Journal of Automated Reasoning (2025) 69:26
https://doi.org/10.1007/s10817-025-09739-4

First-order Hybrid Separation Logic

Frank S. de Boer[1] · **Hans-Dieter A. Hiep**[2]

Received: 24 April 2025 / Accepted: 17 September 2025

Abstract
The basic set-theoretic interpretation of the separating connectives of first-order separation logic allows for an effective, sound and complete axiomatization in a hybrid extension.

Keywords Separation logic · Natural deduction

1 Introduction

Formal reasoning about program correctness is important because memory safety bugs in operating systems and other systems-level software can have far-reaching consequences. For example, the CrowdStrike incident on 19 July 2024 is estimated to have resulted in over US$100 billion in disruption costs [9]. This incident was caused by an error in the operating system driver code of the cyber security platform of the CrowdStrike software company which resulted in an incorrect access of unallocated memory. Consequentially, processors executing the incorrect program crashed early in the boot process, causing machines to enter a reboot loop that required manual intervention to break free from, manifesting as outages and disruptions on a large, world-wide, scale [21].

Separation logic (abbreviated SL) [22, 26] facilitates formal reasoning about the (dynamic) allocation of variables that refer to that part of the memory called the 'heap'. Numerous tools and techniques in the research community are based on it and it is used by large computer technology companies such as Microsoft, Facebook and Amazon [19]. Industrial formal verification tools, such as Microsoft's SLAyer [1] and Facebook's Infer [22], are based on separation logic.

Separation logic extends first-order logic with the connectives of separating conjunction and separating implication for reasoning about structural properties of heap memory. A *separating conjunction* $p * q$ states that the heap can be partitioned in two disjoint sub-heaps which satisfy the formulas p and q, respectively. A *separating implication* $p \mathrel{-\!\!*} q$ states that q holds for any extended heap, where the disjoint extension satisfies p. Both

✉ Frank S. de Boer
frb@cwi.nl

Hans-Dieter A. Hiep
hdh@cwi.nl

[1] Leiden Institute of Advanced Computer Science (LIACS) & Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands

[2] NLnet Foundation & Leiden Institute of Advanced Computer Science (LIACS), Leiden, The hague, The Netherlands

Published online: 29 September 2025

separating connectives thus involve an implicit second-order quantification over heaps. The *graph* of the heap, as a partial function which assigns to each location of the heap its stored value or is undefined if the location is not allocated, is denoted by the so-called (binary) 'points-to' relation $\hookrightarrow$. It is worthwhile to observe here that the representation of a heap in SL as a relation allows to avoid the introduction and formalization of a notion of 'undefined', which would naturally arise in case of a representation of a heap in the logic as a *partial* function. Instead one may require functionality, or univalence, or the points-to relation. See [10] for a comprehensive discussion of the different ways to avoid the 'undefined'.

For their invention of Concurrent Separation Logic, the 2016 EATCS Gödel prize winners, Brookes and O'Hearn, wrote [4]: "It is all too easy to get caught up in completeness and related issues for formal systems that turn out to be too complicated when humans try to apply them; it is more important first to get a sense for the extent to which simple reasoning is or is not supported." In this paper however we show that already some basic valid assertions give rise to incompleteness of existing formalizations of SL, though intuitively these assertions require a simple reasoning about the *set-theoretic* interpretation of the separating connectives in terms of the points-to relation, as informally described above. In his seminal 2002 paper, Reynolds also acknowledges: "Finally, we give axiom schemata for the ['points to'] predicate $\mapsto$. Regrettably, these are far from complete." [26]

Main contribution We introduce in this paper a natural deduction-style proof system which fully supports simple reasoning about the basic set-theoretic interpretation of the 'points-to' relation and the separating connectives.

Proof-theoretically, the main innovation of this paper is a novel first-order generalization of the propositional hybrid extension of SL as introduced and studied in [5]. Quoting [27], most hybrid logics are based on so-called *satisfaction operators* to formalize a statement being true at a particular time, possible world, or some other context. In [5] abstract *nominals* are introduced to denote the heap at which the specified formula holds. In our first-order generalization, the heap as point of reference is denoted by the *extension* of a formula in separation logic itself (such an extension consists of the set of tuples of values that satisfy the formula). The resulting first-order satisfaction operator allows to reason about the separating connectives in a simple set-theoretic way. This operator also highlights the basic *referential opacity* of the separating connectives: we cannot resolve this operator by *substituting* the free occurrences of the points-to relation, which do not appear in the scope of the separating connectives, by the heap specification, because the bound occurrences of the points-to relation, which do appear in the scope of a separating connective, implicitly refer to the same heap. Instead of resolving the satisfaction operator directly by substitution, we resolve it by a compositional analysis of its scope, which involves a straightforward set-theoretic axiomatization of the separating connectives in the context of this operator. The proof system itself is an extension of first-order logic which admits the standard *structural* inference rules of weakening and contraction.

We illustrate the basic set-theoretic reasoning by some examples which have been formalized in Rocq's Calculus of (Co-)Inductive Constructions, available on Zenodo [14]. This prototype formalization extends the type of propositions in Rocq with the separating connectives and the satisfaction operators, and the axioms of the proof system.

Related work Existing formalizations on the one hand syntactically restrict SL, or on the other hand abstract/generalize its semantics.

For example, a tableaux method for a propositional fragment of SL has been developed in [8] which has been proven sound and complete. Limited extensions to first-order SL are discussed which use a labeling mechanism for encoding (finite) heap structures. In [18] a sound modal sequent calculus for propositional SL is introduced, which however is incom-

plete because it provides limited support for equational reasoning about the modal contexts (so-called 'worlds') associated with the SL formulas. This incompleteness is overcome in [5] for a *hybrid* extension of propositional SL. In [7] a sound and complete sequent calculus is described for a quantifier-free subset of SL (however, completeness is restricted to provability of valid assertions, and does not cover general completeness of the consequence relation itself).

On the other hand, examples of further abstractions and generalizations are [15] and [23], and both describe a finitary logic which is sound and complete. In [23], models are based on very general preordered commutative monoids[1] but there is no points-to relation. In [15], special commutative monoids called *separation algebras* are used to give semantics to the separating connectives in the presence of an abstracted version of the points-to predicate. Separation algebras also form the very basis of the Iris project [17] which formalizes the semantics of intuitionistic SL in the higher-order logic of Rocq [16]. The elements of such separation algebras represent the heaps over which implicit quantification in the semantics of separating connectives ranges. However, the semantics of separating conjunction is defined *only* in terms of the underlying abstract monoid, and as such is decoupled from the set-theoretic interpretation of the points-to relation. Additional axioms are required to capture this coupling (as already observed by Reynolds, see above).

Plan of the paper In the next section we introduce the syntax of (the hybrid extension of) SL and describe formally its semantics. In Section 3 we introduce the proof system and provide an example proof. Section 4 provides a general discussion of soundness, completeness and expressivity of SL.

2 Syntax

In this section we introduce the syntax of SL. For an accessible introduction to separation logic, see [25]. We are given a first-order signature of function symbols, predicate symbols; and a set of first-order variables $x, y, z, \ldots$. The first-order terms of this signature are denoted by $t, t', \ldots$. We have the following standard inductive definition of formulas (assertions) of separation logic:

$$p ::= \bot \mid t_1 = t_2 \mid t_1 \hookrightarrow t_2 \mid R(t_1, \ldots, t_n) \mid (p \to q) \mid \forall x p \mid (p * q) \mid (p \mathrel{-\!\!*} q)$$

where R is a n-ary relation symbol. The binary relation symbols for equality $=$ and 'points-to' $\hookrightarrow$ are included in the signature.

By $\bot$ we denote falsity and $(p \to q)$ stands for propositional implication. From these we can derive all other propositional connectives, such as negation $\neg p$, truth $\top$, propositional conjunction $(p \wedge q)$, and propositional disjunction $(p \vee q)$. We have universal quantification $\forall x p$ where the variable x is bound in the usual way, and we can define existential quantification $\exists x p$ as its dual. Separating conjunction is denoted by '$*$' and separating implication (also called the 'magic wand') is denoted by '$-\!\!*$'. Quantification is first-order in the sense that quantification ranges over individuals.

Given a first-order model which provides an interpretation of the first-order signature, including the first-order variables, an assertion p is evaluated with respect to a heap, that is, a partial, unary function, which provides the interpretation of the points-to relation $\hookrightarrow$ as its graph. Thus $(t_1 \hookrightarrow t_2)$ holds in a given heap if the value of t_1 belongs to its domain and applying the heap as a function to that value gives the value of t_2 (this is also called the

[1] A monoid is a set with a single associative binary operation and a neutral element.

weak/loose points-to [3, 20]). The strong/strict version of the points-to relation, denoted by $(t_1 \mapsto t_2)$, additionally states that the domain of the heap only consists of the value of t_1. Note that it can be expressed in terms of the weak points-to by $(t_1 \hookrightarrow t_2) \wedge \forall x, y((x \hookrightarrow y) \rightarrow x = t_1)$ (here x and y are 'fresh' variables not appearing in t_1 and t_2).

The formula $p * q$ states that the given heap can be split into two (disjoint) heaps which satisfy p and q, respectively. As a simple example, we can express $(t_1 \hookrightarrow t_2)$ by $(t_1 \mapsto t_2) * \top$. The formula $p \mathrel{-\!\!*} q$ states the implication, if any disjoint extension heap satisfies p, then that together with the current heap satisfies q. As a simple example, $\top \mathrel{-\!\!*} p$ holds in the empty heap if and only if p holds in every heap.

In order to reason about the validity of SL formulas in a convenient manner, we first generalize the semantics of the points-to relation to *arbitrary* binary relations, instead of binary relations that define the graph of a partial function. The concept of disjoint, binary relations, underlying the generalized semantics of the separating connectives, is defined in terms of the *domains* of the relations. We can model the restriction to heaps simply by *syntactically* restricting occurrences of separating implication to assertions of the form $(p \wedge \mathit{fun}) \mathrel{-\!\!*} q$, where *fun* denotes the assertion $\forall x, y, z((x \hookrightarrow y \wedge x \hookrightarrow z) \rightarrow y = z)$. Let p' denote the result of restricting syntactically all occurrences of separating implication in p to heaps (as described above). It follows that the evaluation of $p' \wedge \mathit{fun}$ is restricted to heaps.

It is straightforward to generalize the semantics of SL to an arbitrary relational interpretation of the points-to relation. Let $M = (D, I)$ denote a first-order model, where D denotes the non-empty universe and I provides an interpretation of the (first-order) variables and the function and predicate symbols, including the points-to relation $\hookrightarrow$. We use the set-theoretic union $\mathcal{R} \cup \mathcal{R}'$ of binary relations $\mathcal{R}$ and $\mathcal{R}'$ considered as sets of pairs. Moreover, by $\mathcal{R} \perp \mathcal{R}'$ we denote that the *domains* of the (binary) relations $\mathcal{R}$ and $\mathcal{R}'$ are *disjoint*.[2] Given $M = (D, I)$ and $d \in D$, we abbreviate the update $(D, I[x := d])$, which assigns the value d to the variable x, by $M[x := d]$. Given $M = (D, I)$ and $\mathcal{R} \subseteq D \times D$, we abbreviate the update $(D, I[\hookrightarrow := \mathcal{R}])$ of the points-to relation by $M[\mathcal{R}]$.

Given a model $M = (D, I)$ we omit the standard inductive definition of the value $I(t) \in D$ of a first-order term t. We denote by $M \models p$ that p holds in the model M, formally defined as follows.

Definition 1 (Semantics of SL) Given $M = (D, I)$ we have

- $M \not\models \bot$.
- $M \models t_1 = t_2$ if and only if $I(t_1) = I(t_2)$.
- $M \models R(t_1, \ldots, t_n)$ if and only if $\langle I(t_1), \ldots, I(t_n)\rangle \in I(R)$.
- $M \models p \rightarrow q$ if and only if $M \models p$ implies $M \models q$.
- $M \models \forall x p$ if and only if $M[x := d] \models p$, for all $x \in D$.
- $M \models (p * q)$ if and only if $M[\mathcal{R}] \models p$ and $M[\mathcal{R}'] \models q$, for some $\mathcal{R}, \mathcal{R}' \subseteq D \times D$ such that $\mathcal{R} \perp \mathcal{R}'$ and $I(\hookrightarrow) = \mathcal{R} \cup \mathcal{R}'$.
- $M \models (p \mathrel{-\!\!*} q)$ if and only if $M[\mathcal{R}] \models p$ implies $M[I(\hookrightarrow) \cup \mathcal{R}] \models q$, for all $\mathcal{R} \subseteq D \times D$ such that $I(\hookrightarrow) \perp \mathcal{R}$.

Note that as a special case of $M \models R(t_1, \ldots, t_n)$ we have that $M \models (t \hookrightarrow t')$ if and only if $\langle I(t), I(t')\rangle \in I(\hookrightarrow)$.

Given this general setting we can introduce and define the semantics of so-called basic *satisfaction formulas* $p@_{x,y}q$.[3] The (distinct) variables x and y are the *formal parameters*

[2] The domain of an arbitrary relation $\mathcal{R} \subseteq D \times D$ is the set comprising exactly those $d \in D$ for which there exists a $d' \in D$ such that $\langle d, d'\rangle \in \mathcal{R}$.

[3] Following the terminology of the hybrid logics overview paper [27].

of the satisfaction operator and instantiated as follows. For $M = (D, I)$ we denote by $M[x, y := d, d']$ the update of the interpretation I which assigns $d \in D$ and $d' \in D$ to the variables x and y, respectively. We denote by $M(q) \subseteq D \times D$ the (binary) relation defined by q which consists of all pairs $\langle d, d' \rangle \in D \times D$ such that $M[x, y := d, d'] \models q$. Given a model $M = (D, I)$, we denote by $M[q]$ the model $M = (D, I[\hookrightarrow := M(q)])$, which thus assigns to the points-to relation the (binary) relation defined by q.

Definition 2 (Semantics basic satisfaction formulas) We have

– $M \models p@_{x,y}q$ if and only if $M[q] \models p$, where $M[q]$ is defined as above with respect to the formal parameters x and y.

Any SL formula p is equivalent to $p@_{x,y}(x \hookrightarrow y)$ because the relation defined by $(x \hookrightarrow y)$, as defined above, clearly equals the points-to relation itself. However, we cannot resolve a basic satisfaction assertion $p@_{x,y}q$ by simply replacing the *free* occurrences of assertions $(t \hookrightarrow t')$, that is, those occurrences which do not fall in the scope of a separating connective, in p by $q(t, t')$, because this fails to take into account the interpretation of the separating connectives in p.

We have the following syntax of general satisfaction formulas ϕ.

$$\phi ::= \bot \mid (p@_{x,y}q) \mid (\phi \to \phi) \mid \forall x \phi$$

(where p and q denote SL formulas as defined above). We reason about SL formulas in terms of these general satisfaction formulas. Alternatively, we could have fully integrated the satisfaction operator in the syntax of SL. This would give rise to complex formulas with nested occurrences of the satisfaction operator and occurrences of the satisfaction operator in the context of the separating connectives. However such complexity is not needed in proving the validity of SL formulas. Therefore we opted for a separate class of general satisfaction formulas. Since we do not have nested occurrences of the satisfaction operator, we can omit for notational convenience the formal parameters, assuming by default the distinguished formal parameters x and y (so that in fact we have a single satisfaction operator instead of a satisfaction operator for each pair of variables). Instead of $p@_{x,y}q$ we write $p@q$.

3 The Proof System

The proof system for deriving the validity of these satisfaction formulas extends the standard proof system for first-order logic. Figure 1 specifies the axioms and proof rules involving constructs of SL in the context of the satisfaction operator (the symbols E and I stand for 'Elimination' and 'Introduction', respectively). Apart from axiom (I), the upper box shows the axioms for formulas $p@q$, where p is not a top-level separating conjunction nor a separating implication. As already observed above, the relation defined by $x \hookrightarrow y$ equals the points-to relation itself, and thus, as stated by axiom (I), any SL formula p is equivalent to $p@(x \hookrightarrow y)$. Axiom (E1) simply substitutes the arguments of the points-to relation for the variables x and y in the formula representing the relation which serves as the reference point. Here $p[x, y := t, t']$ denotes the result of simultaneous substitution of all free occurrences of the variables x and y by t and t', respectively. The notion of free variables is defined as usual but for the clause that defines the free variables of $p@q$ as the union of the set of free variables of q *excluding* the (default) formal parameters x and y, and the set of free variables of p. Free occurrences of the (default) formal parameters x and y in q are thus bound by the satisfaction operator. Axiom (E2) states that the satisfaction operator does not affect any

(I) $p \leftrightarrow (p@(x \hookrightarrow y))$

(E1) $((t \hookrightarrow t')@q) \leftrightarrow q[x, y := t, t']$

(E2) $(p@q) \leftrightarrow p$, where $p ::= \bot \mid t = t' \mid R(\bar{t})$

$(\rightarrow)$ $((p \rightarrow q)@r) \leftrightarrow ((p@r) \rightarrow (q@r))$

$(\forall)$ $((\forall z p)@q) \leftrightarrow \forall z(p@q)$ where z does not occur free in q

$$(*\mathrm{E}) \quad \frac{(p * q)@r \quad (r = R_1 \uplus R_2) \rightarrow (p@R_1) \rightarrow (q@R_2) \rightarrow r'}{r'} \text{ fresh } R_1, R_2$$

$$(*\mathrm{I}) \quad \frac{r = r_1 \uplus r_2 \quad p@r_1 \quad q@r_2}{(p * q)@r}$$

$$(-\!\!*\mathrm{E}) \quad \frac{(p -\!\!* q)@r \quad R = r \uplus r' \quad p@r'}{q@R}$$

$$(-\!\!*\mathrm{I}) \quad \frac{(R' = r \uplus R) \rightarrow (p@R) \rightarrow (q@R')}{((p -\!\!* q)@r)} \text{ fresh } R, R'$$

Fig. 1 Proof system for SL in terms of the @-operator

other basic formula. The axioms $(\rightarrow)$ and $(\forall)$ state that the satisfaction operator distributes over implication and (universal) quantification, respectively. In axiom $(\forall)$ the variable z is supposed not to occur free in q.

At the heart of the proof system for the separating connectives is the logical formalization of the set-theoretic (domain-disjoint) union of (binary) relations in terms of SL formulas. This formalization is abbreviated by $p = q \uplus r$ which states that the domains of the binary relations defined by the formulas q and r are disjoint, and the binary relation defined by p is the union of that of q and r. As a simple example we have that $(x \hookrightarrow y) = R_1(x, y) \uplus R_2(x, y)$ denotes the conjunction of the formulas $\forall x, y, z(\neg R_1(x, y) \vee \neg R_2(x, z)))$ and $\forall x, y((x \hookrightarrow y) \leftrightarrow (R_1(x, y) \vee R_2(x, y)))$. This conjunction thus states that the points-to relation is the domain-disjoint union of the relations denoted by R_1 and R_2.

The proof system for the separating connectives are best explained in terms of the following proof strategies. According to rule $(*\mathrm{E})$, in order to prove that $(p * q)@r$ implies r' one has to introduce fresh relation symbols R_1 and R_2 and prove that r' holds under the assumption that R_1 and R_2 form a partition of r which satisfies p and q, respectively. According to rule $(*\mathrm{I})$, in order to establish $(p * q)@r$ one has to introduce so-called *witnesses* for a partition of r which satisfies the assertions p and q. According to rule $(-\!\!*\mathrm{E})$, we can derive $q@R$ from $(p -\!\!* q)@r$ whenever $R = r \uplus r'$, for some r' such that $p@r'$. Finally, in order to establish $(p -\!\!* q)@r$, which involves a (second-order) universal quantification, we have the rule $(-\!\!*\mathrm{I})$, where the premise requires to prove that $q@R'$, assuming that $R' = R \uplus r$ and $p@R(x, y)$, for fresh binary relation symbols R and R'.

A proof consists of a finite sequence of formulas each of which is either an assumption, an axiom, or follows by an application of an inference rule like modus ponens from preceding formulas. For a proof of a formula $p_1 \rightarrow p_2 \rightarrow \ldots \rightarrow p_n \rightarrow q$, we introduce a box to

Proof.

1. $(x \hookrightarrow y) * \top$ assumption.
2. $((x \hookrightarrow y) * \top)@(x \hookrightarrow y)$ from 1 by (I).
3. $\hookrightarrow = R_1 \uplus R_2$ assumption (R_1, R_2 fresh).
4. $(x \hookrightarrow y)@R_1$ assumption.
5. $R_1(x, y)$ from 4 by (E1).
6. $\top@R_2$ assumption.
7. $(x \hookrightarrow y)$ from 3 and 5.
8. $(\top@R_2) \to (x \hookrightarrow y)$ from 6–7.
9. $((x \hookrightarrow y)@R_1) \to (\top@R_2) \to (x \hookrightarrow y)$ from 4–8.
10. $(\hookrightarrow = R_1 \uplus R_2) \to ((x \hookrightarrow y)@R_1) \to (\top@R_2) \to (x \hookrightarrow y)$ from 3–9.
11. $(x \hookrightarrow y)$ from 2 and 10 by (∗E).
12. $((x \hookrightarrow y) * \top) \to (x \hookrightarrow y)$ from 1–11.
13. $\forall x, y(((x \hookrightarrow y) * \top) \to (x \hookrightarrow y))$ from 1–12 by the rule of universal generalization.

Fig. 2 Proof of $\forall x, y(((x \hookrightarrow y) * \top) \to (x \hookrightarrow y))$

enclose the sub-proof which introduces $p_1, \ldots, p_n$ as *assumptions* and establishes q as the conclusion. Figure 2 presents a proof (of the formula **A1** introduced later in this paper). In this proof, we do not explicitly write down how to do first-order reasoning, instead we focus on the application of the new axioms.

For notational convenience, we use the following abbreviations. A formula $p@_{x,y}q$ is abbreviated by $p@q$. Further, occurrences of just a (binary) relation symbol R abbreviate basic formulas $R(x, y)$, thus omitting the default parameters x and y of the satisfaction operator. Hence, we abbreviate $p@_{x,y}R(x, y)$ by and $p@R$. Similarly, $\hookrightarrow = R_1 \uplus R_2$ abbreviates $(x \hookrightarrow y) = R_1(x, y) \uplus R_2(x, y)$.

Let us explain the construction of this proof in a top-down manner. First, we observe that by the first-order inference rule of *universal generalization* it suffices to prove $((x \hookrightarrow y)*\top) \to (x \hookrightarrow y)$ (see line 12). To prove this we introduce $(x \hookrightarrow y)*\top$ as an assumption (see line 1). The box (see lines 1–11) then shows how to derive $(x \hookrightarrow y)$ from this assumption. In order to apply axiom (∗E) we first apply axiom (I) to obtain $((x \hookrightarrow y)*\top)@(x \hookrightarrow y)$ (see line 2). In order to derive $(x \hookrightarrow y)$ from $((x \hookrightarrow y)*\top)@(x \hookrightarrow y)$ by an application of axiom (∗E), we need to prove that $\hookrightarrow = R_1 \uplus R_2$ implies $((x \hookrightarrow y)@R_1) \to ((\top@R_2) \to (x \hookrightarrow y))$ (see line 10). This proof is presented by the box defined by the lines 3–9 which introduces the assumption $\hookrightarrow = R_1 \uplus R_2$ (see line 3). In the same vein the box defined by the lines 4–8 shows how to derive $(\top@R_2) \to (x \hookrightarrow y)$ from the assumption $(x \hookrightarrow y)@R_1$. Finally, the innermost box defined by the line 6 and 7 shows how to derive $(x \hookrightarrow y)$ from the assumption $\top@R_2$. Note that $\top@R_2$ reduces to $\top$ by the axioms (E2) and $(\to)$. On the other hand, from the assumption $(x \hookrightarrow y)@R_1$ we derive $R_1(x, y)$ by axiom (E1), and so we derive $x \hookrightarrow y$ from the assumption $\hookrightarrow = R_1 \uplus R_2$ by trivial first-order reasoning.

Similar proofs can be given for the formulas **A2** and **A3** (also introduced later in this paper). The interested reader can find these proofs formalized in in Rocq, available on Zenodo [14], which also includes proofs of some basic algebraic laws like $p * q \to q * p$ (commutativity of separating conjunction) and the adjunction property $(p * (p \mathrel{-\!\!*} q)) \to q$. These proofs all use basic set-theoretic properties of domain-disjoint union of binary relations.

Let $(x_i \hookrightarrow -)$ abbreviate $\exists y(x_i \hookrightarrow y)$. The sentences p_n defined by

$$\exists x_1, \ldots, x_n((x_1 \hookrightarrow -) * \ldots * (x_n \hookrightarrow -))$$

then state that there exist at least n allocated locations (elements of the universe of the given first-order model). Note that the semantics of the separating conjunction implies that $x_i \neq x_j$ for $i \neq j$. It is also possible to formulate the same property using propositional conjunction instead of separating conjunction by explicitly stating this fact, that the variables are not aliases. We again refer to [14] for the formalization of a proof of the equivalence between $p_n = (x_1 \hookrightarrow -) * \ldots * (x_n \hookrightarrow -)$ and $q_n = \bigwedge_{i=1}^{n}((x_i \hookrightarrow -) \wedge \bigwedge_{j=1}^{i-1} x_i \neq x_j)$, $n \geq 1$. Here x_n, for $1 \leq n$, is a sequence of distinct variables. Note that both formulas state that the domain of the heap consists of at least n elements. A proof of this equivalence can also be given in the state-of-the-art interactive proof assistant Iris [17] using deduction rules for a recent extension of separation algebras [11]. The proof, however, requires a complicated unraveling of the many layers of the definition of the points-to predicate in Iris.[4]

Another basic example which already exceeds the capability of all the automated separation logic provers in the benchmark competition SL-COMP, is the equivalence between $(x \hookrightarrow -) \wedge ((x = y \wedge z = w) \vee (x \neq y \wedge (y \hookrightarrow z)))$ and $(x \hookrightarrow -) * ((x \hookrightarrow w) \mathrel{-\!\!*} (y \hookrightarrow z))$. Both formulas express the *weakest precondition* of the statement $[x] := w$ which stores the value of w in the location denoted by x with respect to the postcondition $(y \hookrightarrow z)$. Note that the first formula provides a simple explicit description of the aliasing hidden in the semantics of the separating connectives. It is not difficult to prove the equivalence in our system. In fact, this equivalence is expressed in quantifier-free separation logic, for which a complete axiomatization was already known [7]. On the other hand, of all the automated separation logic provers in the benchmark competition SL-COMP, only the CVC tool [24] supports the fragment of separation logic that includes the separating implication. However, from our own experiments with that tool, we found that it produces an incorrect counter-example and reported this as a bug to the maintainers of the project.[5] But also in an interactive tool like Iris it is not obvious how to prove this equivalence: additional axioms are required [6]. The question then arises what axioms and do they provide a complete axiomatization.

We conclude this section with the observation that the points-to relation and the definition of the $\uplus$-operation on formulas can also be viewed as parameters of the proof system. More precisely, we can generalize the points-to relation to an arbitrary n-ary relation $R_{\hookrightarrow}$ and a satisfaction formula accordingly to $p@_{\bar{x}}q$, where $\bar{x}$ denotes a sequence of variables $x_1, \ldots, x_n$. Axiom (I) then generalizes to $p \leftrightarrow (p@_{\bar{x}}R_{\hookrightarrow}(\bar{x}))$ and axiom $(E1)$ to $(R_{\hookrightarrow}(\bar{t})@_{\bar{x}}q) \leftrightarrow q[\bar{x} := \bar{t}]$, where $\bar{t}$ denotes a sequence of terms $t_1, \ldots, t_n$. For example, we can instantiate these parameters to obtain a proof system for reasoning about race-free sharing of the heap between concurrent threads [2]. In this setting the points-to relation is a ternary relation $x \hookrightarrow_t y$, where t denotes a *rational* number (greater than or equal to zero). There, we redefine $p = q \uplus r$ (assuming the fixed parameters x, y, z) by

$$\forall x, y, z(p(x, y, z) \leftrightarrow \exists z_1, z_2(q(x, y, z_1) \wedge q(x, y, z_2) \wedge z = z_1 + z_2 \wedge z \leq 1))$$

to obtain a logical formalization of the model of *fractional permissions* as described in [2]. As another example, consider SL with a *weak* separating conjunction [28], instead of *strong* separating conjunction, by redefining $p = q \uplus r$ as the conjunction of the formulas

$$\forall x((\exists y\, q(x, y) \wedge \exists y\, r(x, y)) \rightarrow \forall y(q(x, y) \leftrightarrow r(x, y)))$$

and $\forall x, y(p(x, y) \leftrightarrow (q(x, y) \vee r(x, y)))$, which express that the (binary) relations defined q and r may overlap in domain if they assign the same values to those overlapping locations.

[4] Personal communication by Robbert Krebbers.

[5] See issue: https://github.com/cvc5/cvc5/issues/12103.

[6] Personal communication by Robbert Krebbers.

4 Soundness, Completeness and Expressivity

A proof of the soundness of the proof system consists of a standard induction on the length of the derivation. The proof is rather straighforward because the axioms and proof rules follow closely the semantics of the separating connectives and the satisfaction operators.

Clearly, the main complexity of SL stems from the (second-order) quantification over heaps (or sub-heaps, as in the case of the separating conjunction). For second-order logic a sound and complete axiomatization can be obtained by generalizing its semantics to so-called *general models* [12]. Such models extend first-order models with a set of possible interpretations of the second-order variables. For example, instead of interpreting a monadic predicate over *all* possible subsets of the given first-order universe, a general model restricts its interpretation to a given set of such subsets. This generalization of the semantics of second-order logic allows for a sound and complete axiomatization by restricting to so-called Henkin models. A Henkin model is a general model for second-order logic which additionally satisfies the comprehension axiom

$$\exists R \forall x_1, \ldots, x_n (R(x_1, \ldots, x_n) \leftrightarrow p(x_1, \ldots, x_n))$$

for any second-order formula $p(x_1, \ldots, x_n)$ which does not contain the n-ary relation symbol R. In the *arithmetic* comprehension axiom, $p(x_1, \ldots, x_n)$ is a first-order formula.

In order to obtain an effective proof system for SL, we first need to generalize the semantics to *infinite* heaps because the semantics restricted to finite heaps gives rise to a *non-compact* logic (as argued in more detail below).

Generalizing the semantics of SL accordingly in terms of a given set of possible (finite and infinite) heaps, which does not necessarily contain *all* heaps, we can formulate in SL the following arithmetic comprehension axiom

$$\blacklozenge(\forall x, y((x \hookrightarrow y) \leftrightarrow p(x, y)))$$

where $\blacklozenge p$ denotes the dual of the modality $\blacksquare p$, and $\blacksquare p$ is an abbreviation of $\top * (\mathbf{emp} \wedge (\top \mathrel{-\!\!*} p))$, which expresses that p holds in *every* heap. The above formula then states the existence of a heap such that its *graph*, as denoted by the points-to relation $\hookrightarrow$, satisfies the 'pure' first-order formula $p(x, y)$, i.e., p does not involve the separating connectives nor the points-to relation. Such an instance of the arithmetic comprehension axiom holds if there exists a heap which is characterized by the formula $p(x, y)$. We cannot generalize this axiom to arbitrary SL formulas because it is not obvious how to avoid contradictions like $\blacklozenge(\forall x, y((x \hookrightarrow y) \leftrightarrow \neg(x \hookrightarrow y)))$. Requiring that the points-to relation does not occur in $p(x, y)$ yields again the arithmetic comprehension axiom. In [6], we therefore introduced a new interpretation of SL that restricts the (second-order) quantification to *first-order definable* heaps. For this new interpretation we developed a *sequent calculus* which is sound and complete. The completeness proof is based on the construction of a model for a *consistent* theory (a theory from which false is not derivable), following [13]. From the completeness proof we further derive that this new interpretation satisfies both compactness and the downward Löwenheim-Skolem theorem. By the seminal theorem of Lindström we then infer that this new interpretation is as expressive as first-order logic. The soundness and completeness proofs in [6] can be extended in a straightforward but tedious manner to the proof system presented in this paper which generalizes the satisfaction operator introduced in [6] to arbitrary SL formulas. Instead of the above comprehension axiom, this result holds for general models of SL for which a semantic comprehension condition holds, such that the set of possible heaps includes for every SL formula $p(x, y)$ the heap described by it.

We continue with a brief discussion of the restriction to *finite* heaps in the standard model of SL. Because of Gödel's incompleteness theorem this extension of the standard first-order model of arithmetic is clearly incomplete as well, due to incompleteness of arithmetic itself. However, the semantics of SL with respect to arbitrary first-order models with finite heaps ranging of an arbitrary universe (not necessarily containing the elements required for arithmetic), still does not in general allow for an effective, sound and complete axiomatization because the corresponding notion of validity is not *compact*: consider the infinite set of sentences p_n which state that the domain of the heap consists of at least n elements (as defined above). Clearly every finite subset has a model but the entire set does not have a model. Suppose, on the other hand, that there exists an effective, sound and complete axiomatization. Completeness then implies the existence of a derivation of a contradiction from the above infinite set of sentences. Since the proof system is effective, there exists a finite subset from which we could derive a contradiction. Since the proof system is sound, this subset does not have a model. But every finite subset has a model. Therefore, an effective, sound and complete proof system for SL requires at least a semantics that allows for infinite heaps too.

Summarizing, we have discussed the following hierarchy of the different semantics of SL (for arbitrary first-order models).

$$\begin{array}{ccc} \textbf{Weak} & \subset & \textbf{General} \\ & & \cup \\ & & \textbf{Complete} \\ & & \cup \\ & & \textbf{Full} \end{array}$$

Here **Weak** denotes the class of models restricted to finite heaps, **Full** denotes the class of models which include *all* heaps, both finite and infinite, **Complete** denotes the class of models which specify a subset of the set of all possible heaps that includes for every SL formula $p(x, y)$ the heap described by it, and finally, **General** denotes the class of models which specify a subset of the set of all possible heaps.

As already observed above, for the class of models **Weak** there does not exist an effective, sound and complete proof system, because the corresponding notion of validity is not compact. It is worthwhile to observe that also the notion of validity defined by the class of models **Full** is also not compact, because the above $\blacksquare$-modality allows to express that the universe D of the first-order model is finite by asserting that every injective function $f : D \to D$ is a surjection. This can be formalized as follows.

- Let *inj* be $\forall x, y, z((x \hookrightarrow z \wedge y \hookrightarrow z) \to x = y)$.
- Let *tot* be $\forall x \exists y(x \hookrightarrow y)$.
- Let *sur* be $\forall x \exists y(y \hookrightarrow x)$.

The formula $tot \wedge inj \wedge fun$ states that the points-to relation is a total injective function. And so $\blacksquare(tot \wedge inj \wedge fun \to sur)$ states that every total injective function is surjective, and thus that the universe D is finite (for infinite universes there exists total injective functions which are not surjective). Note that the occurrences of $\hookrightarrow$ in the scope of the $\blacksquare$-modality are universally bounded, and the interpretation of $\hookrightarrow$ thus ranges over *all* functions. An interesting question that naturally arises, is whether we can express the finiteness of the *domain* of a heap. We conjecture that this is not the case.

It is also worthwhile to observe that the class of models **Full** of SL *without* separating implication is already not compact. Let $\Box p$ abbreviate $\neg(\top * \neg p)$, which states that p holds in every sub-heap (of the given heap). Then

$$\Box(\forall x, y(x \not\hookrightarrow)y \vee \exists x((x \hookrightarrow -) \wedge \forall y((y \hookrightarrow -) \to (y \not\hookrightarrow x)))$$

Fig. 3 Categorical axiomatization of the points-to relation

$$\begin{array}{l}\mathbf{A1}\ \forall x, y.\ ((x \hookrightarrow y) * \mathbf{true}) \rightarrow (x \hookrightarrow y)\\ \mathbf{A2}\ \forall x, y((x \hookrightarrow y) \rightarrow \neg((x \not\hookrightarrow y) * (x \not\hookrightarrow y)))\\ \mathbf{A3}\ \forall x, y, z\neg((x \hookrightarrow y) * (x \hookrightarrow z))\\ \mathbf{A4}\ \forall x, y, z(((x \hookrightarrow y) \wedge (x \hookrightarrow z)) \rightarrow y = z)\end{array}$$

expresses that the points-to relation $\hookrightarrow$ is *well-founded*. Note that a relation is well-founded iff *every* (non-empty) sub-relation has a minimal element (with respect to that sub-relation). Every finite subset of the infinite set of formulas $(x_{n+1} \hookrightarrow x_n)$, $n \in \mathbb{N}$, plus the above formula, has a model. But clearly, the entire set does not.

4.1 Separation algebras

Using separation algebras, an SL model provides an interpretation of separating conjunction in terms of the (partial) monoid operation $\circ$ of a separation algebra, and the abstract elements of the separation algebra are taken as heaps (e.g., sets of location/value pairs). Interestingly, we can define separating conjunction in SL as the set-theoretic union of disjoint heaps by the axioms in Figure 3.

For any element h of a given separation algebra let $\vec{h}$ denote the associated heap. Axiom **A1** states that for any element h of the separation algebra whenever $h = h_1 \circ h_2$ then $\vec{h_1} \cup \vec{h_2} \subseteq \vec{h}$ (by commutativity of $\circ$). Axiom **A2** states that whenever $h = h_1 \circ h_2$ then $\vec{h} \subseteq \vec{h_1} \cup \vec{h_2}$. Axiom **A3** states that whenever $h = h_1 \circ h_2$ then $dom(\vec{h_1}) \cap dom(\vec{h_2}) = \emptyset$, where *dom* applied to a (binary) relation gives its domain, the set of first elements of its pairs. Axiom **A4** states that the points-to relation denotes the graph of a (partial) function. So we have that $h = h_1 \circ h_2$ implies that $dom(\vec{h_1}) \cap dom(\vec{h_2}) = \emptyset$ and $\vec{h} = \vec{h_1} \cup \vec{h_2}$, for any model of these axioms. In other words, these axioms form a *categorical* theory in the sense that any model is *isomorphic* to the standard model that consists of a monoid of heaps as partial functions with the (partial) monoid operation of disjoint union.

This justifies our focus on how one can reason directly about the standard set-theoretic interpretation of the separating connectives in terms of the points-to relation. Further, we conclude that the class of SL models based on separation algebras which satisfy the axioms **A1**,**A2**,**A3**,**A4** and which allow for infinite heaps coincides with the class **General**.

5 Conclusion and future work

We have introduced in this paper a natural deduction proof system for classical separation logic that axiomatizes the basic, set-theoretic semantics of the separating connectives in terms of a first-order hybrid extension. This extension allows to reason about the separating connectives in the context of a description of the current heap in separation logic itself.

Further development focuses on an inter-active proof assistant and its application to program correctness, using suitable extensions of Reynolds original programming logic as described in [26]. In particular, we want to prove the correctness of pointer-programs manipulating (recursive) page tables.

Other future work is the investigation of a sound, complete and effective proof system for intuitionistic separation logic.

Acknowledgements We thank the anonymous reviewers for their constructive comments. We also thank Robbert Krebbers for the interesting discussions on Iris.

Author Contributions The two authors worked closely together in the development of the presented proof system.

Data Availibility No datasets were generated or analysed during the current study.

Statements and Declarations

Competing interests The authors declare to have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper. The results/data/figures in this manuscript have not been published elsewhere, nor are they under consideration by another publisher. The manuscript does not contain any material from third parties. This manuscript does not report data generation or analysis. The authors Frank S. de Boer and Hans-Dieter A. Hiep contributed equally to this work. There is no funding to declare.

References

1. Berdine, J., Cook, B., Ishtiaq, S.: SLAyer: Memory safety for systems-level code. In Computer Aided Verification: 23rd International Conference, CAV 2011, volume 6806 of LNCS, pages 178–183. Springer, (2011)
2. Bornat, R., Calcagno, C., O'Hearn, P., Parkinson, M.: Permission accounting in separation logic. SIGPLAN Notices **40**(1), 259–270 (2005)
3. Brochenin, R., Demri, S., Lozes, E.: On the almighty wand. Inf. Comput. **211**, 106–137 (2012)
4. Brookes, S., O'Hearn, P.W.: Concurrent separation logic. ACM SIGLOG News **3**(3), 47–65 (2016)
5. Brotherston, J., Villard, J: Parametric completeness for separation theories. In Suresh Jagannathan and Peter Sewell, editors, The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014, pages 453–464. ACM, (2014)
6. de Boer, F. S., Hiep, H.-D. A., de Gouw, S.: The logic of separation logic: Models and proofs. In Automated Reasoning with Analytic Tableaux and Related Methods - 32nd International Conference TABLEAUX, volume 14278 of LNCS, pages 407–426. Springer, (2023)
7. Demri, S., Lozes, É., Mansutti, A.: A complete axiomatisation for quantifier-free separation logic. Logical Methods in Computer Science, 17(3), (2021)
8. Galmiche, D., Méry, D.: Tableaux and resource graphs for separation logic. J. Log. Comput. **20**(1), 189–231 (2010)
9. George, A.S.: When trust fails: Examining systemic risk in the digital economy from the 2024 CrowdStrike outage. Partners Universal Multidisciplinary Research Journal **1**(2), 134–152 (2024)
10. Gries, D., Schneider, F.B.: Avoiding the undefined by underspecification. In Computer Science Today: Recent Trends and Developments, volume 1000 of LNCS, pages 366–373. Springer, (1995)
11. Hance, T., Howell, J., Padon, O., Parno, B.: Leaf: Modularity for temporary sharing in separation logic. Proceedings of the ACM on Programming Languages, 7(OOPSLA2), (2023)
12. Henkin, L: Completeness in the theory of types. The Journal of Symbolic Logic, 15(2), (1950)
13. Henkin, L.: The completeness of the first-order functional calculus. J. Symb. Log. **14**(3), 159–166 (1949)
14. Hiep, H.-D.A.: The logic of separation logic: First-order hybrid separation logic (Coq artifact) (January 2025). https://doi.org/10.5281/zenodo.14635180

15. Hou, Z., Tiu, A.: Completeness for a first-order abstract separation logic. In Programming Languages and Systems - 14th Asian Symposium, APLAS 2016, volume 10017 of LNCS, pages 444–463, (2016)
16. Huet, G. P., Herbelin, H.: 30 years of research and development around Coq. In The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), pages 249–250. ACM, (2014)
17. Jung, R., Krebbers, R., Jourdan, J.-H., Bizjak, A., Birkedal, L., Dreyer, D.: Iris from the ground up: A modular foundation for higher-order concurrent separation logic. Journal of Functional Programming, 28, (2018)
18. Krishnaswami, N. R.: A modal sequent calculus for propositional separation logic, (2008). (Draft)
19. Luca Aceto. Interview with Stephen Brookes and Peter W. O'Hearn, recipients of the: Gödel prize. Bulletin of EATCS **1**(119), 2016 (2016)
20. Mansutti, A.: Reasoning with Separation Logics. PhD thesis, École Normale Supérieure, France, (2020)
21. Mugu, S R., Zhang, B., Kolla, H., Balaji, S.R.A., Ranganathan, P.: Lessons from the CrowdStrike incident: Assessing endpoint security vulnerabilities and implications. In 2024 Cyber Awareness and Research Symposium (CARS), 1–10, (2024)
22. O'Hearn, P.: Separation logic. Communications of the ACM **62**(2), 86–95 (2019)
23. Pym, D.J.: The semantics and proof theory of the logic of bunched implications, volume 26 of Applied Logic Series. Springer, (2002)
24. Reynolds, A., Iosif, R., Serban, C., King,T.: A decision procedure for separation logic in SMT. In International Symposium on Automated Technology for Verification and Analysis, volume 9938 of LNCS, pages 244–261. Springer, (2016)
25. Reynolds, J.C.: An overview of separation logic. In Verified Software: Theories, Tools, Experiments, First IFIP TC 2/WG 2.3 Conference, VSTTE 2005, volume 4171 of LNCS, pages 460–469. Springer, (2005)
26. Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In 17th IEEE Symposium on Logic in Computer Science (LICS 2002), pages 55–74. IEEE Computer Society, (2002)
27. Torben Braüner. Hybrid Logic. In Edward N. Zalta and Uri Nodelman, editors, The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, Summer 2024 edition, 2024
28. van Starkenburg, B., Basold, H., Ford, C.: Separation logic of generic resources via sheafeology. arXiv preprint arXiv:2508.01866, (2025)