::= p \mid \top \mid \bot \mid \neg A \mid A \lor A \mid A \land A \mid A \rightarrow A \mid \top^* \mid A * A \mid A \rightarrow A$

PASL-formulae are interpreted according to the semantics below.

Definition 2.1. A separation algebra, or partial commutative combinatorial monoid, is a triple $(H, \circ, e)$ where $H$ is a non-empty set, $\circ$ is a partial binary function $H \times H \rightarrow H$ written infix, and $e \in H$, satisfying the following conditions, where ‘$\Rightarrow$’ is interpreted as ‘both sides undefined, or both sides defined and equal’:

identity: $\forall h \in H. h \circ e = h$

commutativity: $\forall h, h_2 \in H. h \circ h_2 = h_2 \circ h_1$

associativity: $\forall h_1, h_2, h_3 \in H. (h_1 \circ h_2) \circ h_3 = h_1 \circ (h_2 \circ h_3)$

1 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.

2 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.

cancellativity: $\forall h_1, h_2, h_3, h_4 \in H. \text{if } h_1 \circ h_2 = h_3 \text{ and } h_1 \circ h_4 = h_3 \text{ then } h_2 = h_4$

The paradigmatic example of a separation algebra is the set of heaps; here $\circ$ is the combination of two heaps with disjoint domain, and $e$ is the empty heap.

In the paper we prefer to express PASL semantics in the style of ternary relations, to maintain consistency with the earlier work of H\o u 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 $(H, R, e)$, where $H$ is a non-empty set of worlds, $R \subseteq H \times H \times H$, and $e \in H$, satisfying the following conditions for all $h_1, h_2, h_3, h_4, h_5 \in H$:

identity: $R(h_1, e, h_2) \iff h_1 = h_2$

commutativity: $R(h_1, h_2, h_3) \iff R(h_2, h_1, h_3)$

associativity: if $R(h_1, h_2, h_3)$ and $R(h_2, h_3, h_5)$ then there exists $h_4$ such that $R(h_4, h_3, h_5)$ and $R(h_1, h_4, h_5)$

cancellativity: if $R(h_1, h_2, h_3)$ and $R(h_3, h_1, h_4)$ then $h_2 = h_4$

partial-determinism: if $R(h_1, h_2, h_3)$ and $R(h_1, h_3, h_4)$ then $h_2 = h_4$

A PASL Kripke relational model is a tuple $(H, R, e, \nu)$ of a PASL Kripke relational frame $(H, R, e)$ and a valuation function $\nu : Var \rightarrow P(H)$, where $P(H)$ is the power set of $H$. The forcing relation $\models$ between a model $M = (H, R, e, \nu)$ and a formula is defined in Table 1, where we write $M, h \not\models A$ for the negation of $M, h \models A$. Given a model $M = (H, R, e, \nu)$, a formula is true at a world $h$ iff $M, h \models A$. The formula $A$ is valid iff it is true at all worlds of all models.

2.2 The labelled sequent calculus $LSP_{PASL}$

Let $Var$ be an infinite set of label variables, and let the set $\mathcal{L}$ of labels be $Var \cup \{e\}$, where $e$ is a label constant not in $Var$; 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 $(a, b, c)$, where $a, b, c$ are labels. A sequent takes the form

$G ; \Gamma \vdash \Delta$

where $G$ is a set of relational atoms, and $\Gamma$ and $\Delta$ are multisets of labelled formulae. Then, $\Gamma : \Delta$ is the multisets union of $\Gamma$ and $\Delta$.

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 $LSP_{PASL}$, by adding the rules $P$ (partial-determinism) and $C$ (cancellativity) to $LS_{SBFI}$ [16]. The rules for $LSP_{PASL}$ 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 $\Gamma [y/x]$ (resp. $G[y/x]$) for the set of labelled formulae (resp. relational atoms) for which the label variable $x$ has been uniformly replaced by the label $y$. In each rule, the formula (resp. relational atom) shown explicitly 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 if it satisfies $\rho(e) = e$, mapping the label constant $e$ to the identity world of $H$. Thus we define an extended PASL Kripke relational model $(H, R, e, \nu, \rho)$ as a model equipped with a label mapping.

Definition 2.3 (Sequent Falsifiability). A sequent $G ; \Gamma \vdash \Delta$ is falsifiable in an extended model $M = (H, R, e, \nu, \rho)$ if for every
Refer the interested reader to a similar proof for upwards. The proof is straightforward so we omit the details.

**Proof.**

\[ M, h \models p \iff p \in \text{Var} \text{ and } h \in v(p) \]

\[ M, h \models \bot \iff \text{never} \]

\[ M, h \models A \land B \iff M, h \models A \text{ and } M, h \models B \]

\[ M, h \models A \rightarrow B \iff M, h \models A \text{ or } M, h \models B \]

\[ M, h \models A \ast B \iff \exists h_1, h_2. (R(h_1, h_2, h) \text{ and } M, h_1 \models A \text{ and } M, h_2 \models B) \]

\[ M, h \models A \rightarrow B \text{ iff } \forall h_1, h_2. ((R(h, h_1, h_2) \text{ and } M, h_1 \models A) \text{ implies } M, h_2 \models B) \]

**Table 1.** Semantics of PASL, where \( M = (H, R, \epsilon, \nu) \).

---

**Identity and Cut:**

\[
\frac{\rho; \Gamma, \alpha \vdash \omega}{\rho; \Gamma, \alpha \vdash \omega}
\]

**Logical Rules:**

\[
\frac{\rho; \Gamma, \alpha \vdash \omega}{\rho; \Gamma, \alpha \vdash \omega}
\]

\[
\frac{\rho; \Gamma, \alpha \vdash \omega}{\rho; \Gamma, \alpha \vdash \omega}
\]

\[
\frac{\rho; \Gamma, \alpha \vdash \omega}{\rho; \Gamma, \alpha \vdash \omega}
\]

**Structural Rules:**

\[
\frac{\rho, \Gamma, \alpha \vdash \omega}{\rho, \Gamma, \alpha \vdash \omega}
\]

**Side conditions:**

Only label variables (not \( \epsilon \)) may be substituted for.

In the rule A, AC, the label \( w \) does not occur in the conclusion.

**Figure 1.** The labelled sequent calculus \( LSP_{PASL} \) for Propositional Abstract Separation Logic.

\[
x : A \in \Gamma, (a, b \in \Delta) \in \rho, \text{ and for every } y : B \in \Delta, \text{ we have } (M, \rho(x) \models A), (R(\rho(a), \rho(b), \rho(c)) \text{ and } (M, \rho(y) \not\models B)) \text{ is falsifiable if it is falsifiable in some extended model.}
\]

**Theorem 2.1** (Soundness). For any formula \( A \), and for an arbitrary label \( w \), if the labelled sequent \( \Gamma \vdash w : A \) is derivable in \( LSP_{PASL} \), then \( A \) is valid.

**Proof.** We prove that the rules of \( LSP_{PASL} \) preserve falsifiability upwards. The proof is straightforward so we omit the details; but refer the interested reader to a similar proof for \( LSP_{BBI} \) [16].

**2.3 Cut-elimination**

The only differences between \( LSP_{PASL} \) and \( LSP_{BBI} \) [16] are the additions of the structural rules \( P \) and \( C \), so we may prove cut-elimination by the same route, which in turn follows from the usual cut-elimination procedure for labelled sequent calculi for modal logics [25]. We therefore omit full proof details and simply list the necessary lemmas. We use \( ht(\Pi) \) to denote the height of the derivation \( \Pi \).
Lemma 2.3 (Admissibility of weakening). If \( \mathcal{G}; \Gamma \vdash \Delta \) is derivable in \( \text{LS}_{\text{BBI}} \), then for all structures \( \mathcal{G}, \Gamma \) and \( \Delta' \), the sequent \( \mathcal{G}; \Gamma \vdash \Delta; \Delta' \) is derivable with the same height in \( \text{LS}_{\text{BBI}} \).

Lemma 2.4 (Invertibility). If \( \Pi \) is a cut-free \( \text{LS}_{\text{BBI}} \), derivation of the conclusion of a rule, then there is a cut-free \( \text{LS}_{\text{BBI}} \) derivation for each premise, with height at most \( \text{ht}(\Pi) \).

Lemma 2.5 (Admissibility of contraction). If \( \mathcal{G}; \Gamma \vdash \Delta \) is derivable in \( \text{LS}_{\text{BBI}} \), then \( \mathcal{G}; \Gamma \vdash \Delta \) is derivable with the same height in \( \text{LS}_{\text{BBI}} \).

Theorem 2.6 (Cut-elimination). If \( \mathcal{G}; \Gamma \vdash \Delta \) is derivable in \( \text{LS}_{\text{BBI}} \) then it is derivable without using the cut rule.

Proof. The proof follows the same structure as that for \( \text{LS}_{\text{BBI}} \), utilising the lemmas above. The additional cases we need to consider are those involving the rules \( \Pi \) and \( \Pi' \); their treatment is similar to that for \( \text{LS}_{\text{BBI}} \) in the proof for \( \text{LS}_{\text{BBI}} \) [16].

3. Completeness of \( \text{LS}_{\text{BBI}} \)

We prove the completeness of \( \text{LS}_{\text{BBI}} \) with respect to the Kripke relational semantics by a counter-model construction. A standard way to construct a counter-model for an unprovable sequent is to show that it can be saturated by repeatedly applying all applicable inference rules to reach a limit sequent where a counter-model is shown by Hôu et al. [16]. Properties such as contraction admissibility, closure under substitution etc. also hold for \( \text{LS}_{\text{BBI}} \).

Lemma 3.1. Let \( \mathcal{G} \) be a set of relational atoms, if \( \mathcal{G} \vdash_E (a = b) \) by applying \( \sigma_1 \) and \( \mathcal{G} \vdash_E (c = d) \) by applying \( \sigma_2 \), then \( 3\sigma_3 \) such that \( S(\mathcal{G}, \sigma_1) \vdash_E (c = d) \) by \( \sigma_3 \), where \( \theta = \text{subt}(\sigma_1) \).

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

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

Proof. We show that \( =_\mathcal{G} \) satisfies the following conditions:

1. Reflexivity: for any label \( a \) that occurs in \( \mathcal{G} \), we have \( \mathcal{G} \vdash_E (a = a) \) by applying an empty sequence of \( \text{Eq}_1, \text{Eq}_2, P, C \) rules.

2. Symmetry: if \( \mathcal{G} \vdash_E (x = y) \), via a sequence \( \sigma \) of \( \text{Eq}_1, \text{Eq}_2, P, C \) applications. Let \( \theta = \text{subt}(\sigma) \), then by definition \( x\theta \equiv y\theta \) in \( \mathcal{G}\theta \). Thus \( y\theta \equiv x\theta \), and we obtain \( \mathcal{G} \vdash_E (y = x) \).

3. Transitivity: if \( \mathcal{G} \vdash_E (x = y) \) and \( \mathcal{G} \vdash_E (y = z) \), then by Lemma 3.1 we obtain a sequence \( \sigma \) of \( \text{Eq}_1, \text{Eq}_2, P, C \) applications, and let \( \theta = \text{subt}(\sigma) \), then \( x\theta \equiv y\theta \equiv z\theta \).

The intermediate system \( \text{LS}_{\text{BBI}} \) is equivalent to \( \text{LS}_{\text{BBI}} \), i.e., every sequent provable in \( \text{LS}_{\text{BBI}} \) is also provable in \( \text{LS}_{\text{BBI}} \), and vice versa. This connection is easy to make, as is shown by Hôu et al. [16]. Properties such as contraction admissibility, closure under substitution etc. also hold for \( \text{LS}_{\text{BBI}} \).

Lemma 3.3. The intermediate labelled calculus \( \text{LS}_{\text{BBI}} \) is equivalent to \( \text{LS}_{\text{BBI}} \).

We now give a counter-model construction procedure for \( \text{LS}_{\text{BBI}} \) which, by Lemma 3.3, applies to \( \text{LS}_{\text{BBI}} \) 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 our calculus.

As the counter-model construction involves infinite sets and sequents, we extend the definition of \( =_\mathcal{G} \) appropriately as below.

Definition 3.2. A (possibly infinite) set \( \mathcal{G} \) of relational atoms satisfies \( \mathcal{G} \vdash_E (x = y) \) iff \( \mathcal{G} \vdash_E (x = y) \) 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 \( a, b, c, d, e, z \):

1. It is not the case that \( a \in \Gamma, b \in \Delta \) and \( a =_\mathcal{G} b \).
2. If \( a : A \land B \in \Gamma \) then \( a : A \in \Gamma \) and \( a : B \in \Delta \).
3. If \( a : A \land B \in \Delta \) then \( a : A \in \Delta \) or \( a : B \in \Gamma \).
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 \( a =_\mathcal{G} \varepsilon \).
7. If \( a : \top \in \Delta \) then \( a =_\mathcal{G} \varepsilon \).
8. If \( z : A \leftrightarrow B \in \Gamma \) then \( \exists x, y, z' \) such that \( (x, y \leftrightarrow z') \in \mathcal{G} \) and \( z =_\mathcal{G} z' \) then \( x : A \in \Delta \) or \( y : B \in \Gamma \).
10. If \( z : A \rightarrow B \in \Gamma \) then \( \forall x, y, z' \) if \( (x, z' \triangleright y) \in \mathcal{G} \) and \( z = \triangleright z' \), then \( x : A \in \Delta \) or \( y \in \Delta \).

11. If \( z : A \rightarrow B \in \Delta \) then \( \exists x, y, z' \) such that \( (x, z' \triangleright y) \in \mathcal{G} \), 
\( z = \triangleright z' \), \( x : A \in \Gamma \) and \( y : B \in \Delta \).

12. For any label \( m \in \mathcal{L} \), \( m \in \mathcal{G} \).

13. If \( (a, b, c) \in \mathcal{G} \) then \( b \in \mathcal{G} \).

14. If \( (a, b, c) \in \mathcal{G} \) and \( (d, e, f) \in \mathcal{G} \) and \( a = \triangleright a' \), then \( 3f, f' \) such that \( (d, f \triangleright d) \in \mathcal{G} \), \( (b, e \triangleright f) \in \mathcal{G} \) and \( f = \triangleright f' \).

15. \( a : \bot \notin \Gamma \) and \( a : \top \notin \Delta \).

**Lemma 3.4. Every Hintikka sequent is falsifiable.**

**Proof.** Let \( \mathcal{G} : \Gamma \vdash \Delta \) be a Hintikka sequent. We construct an extended model \( M' = (H, \mathcal{G}, b, \nu, p) \) as follows:

- \( H = \{ [a]_\mathcal{G} | a \in \mathcal{L} \} \)
- \( b_\mathcal{G}([a]_\mathcal{G}, [b]_\mathcal{G}, [c]_\mathcal{G}) \) if \( \exists a', b', c'. (a', b' \triangleright c') \in \mathcal{G} \), \( a = \triangleright a' \), \( b = \triangleright b' \), \( c = \triangleright c' \)
- \( \nu([a]_\mathcal{G}) = \{ a : p \in \Gamma \} \) for every \( p \in \forall ar \)
- \( \nu((a) = [a]_\mathcal{G} \) for every \( a \in \mathcal{L} \).

To reduce clutter, we shall drop the subscript \( \mathcal{G} \) from \( [a]_\mathcal{G} \) and write \( [a], [b] \) instead of \( [a]_\mathcal{G}, [b]_\mathcal{G} \) as needed.

We first show that \( F' = (H, \nu, \nu) \) is a PASL Kripke relational frame. The identity, commutativity and associativity properties of \( F \) follow immediately from Definition 3.3, clause 12, 13, and 14, respectively. We next show partial-determinism and partial-determinism:

**Partial-determinism:** If \( [a], [b] \triangleright [c] \) and \( [a], [b] \triangleright [d] \), then there exists some \( a', b', c', d' \in \mathcal{G} \) and \( a', b', c', d' \in \mathcal{G} \) such that \( [a] = [a'] = [a''] \), \( [b] = [b'] = [b''] \), \( [c] = [c'] = [c''] \), \( [d] = [d'] = [d''] \). Then by Lemma 3.1, \( \mathcal{G} \vdash \mathcal{G} (c' = d') \) by using rule \( P \) to unify \( c' \) and \( d' \), thus we obtain that \( [c'] = [c''] \) and \( [d'] = [d''] \).

**Cancellativity:** If \( [a], [b] \triangleright [c] \) and \( [a], [d] \triangleright [c] \), then we can find some \( a', b', c', d' \in \mathcal{G} \) and \( a', b', c', d' \in \mathcal{G} \) such that \( [a] = [a'] = [a''] \), \( [c] = [c'] = [c''] \), \( [b] = [b'] = [b''] \), \( [d] = [d'] = [d''] \). Then by Lemma 3.1, \( \mathcal{G} \vdash \mathcal{G} (b' = c') \) by using \( C \) to unify \( b' \) and \( c' \), thus we obtain that \( [b'] = [b''] \) and \( [c'] = [c''] \).

So \( M' \) is indeed a model based on a PASL Kripke relational frame. We show next that \( \mathcal{G} : \Gamma \vdash \Delta \) is falsifiable in \( M' \). We need to show the following (where \( p \in m)\) for \( p \in m)\):

1. If \( (a, b, c) \in \mathcal{G} \) then \( [a], [b] \triangleright [c] \).
2. If \( m : A \in \Gamma \) then \( M, \nu(m) \vdash A \).
3. If \( m : A \in \Delta \) then \( M, \nu(m) \not\vdash A \).

Item (1) follows from the definition of \( \triangleright \). We prove (2) and (3) simultaneously by induction on the size of \( A \). In the following, to simplify presentation, we omit the \( M \) 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 \( \nu \), so \( \models p \).
- Suppose \( m : p \in \Delta \), then \( m' : p \in \Gamma \), for some \( m' \) such that \( m' = \triangleright m \). This violates condition 1 in Def. 3.3. Thus \( [m'] \not\models p \).

**Inductive cases:** when \( A \) is a compound formula. We show here the interesting cases involving multiplicative connectives:

1. If \( m' \in \Delta \), then \( \models \gamma \in \Delta \) by condition 6 in Def. 3.3. Since \( \langle \gamma \rangle \not\models \tau \), we obtain \( \models \gamma \not\models \tau \).
2. If \( m' : \tau \in \Delta \), by condition 7 in Def. 3.3, \( [m'] \not\models \tau \).
3. If \( m : \tau \in \Delta \), by condition 8 in Def. 3.3, \( a \triangleright b' \in \gamma \) such that \( (a, b' \triangleright m') \in \mathcal{G} \) and \( [m] = [m'] \) and \( a : A \in \Gamma \) and \( b : B \in \Delta \). By the induction hypothesis, \( [a] \not\vdash A \) and \( [b] \not\vdash B \). Thus \( [a], [b] \triangleright [m] \) holds and \( [m] \not\models \tau \).
4. If \( m : \tau \in \Delta \), by condition 9 in Def. 3.3, \( a \triangleright b' \in \gamma \) such that \( (a, b' \triangleright m') \in \mathcal{G} \) 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 \( [a], [b] \triangleright [m] \), there must be some \( a', b' \triangleright m'' \in \gamma \) such that \( [a'] \not\vdash A \) or \( [b'] \not\vdash B \). Thus \( [m] \not\models \tau \).
5. If \( m : A \triangleright B \in \Delta \), by condition 10 in Def. 3.3, \( a \triangleright b' \in \gamma \) such that \( (a, b' \triangleright m') \in \mathcal{G} \) and \( [m] = [m'] \) and \( a : A \in \Gamma \) and \( b : B \in \Delta \). By the induction hypothesis, \( [a] \not\vdash A \) and \( [b] \not\vdash B \). Consequently, \( [m] \not\models \tau \).

To prove the completeness of \( ILS_{PASL} \), 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 calculi 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:

\[
ExF ::= F \mid U \mid E \mid A \mid AC
\]

where \( F \) is a formula, and \( U, E, A, AC \) are constants. The intention is that \( U, E, A, AC \) will be as used as “dummy” principal formulae for the structural rules \( U, E, A, AC \), respectively. A scheduler then determines the sequence of rule applications to apply.
Definition 3.4. 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 \(S\) denote the set of all schedules. A scheduler is a function from the set of natural numbers, \(\mathbb{N}\), to \(S\). A scheduler \(\phi\) 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 follows a similar proof in [20]. To adapt their proof, we need to show that the set \(S\) is countable, which follows from the fact that \(S\) is a finite product of countable sets.

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

We say the formula \(F\) is not cut-free provable in \(ILS_{PASL}\) if the sequent \(\vdash w : F\) is not cut-free derivable in \(ILS_{PASL}\) 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 cut-free provable in \(ILS_{PASL}\). We construct a series of finite sequents \(\{\mathcal{G}_i; \Gamma_i \vdash \Delta_i\}_{i \in \mathbb{N}}\) from \(F\) where \(\mathcal{G}_i = \emptyset\) and \(\Delta_i = a_1 : F\). 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 we have the hypothesis \(\phi(i) = (O_i, m_i, ExF_i, R_i)\).

- If \(O_i = 0\), \(ExF_i\) is a PASL formula \(C_i\) and \(m_i : C_i \in \Gamma_i\):
  - If \(C_i = F_1 \land F_2\), 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 \(C_i = F_1 \rightarrow F_2\), then \(\mathcal{G}_{i+1} = \mathcal{G}_i, \Gamma_{i+1} = \Gamma_i \cup \{m_i : F_1\}, \Delta_{i+1} = \Delta_i \cup \{m_i\} \setminus \{F_1\}\).
- If \(O_i = 1\), \(ExF_i\) is a PASL formula \(C_i\) and \(m_i : C_i \in \Delta_i\):
  - If \(C_i = F_1 \land F_2\), then \(\mathcal{G}_{i+1} = \mathcal{G}_i, \Gamma_{i+1} = \Gamma_i \cup \{m_i : F_1\}, \Delta_{i+1} = \Delta_i \cup \{m_i\} \setminus \{F_2\}\).
  - If \(C_i = F_1 \rightarrow F_2\), then \(\mathcal{G}_{i+1} = \mathcal{G}_i, \Gamma_{i+1} = \Gamma_i \cup \{m_i : F_1\}, \Delta_{i+1} = \Delta_i \cup \{y : F_2\}\).

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 \(ILS_{PASL}\). The limit sequent for \(F\) is the sequent \(\Gamma^\omega \vdash \Delta^\omega\) where \(\Gamma^\omega = \bigcup_{i \in \mathbb{N}} \mathcal{G}_i\) and \(\Delta^\omega = \bigcup_{i \in \mathbb{N}} \Delta_i\), and where \(\mathcal{G}_i; \Gamma_i \vdash \Delta_i\) is as defined in Definition 3.5.

**Lemma 3.7. If \(F\) is a formula unprovable in \(ILS_{PASL}\), then the limit labelled sequent for \(F\) is a Hintikka sequent.**

Proof. Let \(\Gamma^\omega; \Delta^\omega\) be the limit sequent. First we show that \(\Gamma^\omega; \Delta^\omega\) is finitely-consistent. Consider any \(\Gamma, \Delta \subseteq \Gamma^\omega; \Delta^\omega\), we show that \(\mathcal{G}_i; \Gamma \vdash \Delta\) has no derivation. Since \(\mathcal{G}_i, \Gamma \subseteq \Gamma^\omega, \Delta \subseteq \Delta^\omega\), and where \(\mathcal{G}_i; \Gamma \vdash \Delta\) is as defined in Definition 3.6. 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 obvious properties of the construction of the sequents \(\mathcal{G}_i; \Gamma_i \vdash \Delta_i\). This can be proved by a simple induction on \(i\).

**Lemma 3.6.** For any \(i \in \mathbb{N}\), the following properties hold:
1. \(\mathcal{G}_i; \Gamma_i \vdash \Delta_i\) has no derivation
2. \(\mathcal{L}_i \subseteq \{a_0, a_1, \ldots, a_{2i-1}\}\)
3. \(\mathcal{G}_i; \Gamma_i \vdash \Delta_i \subseteq \mathcal{G}_{i+1}; \Gamma_{i+1} \vdash \Delta_{i+1}\)

We show the proofs of the other conditions in Definition 3.3 involving multiplicative connectives and structural rules are also satisfied by the limit sequent; the cases (1-5) and (15) for the additives are straightforward and are therefore omitted here.

6. If \(m : T^+ \in \Gamma^\omega\), then \(m : T^+ \in \Gamma_i\), for some \(i \in \mathbb{N}\) since each labelled formula from \(\Gamma^\omega\) must appear somewhere in the sequence. Then there exists \(j > i\) such that \(\phi(j) = (0, m, T^+, R)\) where this formula becomes principal. By construction \((\epsilon, m \in \epsilon) \in \mathcal{G}_j \subseteq \Gamma^\omega\). Then \(\Gamma^\omega \vdash \epsilon \vdash \epsilon\) because \(\mathcal{G}_j \vdash \epsilon \vdash \epsilon\) (for \(m = \epsilon\)). So \(\epsilon = \epsilon\).

8. If \(m : F_1 \ast F_2 \in \Gamma^\omega\), then it is in some \(\Gamma_i\), where \(i \in \mathbb{N}\). Then there exists \(j > i\) such that \(\phi(j) = (0, m, F_1 \ast F_2, R)\).
By construction $G_{j+1} = G_j \cup \{(a_{2j}, a_{2j+1} \bowtie m)\} \subseteq G^ω$, and $Γ_{j+1} = Γ_j \cup \{(a_{2j}, F_1, a_{2j+1} : F_2)\} \subseteq Γ^ω$.

9. If $m : F_1 \ast F_2 \in Δ^ω$, then it is in some $Δ_i$, where $i \in N$. For any $(x, y \bowtie m') \in G^ω$ such that $G^ω \vdash_{IE} (m = m')$, there exists $j > i$ such that $(x, y \bowtie m') \in G_j$ and $G_j \vdash_{IE} (m = m')$. Also, there exists $j > i$ such that $\phi(k) = (1, m, F_1, F_2, \{(x, y \bowtie m')\})$ where the labelled formula becomes principal. Since $(x, y \bowtie m') \in G_k$ and $G_k \vdash (m = m')$, we have either $x : F_1 \in Δ^ω$ or $y : F_2 \in Δ^ω$, similar to case 8.

10. If $m : F_1 \ast F_2 \in Δ^ω$, similar to case 9.

11. For each $a_n \in L$, there is a $j > n$ such that $\phi(j) = (O, m, U, \{(a_n, a_n)\})$ where $U$ is applied to $a_n$. Then $G_{j+1} = G_j \cup \{(a_n, a_n)\} \subseteq G^ω$, because $n \leq 2j + 1$.

12. If $(x, y \bowtie z) \in G^ω$, then it is in some $G_i$, where $i \in N$. Then there is a $j > i$ such that $\phi(j) = (O, m, U, \{(x, y \bowtie z)\})$ where $E$ is applied. Then $G_{j+1} = G_j \cup \{(y, x \bowtie z)\} \subseteq G^ω$.

13. If $(x, y \bowtie z) \in G^ω$, then it is in some $G_i$, where $i \in N$ such that $(x, y \bowtie z) \in G_i$, and $G_j \vdash (x = x')$. There are two cases to consider, depending on whether $(x, y \bowtie z)$ and $(u, v \bowtie x')$ are the same relational atoms. Suppose they are distinct. Then there must be some $j > i$ such that $\phi(j) = (O, m, U, \{(x, y \bowtie z), (u, v \bowtie x')\})$. Then $(x, y \bowtie z), (u, v \bowtie x') \in G_j$, and $G_j \vdash (x = x')$. By construction we obtain that $G_{j+1} = G_j \cup \{(a_{2j}, a_{2j+1} \bowtie z), (u, v \bowtie a_{2j+1})\} \subseteq G^ω$. If $(x, y \bowtie z)$ and $(u, v \bowtie x')$ are the same relational atom, then a similar argument can be applied, but in this case the schedule is choose one which selects $\mathcal{A}_C$ rather than $\mathcal{A}_H$.

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

**Proof.** We construct a limit sequent $G^ω; Γ^ω \vdash Δ^ω$ for $F$ following Definition 3.6. By the construction of the limit sequent, we have $a_1 : F \in Δ^ω$. By Lemma 3.7, this limit sequent is a Hintikka sequent, and therefore by Lemma 3.4, $G^ω; Γ^ω \vdash Δ^ω$ is falsifiable. This means there exists a model $(F, ν, ρ)$ that satisfies $G^ω$ and $Γ^ω$ and falsifies every element of $Δ^ω$, including $a_1 : F$, which means that $F$ is false at world $ρ(a_1)$. Thus $F$ is not valid.

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.

**Indivisible unit.** The unit $e$ in a commutative monoid $(H, \bowtie, e)$ is indivisible if the following holds for any $h_1, h_2 : H$:

\[ \forall h_1, h_2 : H. \text{if } h_1 \bowtie h_2 = e \text{ then } h_1 = e. \]

Relationally, this corresponds to the first-order condition:

\[ \forall h_1, h_2 : H. \text{if } R(h_1, h_2, e) \text{ then } h_1 = e. \]  

Note that this also means that $h_2 = e$ whenever $h_1 \bowtie h_2 = e$. Most memory models in the literature have indivisible unit [5], so this property seems appropriate for reasoning about concrete applications of separation logic. Indivisible unit can be axiomatised by the following formula [6]:

\[ T^+ \wedge (A \ast B) \rightarrow A \]

We use the following sound rule to capture this property:

\[ (e, y \bowtie y); G^ω[x/z]; Γ^ω[x/z] \vdash Δ^ω/x \]

Note that we can then instantiate the label $y$ to $e$ by applying $Eq_1$, upwards. Recall that the sequent calculus $LS_{BB1}$ [16] is just the sequent calculus $LS_{PASL}$ minus the rules $C$ and $P$.

**Proposition 4.1.** The formula $T^+ \wedge (A \ast B) \rightarrow A$ is provable in $LS_{BB1} + IU$. The proof is as shown in Figure 3 (a).

**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.

**Proof.** Soundness is straightforward as the rule $IU$ essentially just encodes the first-order formula (2) into the labelled sequent calculus. Completeness can be proved via the same counter-model construction for $LS_{PASL}$ (Theorem 3.8). That is, we first define an intermediate calculus $LS_{PASL} + IU$ that is equivalent to $LS_{PASL} + IU$, and do counter-model construction in $LS_{PASL} + IU$. Since the $IU$ rule contains substitution, the rule will be localised into the entailment relation $\vdash_{IE}$, so the definition of $\vdash_{IE}$ in Definition 3.1 is modified to allowed $IU$ in addition to $Eq_1, Eq_2, P$ and $C$. Thus the rules of $LS_{PASL} + IU$ are exactly the same as $LS_{PASL}$, and the only change is in the definition of $\vdash_{IE}$. The equivalence between $LS_{PASL} + IU$ and $LS_{PASL} + IU$ can be proved as in Lemma 3.3.

We then show that a Hintikka sequent yields a Kripke relational frame that corresponds to a separation algebra with indivisible unit. No additional clauses are needed in the definition of Hintikka sequence since it is parametric on the entailment relation $\vdash_{IE}$.

For a Hintikka sequent $G; Γ \vdash Δ$, suppose $(H, V \otimes [\cdot])$ is the PASL Kripke relational frame generated by $G$. Given any $[a], [b] \not\in [\cdot]$, we can find a $(a', b' \bowtie c') \in G$ such that $[a] = [a']$, $[b] = [b']$, $[c'] = [c]$. Also, we can use the rule $IU$ to derive $G \vdash_{IE} (a' = e)$. Thus by Lemma 3.1, we obtain $[a] = [a'] = [e]$. So the structure $(H, V \otimes [\cdot])$ generated by $G$ is indeed a PASL Kripke relational frame that obeys indivisible unit.

The saturation with logical and structural rules $E, U, A, AC$ is then the same as in Sec. 3.

**Disjointness.** The separating conjunction $\ast$ in separation logic requires that the two combined heaps have disjoint domains [29]. In a separation algebra $(H, \bowtie, e)$, disjointness is defined by the following additional requirement:

\[ \forall h_1, h_2 : H. \text{if } h_1 \bowtie h_2 = e \text{ then } h_1 = e \]

Relationally, this corresponds to the first-order condition:

\[ \forall h_1, h_2 : H. \text{if } R(h_1, h_2, e) \text{ then } h_1 = e \]

The disjointness condition is captured in labelled sequent calculus by the following rule, where $x, y$ are labels:

\[ (e, y \bowtie y); G^ω[x/z]; Γ^ω[x/z] \vdash Δ^ω/x \]

In fact disjointness implies indivisible unit (but not vice versa), as shown by Dockins et al. [11]. Thus we can prove the axiom for indivisible unit by using $LS_{BB1} + D$.

**Proposition 4.3.** The formula $T^+ \wedge (A \ast B) \rightarrow A$ is provable in $LS_{BB1} + D$. Figure 3(b) shows the derivation of the formula; we highlight the principal relational atoms where they are not obvious.

**Theorem 4.4.** $LS_{PASL} + D$ is sound and cut-free complete with respect to the class of 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 monoid \((H, \circ, e)\) has splittability if for every \(h_0 \in H \setminus \{e\}\) there are \(h_1, h_2 \in H \setminus \{e\}\) such that \(h_1 \circ h_2 = h_0\). Relationally, this corresponds to: if \(h_0 \neq e\) then there exist \(h_1 \neq e, h_2 \neq e\) such that \(R(h_1, h_2, h_0)\). This property can be axiomatised as the BBI formula \(\neg T^* \rightarrow (\neg T^* \circ \neg T^*)\) [6]. We give the following rule for splittability, where \(x, y\) are fresh:

\[
\frac{(x, y; z; \Gamma; x : T^*; y : T^*; z : T^* \Delta)}{\Gamma ; \Gamma \vdash z : T^* \Delta} \quad S
\]

**Proposition 4.5.** The rule \(S\) for splittability is sound.

**Proof.** Assume the conclusion of \(S\) is falsifiable in an extended relational model \((H, R, e, v, \rho)\); we show that the premise is also falsifiable. So suppose that everything in \(\Gamma \cup \Gamma\) is true, and everything in \(\{z : T^* \cup \Delta\} \) is false in the model. Specifically, \(z : T^* \) is false, meaning \(\rho(z) \nmid T^*\), thus \(\rho(z) \neq e\). By splittability, there exist some \(h_1, h_2 \in H \setminus \{e\}\) such that \(R(h_1, h_2, h_0)\) holds. Let \(\rho' = \rho \cup \{x \mapsto h_1, y \mapsto h_2\}\). Then \((x, y; z)\) is true under \(\rho'\). Also, since \(\rho'(x) \neq e\) and \(\rho'(y) \neq e\), the labelled formulae \(x : T^*\) and \(y : T^*\) are false under \(\rho'\). Thus the model with frame \((H, R_\rho, e, v, \rho)\), valuation \(v\), and label mapping \(\rho'\) falsifies the premise. \(\square\)

**Proposition 4.6.** The axiom \(\neg T^* \rightarrow (\neg T^* \circ \neg T^*)\) for splittability is provable in \(L_{S_{BB}L_1} + S\).

**Proof.** We start from \(\neg T^* \rightarrow (\neg T^* \circ \neg T^*)\), and obtain the following derivation backward:

\[
\frac{(a_1, a_2 \circ a_0); \vdash a_1 : T^*; a_2 : T^*; a_0 : T^* \circ \neg T^*}{\vdash a_0 : T^* \circ a_0 : \neg T^* \circ \neg T^* \circ \neg T^* \circ \neg T^*} \quad S
\]

Then we apply \(\circ R\) (backwards) on the top sequent using the relational atom \((a_1, a_2 \circ a_0)\) to obtain the premises \(S_L\) and \(S_R\).

The right premise \(S_R\) can be proved similarly.

Note that the rule \(S\) creates new labels in the premise, as the rule \(A\) does. However, it also has a principal formula, so it is not a structural rule. Unlike the rules \(IU\) and \(D\), the rule \(S\) cannot simply be localised into the entailment relation \(\vdash \), so we have to extend the notion of a Hintikka sequent to add a condition corresponding to splittability. The counter-model construction then has to ensure that this condition is satisfied by the limit sequent by ensuring that the rule \(S\) is applied in a fair way. We therefore leave the details of this completeness proof for \(L_{S_{BB}L_1} + S\) to future work.

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 monoid \((H, \circ, e),\) 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 \(h_1 \circ h_4 = h_{14}, h_2 \circ h_3 = h_{23}, h_{14} \circ h_{23} = h_3\), and \(h_{14} \circ h_{23}\) = \(h_4\). 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:

\[
\frac{R; (x, y; z); (u, v; z); \Gamma; \Gamma \vdash \Delta}{(x, y; z); (u, v; z); \Gamma; \Gamma \vdash \Delta} \quad \text{C.S}
\]

where \(R := (p, q; x; z); (s, t; u; v)\) and \(p, q, s, t\) do not occur in the conclusion

We conjecture that this rule can be handled in the counter-model construction by treating it similarly to the rule \(A\). We then need to change the indexing of labels when constructing the limit sequent. Let the rule \(A\) choose \(a_0\), the rules \(\ast L\) and \(\ast R\) choose \(a_4, a_{4+1}\), and the rule \(CS\) choose \(a_{4i}, a_{4i+1}, a_{4i+2}, a_{4i+3}\) as new labels. Further, let the rule \(U\) create the identity relational atom \((a_{4i}, e \circ a_{4i})\) only when \(n \leq 4i + 3\). Then each \(L_i\) is a subset of \(\{a_{4i}, \ldots, a_{4i+1}\}\).

Splittability and cross-split are not as frequently used in separation logic as the other conditions, and they require non-trivial modifications in our current proofs. 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. For these reasons we omit these from our modular framework of proof systems for PASL.
5. Tailoring the labelled calculus

We now consider various labelled calculi obtained by extending \( \text{LS}_{BBI} \) 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 [6, 11], it is not surprising that our structural rules can also be added modularly to \( \text{LS}_{BBI} \), 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 \( \text{LS}_{PASL} \) gives an interesting system. The proofs in Sec. 3 still work if we just insist on a partial commutative monoid, and drop \( C \) in \( +_E \).

**Theorem 5.1.** The labelled sequent calculus \( \text{LS}_{BBI} + P \) is sound and cut-free complete with respect to the partial commutative monoidal semantics for \( \text{BBI} \).

As a result, it is easy to obtain the following sound and complete labelled calculus for the corresponding semantics: \( \text{LS}_{BBI} + P + IU \) and \( \text{LS}_{BBI} + P + D \). The proofs are similar to that for Theorem 4.2.

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

**Theorem 5.2.** The labelled sequent calculus \( \text{LS}_{BBI} + C \) is sound and cut-free complete with respect to the partial commutative monoidal semantics for \( \text{BBI} \).

Again, using a similar argument as in Theorem 4.2, we can obtain sound and complete labelled calculi \( \text{LS}_{BBI} + C + IU \) and \( \text{LS}_{BBI} + C + D \).

**Calculi without partial-determinism and cancellativity.** The labelled calculus \( \text{LS}_{BBI} + IU \) is sound and complete by Prop. 4.1, and cut-elimination holds.

**Theorem 5.3.** The labelled sequent calculus \( \text{LS}_{BBI} + IU \) is sound and cut-free complete with respect to the commutative monoidal semantics for \( \text{BBI} \) with indivisible unit.

To prove the completeness of the calculus \( \text{LS}_{BBI} + D \), 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 \( +_E \) by using \( E_Q, E_Q, D \) 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 \( \text{LS}_{BBI} + D \) is sound and cut-free complete with respect to the commutative monoidal semantics for \( \text{BBI} \) with disjointness.

To summarise, our approach offers a sound and cut-free calculus for the extension of \( \text{BBI} \) with every combination of the properties \( P, C, IU, D \). The case where none of the properties hold, i.e. regular \( \text{BBI} \), 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:

\[
\begin{align*}
\text{LS}_{BBI} + IU & \quad \text{LS}_{BBI} + C \\
\text{LS}_{BBI} + P & \quad \text{LS}_{BBI} + D \\
\text{LS}_{BBI} + P + IU & \quad \text{LS}_{BBI} + C + IU \\
\text{LS}_{BBI} + P + D & \quad \text{LS}_{BBI} + C + D \\
\end{align*}
\]

\[
\begin{align*}
\text{LS}_{PASL} = \text{LS}_{BBI} + P + C & \quad \text{LS}_{BBI} + P + C \quad \text{LS}_{PASL} + IU \\
\text{LS}_{PASL} + D & \quad \text{LS}_{PASL} + D
\end{align*}
\]

6. Implementation and experiment

We discuss here an implementation of the proof system \( \text{LS}_{PASL} + D \). It turns out that the \( A_C \) rule is admissible in this system; in fact it is admissible in the subsystem \( \text{LS}_{BBI} + C \), as shown next, so we do not implement the \( A_C \) rule.

**Proposition 6.1.** The \( A_C \) rule is admissible in \( \text{LS}_{BBI} + C \).

**Proof.** We show that every derivation in \( \text{LS}_{BBI} + C \) 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 \( \text{LS}_{BBI} + C \) of the form:

\[
\Pi \quad \frac{\Delta}{\Gamma \vdash \Delta}
\]

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

\[
\Pi' \quad \frac{(x, \epsilon \triangleright x); (\epsilon, \epsilon \triangleright \epsilon); \epsilon[y/y]; \Gamma \vdash \Delta[e/y]}{(x, \epsilon \triangleright x); (x, y \triangleright x); \Gamma \vdash \Delta[e/y]}
\]

where \( \Pi' \) is obtained by applying the substitutions \([e/x]\) and \([e/w]\) to \( \Pi \) (Lemma 2.2). Note that since \( w \) does not occur in the root sequent, \( \Pi'[e/x] = \Pi[e/x] \) and \( \Pi'[e/w] = \Pi[e/w] \). These substitutions do not introduce new instances of \( A_C \).

Our implementation uses the following strategy when applying rules on a sequent:

1. Try to close the branch by rules \( id, \perp_L, \perp^*_R, \perp^*_R \).
2. If (1) not applicable, apply all possible \( E_Q, E_Q, P, C, IU, D \) rules to unify labels.
3. If (1-2) not applicable, apply invertible rules \( \land_L, \land_R, \rightarrow_L, \rightarrow_R, \star_L, \rightarrow_R, \perp^*_L \) in all possible ways.
4. If (1-3) not applicable, try rules \( \star_R \) or \( \rightarrow_L \) by choosing existing relational atoms.
5. If none of the existing relational atoms are applicable, or all combinations of \( \star_R \rightarrow L \) formulae and relational atoms are already applied in (4), apply structural rules on the set \( G_0 \) of relational atoms in the sequent as follows:
   a. Use \( E \) to generate all commutative variants of existing relational atoms in \( G_0 \), giving a set \( G_1 \).
   b. Apply \( A \) for each applicable pair in \( G_1 \), generating a set \( G_2 \).
   c. Use \( U \) to generate all identity relational atoms for each label in \( G_2 \), giving the set \( G_3 \).
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 \( G_1 = G_0 \). It is also clear that step (5) is terminating. We forbid applications of the rule \( A \) to the pair \( (x, y \triangleright z), (u, v \triangleright x) \) when \( (u, w \triangleright z), (y, v \triangleright w) \), for some label \( w \); (or any commutative variants of this pair, e.g., \( (w, w \triangleright z), (v, y \triangleright w) \)) 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 \( \Gamma, \Delta \) in a sequent \( \Gamma \vdash \Delta \) 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

\footnote{Although \( IU \) is admissible, we keep it because it simplifies proof search.}
always tries to decompose the subformulae of a principal formula if possible. To guarantee completeness, each time we apply a \( * \) \( R \) or \( \rightarrow \) \( L \) rule, the principal formula is moved to the end of the list, so that each principal formula for non-determinism rules \( * \) \( R \), \( \rightarrow \) \( L \) 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 \( \rightarrow \) \( L \) 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 BBeye has shown that associativity of \( * \) is a source of inefficiency in proof search [16, 27]. 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 BBeye, BBeye [27] and the incomplete heuristic-based \( FVL SBBI \) [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 \( FVL SBBI \). 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 \( FVL SBBI \) with heuristics to terminate within the timeout but without finding a proof due to its incompleteness. Formula (18) is valid only when the monoid is partial [22], and formula (19) is the axiom of indi-visible 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 \( \rightarrow \), which allows direct reference to the contents of the heap. \( E \mapsto E' \) is satisfied by a heap if 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 \( e, e_1, e_2, \ldots \) only, as is done for example by Berdine et al [3]. The rules for quantifiers are straightforward, e.g.: 

\[
\frac{G; \Gamma \vdash h : F[e/x] \rightarrow \Delta}{G; \Gamma \vdash h : \exists x, F : \Delta} \quad \text{(3)} \\
\frac{G; \Gamma \vdash h : F[e/x] \rightarrow \Delta}{\exists \Delta L} \\
\frac{G; \Gamma \vdash h : F[e/x] \rightarrow \Delta}{\exists \Delta R}
\]

where \( e \) does not appear free in the conclusion of \( \exists \Delta L \).

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

\[
\frac{G; \Gamma \vdash [e_2/e_1] \rightarrow \Delta}{G; \Gamma \vdash h : e_1 = e_2 \rightarrow \Delta} \quad \text{(4)} \\
\frac{G; \Gamma \vdash h : e = e_1 \rightarrow \Delta}{\exists \Delta L} \\
\frac{G; \Gamma \vdash h : e = e_2 \rightarrow \Delta}{\exists \Delta R}
\]

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 \( e_1 \rightarrow e_2 \) 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 \( e \), and cannot be split into two non-\( e \) worlds. This motivates rules \( \rightarrow L_1 \) and \( \rightarrow L_2 \). The rules \( \rightarrow L_3 \) and \( \rightarrow L_4 \) address the content of heaps: \( \rightarrow L_3 \) says that two heaps with the same address (value of \( e_1 \)) must be the same heap, and \( \rightarrow L_4 \) says that a singleton heap makes a unique assignment.

Our implementation of the calculus defined by adding these rules to \( LSP_{ASL} + D \) 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:

\[
T^* \rightarrow \neg((e_1 \rightarrow e_2) \rightarrow \neg(e_1 \rightarrow e_2)) \quad (5)
\]

This formula essentially asserts that a heap satisfying \( e_1 \rightarrow e_2 \) 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]:

\[
\frac{(e, e \triangleright a_0); e : (e_1 \rightarrow e_2) \vdash a_0 : \bot}{\rightarrow L_3} \\
\frac{(a_1, a_1 \triangleright a_0); a_1 : (e_1 \rightarrow e_2) \vdash a_0 : \bot}{\rightarrow L_3} \\
\frac{(a_1, a_2 \triangleright a_0); a_1 : (e_1 \rightarrow e_2); a_2 : (e_1 \rightarrow e_2) \vdash a_0 : \bot}{\rightarrow L_3} \\
\frac{; a_0 : (e_1 \rightarrow e_2) \neq (e_1 \rightarrow e_2) \vdash a_0 : \bot}{\rightarrow L_3} \\
\frac{; a_0 : (e_1 \rightarrow e_2) \neq (e_1 \rightarrow e_2) \vdash a_0 : \bot}{\rightarrow L_3}
\]

Our experimental prover Separata+, extending \( LSP_{ASL} + D \) 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 Galmiche 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, \( \neg(A \rightarrow \neg B) \) [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 [8] 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 \( \rightarrow \) 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 \( \rightarrow \) identified by Calcagno et al [7], 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 \( \rightarrow \) is SmallfootRG [9], which supports automation yet includes septraction [32], the De Morgan dual of \( \rightarrow \). However SmallfootRG does not support additive negation nor implication, and so \( \rightarrow \) 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 \( \rightarrow \), has found application. A non-exhaustive list follows: generating weakest preconditions via backwards reasoning [17]; specifying iterators [15, 19, 28]; 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 \( \rightarrow \) (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 Hôu et al [16]. Their prover \( FV/LSP_{BBI} \) implements a free-variable calculus for BBI but is incomplete. Our extensions to Hôu 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 calculi for Reynolds’ heap model. Their calculus, like ours, cannot prove the formula (5)\(^4\) 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 [20, 21], 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.

\(^4\) Confirmed by private communications with authors.
\[
\frac{\varrho; \Gamma; \epsilon : e_1 \Rightarrow e_2 \vdash \Delta}{\varrho; \Gamma; \epsilon : e_1 \Rightarrow e_2 \vdash \Delta}
\]

\[
\frac{\varrho[h/h']; \Gamma[h/h']; \epsilon : e_1 \Rightarrow e_2; h : e_1 \Rightarrow e_3 \vdash \Delta[h/h']}{\varrho[h/h']; \Gamma[h/h']; \epsilon : e_1 \Rightarrow e_2 \vdash \Delta}\]

\[
\frac{\varrho[h/h']; \Gamma[h/h']; \epsilon : e_1 \Rightarrow e_2; h : e_1 \Rightarrow e_3 \vdash \Delta[h/h']}{\varrho[h/h']; \Gamma[h/h']; \epsilon : e_1 \Rightarrow e_2 \vdash \Delta}
\]

\[
\frac{\varrho[h/h']; \Gamma[h/h']; \epsilon : e_1 \Rightarrow e_2; h : e_1 \Rightarrow e_3 \vdash \Delta[h/h']}{\varrho[h/h']; \Gamma[h/h']; \epsilon : e_1 \Rightarrow e_2 \vdash \Delta}
\]

\[
\frac{\varrho[h/h']; \Gamma[h/h']; \epsilon : e_1 \Rightarrow e_2; h : e_1 \Rightarrow e_3 \vdash \Delta[h/h']}{\varrho[h/h']; \Gamma[h/h']; \epsilon : e_1 \Rightarrow e_2 \vdash \Delta}
\]

Figure 4. Some rules for the predicate \( \Rightarrow \) in separation logic.

<table>
<thead>
<tr>
<th>Formula</th>
<th>Separata+</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) ( ((\epsilon_1 \Rightarrow \epsilon_2) * (\epsilon_1 \Rightarrow \epsilon_2)) \Rightarrow \bot )</td>
<td>0.004</td>
</tr>
<tr>
<td>(2) ( (\exists x : x.1.(x \Rightarrow x2) * (x \Rightarrow x1)) \Rightarrow (x \Rightarrow x2) * (x \Rightarrow x1) )</td>
<td>0.002</td>
</tr>
<tr>
<td>(3) ( \neg((\epsilon_1 \Rightarrow \epsilon_2) \Rightarrow \neg(\epsilon_3 \Rightarrow \epsilon_4)) \Rightarrow (\epsilon_1 = \epsilon_3) \land ((\epsilon_2 = \epsilon_4) \land \top^*) )</td>
<td>0.001</td>
</tr>
<tr>
<td>(4) ( \neg(((\epsilon_1 \Rightarrow \epsilon_2) \Rightarrow \neg(\epsilon_3 \Rightarrow \epsilon_4)) \Rightarrow (\epsilon_1 = \epsilon_3) \land ((\epsilon_2 = \epsilon_4) \land \top^*)) )</td>
<td>0.004</td>
</tr>
<tr>
<td>(5) ( \neg(((\epsilon_1 \Rightarrow p) * (\epsilon_2 \Rightarrow q)) \Rightarrow \neg(\epsilon_3 \Rightarrow r)) \Rightarrow \neg(((\epsilon_1 \Rightarrow p) \Rightarrow \neg((\epsilon_2 \Rightarrow q) \Rightarrow \neg(\epsilon_3 \Rightarrow r)))) )</td>
<td>0.002</td>
</tr>
<tr>
<td>(6) ( \neg(((\epsilon_1 \Rightarrow p) \Rightarrow \neg(\epsilon_2 \Rightarrow q)) \Rightarrow \neg(((\epsilon_1 \Rightarrow p) \Rightarrow \neg((\epsilon_2 \Rightarrow q) \Rightarrow \neg(\epsilon_3 \Rightarrow r)))) )</td>
<td>0.003</td>
</tr>
</tbody>
</table>

Table 3. Experimental results from the prover Separata+. 

References