Proof Search for Propositional Abstract Separation Logics via Labelled Sequents

Zhé Hóu\textsuperscript{1}, Ranald Clouston\textsuperscript{1}, Rajeev Goré\textsuperscript{1}, and Alwen Tiu\textsuperscript{2}

\textsuperscript{1} The Australian National University
\textsuperscript{2} Nanyang Technological University

January 22, 2014
Reasoning about programs that alter data structures

In C++, we can write: ...; \( y = 5; x = &y; *x = 10; \) ...

Programming in this style can be very confusing, verification of those programs is also hard. Separation logic [Reynolds 2002] provides a way to reason about such programs:

Stores: \( \text{Var} \rightarrow \text{Value} \)  
Heaps: \( \text{Addr} \rightarrow_{\text{fin}} \text{Value} \)

Pointers can be expressed explicitly: define \( h \in \text{Heaps}, s \in \text{Stores} \) as \( s(x) = &y, s(y) = 5, (s, h) \models x \mapsto y \)

Two basic operations \( \ast \) and \( \rightarrow \ast \) allow us to:
- split a heap: \( (x \mapsto y) \ast (x' \mapsto y') \)
- extend a heap: \( (w \mapsto z) \rightarrow \ast P \)

Express more complex things, e.g.,
\[ (((1 \mapsto 5) \ast (2 \mapsto 6)) \rightarrow \ast \bot) \rightarrow ((1 \mapsto 5) \ast ((2 \mapsto 6) \rightarrow \bot)) \]
Separation logic

There have been so many separation logics that it has become a “recipe” [Jensen 2013]:

Choose a programming language
Design an assertion logic
Design a specification logic

People then sought abstractions: e.g., [Calcagno, O’Hearn, Yang 2007] defined separation algebra as a cancellative, partial, commutative monoid.

But now there are many variations on this theme of separation algebra...
Separation algebras

Identity: \( \forall a \in M, a \circ \epsilon = a. \)
Commutativity: \( \forall a, b \in M, a \circ b = b \circ a. \)
Associativity: \( \forall a, b, c \in M, a \circ (b \circ c) = (a \circ b) \circ c. \)
Partial-determinism: \( \forall a, b, c, d \in M, c, d \in a \circ b \Rightarrow c = d. \)
Total-determinism: \( \forall a, b \in M, \exists c \in M \text{ s.t. } a \circ b = c. \)
Cancellativity: \( \forall a, b, c \in M, a \circ b = a \circ c \Rightarrow b = c. \)
Indivisible unit: \( \forall a, b \in M, a \circ b = \epsilon \Rightarrow a = \epsilon. \)
Disjointness: \( \forall a, b \in M, a \circ a = b \Rightarrow a = \epsilon. \)
Splittability: \( \forall a \neq \epsilon \in M, \exists b \neq \epsilon, c \neq \epsilon \in M \text{ s.t. } b \circ c = a. \)
Cross-split: \( \forall a, b, c, d \in M, a \circ b = c \circ d \Rightarrow \exists u, v, x, y \in M \text{ s.t. } u \circ v = a, x \circ y = b, u \circ x = c, \text{ and } v \circ y = d. \)
Propositional Abstract Separation Logics

Our base logic PASL is defined based on Calcagno et al.’s separation algebra, with the syntax from Boolean BI.

Concrete formula:
\[((1 \mapsto \to 5) \ast (2 \mapsto \to 6)) \ast \bot \to (1 \mapsto \to 5) \ast ((2 \mapsto \to 6) \ast \bot)\]

Abstract formula:
\[((p \ast q) \ast \bot) \to (p \ast (q \ast \bot))\]

By adding and/or removing certain conditions for the monoid, we obtain a framework of variants of PASL, thus the plural in the title.

The following problems have been open since 2007:

How to determine whether a formula is valid in PASL?
What if the formula is not valid?
How about the validity problem for a variant of PASL?
Can we do all this in a modular way?
Our answers, on the top level...

How to determine whether a formula is valid in PASL?
We give a sequent-style proof system for PASL, called $LSPASL$. We can easily obtain the following results for this system:

- **Soundness**: the system is correct wrt the semantics of PASL.
- **Cut-elimination**: (backward) proof search can be systematic.

What if the formula is not valid?
The difficulty: since PASL is not axiomatisable, cut-elimination doesn’t give us completeness. We prove the completeness by a quite complicated counter-model construction.

- **Completeness**: if a formula is not provable, we can find a counter-model.

How about the validity problem for a variant of PASL?
Can we do all this in a modular way?
Modularity: our proof system can easily be extended or tailored to handle many (not all) other variants of PASL.
Boolean BI

Syntax:

\[
F := p \mid \top \mid \bot \mid F \land F \mid F \rightarrow F \mid \top^* \mid F \ast F \mid F \rightarrow F
\]

Semantics:

A non-deterministic monoid \((M, \circ, \epsilon)\): a set \(M\), binary operator \(\circ\), unit \(\epsilon\).

\(m \vDash P\) iff \(P \in \text{Var}\) and \(m \in v(P)\)
\(m \vDash \bot\) iff never
\(m \vDash \top\) iff always
\(m \vDash \neg A\) iff \(m \not\vDash A\)
\(m \vDash \top^*\) iff \(m = \epsilon\)
\(m \vDash A \ast B\) iff \(\exists a, b. (a \circ b = m \text{ and } a \vDash A \text{ and } b \vDash B)\)
\(m \vDash A \rightarrow B\) iff \(\forall a, b. ((m \circ a = b \text{ and } a \vDash A) \implies b \vDash B)\)
Conditions for $\circ$

Identity: $\forall a \in M, a \circ \varepsilon = a.$

Commutativity: $\forall a, b \in M, a \circ b = b \circ a.$

Associativity: $\forall a, b, c \in M, a \circ (b \circ c) = (a \circ b) \circ c.$
Conditions for $\circ$

Identity: $\forall a \in M, a \circ \epsilon = a$.

Commutativity: $\forall a, b \in M, a \circ b = b \circ a$.

Associativity: $\forall a, b, c \in M, a \circ (b \circ c) = (a \circ b) \circ c$.

Partial-determinism: $\forall a, b, c, d \in M, c, d \in a \circ b \Rightarrow c = d$. 
Conditions for $\circ$

Identity: $\forall a \in M, a \circ \epsilon = a$.

Commutativity: $\forall a, b \in M, a \circ b = b \circ a$.

Associativity: $\forall a, b, c \in M, a \circ (b \circ c) = (a \circ b) \circ c$.

Partial-determinism: $\forall a, b, c, d \in M, c, d \in a \circ b \Rightarrow c = d$.

Total-determinism: $\forall a, b \in M, \exists c \in M \text{ s.t. } a \circ b = c$. 

Conditions for $\circ$

Identity: $\forall a \in M, a \circ \epsilon = a$.

Commutativity: $\forall a, b \in M, a \circ b = b \circ a$.

Associativity: $\forall a, b, c \in M, a \circ (b \circ c) = (a \circ b) \circ c$.

Partial-determinism: $\forall a, b, c, d \in M, c, d \in a \circ b \Rightarrow c = d$.

Total-determinism: $\forall a, b \in M, \exists c \in M$ s.t. $a \circ b = c$.

Cancellativity: $\forall a, b, c \in M, a \circ b = a \circ c \Rightarrow b = c$.

Indivisible unit: $\forall a, b \in M, a \circ b = \epsilon \Rightarrow a = \epsilon$.

Disjointness: $\forall a, b \in M, a \circ a = b \Rightarrow a = \epsilon$.

Splittability: $\forall a \neq \epsilon \in M, \exists b \neq \epsilon, c \neq \epsilon \in M$ s.t. $b \circ c = a$.

Cross-split: $\forall a, b, c, d \in M, a \circ b = c \circ d \Rightarrow \exists u, v, x, y \in M$ s.t. $u \circ v = a, x \circ y = b, u \circ x = c,$ and $v \circ y = d$. 
Attempt to refute a formula semantically

\(((p \ast q) \ast \bot) \rightarrow (p \ast (q \ast \bot))\)

Prove by contradiction, pick an arbitrary element \(h_0\) in the monoid.

\[\begin{array}{c}
\times (q \ast \bot) \\
\hline
h_2 \\
\hline
h_1 & h_0 \\
\checkmark p & \checkmark ((p \ast q) \ast \bot) & \times (p \ast (q \ast \bot))
\end{array}\]

\(m \models A \ast B\) iff \(\forall a, b. ((m \circ a = b \text{ and } a \models A) \implies b \models B)\)

\(h_0 \not\models p \ast (q \ast \bot)\) iff \(\exists h_1, h_2. ((h_0 \circ h_1 = h_2 \text{ and } h_1 \models p) \text{ and } h_2 \not\models (q \ast \bot))\)
Attempt to refute a formula semantically

\(((p \ast q)\ast \bot) \rightarrow (p \ast (q \ast \bot))\)

\[
\begin{align*}
\times (q \ast \bot) & \\
\times \bot & \\
h_4 & \\
\hline
h_3 & \checkmark q \\
\hline
h_1 & \checkmark p \\
\hline
h_0 & \checkmark ((p \ast q) \ast \bot) \\
& \times (p \ast (q \ast \bot))
\end{align*}
\]

\[m \models A \ast B \iff \forall a, b.((m \circ a = b \text{ and } a \models A) \implies b \models B)\]

\[h_2 \not\models q \ast \bot \iff \exists h_3, h_4.((h_2 \circ h_3 = h_4 \text{ and } h_3 \models q) \text{ and } h_4 \not\models \bot)\]
Attempt to refute a formula semantically

$$((p \ast q) \rightarrow \bot) \rightarrow (p \rightarrow (q \rightarrow \bot))$$

$m \models A \ast B$ iff $\exists a, b. (a \circ b = m$ and $a \models A$ and $b \models B$)

$h_5 \models p \ast q$ iff $\exists h_1, h_3. (h_1 \circ h_3 = h_5$ and $h_1 \models p$ and $h_3 \models q$)

Contradiction!
Why labelled sequents?

_Hilbert systems_ are not easy to automate (in fact, don’t exist for many separation algebras [Brotherston & Villard, 2013]).

_Display calculus_ [Brotherston 2009] is also hard to implement.

_Nested sequent calculus_ [Seo, Park, Park 2013] can be very complicated even for BBI only.

Seems that the proof theory is hard. But we can try to work on the semantics.

We used _labelled sequent calculus_ to handle BBI, and proposed rules for cancellativity and partial-determinism already [TABLEAUX 2013].

- Directly encodes the semantics.
- Easy to understand and use.
- Very extensible.
- Doesn’t give much insight on decidability, though...
The labelled sequent calculus $LS_{PASL}$

A labelled sequent is of the form $\mathcal{G}; \Gamma \vdash \Delta$, where $\mathcal{G}$ is a set of relational atoms, $\Gamma, \Delta$ are multisets of labelled formulae.

A relational atom $(x, y \triangleright z)$ encodes the relation $x \circ y = z$ in the semantics.

A labelled formula $x : F$ in $\Gamma$ encodes $x \not\vdash F$, and in $\Delta$ encodes $x \not\not\vdash F$ in the semantics.

$$LS_{PASL} = LS_{BBI} + P + C$$
Sample inference rules

\[
(\epsilon, w \triangleright \epsilon); \mathcal{G}; \Gamma \vdash \Delta \\
\mathcal{G}; \Gamma; w : \top^* \vdash \Delta \quad \text{\textsuperscript{T*}_L}
\]

\[
(\epsilon, w \triangleright w); \mathcal{G}[w/w']; \Gamma[w/w'] \vdash \Delta[w/w'] \\
(\epsilon, w' \triangleright w); \mathcal{G}; \Gamma \vdash \Delta \quad \text{Eq\textsubscript{1}}
\]
Sample inference rules

\[
(x, y △ z); \mathcal{G}; \Gamma; x : A \vdash z : B; Δ \quad \rightarrow^* R
\]
\[
\mathcal{G}; Γ \vdash y : A \rightarrow B; Δ
\]

\[
x, z \text{ do not occur in the conclusion.}
\]

\[
(x, y △ z); \mathcal{G}; \Gamma; y : A \rightarrow B \vdash x : A; Δ
go to line 10
\]
\[
(x, y △ z); \mathcal{G}; Γ; y : A \rightarrow B; z : B \vdash Δ \quad \rightarrow^* L
\]
\[
(x, y △ z); \mathcal{G}; Γ; y : A \rightarrow B \vdash Δ
\]
Identity: \( \forall a \in M, a \circ \epsilon = a. \)

\[
\frac{(a, \epsilon \triangleright a); \mathcal{G}; \Gamma \vdash \Delta}{\mathcal{G}; \Gamma \vdash \Delta} \quad \text{U}
\]

Commutativity: \( \forall a, b \in M, a \circ b = b \circ a. \)

\[
\frac{(b, a \triangleright c); (a, b \triangleright c); \mathcal{G}; \Gamma \vdash \Delta}{(a, b \triangleright c); \mathcal{G}; \Gamma \vdash \Delta} \quad \text{E}
\]
Sample inference rules

Partial-determinism: $\forall a, b, c, d \in M, c, d \in a \circ b \Rightarrow c = d$.

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

Cancellativity: $\forall a, b, c \in M, a \circ b = a \circ c \Rightarrow b = c$.

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

\[ \vdash h_1 : p ; \ldots \vdash h_3 : q \vdash h_3 : q ; \ldots \]

\[ (h_1 , h_3 \triangleright h_5) \vdash h_1 : p ; h_3 : q \vdash h_5 : p \ast q ; h_4 : \bot \]

\[ (h_1 , h_3 \triangleright h_5) ; (h_5 , h_0 \triangleright h_4) ; h_0 : (p \ast q) \dashv \bot \vdash h_1 : p ; h_3 : q \vdash h_4 : \bot \]

\[ (h_1 , h_0 \triangleright h_2) ; (h_3 , h_2 \triangleright h_4) ; h_0 : (p \ast q) \dashv \bot \vdash h_1 : p ; h_3 : q \vdash h_4 : \bot \]

\[ (h_1 , h_0 \triangleright h_2) ; h_0 : (p \ast q) \dashv \bot \vdash h_1 : p \vdash h_2 : q \dashv \bot \]

\[ ; h_0 : (p \ast q) \dashv \bot \vdash h_0 : p \dashv (q \dashv \bot) \]

\[ ; \vdash h_0 : ((p \ast q) \dashv \bot) \rightarrow (p \dashv (q \dashv \bot)) \rightarrow R \]
Counter-model construction

The rules with label substitutions induce an equivalence relation $\equiv_G$ on labels. We then define a saturated structure using $\equiv_G$.

Define a **Hintikka sequent** as a saturated structure which is also **consistent**. For example, it satisfies:

- It is not the case that $a : A \in \Gamma$, $b : A \in \Delta$ and $a \equiv_G b$.
- If $a : A \land B \in \Gamma$ then $a : A \in \Gamma$ and $a : B \in \Gamma$.

...
Every Hintikka sequent is falsifiable, thus can be used as a counter-model for the formulae in the succedent.

Start with $\vdash a_0 : F$, use a fair scheduler to apply each rule infinitely often, obtaining a limit sequent.

The limit sequent is a saturated structure, i.e., is a Hintikka sequent.

If $F$ is unprovable, we can find a counter-model in the semantics, so $F$ is invalid (completeness).

Inspired by (but different from) the completeness proof of Tableaux system for partial-BBI [Larchey-Wendling, 2013].
Extensions of PASL

Indivisible unit: if $a \circ b = \epsilon$, then $a = \epsilon$.

$$(\epsilon, b \triangleright \epsilon); G[\epsilon/a]; \Gamma[\epsilon/a] \vdash \Delta[\epsilon/a]$$

$$\frac{\rule{0pt}{1.8ex}}{(a, b \triangleright \epsilon); G; \Gamma \vdash \Delta} _{IU}$$

Disjointness: if $a \circ a = b$, then $a = \epsilon$.

$$(\epsilon, \epsilon \triangleright b); G[\epsilon/a]; \Gamma[\epsilon/a] \vdash \Delta[\epsilon/a]$$

$$\frac{\rule{0pt}{1.8ex}}{(a, a \triangleright b); G; \Gamma \vdash \Delta} _{D}$$
Summary

The main result: $LS_{BBI} + \text{any subset of } \{P, C, IU, D\}$ is sound and cut-free complete w.r.t. the corresponding abstract semantics.

A modular theorem prover is implemented based on the proof theory with some optimisations and heuristics. Available online: http://users.cecs.anu.edu.au/~zhehou/
Happy to do a demo in a coffee break if you wish!

Detailed and extended version of this paper:
Future work

Other properties in separation theories

Splittability/Divisibility: every non-unit element can be split into two non-unit elements. *Not true in Reynolds’ separation logic.*

Cross-split:

![Diagram](image)

Treatments for these two properties can be found in the extended version.
Future work (cont.)

Reynolds’ semantics

\[ \Gamma; \epsilon : e_1 \leftrightarrow e_2 \vdash \Delta \rightarrow L_1 \]

\((e_1 \leftrightarrow e_2)\) must be a singleton heap, so it can’t be empty.
Future work (cont.)

Reynolds’ semantics

\[(\epsilon, h_0 \triangleright h_0); G[\epsilon/h_1][h_0/h_2]; \Gamma[\epsilon/h_1][h_0/h_2]; h_0 : e_1 \mapsto e_2 \vdash \Delta[\epsilon/h_1][h_0/h_2] \]

\[(h_0, \epsilon \triangleright h_0); G[\epsilon/h_2][h_0/h_1]; \Gamma[\epsilon/h_2][h_0/h_1]; h_0 : e_1 \mapsto e_2 \vdash \Delta[\epsilon/h_2][h_0/h_1] \]

\[(h_1, h_2 \triangleright h_0); G; \Gamma; h_0 : e_1 \mapsto e_2 \vdash \Delta \]

\[(e_1 \mapsto e_2) \text{ must be a singleton heap, so it can’t be a composite heap.}\]
A singleton heap is a function mapping from exactly one address to one value.
The rule $\mapsto L_3$ is unsound. In Reynolds’ model, the set Heaps contains two heaps such as $(5 \mapsto 4)$ and $(5 \mapsto 6)$. 

\[
\frac{G[h/h']; \Gamma[h/h']; h : e_1 \mapsto e_2; h : e_1 \mapsto e_3 \vdash \Delta[h/h']}{} \quad \mapsto L_3
\]
Future work (cont.)

Reynolds’ semantics

\[
(h, h' \triangleright h''); \mathcal{G}; \Gamma; h : e_1 \mapsto e_2; h' : e_1 \mapsto e_3 \vdash \Delta \to L_3
\]

These rules are not enough...