BI Hyperdoctrines and Higher-Order Separation Logic

Bodil Biering, Lars Birkedal*, and Noah Torp-Smith*

Department of Theoretical Computer Science, IT University of Copenhagen, Denmark
{biering, birkedal, noah}@itu.dk

Abstract. We present a precise correspondence between separation logic and a new simple notion of *predicate* BI, extending the earlier correspondence given between part of separation logic and *propositional* BI [14]. Moreover, we introduce the notion of a BI hyperdoctrine and show that it soundly models classical and intuitionistic first- and higher-order predicate BI, and use it to show that we may easily extend separation logic to *higher-order*. We argue that the given correspondence may be of import for formalizations of separation logic.

1 Introduction

Separation logic [20, 19, 5, 6, 22, 9, 2] is a Hoare-style program logic, and variants of it have been applied to prove correct interesting pointer algorithms such as copying a dag, disposing a graph, the Schorr-Waite graph algorithm, and Cheney's copying garbage collector. Different extensions of core separation logic were employed to conduct these proofs. For example, Yang [21] extended the core logic with lists and trees, and in [2] the logic included finite sets and relations. Thus it is natural to ask whether one has to make a new extension of separation logic for every proof one wants to make [17]. This would be unfortunate for formal verification of proofs in separation logic since it would make the enterprise of formal verification burdensome and dubious. We argue that there is a natural single underlying logic in which it is possible to *define* the various extensions and prove the expected properties thereof; this is then the single logic that should be employed for formal verification.

Part of the pointer model of separation logic, namely that given by heaps (but not stacks), has been related to *propositional* BI, the logic of bunched implications introduced by O'Hearn and Pym [12]. In this paper we show how the correspondence may be extended to a precise correspondence between all of the pointer model (including stacks) and a simple notion of *predicate* BI. We introduce the notion of a *BI hyperdoctrine*, a simple extension of Lawvere's notion of hyperdoctrine [8], and show that it soundly models predicate BI. We consider a different notion of predicate BI than that of [15, 16], which has a BI structure on contexts. However, we believe that our notion of predicate BI with its class of BI hyperdoctrine models is the right one for separation logic (Pym aimed to model mulitiplicative quantifiers; separation logic only uses additive quantifiers).

* Partially supported by Danish Technical Research Council Grant 56-00-0309.

To make this point, we show that the pointer model of separation logic exactly corresponds to the interpretation of predicate BI in a simple BI hyperdoctrine. This correspondence also allows us to see that it is simple to extend separation logic to *higher-order* separation logic. We explain this extension and suggest that it may be useful for program proving.

Before proceeding with the technical development we give an intuitive justification of the use of BI hyperdoctrines to model higher-order predicate BI. A powerful way of obtaining models of BI is by means of functor categories (presheaves), using Day's construction to obtain a doubly-closed structure on the functor category [14]. Such functor categories can be used to model *propositional* BI in two different senses: In the first sense, one models *provability*, entailment between propositions, and it works because the lattice of subobjects of the terminal object in such functor categories form a BI algebra (a doubly cartesian closed preorder). In the second sense, one models *proofs*, and it works because the whole functor category is doubly cartesian closed. Here we seek models of provability of *predicate* BI. Since the considered functor categories are toposes and hence model higher-order predicate logic, one might think that a straightforward extension is possible. But, alas, it is not the case. In general, for this to work, *every* lattice of subobjects (for any object, not only for the terminal object) should be a BI algebra and, moreover, to model substitution correctly the BI algebra structure should be preserved by pulling back along any morphism. We show that this can only be the case if the BI algebra structure is trivial, that is, coincides with the cartesian structure (see Theorem 7). Our theorem holds for any topos, not just for the functor categories considered earlier. Hence we need to consider a wider class of models for predicate BI than just toposes and this justifies the notion of a BI hyperdoctrine. The intuitive reason that BI hyperdoctrines work, is that predicates are not required to be modeled by subobjects, they can be something more general. Another important point of BI hyperdoctrines is that they are easy to come by: given any complete BI algebra B, we can define a canonical BI hyperdoctrine in which predicates are modeled as B-valued functions; we explain this in detail in Example 6.

The remainder of this paper is organized as follows. In Section 2 we recall the notion of a (first-order) hyperdoctrine and explain how it soundly models predicate logic. We then define the concept of a (first-order) BI hyperdoctrine and explain how it soundly models predicate BI. In Section 3 we briefly recall the standard pointer model of separation logic and show how it can be construed as a first-order BI hyperdoctrine. In Section 4 we discuss some consequences for separation logic, and in particular, we use the higher-order logic to give logical characterizations of interesting classes of predicates. Finally, we conclude in Section 5.

2 BI Hyperdoctrines

In this section we introduce Lawvere's notion of a *hyperdoctrine* [8] and briefly recall how it can be used to model intuitionistic and classical first- and higher-

order predicate logic (see, for example, [13] and [7] for more explanations than can be included here). We then define the notion of a BI hyperdoctrine, which is a straightforward extension of the standard notion of hyperdoctrine, and explain how it can be used to model predicate BI logic.

Hyperdoctrines. A first-order hyperdoctrine is a categorical structure tailored to model first-order predicate logic with equality. The structure has a base category $\mathcal{C}$ for modeling the types and terms, and a $\mathcal{C}$-indexed category $\mathcal{P}$ for modeling formulas.

Definition 1 (First-order hyperdoctrines) *Let $\mathcal{C}$ be a category with finite products. A* first-order hyperdoctrine *$\mathcal{P}$ over $\mathcal{C}$ is a contravariant functor $\mathcal{P} : \mathcal{C}^{op} \to$ Poset from $\mathcal{C}$ into the category of partially ordered sets and monotone functions, with the following properties.*

1. *For each object X, the partially ordered set $\mathcal{P}(X)$ is a Heyting algebra.*
2. *For each morphism $f : X \to Y$ in $\mathcal{C}$, the monotone function $\mathcal{P}(f) : \mathcal{P}(Y) \to \mathcal{P}(X)$ is a Heyting algebra homomorphism.*
3. *For each diagonal morphism $\Delta_X : X \to X \times X$ in $\mathcal{C}$, the left adjoint to $\mathcal{P}(\Delta_X)$ at the top element $\top \in \mathcal{P}(X)$ exists. In other words, there is an element $=_X$ of $\mathcal{P}(X \times X)$ satisfying that for all $A \in \mathcal{P}(X \times X)$,*

$$\top \leq \mathcal{P}(\Delta_X)(A) \quad \textit{iff} \quad =_X \leq A.$$

4. *For each product projection $\pi : \Gamma \times X \to \Gamma$ in $\mathcal{C}$, the monotone function $\mathcal{P}(\pi) : \mathcal{P}(\Gamma) \to \mathcal{P}(\Gamma \times X)$ has both a left adjoint $(\exists X)_\Gamma$ and a right adjoint $(\forall X)_\Gamma$:*

$$A \leq \mathcal{P}(\pi)(A') \quad \textit{if and only if} \quad (\exists X)_\Gamma(A) \leq A'$$
$$\mathcal{P}(\pi)(A') \leq A \quad \textit{if and only if} \quad A' \leq (\forall X)_\Gamma(A).$$

Moreover, these adjoints are natural in Γ, i.e., given $s : \Gamma \to \Gamma'$ in $\mathcal{C}$, we have

$$\begin{array}{ccc} \mathcal{P}(\Gamma' \times X) & \xrightarrow{\mathcal{P}(s \times \mathrm{id}_X)} & \mathcal{P}(\Gamma \times X) \\ {\scriptstyle (\exists X)_{\Gamma'}}\downarrow & & \downarrow{\scriptstyle (\exists X)_\Gamma} \\ \mathcal{P}(\Gamma') & \xrightarrow[\mathcal{P}(s)]{} & \mathcal{P}(\Gamma) \end{array} \qquad \begin{array}{ccc} \mathcal{P}(\Gamma' \times X) & \xrightarrow{\mathcal{P}(s \times \mathrm{id}_X)} & \mathcal{P}(\Gamma \times X) \\ {\scriptstyle (\forall X)_{\Gamma'}}\downarrow & & \downarrow{\scriptstyle (\forall X)_\Gamma} \\ \mathcal{P}(\Gamma') & \xrightarrow[\mathcal{P}(s)]{} & \mathcal{P}(\Gamma). \end{array}$$

The elements of $\mathcal{P}(X)$, where X ranges over objects of $\mathcal{C}$, will be referred to as $\mathcal{P}$-predicates.

Interpretation of first-order logic in a first-order hyperdoctrine. Given a (first order) signature with types X, function symbols $f : X_1, \ldots, X_n \to X$, and relation symbols $R \subset X_1, \ldots, X_n$, a *structure* for the signature in a first-order hyperdoctrine $\mathcal{P}$ over $\mathcal{C}$ assigns an object $[\![X]\!]$ in $\mathcal{C}$ to each type, a morphism $[\![f]\!] : [\![X_1]\!] \times \cdots \times [\![X_n]\!] \to [\![X]\!]$ to each function symbol, and a $\mathcal{P}$-predicate

$[\![R]\!] \in \mathcal{P}([\![X_1]\!] \times \cdots \times [\![X_n]\!])$ to each relation symbol. Any term t over the signature, with free variables in $\Gamma = \{x_1 : X_1, \ldots, x_n : X_n\}$ and of type X say, is interpreted as a morphism $[\![t]\!] : [\![\Gamma]\!] \to [\![X]\!]$, where $[\![\Gamma]\!] = [\![X_1]\!] \times \cdots \times [\![X_n]\!]$, by induction on the structure of t (in the standard manner in which terms are interpreted in categories).

Each formula ϕ with free variables in Γ is interpreted as a $\mathcal{P}$-predicate $[\![\phi]\!] \in \mathcal{P}([\![\Gamma]\!])$ by induction on the structure of ϕ using the properties given in Definition 1. For atomic formulas $R(t_1, \ldots, t_n)$, the interpretation is given by $\mathcal{P}(\langle [\![t_1]\!], \ldots, [\![t_n]\!] \rangle)([\![R]\!])$. In particular, the atomic formula $t =_X t'$ is interpreted by the $\mathcal{P}$-predicate $\mathcal{P}(\langle [\![t]\!], [\![t']\!] \rangle)(=_{[\![X]\!]})$. The interpretation of other formulas is given by structural induction. Assume ϕ, ϕ' are formulas with free variables in Γ and that ψ is a formula with free variables in $\Gamma \cup \{x : X\}$. Then,

$$
\begin{array}{ll}
[\![\top]\!] = \top_H & [\![\phi \wedge \phi']\!] = [\![\phi]\!] \wedge_H [\![\phi']\!] \\
[\![\bot]\!] = \bot_H & [\![\phi \vee \phi']\!] = [\![\phi]\!] \vee_H [\![\phi']\!] \\
 & [\![\phi \to \phi']\!] = [\![\phi]\!] \to_H [\![\phi']\!]
\end{array}
$$

$$
\begin{array}{l}
[\![\forall x : X.\psi]\!] = (\forall [\![X]\!])_{[\![\Gamma]\!]}([\![\psi]\!]) \in \mathcal{P}([\![\Gamma]\!]) \\
[\![\exists x : X.\psi]\!] = (\exists [\![X]\!])_{[\![\Gamma]\!]}([\![\psi]\!]) \in \mathcal{P}([\![\Gamma]\!]),
\end{array}
$$

where $\wedge_H, \vee_H$, etc., is the Heyting algebra structure on $\mathcal{P}([\![\Gamma]\!])$.

We say that a formula ϕ with free variables in Γ is *satisfied* if $[\![\phi]\!]$ is the top element of $\mathcal{P}([\![\Gamma]\!])$. This notion of satisfaction is *sound* for intuitionistic predicate logic, in the sense that all provable formulas are satisfied. Moreover, it is also *complete* in the sense that a formula is provable if it is satisfied in all structures in first-order hyperdoctrines. A first-order hyperdoctrine $\mathcal{P}$ is sound for *classical* predicate logic in case all the fibres $\mathcal{P}(X)$ are Boolean algebras and all the reindexing functions $\mathcal{P}(f)$ are Boolean algebra homomorphisms.

Definition 2 (Hyperdoctrines) *A (general)* hyperdoctrine *is a first-order hyperdoctrine with the following additional properties: $\mathcal{C}$ is cartesian closed, and there is a Heyting algebra H and a natural bijection $\Theta_X : Obj(\mathcal{P}(X)) \simeq \mathcal{C}(X, H)$.*

A hyperdoctrine is sound for higher-order intuitionistic predicate logic: the Heyting algebra H is used to interpret the type, call it prop, of propositions and higher types (e.g., prop^X, the type for predicates over X), are interpreted by exponentials in $\mathcal{C}$. The natural bijection Θ_X is used to interpret substitution of formulas in formulas: Suppose ϕ is a formula with a free variable q of type prop and with remaining free variables in Γ, and that ψ is a formula with free variables in Γ. Then $[\![\psi]\!] \in \mathcal{P}([\![\Gamma]\!])$, $[\![\phi]\!] \in \mathcal{P}([\![\Gamma]\!] \times H)$, and $\phi[\psi/q]$ (ϕ with ψ substituted in for q) is interpreted by $\mathcal{P}(\langle \mathrm{id}, \Theta([\![\psi]\!]) \rangle)([\![\phi]\!])$. For more details see, e.g., [13].

Again it is the case that a hyperdoctrine $\mathcal{P}$ is sound for *classical* higher-order predicate logic in case all the fibres $\mathcal{P}(X)$ are Boolean algebras and all the reindexing functions $\mathcal{P}(f)$ are Boolean algebra homomorphisms.

Example 3 (Canonical hyperdoctrine over a topos) *Let $\mathcal{E}$ be a topos. It is well-known that $\mathcal{E}$ models higher-order predicate logic, by interpreting types as*

objects in $\mathcal{E}$, terms as morphisms in $\mathcal{E}$ and predicates as subobjects in $\mathcal{E}$. The topos $\mathcal{E}$ induces a canonical $\mathcal{E}$-indexed hyperdoctrine $\mathrm{Sub}_{\mathcal{E}} : \mathcal{E}^{op} \to \mathrm{Poset}$*, which maps an object X in $\mathcal{E}$ to the poset of subobjects of X in $\mathcal{E}$ and a morphisms $f : X \to Y$ to the pullback functor $f^* : \mathrm{Sub}(Y) \to \mathrm{Sub}(X)$. Then the standard interpretation of predicate logic in $\mathcal{E}$ coincides with the interpretation of predicate logic in the hyperdoctrine* $\mathrm{Sub}_{\mathcal{E}}$*. Compared to the standard interpretation in toposes, however, hyperdoctrines allow that predicates are not always modeled by subobjects but can come from some other universe. Thus hyperdoctrines describe a wider class of models than toposes do.*

BI Hyperdoctrines. Recall that a Heyting algebra is a bi-cartesian closed partial order, i.e., a partial order, which, when considered as a category, is cartesian closed ($\top$, $\wedge$, $\to$) and has finite coproducts ($\bot$, $\vee$). Further recall that a *BI algebra* is a Heyting algebra, which has an additional symmetric monoidal closed structure (I, $*$, $-\!\!*$) [15].

We now present a straightforward extension of (first-order) hyperdoctrines, which models first and higher-order predicate BI.

Definition 4 (BI Hyperdoctrines)

- *A first-order hyperdoctrine $\mathcal{P}$ over $\mathcal{C}$ is a* first-order BI hyperdoctrine *in case all the fibres $\mathcal{P}(X)$ are BI algebras and all the reindexing functions $\mathcal{P}(f)$ are BI algebra homomorphisms.*
- *A* BI hyperdoctrine *is a first-order BI hyperdoctrine with the additional properties that $\mathcal{C}$ is cartesian closed, and there is a BI algebra B and a natural bijection $\Theta_X : Obj(\mathcal{P}(X)) \simeq \mathcal{C}(X, B)$.*

First-order predicate BI is first-order predicate logic with equality, extended with formulas I, $\phi * \psi$, $\phi -\!\!* \psi$ satisfying the following rules (in any context Γ including the free variables of the formulas):

$$(\phi * \psi) * \theta \vdash_\Gamma \phi * (\psi * \theta) \qquad \phi * (\psi * \theta) \vdash_\Gamma (\phi * \psi) * \theta \qquad \vdash_\Gamma \phi \leftrightarrow \phi * \mathrm{I}$$

$$\phi * \psi \vdash_\Gamma \psi * \phi \qquad \frac{\phi \vdash_\Gamma \psi \quad \theta \vdash_\Gamma \omega}{\phi * \theta \vdash_\Gamma \psi * \omega} \qquad \frac{\phi * \psi \vdash_\Gamma \theta}{\phi \vdash_\Gamma \psi -\!\!* \theta}$$

Our notion of predicate BI should not be confused with the one presented in [15]; the latter seeks to also include a BI structure on contexts but we do not attempt to do that here, since that is not what is used in separation logic. In particular, weakening at the level of variables is always allowed:

$$\frac{\phi \vdash_\Gamma \psi}{\phi \vdash_{\Gamma \cup \{x:X\}} \psi}$$

We can interpret first-order predicate BI in a first-order BI hyperdoctrine simply by extending the interpretation of first-order logic in first-order hyperdoctrine given above by:

$$\begin{aligned} [\![\mathrm{I}]\!] &= \mathrm{I}_B \\ [\![\phi * \psi]\!] &= [\![\phi]\!] *_B [\![\psi]\!] \\ [\![\phi -\!\!* \psi]\!] &= [\![\phi]\!] -\!\!*_B [\![\psi]\!], \end{aligned}$$

where I_B, $*_B$ and $-\!\!*_B$ is the monoidal closed structure in the BI algebra $\mathcal{P}([\![\Gamma]\!])$. We then have:

Theorem 5 *The interpretation of first-order predicate BI given above is sound and complete.*

Likewise, BI hyperdoctrines form sound and complete models for higher-order predicate BI. Of course, a (first-order) BI hyperdoctrine is sound for classical BI in case all the fibres $\mathcal{P}(X)$ are Boolean BI algebras and all the reindexing functions $\mathcal{P}(f)$ are Boolean BI algebra homomorphisms. Here is a canonical example of a BI hyperdoctrine.

Example 6 (BI hyperdoctrine over a complete BI algebra) *Let B be a complete BI algebra, i.e., it has all joins and meets. It determines a BI hyperdoctrine over the category* **Set** *as follows. For each set X, let $\mathcal{P}(X) = B^X$, the set of functions from X to B, ordered pointwise. Given $f : X \to Y$, $\mathcal{P}(f) : B^Y \to B^X$ is the BI algebra homomorphism given by composition with f. For example if $s, t \in \mathcal{P}(Y)$, i.e., $s, t : Y \to B$, then $\mathcal{P}(f)(s) = s \circ f : X \to B$ and $s * t$ is defined pointwise as $(s * t)(y) = s(y) * t(y)$. Equality predicates $=_X$ in $B^{X \times X}$ are given by*

$$=_X (x, x') \stackrel{def}{=} \begin{cases} \top \text{ if } x = x' \\ \bot \text{ if } x \neq x' \end{cases}$$

where $\top$ and $\bot$ are the greatest and least elements of B, respectively. The quantifiers use set-indexed joins ($\bigvee$) and meets ($\bigwedge$). Specifically, given $A \in B^{\Gamma \times X}$ one has

$$(\exists X)_\Gamma(A) \stackrel{def}{=} \lambda i \in \Gamma. \bigvee_{x \in X} A(i, x) \qquad (\forall X)_\Gamma(A) \stackrel{def}{=} \lambda i \in \Gamma. \bigwedge_{x \in X} A(i, x)$$

in B^Γ. The conditions in Definition 2 are trivially satisfied (Θ is the identity).

There are plenty of examples of complete BI algebras: for any Grothendieck topos $\mathcal{E}$ with an additional symmetric monoidal closed structure, $\mathrm{Sub}_\mathcal{E}(1)$ is a complete BI algebra, and for any monoidal category $\mathcal{C}$ such that the monoid is cover preserving w.r.t. the Grothendieck topology J, $\mathrm{Sub}_{\mathrm{Sh}(\mathcal{C},J)}(1)$ is a complete BI algebra [1, 14].

The following theorem shows that to get interesting models of higher-order predicate BI, it does not suffice to consider BI hyperdoctrines arising as the canoncial hyperdoctrine over a topos (as in Example 3). Indeed this is the reason for introducing the more general BI hyperdoctrines. For reasons of space, we omit the proof in this exposition.

Theorem 7 *Let $\mathcal{E}$ be a topos and suppose* $\mathrm{Sub}_\mathcal{E} : \mathcal{E}^{op} \to$ Poset *is a BI hyperdoctrine. Then the BI structure on each lattice $\mathrm{Sub}_\mathcal{E}(X)$ is trivial, i.e., for all $\varphi, \psi \in \mathrm{Sub}_\mathcal{E}(X)$, $\varphi * \psi \leftrightarrow \varphi \wedge \psi$.*

3 Separation Logic modeled by BI-hyperdoctrines

We briefly recall the standard pointer model of separation logic (for a more thorough presentation see, for instance, [20]) and then show how it can be construed as a BI hyperdoctrine over **Set**.

The core assertion language of separation logic (which we will henceforth also call separation logic) is often defined as follows. There is a single type Val of values. Terms t are defined by a grammar

$$t ::= x \mid n \mid t + t \mid t - t \mid \cdots,$$

where $n : \mathrm{Val}$ are constants for all integers n. Formulas, also called assertions, are defined by

$$\phi ::= \top \mid \bot \mid t = t \mid t \mapsto t \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \to \phi \mid \phi * \phi \mid \phi \mathrel{-\!\!*} \phi \mid \mathbf{emp} \mid \forall x.\phi \mid \exists x.\phi$$

The symbol **emp** is used in separation logic for the unit of BI.

Note that the above is just another way of defining a signature (specification of types, function symbols and predicate symbols) for first-order predicate BI with a single type Val, function symbols $+, -, \ldots : \mathrm{Val}, \mathrm{Val} \to \mathrm{Val}$, constants $n : \mathrm{Val}$, and relation symbol $\mapsto \subseteq \mathrm{Val}, \mathrm{Val}$.

The pointer model. The standard pointer model of separation logic is usually presented as follows. It consists of a set $[\![\mathrm{Val}]\!]$ interpreting the type Val and a set $[\![\mathrm{Loc}]\!]$ of locations such that $[\![\mathrm{Loc}]\!] \subseteq [\![\mathrm{Val}]\!]$ and binary functions on $[\![\mathrm{Val}]\!]$ interpreting the function symbols $+, -$. The set $H = [\![\mathrm{Loc}]\!] \rightharpoonup_{fin} [\![\mathrm{Val}]\!]$ of finite partial functions from $[\![\mathrm{Loc}]\!]$ to $[\![\mathrm{Val}]\!]$, ordered discretely, is referred to as the set of *heaps*. The set of heaps has a partial binary operation $*$ defined by

$$h_1 * h_2 = \begin{cases} h_1 \cup h_2 & \text{if } h_1 \# h_2 \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where $\#$ is the binary relation on heaps defined by $h_1 \# h_2$ iff $\mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) = \emptyset$. The interpretation of the relation $\mapsto \subseteq [\![\mathrm{Val}]\!] \times [\![\mathrm{Val}]\!]$ is the subset of singleton heaps, that is, for $h \in H$, $h \in \mapsto$ iff $h = \{(v_1, v_2)\}$ for some values v_1, v_2. To define the standard interpretation of terms and formulas, one assumes a partial function $s : \mathrm{Var} \rightharpoonup_{fin} [\![\mathrm{Val}]\!]$, called a stack (also called a store in the literature). The interpretation of terms depends on the stack and is defined by

$$\begin{aligned} [\![x]\!]s &= s(x) \\ [\![n]\!]s &= [\![n]\!] \\ [\![t_1 \pm t_2]\!]s &= [\![t_1]\!]s \pm [\![t_2]\!]s \end{aligned}$$

The interpretation of formulas is standardly given by a forcing relation $s, h \Vdash \phi$, where $\mathrm{FV}(\phi) \subseteq \mathrm{dom}(s)$, as follows

$$
\begin{array}{lll}
s,h \Vdash t_1 = t_2 & \text{iff} & [\![t_1]\!]s = [\![t_2]\!]s \\
s,h \Vdash t_1 \mapsto t_2 & \text{iff} & \mathrm{dom}(h) = \{[\![t_1]\!]s\} \text{ and } h([\![t_1]\!]s) = [\![t_2]\!]s \\
s,h \Vdash \mathbf{emp} & \text{iff} & h = \emptyset \\
s,h \Vdash \top & & \text{always} \\
s,h \Vdash \bot & & \text{never} \\
s,h \Vdash \phi * \psi & \text{iff} & \text{there exists } h_1, h_2 \in H.\, h_1 * h_2 = h \text{ and} \\
 & & s, h_1 \Vdash \phi \text{ and } s, h_2 \Vdash \psi \\
s,h \Vdash \phi \mathrel{-\!\!*} \psi & \text{iff} & \text{for all } h', h' \# h \text{ and } s, h' \Vdash \phi \text{ implies } s, h * h' \Vdash \psi \\
s,h \Vdash \phi \vee \psi & \text{iff} & s,h \Vdash \phi \text{ or } s,h \Vdash \psi \\
s,h \Vdash \phi \wedge \psi & \text{iff} & s,h \Vdash \phi \text{ and } s,h \Vdash \psi \\
s,h \Vdash \phi \to \psi & \text{iff} & s,h \Vdash \phi \text{ implies } s,h \Vdash \psi \\
s,h \Vdash \forall x.\phi & \text{iff} & \text{for all } v \in [\![\mathrm{Val}]\!].s[x \mapsto v], h \Vdash \phi \\
s,h \Vdash \exists x.\phi & \text{iff} & \text{there exists } v \in [\![\mathrm{Val}]\!].s[x \mapsto v], h \Vdash \phi
\end{array}
$$

We now show how this pointer model is an instance of a BI-hyperdoctrine of a complete Boolean BI algebra (cf. Example 6).

The pointer model as a BI hyperdoctrine. Let $(H_\bot, *)$ be the discretely ordered set of heaps with a bottom element added to represent undefined, and let $* : H_\bot \times H_\bot \to H_\bot$ be the total extension of $* : H \times H \rightharpoonup H$ satisfying $\bot * h = h * \bot = \bot$, for all $h \in H_\bot$. This defines a partially ordered commutative monoid with the empty heap $\{\}$ as the unit for $*$. The powerset of H, $\mathcal{P}(H)$ (without $\bot$) is a complete Boolean BI algebra, ordered by inclusion and with monoidal closed structure given by (for $U, V \in \mathcal{P}(H)$):

- I is $\{\emptyset\}$
- $U * V := \{h * h' \mid h \in U \wedge h' \in V\} \setminus \{\bot\}$
- $U \mathrel{-\!\!*} V := \bigcup\{W \subseteq H \mid (W * U) \subseteq V\}$.

It can easily be verified directly that this defines a complete Boolean BI algebra; it also follows from more abstract arguments in [14, 1].

Let S be the BI hyperdoctrine induced by the complete Boolean BI algebra $\mathcal{P}(H)$ as in Example 6. To show that the interpretation of separation logic in this BI hyperdoctrine exactly corresponds to the standard pointer model presented above we spell out the interpretation of separation logic in S.

A term t in a context $\Gamma = \{x_1 : \mathrm{Val}, \ldots, x_n : \mathrm{Val}\}$ is interpreted as a morphism between sets:

- $[\![x_i : \mathrm{Val}]\!] = \pi_i$, where $\pi_i : \mathrm{Val}^n \to \mathrm{Val}$ is the i'th projection,
- $[\![n]\!]$ is the map $[\![n]\!] : [\![\Gamma]\!] \to 1 \to [\![\mathrm{Val}]\!]$ which sends the unique element of the one-point set 1 to $[\![n]\!]$,
- $[\![t_1 \pm t_2]\!] = [\![t_1]\!] \pm [\![t_2]\!] : [\![\Gamma]\!] \to [\![\mathrm{Val}]\!] \times [\![\mathrm{Val}]\!] \to [\![\mathrm{Val}]\!]$, where $[\![t_i]\!] : [\![\Gamma]\!] \to [\![\mathrm{Val}]\!]$, for $i = 1, 2$.

The interpretation of a formula ϕ in a context $\Gamma = \{x_1 : \mathrm{Val}, \ldots, x_n : \mathrm{Val}\}$ is given inductively as follows. Let $I = [\![\mathrm{Val}]\!] \times \cdots \times [\![\mathrm{Val}]\!] = [\![\mathrm{Val}]\!]^n$ and write $\overline{v}$ for elements of I. Then ϕ is interpreted as an element of $\mathcal{P}(I)$ as follows:

$$
\begin{array}{ll}
[\![t_1 \mapsto t_2]\!](\overline{v}) & = \{h \mid \mathrm{dom}(h) = \{[\![t_1]\!](\overline{v})\} \text{ and } h([\![t_1]\!](\overline{v})) = [\![t_2]\!](\overline{v})\} \\
[\![t_1 = t_2]\!](\overline{v}) & = H \text{ if } [\![t_1]\!](\overline{v}) = [\![t_2]\!](\overline{v}),\ \emptyset \text{ otherwise} \\
[\![\top]\!](*) & = H \\
[\![\bot]\!](*) & = \emptyset \\
[\![\mathbf{emp}]\!](*) & = \{h \mid \mathrm{dom}(h) = \emptyset\} \\
[\![\phi \wedge \psi]\!](\overline{v}) & = [\![\phi]\!](\overline{v}) \cap [\![\psi]\!](\overline{v}) \\
[\![\phi \vee \psi]\!](\overline{v}) & = [\![\phi]\!](\overline{v}) \cup [\![\psi]\!](\overline{v}) \\
[\![\phi \to \psi]\!](\overline{v}) & = \{h \mid h \in [\![\phi]\!](\overline{v}) \text{ implies } h \in [\![\psi]\!](\overline{v})\} \\
[\![\phi * \psi]\!](\overline{v}) & = [\![\phi]\!](\overline{v}) * [\![\psi]\!](\overline{v}) \\
 & = \{h_1 * h_2 \mid h_1 \in [\![\phi]\!](\overline{v}) \text{ and } h_2 \in [\![\psi]\!](\overline{v})\} \setminus \{\bot\} \\
[\![\phi \mathrel{-\!\!*} \psi]\!](\overline{v}) & = [\![\phi]\!](\overline{v}) \mathrel{-\!\!*} [\![\psi]\!](\overline{v}) \\
 & = \{h \mid [\![\phi]\!](\overline{v}) * \{h\} \subseteq [\![\psi]\!](\overline{v})\} \\
[\![\forall x : \mathrm{Val}\,.\phi]\!](\overline{v}) & = \bigcap_{v_x \in [\![\mathrm{Val}]\!]} ([\![\phi]\!](v_x, \overline{v})) \\
[\![\exists x : \mathrm{Val}\,.\phi]\!](\overline{v}) & = \bigcup_{v_x \in [\![\mathrm{Val}]\!]} ([\![\phi]\!](v_x, \overline{v}))
\end{array}
$$

Now it is easy to verify by structural induction on formulas ϕ that the interpretation given in the BI hyperdoctrine S corresponds exactly to the forcing semantics given earlier:

Theorem 8 *$h \in [\![\phi]\!](v_1, \ldots, v_n)$ iff $[x_1 \mapsto v_1, \ldots, x_n \mapsto v_n], h \Vdash \phi$.*

As a consequence, we of course obtain the well-known result that separation logic is sound for classical first-order BI. But, more interestingly, the correspondence also shows that we may easily extend separation logic to higher-order since the BI hyperdoctrine S soundly models higher-order BI. We expand on this in the next section, which also discusses other consequences of the above correspondence. First, however, we explain that one can also obtain such a correspondence for other versions of separation logic.

An intuitionistic model. Consider again the set of heaps $(H_\bot, *)$ with an added bottom $\bot$, as above. We now define the order by

$$h_1 \sqsupseteq h_2 \qquad \text{iff} \qquad \mathrm{dom}(h_1) \subseteq \mathrm{dom}(h_2) \text{ and for all } x \in \mathrm{dom}(h_1).\ h_1(x) = h_2(x).$$

Let I be the set of sieves on H, i.e., downwards closed subsets of H, ordered by inclusion. This is a complete BI algebra, as can be verified directly or by an abstract argument [1, 14].

Now let T be the BI hyperdoctrine induced by the complete BI algebra I as in Example 6. The interpretation of predicate BI in this BI hyperdoctrine corresponds exactly to the intuitionistic pointer model of separation logic, which is presented using a forcing style semantics in [6].

The permissions model. It is also possible to fit the permissions model of separation logic from [4] into the framework presented here. The main point is that the set of heaps, which in that model map locations to values and permissions, has a binary operation $*$, which makes $(H_\perp, *)$ a partially ordered commutative monoid.

Remark 9 *The correspondences between separation logic and BI hyperdoctrines given above illustrate that what matters for the interpretation of separation logic is the choice of BI algebra. Indeed, the main relevance of the topos-theoretic constructions in [14] for models of separation logic is that they can be used to construct suitable BI-algebras (as subobject lattices in categories of sheaves).*

4 Consequences for Separation Logic

We have shown above that it is completely natural and straightforward to interpret first-order predicate BI in first-order BI-hyperdoctrines and that the standard pointer model of separation logic corresponds to a particular case of BI-hyperdoctrine. Based on this correspondence, in this section we draw some further consequences for separation logic.

4.1 Formalizing Separation Logic

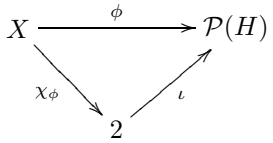
The strength of separation logic has been demonstrated in numerous papers before. In the early days of separation logic, it was shown that it could handle simple programs for copying trees, deleting lists, etc. The first proof of a more realistic program appeared in Yang's thesis [21], in which he showed correctness of the Schorr-Waite graph marking algorithm. Later, a proof of correctness of Cheney's garbage collection algorithm was published in [2], and other examples of correctness proofs of non-trivial algorithms may be found in [3]. In all of these papers, different simple extensions of core separation logic were used. For example, Yang used lists and binary trees as parts of his term language, and Birkedal et. al. introduced expression forms for finite sets and relations. It would seem that it is a weakness of separation logic that one has to come up with suitable extensions of it every time one has to prove a new program correct. In particular, it would make machine-verifiable formalizations of such proofs more burdensome and dubious if one would have to alter the underlying logic for every new proof.

We argue that the right way to look at these "extensions" is that they are really trivial definitional extensions of one and the same logic, namely the internal logic of the classical BI hyperdoctrine S presented in Section 3. The internal language of a BI hyperdoctrine $\mathcal{P}$ over $\mathcal{C}$ is formed as follows: to each object of $\mathcal{C}$ one associates a type, to each morphism of $\mathcal{C}$ one associates a function symbol, and to each predicate in $\mathcal{P}(X)$ one associates a relation symbol. The terms and formulas over this signature (considered as a higher-order signature [7]) form the internal language of the BI hyperdoctrine. There is an obvious structure for this language in $\mathcal{P}$.

Let $2 = \{\bot, \top\}$ be a two-element set (the subobject classifier of **Set**). There is a canonical map $\iota : 2 \to \mathcal{P}(H)$ that maps $\bot$ to $\{\}$ (the bottom element of the BI algebra $\mathcal{P}(H)$) and $\top$ to H (the top element of $\mathcal{P}(H)$).

Definition 10 *Let ϕ be an S-predicate over a set X, i.e., a function $\phi : X \to \mathcal{P}(H)$. Call ϕ* pure *if ϕ factors through ι.*

Thus $\phi : X \to \mathcal{P}(H)$ is pure if there exists a map $\chi_\phi : X \to 2$ such that



commutes. This corresponds to the notion of pure predicate traditionally used in separation logic [20].

The sub-logic of pure predicates is simply the standard classical higher-order logic of **Set**, and thus it is sound for classical higher-order logic. Hence one can use classical higher-order logic for defining lists, trees, finite sets and relations in the standard manner using pure predicates and prove the standard properties of these structures, as needed for the proofs presented in the papers referred to above. In particular, notice that recursive definitions of predicates, which in [21, 2, 3] are defined at the meta level, can be defined inside the higher-order logic itself. For machine verification one would thus only need to formalize one and the same logic, namely a sufficient fragment of the internal logic of the BI hyperdoctrine (with obvious syntactic rules for when a formula is pure). The internal logic itself is "too big" (it can have class-many types and function symbols, e.g.); hence the need for a fragment thereof, say classical higher-order logic with natural numbers.

4.2 Higher-order Separation Logic

As mentioned in Section 3, the interpretation of separation logic in BI hyperdoctrines shows that we may extend separation logic to higher-order. Specifically, we may quantify over *any* set X in $\forall x : X.\phi$ and $\exists x : X.\phi$, including "pure sets" of trees, lists, etc., but also including propositions — the BI algebra $\mathcal{P}(H)$ — and predicates — sets of the form $(\mathcal{P}(H))^Y$, for some set Y. The quantification over "pure sets" has been usefully applied in program proving with separation logic, as mentioned in the previous section (it has also been usefully applied in Hoare logic, as pointed out to us by John Reynolds, see [18]). It remains to be seen to what extent quantification over general propositions and predicates is useful in actual program proving. But let us consider a simple example, which indicates that it may be useful. Consider the assertion $\exists P : \mathsf{prop}\,.\ P \mathrel{-\!\!*} Q$. Intuitively, is says that for *some* extension of the current heap, described by P, the combined heap will satisfy Q. Consider a canonical algorithm for copying a tree. To describe the invariant, we look at the snapshot in Fig. 1. Suppose the

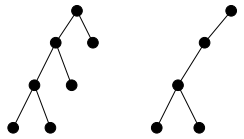


Fig. 1. Copying a tree.

predicate $\mathsf{tree}\ \tau$ asserts that "the tree on the left in Fig. 1 is represented in the heap" (we are a bit informal here, but a formal presentation would clutter the point). Then the following assertion describes the situation in Fig. 1:

$$\mathsf{tree}\ \tau * ((\exists l_1, l_2, v_1, v_2.\ l_1 \mapsto v_1 * l_2 \mapsto v_2) \mathrel{-\!\!*} \mathsf{tree}\ \tau).$$

However, we might not care about the number of actual values and locations that are missing in the "new" tree in the heap, but just wish to express that some of the original tree has been copied, and that the original tree has not been manipulated. This can be done with the assertion

$$\mathsf{tree}\ \tau * (\exists P : \mathsf{prop}.\ P \mathrel{-\!\!*} \mathsf{tree}\ \tau).$$

Future research will show how useful *higher-order* separation logic is in actual proofs of programs.

4.3 Logical Characterizations of Classes of Assertions

Different classes of assertions, precise, monotone, and pure, were introduced in [20], and it was noted that special axioms for these classes of assertions are valid. Such special axioms were further exploited in [2], where pure assertions were moved in and out of the scope of iterated separating conjunctions, and in [11], where precise assertions were crucially used to verify soundness of the hypothetical frame rule. The different classes of assertions were defined semantically and the special axioms were also validated using the semantics. We now show how the higher-order features of higher-order separation logic may be used to logically characterize the classes of assertions and logically prove the properties earlier taken as axioms. This is, of course, important for machine verification, since it means that the special classes of assertions and their properties can be expressed *in the logic*.

To simplify notation we just present the characterizations for *closed* assertions, the extension to open assertions is straightforward. Recall that closed assertions are interpreted in S as functions from 1 to $\mathcal{P}(H)$, i.e., as subsets of H.

In the proofs below, we use assertions which describe heaps in a canonical way. Since a heap h has finite domain, there is a unique (up to permutation) way to write an assertion $p_h \equiv l_1 \mapsto n_1 * \ldots * l_k \mapsto n_k$ such that $[\![p_h]\!] = \{h\}$.

Precise assertions. The traditional definition of a precise assertion is semantic, in that an assertion q is precise if, and only if, for all states s, h, there is at most one subheap h_0 of h such that $s, h_0 \Vdash q$. The following proposition logically characterizes closed precise assertions (at the semantic level, this characterization of precise predicates was mentioned in [10]).

Proposition 11 *The closed assertion q is precise if, and only if, the assertion*

$$\forall p_1, p_2 : \mathsf{prop}\,.\ (p_1 * q) \wedge (p_2 * q) \rightarrow (p_1 \wedge p_2) * q \tag{1}$$

is valid in the BI hyperdoctrine S.

Proof: The "only-if" direction is trivial, so we focus on the other implication. Thus suppose (1) holds for q, and let h be a heap with two different subheaps h_1, h_2 for which $h_i \in [\![q]\!]$. Let p_1, p_2 be canonical assertions that describe the heaps $h \setminus h_1$ and $h \setminus h_2$, respectively. Then $h \in [\![(p_1 * q) \wedge (p_2 * p)]\!]$, so $h \in [\![(p_1 \wedge p_2) * q]\!]$, whence there is a subheap $h' \subseteq h$ with $h' \in [\![p_1 \wedge p_2]\!]$. This is a contradiction. □

One can verify properties that hold for precise assertions *in the logic* without using semantical arguments. For example, one can show that $q_1 * q_2$ is precise if q_1 and q_2 are by the following logical argument: Suppose (1) holds for q_1, q_2. Then,

$$\begin{array}{ll} & (p_1 * (q_1 * q_2)) \wedge (p_2 * (q_1 * q_2)) \Rightarrow ((p_1 * q_1) * q_2) \wedge ((p_2 * q_1) * q_2)) \\ \Rightarrow & ((p_1 * q_1) \wedge (p_2 * q_1)) * q_2 \qquad \Rightarrow ((p_1 \wedge p_2) * q_1) * q_2 \\ \Rightarrow & (p_1 \wedge p_2) * (q_1 * q_2), \end{array}$$

as desired.

Monotone assertions. A closed assertion q is defined to be *monotone* if, and only if, whenever $h \in [\![q]\!]$ then also $h' \in [\![q]\!]$, for all extensions $h' \supseteq h$.

Proposition 12 *The closed assertion q is* monotone *if, and only if, the assertion $\forall p : \mathsf{prop}\,.\ p * q \rightarrow q$ is valid in the BI hyperdoctrine S.*

This is also easy to verify, and again, one can show the usual rules for monotone assertions in the logic (without semantical arguments) using this characterization.

Pure assertions. Recall from above that an assertion q is pure iff its interpretation factors through 2. Thus a closed assertion is pure iff its interpretation is either $\emptyset$ or H.

Proposition 13 *The closed assertion q is pure if, and only if, the assertion*

$$\forall p_1, p_2 : \mathsf{prop}\,.\ (q \wedge p_1) * p_2 \leftrightarrow q \wedge (p_1 * p_2) \tag{2}$$

is valid in the BI hyperdoctrine S.

Proof: Again, the interesting direction here is the "if" implication. Hence, suppose (2) holds for the assertion q, and that $h \in [\![q]\!]$. For any heap h_0, we must then show that $h_0 \in [\![q]\!]$. This is done via the verification of two claims.
Fact 1: For all $h' \subseteq h$, $h' \in [\![q]\!]$. Proof: Let p_1 be a canonical description of h', and p_2 a canonical description of $h \setminus h'$. Then $h \in [\![q \wedge (p_1 * p_2)]\!]$, so $h \in [\![(q \wedge p_1) * p_2]\!]$. This means that there is a split $h_1 * h_2 = h$ with $h_1 \in [\![q \wedge p_1]\!]$ and $h_2 \in [\![p_2]\!]$. But then, $h_2 = h \setminus h'$, so $h_1 = h'$, and thus, $h' \in [\![q]\!]$.
Fact 2: For all $h' \supseteq h$, $h' \in [\![q]\!]$. Proof: Let p_1 and p_2 be canonical descriptions of h and $h' \setminus h$, respectively. Then, $h' \in [\![(q \wedge p_1) * p_2]\!]$, so $h' \in [\![q \wedge (p_1 * p_2)]\!]$, and in particular, $h' \in [\![q]\!]$, as desired.

Using Facts 1 and 2, we deduce $h \in [\![q]\!] \;\Rightarrow\; \emptyset \in [\![q]\!] \;\Rightarrow\; h_0 \in [\![q]\!]$. □

4.4 Separation Logic for Richer Languages

Separation logic has mostly been used for low-level languages with a simple set-theoretic operational semantics. Yang [21, Ch. 9] has made some initial work on separation logic for richer languages such as Idealized Algol with heaps. For such richer languages, the semantics is typically given using more involved semantic structures such as functor categories and domains. We emphasize that all the developments in the present paper easily generalize to such more involved settings. Specifically, given any cartesian closed category $\mathcal{C}$ with an *internal* complete BI algebra B, one may construct a $\mathcal{C}$-indexed BI hyperdoctrine just as in Example 6.

5 Conclusion

We have introduced the notion of a (first-order) BI hyperdoctrine and shown that it soundly models classical and intuitionistic first- and higher-order predicate BI, thus connecting models of prediate BI with standard categorical notions of models of predicate logic. Moreover, we have shown that the standard pointer model of separation logic exactly corresponds to the interpretation of predicate BI in a BI hyperdoctrine. Finally, we have argued that this correspondence is of import for formalizations of separation logic, and that one can extend separation logic to higher-order.

Acknowledgements: The authors wish to thank Carsten Butz and the anonymous referees for their helpful comments and suggestions.

References

1. B. Biering. On the logic of bunched implications and its relation to separation logic. Master's thesis, University of Copenhagen, 2004.
2. L. Birkedal, N. Torp-Smith, and J. C. Reynolds. Local reasoning about a copying garbage collector. In *Proceedings of the 31-st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'04)*, pages 220 – 231, Venice, Italy, 2004.

3. R. Bornat. Local reasoning, separation and aliasing. In *Proceedings of the Second Workshop on Semantics, Program Analysis and Computing Environments for Memory Management (SPACE'04)*, Venice, Italy, January 2004.
4. R. Bornat, C. Calcagno, P. O'Hearn, and M. Parkinson. Permission accounting in separation logic. In *Proceedings of POPL'05*, Long Beach, CA, USA, January 2005. ACM. Accepted for publication.
5. C. Calcagno, P. W. O'Hearn, and R. Bornat. Program logic and equivalence in the presence of garbage collection. *Theoretical Computer Science*, 298(3):557 – 587, 2003.
6. S. Ishtiaq and P. W. O'Hearn. BI as an assertion language for mutable data structures. In *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'01)*, 2001.
7. B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, 1999.
8. F.W. Lawvere. Adjointness in foundations. *Dialectica*, 23(3/4):281–296, 1969.
9. P. W. O'Hearn, H. Yang, and J. C. Reynolds. Local reasoning about programs that alter data structures. In *Proceedings of CSL'01*, pages 1 – 19, Paris, France, September 2001. Springer Verlag.
10. P. W. O'Hearn, H. Yang, and J. C. Reynolds. Separation and information hiding (work in progress). Extended version of [11], 2003.
11. P. W. O'Hearn, H. Yang, and J. C. Reynolds. Separation and information hiding. In *Proceedings of the 31-st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'04)*, pages 268 – 280, Venice, Italy, 2004.
12. P.W. O'Hearn and D.J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, June 99.
13. A. M. Pitts. Categorical logic. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 5. Algebraic and Logical Structures*, chapter 2. Oxford University Press, 2000.
14. D. Pym, P. W. O'Hearn, and H. Yang. Possible worlds and resources: The semantics of BI. *Theoretical Computer Science*, 315(1):257 – 305, May 2004.
15. D. J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*. Kluwer Academic Publishers, 2002.
16. D. J. Pym. Errata and remarks for the semantics and proof theory of the logic of bunched implications. 2004. Available at `http://www.cs.bath.ac.uk/~pym/BI-monograph-errata.pdf`.
17. J. C. Reynolds. On extensions of separation logic. Private Communication.
18. J. C. Reynolds. *The Craft of Programming*. Prentice-Hall, 1981.
19. J. C. Reynolds. Intuitionistic reasoning about shared mutable data structure. In J. Davies, B. Roscoe, and J. Woodcock, editors, *Millennial Perspectives in Computer Science*, pages 303–321. Palgrave, Houndsmill, Hampshire, 2000.
20. J. C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Seventeenth Annual IEEE symposium on Logic in Computer Science (LICS'02)*, pages 55 – 74, Copenhagen, Denmark, 2002.
21. H. Yang. *Local Reasoning for Stateful Programs*. PhD thesis, University of Illinois, Urbana-Champaign, 2001.
22. H. Yang and U. Reddy. Correctness of data representations involving heap data structures. *Science of Computer Programming*, 50(1):129 – 160, March 2004.