Internal Calculi for Separation Logics

Stéphane Demri
LSV, CNRS, ENS Paris-Saclay, Université Paris-Saclay, France

Etienne Lozes
Université Côte d'Azur, CNRS, I3S, France

Alessio Mansutti
LSV, CNRS, ENS Paris-Saclay, Université Paris-Saclay, France

Abstract

We present a general approach to axiomatise separation logics with heaplet semantics with no external features such as nominals/labels. To start with, we design the first (internal) Hilbert-style axiomatisation for the quantifier-free separation logic $\mathsf{SL}(*, -\!\!*)$. We instantiate the method by introducing a new separation logic with essential features: it is equipped with the separating conjunction, the predicate `ls`, and a natural guarded form of first-order quantification. We apply our approach for its axiomatisation. As a by-product of our method, we also establish the exact expressive power of this new logic and we show PSpace-completeness of its satisfiability problem.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Separation logic, internal calculus, adjunct/quantifier elimination

Digital Object Identifier 10.4230/LIPIcs.CSL.2020.19

Acknowledgements We would like to thank the anonymous reviewers for their suggestions and remarks that help us to improve the quality of this paper.

1 Introduction

The virtue of axiomatising program logics. Designing a Hilbert-style axiomatisation for your favourite logic is usually quite challenging. This does not lead necessarily to optimal decision procedures, but the completeness proof usually provides essential insights to better understand the logic at hand. That is why many logics related to program verification have been axiomatised, often requiring non-trivial completeness proofs. By way of example, there exist axiomatisations for the linear-time μ-calculus [28, 19], the modal μ-calculus [39] or for the alternating-time temporal logic ATL [23]. Concerning the separation logics that extend Hoare-Floyd logic to verify programs with mutable data structures (see e.g. [34, 38, 27, 33, 37]), a Hilbert-style axiomatisation of Boolean BI has been introduced in [21], but remained at the abstract level of Boolean BI. More recently, HyBBI [8], a hybrid version of Boolean BI has been introduced in order to axiomatise various classes of separation logics; HyBBI naturally considers classes of abstract models (typically preordered partial monoids) but it does not fit exactly the heaplet semantics of separation logics. Furthermore, the addition of nominals (in the sense of hybrid modal logics, see e.g. [1]) extends substantially the object language. Other frameworks to axiomatise classes of abstract separation logics can be found in [18] and in [25], respectively with labelled tableaux calculi and with sequent-style proof systems.

Our motivations. Since the birth of separation logics, there has been a lot of interest in the study of decidability and computational complexity issues, see e.g. [3, 10, 11, 7, 15, 32], and comparatively a bit less attention to the design of proof systems, and even less with the puristic approach that consists in discarding any external feature such as nominals or labels in the calculi. The well-known advantages of such an approach include an exhaustive understanding of the expressive power of the logic and discarding the use of any external

28th EACSL Annual Conference on Computer Science Logic (CSL 2020).
Editors: Maribel Fernández and Anca Muscholl; Article No. 19; pp. 19:1–19:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

artifact referring to semantical objects. For instance, a complete tableaux calculus with labels for quantifier-free separation logic is designed in [22] –with an extension of the calculus to handle quantifiers, whereas Hilbert-style calculi for abstract separation logics with nominals are defined in [8] (see also in [26] a proof system for a first-order abstract separation logic with an abstracted version of the points-to predicate). Similarly, display calculi for bunched logics are provided in [5] and such calculi extend Gentzen-style proof systems by allowing new structural connectives. In this paper, we advocate a puristic approach and aim at designing Hilbert-style proof systems for quantifier-free separation logic $\mathsf{SL}(*, \mathrel{-\!\!*})$ (which includes the separating conjunction $*$ and implication $\mathrel{-\!\!*}$, as well as all Boolean connectives) and more generally for other separation logics, while remaining within the very logical language. Consequently, in this work we only focus on axiomatising the separation logics, and we have no claim for practical applications in the field of program verification. Aiming at internal calculi is a non-trivial task as the general frameworks for abstract separation logics make use of labels, see e.g. [18, 25]. We cannot fully rely on label-free calculi for BI, see e.g. [36, 21], as separation logics are usually understood as Boolean BI interpreted on models of heap memory and therefore require calculi that handle specifically the stack-and-heap models. Finally, we know translations from separation logics into logics/theories, see e.g. [9, 35, 4], but completeness cannot always be inherited by sublogics as the proof system should only use the sublogic and therefore their axiomatisation may lead to different methods.

Our contribution. Though our initial motivation is to design an internal Hilbert-style axiomatisation for $\mathsf{SL}(*, \mathrel{-\!\!*})$, we go beyond this, and we propose a method to axiomatise other separation logics assuming that key properties are satisfied. Hence, we consider a broader perspective and we use our approach on two separation logics: quantifier-free separation logic and a new separation logic that admits a form of guarded first-order quantification. Our results are not limited to (internal) axiomatisation, as we provide a complexity analysis based on the properties of the derivations in the proof system. Let us be a bit more precise.

In Section 3, we provide the first Hilbert-style proof system for $\mathsf{SL}(*, \mathrel{-\!\!*})$ that uses axiom schemas and rules involving only formulae of this logic. Each formula of $\mathsf{SL}(*, \mathrel{-\!\!*})$ is equivalent to a Boolean combination of *core formulae*: simple formulae of the logic expressing elementary properties about the models [30]. Though core formulae (also called *test formulae*) have been handy in several occasions for establishing complexity results for separation logics, see e.g. [14, 15, 20], in the paper, these formulae are instrumental for the axiomatisation. Indeed, we distinguish the axiomatisation of Boolean combinations of core formulae from the transformation of formulae into such Boolean combinations. Thus, we show how to introduce axioms to transform every formula into a Boolean combination of core formulae, together with axioms to deal with these simple formulae. Schematically, for a valid formula φ, we conclude $\vdash \varphi$ from $\vdash \varphi'$ and $\vdash \varphi' \Leftrightarrow \varphi$, where φ' is a Boolean combination of core formulae. Another difficulty arises as we have to design an axiomatisation for such Boolean combinations. So, the calculus is divided in three parts: the axiomatisation of Boolean combinations of core formulae, axioms and inference rules to simulate a bottom-up elimination of separating connectives, and finally axioms and inference rules from propositional calculus and Boolean BI. Such an approach that consists in first axiomatising a syntactic fragment of the whole logic (in our case, the core formulae), is best described in [19] (see also [39, 40, 31, 13]).

In Section 4, our intention is to add standard features to the logic such as first-order quantification and inductive predicates, and to apply our method for axiomatisation. As $\mathsf{SL}(*, \mathrel{-\!\!*}, \mathtt{ls})$ (i.e. $\mathsf{SL}(*, \mathrel{-\!\!*})$ enriched with the predicate $\mathtt{ls}$) is already non-finitely axiomatisable [16], we need to fine-tune the logical formalism. That is why, we introduce a new separation logic $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ that admits the separating conjunction $*$ (no $\mathrel{-\!\!*}$) and a guarded form of first-order quantification. In the formula $\exists \mathtt{z}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle \varphi$, the variable $\mathtt{z}$ is existentially quantified over the set of locations in the minimal non-empty path from $\mathtt{x}$ to $\mathtt{y}$, if any. The logic $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ contains the symbolic heap fragment [2, 11] but also richer logics such as $\mathsf{SL}(*, \mathtt{reach}^+)$ from [15]. Hence, the logic $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ captures the list segment predicate $\mathtt{ls}$ but also allows us to quantify in a guarded form over locations in a minimal path, which makes it a promising language. We provide an internal Hilbert-style axiomatisation for $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$, illustrating the flexibility of our method. It requires the design of an adequate family of core formulae that captures $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$. The axiomatisation of Boolean combinations of core formulae reveals to be challenging, and the elimination of guarded quantification or separating conjunction happens also to require complex developments. We analyse the derivations from the calculus to establish a small model property for the logic and, together with a symbolic model-checking algorithm, prove that the satisfiability problem for $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ is in PSPACE.

2 Preliminaries

Quantifier-free separation logic $\mathsf{SL}(*, \mathrel{-\!\!*})$. We present the quantifier-free separation logic $\mathsf{SL}(*, \mathrel{-\!\!*})$, that includes standard features such as the separating conjunction $*$ and the separating implication $\mathrel{-\!\!*}$. Let $\mathsf{VAR} = \{\mathtt{x}, \mathtt{y}, \ldots\}$ be a countably infinite set of *program variables*. The formulae φ of $\mathsf{SL}(*, \mathrel{-\!\!*})$ and its atomic formulae π are built from the grammars below (where $\mathtt{x}, \mathtt{y} \in \mathsf{VAR}$ and the connectives $\Rightarrow$, $\Leftrightarrow$ and $\vee$ are defined as usually).

$$\pi ::= \mathtt{x} = \mathtt{y} \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \mathtt{emp} \qquad \varphi ::= \pi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi * \varphi \mid \varphi \mathrel{-\!\!*} \varphi.$$

In the heaplet semantics, the formulae of $\mathsf{SL}(*, \mathrel{-\!\!*})$ are interpreted on *memory states* that are pairs (s, h) where $s : \mathsf{VAR} \to \mathsf{LOC}$ is a variable valuation (the *store*) from the set of program variables to a countably infinite set of *locations* $\mathsf{LOC} = \{\ell_0, \ell_1, \ell_2, \ldots\}$ whereas $h : \mathsf{LOC} \to_{\mathsf{fin}} \mathsf{LOC}$ is a partial function with finite domain (the *heap*). We write $\mathrm{dom}(h)$ to denote its domain and $\mathrm{ran}(h)$ to denote its range. A *memory cell* of h is understood as a pair of locations (ℓ, ℓ') such that $\ell \in \mathrm{dom}(h)$ and $\ell' = h(\ell)$. As usual, the heaps h_1 and h_2 are said to be *disjoint*, written $h_1 \perp h_2$, if $\mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) = \emptyset$; when this holds, we write $h_1 + h_2$ to denote the heap corresponding to the disjoint union of the graphs of h_1 and h_2, hence $\mathrm{dom}(h_1 + h_2) = \mathrm{dom}(h_1) \uplus \mathrm{dom}(h_2)$. Moreover, we write $h' \sqsubseteq h$ to denote that $\mathrm{dom}(h') \subseteq \mathrm{dom}(h)$ and for all locations $\ell \in \mathrm{dom}(h')$, we have $h'(\ell) = h(\ell)$. Given a heap h, we define a family of $(h^\delta)_{\delta \in \mathbb{N}}$ of partial functions such that h^0 is the identity function on LOC, $h^1 = h$ and for all $\delta \geq 2$ and $\ell \in \mathsf{LOC}$, we have $h^\delta(\ell) \stackrel{\text{def}}{=} h(h^{\delta-1}(\ell))$, assuming that $h^{\delta-1}(\ell)$ is defined and belongs to $\mathrm{dom}(h)$, otherwise $h^\delta(\ell)$ is undefined. The satisfaction relation $\models$ is defined as follows (omitting standard clauses for $\neg$, $\wedge$):

$$(s,h) \models \mathtt{x} = \mathtt{y} \stackrel{\text{def}}{\Leftrightarrow} s(\mathtt{x}) = s(\mathtt{y}) \qquad (s,h) \models \mathtt{emp} \stackrel{\text{def}}{\Leftrightarrow} \mathrm{dom}(h) = \emptyset$$
$$(s,h) \models \mathtt{x} \hookrightarrow \mathtt{y} \stackrel{\text{def}}{\Leftrightarrow} s(\mathtt{x}) \in \mathrm{dom}(h) \text{ and } h(s(\mathtt{x})) = s(\mathtt{y})$$
$$(s,h) \models \varphi_1 * \varphi_2 \stackrel{\text{def}}{\Leftrightarrow} \exists h_1, h_2.\ h_1 \perp h_2,\ (h_1 + h_2) = h,\ (s, h_1) \models \varphi_1 \text{ and } (s, h_2) \models \varphi_2$$
$$(s,h) \models \varphi_1 \mathrel{-\!\!*} \varphi_2 \stackrel{\text{def}}{\Leftrightarrow} \forall h_1.\ (h_1 \perp h \text{ and } (s, h_1) \models \varphi_1) \text{ implies } (s, h + h_1) \models \varphi_2.$$

We denote with $\perp$ the contradiction $\mathtt{x} \neq \mathtt{x}$, and with $\top$ its negation $\neg\perp$. The septraction operator $\mathrel{-\!\!\circledast}$ (kind of dual of $\mathrel{-\!\!*}$), defined by $\varphi \mathrel{-\!\!\circledast} \psi \stackrel{\text{def}}{=} \neg(\varphi \mathrel{-\!\!*} \neg\psi)$, has the following semantics:

$$(s,h) \models \varphi \mathrel{-\!\!\circledast} \psi \Leftrightarrow \text{there is a heap } h' \text{ such that } h \perp h',\ (s, h') \models \varphi, \text{ and } (s, h + h') \models \psi.$$

Moreover, we introduce the following (important) shortcuts:

- $\mathtt{alloc}(\mathtt{x})$ which is satisfied by (s, h) iff $s(\mathtt{x}) \in \mathrm{dom}(h)$. It is defined as $(\mathtt{x} \hookrightarrow \mathtt{x}) \mathrel{-\!\!*} \perp$.

- $\mathtt{size} \geq \beta$ which is satisfied by (s,h) iff $\mathrm{card}(\mathrm{dom}(h)) \geq \beta$, where $\beta \in \mathbb{N}$ and $\mathrm{card}(X)$ denotes the cardinality of the set X. This shortcut is inductively defined as $\mathtt{size} \geq 0 \stackrel{\text{def}}{=} \top$, $\mathtt{size} \geq 1 \stackrel{\text{def}}{=} \neg\mathtt{emp}$ and, for each $\beta \in \mathbb{N}$, $\mathtt{size} \geq \beta{+}2 \stackrel{\text{def}}{=} \neg\mathtt{emp} * \mathtt{size} \geq \beta{+}1$.

We use $\mathtt{size}{=}\beta$ as a shorthand for $\mathtt{size}{\geq}\beta \wedge \neg\mathtt{size}{\geq}\beta{+}1$. A formula φ is *valid* if $(s,h) \models \varphi$ for all (s,h) (and we write $\models \varphi$). For a complete description of separation logic, see e.g. [38].

Hilbert-style proof systems. A *Hilbert-style proof system* $\mathcal{H}$ is defined as a set of *derivation step schemata* $((\Phi_1,\ldots,\Phi_n),\Psi)$ with $n \geq 0$, where $\Phi_1,\ldots,\Phi_n,\Psi$ are *formula schemata*. When $n \geq 1$, $((\Phi_1,\ldots,\Phi_n),\Psi)$ is called an *inference rule*, otherwise it is an *axiom*. As usual, formula schemata generalise the notion of formulae by allowing metavariables for formulae (typically φ,ψ,χ), for program variables (typically $\mathtt{x},\mathtt{y},\mathtt{z}$) or for any type of syntactic objects in formulae, depending on the context. The set of formulae *derivable* from $\mathcal{H}$ is the least set S such that for all $((\Phi_1,\ldots,\Phi_n),\Psi) \in \mathcal{H}$ and for all substitutions σ such that $\Phi_1\sigma,\ldots,\Phi_n\sigma \in S$, $\Psi\sigma \in S$. We write $\vdash_{\mathcal{H}} \varphi$ if φ is derivable from $\mathcal{H}$. A proof system $\mathcal{H}$ is *sound* if all derivable formulae are valid. $\mathcal{H}$ is *complete* if all valid formulae are derivable. $\mathcal{H}$ is *strongly complete* iff for all sets of formulae Γ and formulae φ, we have $\Gamma \models \varphi$ (semantical entailment) iff $\vdash_{\mathcal{H}\cup\Gamma} \varphi$.

Interestingly enough, there is no strongly complete proof system for separation logic, as strong completeness implies compactness and separation logic is not compact. Indeed, $\{\mathtt{size} \geq \beta \mid \beta \in \mathbb{N}\}$ is unsatisfiable, as heaps have finite domains, but all finite subsets of it are satisfiable. Even for the weaker notion of completeness, deriving an Hilbert-style axiomatisation for $\mathsf{SL}(*,\mathrel{-\!\!*})$ remains challenging. Indeed, the satisfiability problem for $\mathsf{SL}(*,\mathrel{-\!\!*})$ reduces to its validity problem, making $\mathsf{SL}(*,\mathrel{-\!\!*})$ an unusual logic from a proof-theoretical point of view. Let us develop a bit further this point. Let φ be a formula with program variables in $\mathtt{X} \subseteq_{\mathsf{fin}} \mathsf{VAR}$, and let $\approx$ be an equivalence relation on $\mathtt{X}$. The formula $\psi_{\approx} \stackrel{\text{def}}{=} (\mathtt{emp} \wedge \bigwedge_{\mathtt{x}\approx\mathtt{y}} \mathtt{x} = \mathtt{y} \wedge \bigwedge_{\mathtt{x}\not\approx\mathtt{y}} \mathtt{x} \neq \mathtt{y}) \Rightarrow (\varphi \mathrel{-\!\circledast} \top)$ can be shown to be valid iff for every store s agreeing on $\approx$, there is a heap h such that $(s,h) \models \varphi$. It is known that for all stores s,s' agreeing on $\approx$, and every heap h, (s,h) and (s',h) satisfy the same set of formulae having variables from $\mathtt{X}$. Since the antecedent of $\psi_{\approx}$ is satisfiable, we conclude that $\psi_{\approx}$ is valid iff there are a store s agreeing on $\approx$ and a heap h such that $(s,h) \models \varphi$. To check whether φ is satisfiable, it is sufficient to find an equivalence relation $\approx$ on $\mathtt{X}$ such that $\psi_{\approx}$ is valid. As the number of equivalence relations on $\mathtt{X}$ is finite, we obtain a Turing reduction from satisfiability to validity. Consequently, it is not possible to define sound and complete axiom systems for any extension of $\mathsf{SL}(*,\mathrel{-\!\!*})$ admitting an undecidable validity problem (as long as there is a reduction from satisfiability to validity, as above). A good example is $\mathsf{SL}(*,\mathrel{-\!\!*},\mathtt{ls})$ [16] (extension of $\mathsf{SL}(*,\mathrel{-\!\!*})$ with $\mathtt{ls}$). Indeed, in order to obtain a sound and complete axiom system, the validity problem has to be recursively enumerable (r.e.). However, this would imply that the satisfiability problem is also r.e.. As φ is not valid iff $\neg\varphi$ is satisfiable, we then conclude that the set of valid formulae is recursive, hence decidable, a contradiction.

It is worth also noting that quantifier-free $\mathsf{SL}(*,\mathrel{-\!\!*})$ axiomatised below admits a PSPACE-complete validity problem, see e.g. [10], and should not be confused with propositional separation logic with the stack-heap models shown undecidable in [6, Corollary 5.1] (see also [12]), in which there are propositional variables interpreted by sets of memory states.

3 Hilbert-style proof system for $\mathsf{SL}(*,\mathrel{-\!\!*})$

We define a proof system for $\mathsf{SL}(*,\mathrel{-\!\!*})$, namely $\mathcal{H}_{\mathsf{C}}(*,\mathrel{-\!\!*})$, by relying on its *core formulae*: simple $\mathsf{SL}(*,\mathrel{-\!\!*})$ formulae capturing essential properties of the models, see e.g. [29, 41]. It is known that every $\mathsf{SL}(*,\mathrel{-\!\!*})$ formula is logically equivalent to a Boolean combination of

(System 1) $\mathcal{H}_{\mathsf{C}}$: Axioms for Boolean combinations of core formulae

$(\mathbf{A^{C}_{1}})$ $\mathtt{x} = \mathtt{x}$	$(\mathbf{A^{C}_{4}})$ $\mathtt{x} \hookrightarrow \mathtt{y} \wedge \mathtt{x} \hookrightarrow \mathtt{z} \Rightarrow \mathtt{y} = \mathtt{z}$
$(\mathbf{A^{C}_{2}})$ $\varphi \wedge \mathtt{x} = \mathtt{y} \Rightarrow \varphi[\mathtt{y}{\leftarrow}\mathtt{x}]$	$(\mathbf{I^{C}_{5}})$ $\mathtt{size} \geq \beta{+}1 \Rightarrow \mathtt{size} \geq \beta$
$(\mathbf{A^{C}_{3}})$ $\mathtt{x} \hookrightarrow \mathtt{y} \Rightarrow \mathtt{alloc}(\mathtt{x})$	$(\mathbf{I^{C}_{6}})$ $\bigwedge_{\mathtt{x}\in\mathtt{X}}(\mathtt{alloc}(\mathtt{x}) \wedge \bigwedge_{\mathtt{y}\in\mathtt{X}\setminus\{\mathtt{x}\}} \mathtt{x} \neq \mathtt{y}) \Rightarrow \mathtt{size} \geq \mathrm{card}(\mathtt{X})$

(System 2) Axioms and inference rule for the separating conjunction

$(\mathbf{A^{*}_{7}})$ $(\varphi * \psi) \Leftrightarrow (\psi * \varphi)$	$(\mathbf{A^{*}_{14}})$ $\mathtt{e} * \top \Rightarrow \mathtt{e}$ $\blacktriangleleft$e is $\neg\mathtt{emp}$, $\mathtt{x} = \mathtt{y}$, $\mathtt{x} \neq \mathtt{y}$ or $\mathtt{x} \hookrightarrow \mathtt{y}$]
$(\mathbf{A^{*}_{8}})$ $(\varphi * \psi) * \chi \Leftrightarrow \varphi * (\psi * \chi)$	$(\mathbf{A^{*}_{15}})$ $\neg\mathtt{alloc}(\mathtt{x}) * \neg\mathtt{alloc}(\mathtt{x}) \Rightarrow \neg\mathtt{alloc}(\mathtt{x})$
$(\mathbf{I^{*}_{9}})$ $(\varphi \vee \psi) * \chi \Rightarrow (\varphi * \chi) \vee (\psi * \chi)$	$(\mathbf{A^{*}_{16}})$ $(\mathtt{alloc}(\mathtt{x}) \wedge \neg\mathtt{x} \hookrightarrow \mathtt{y}) * \top \Rightarrow \neg\mathtt{x} \hookrightarrow \mathtt{y}$
$(\mathbf{I^{*}_{10}})$ $(\bot * \varphi) \Leftrightarrow \bot$	$(\mathbf{A^{*}_{17}})$ $\mathtt{alloc}(\mathtt{x}) \Rightarrow (\mathtt{alloc}(\mathtt{x}) \wedge \mathtt{size} = 1) * \top$
$(\mathbf{A^{*}_{11}})$ $\varphi \Leftrightarrow \varphi * \mathtt{emp}$	$(\mathbf{A^{*}_{18}})$ $\neg\mathtt{emp} \Rightarrow \mathtt{size} = 1 * \top$
$(\mathbf{I^{*}_{12}})$ $\mathtt{alloc}(\mathtt{x}) * \top \Rightarrow \mathtt{alloc}(\mathtt{x})$	$(\mathbf{A^{*}_{19}})$ $\neg\mathtt{size} \geq \beta_1 * \neg\mathtt{size} \geq \beta_2 \Rightarrow \neg\mathtt{size} \geq \beta_1{+}\beta_2{\dot{-}}1$
$(\mathbf{I^{*}_{13}})$ $(\mathtt{alloc}(\mathtt{x}) * \mathtt{alloc}(\mathtt{x})) \Leftrightarrow \bot$	$(\mathbf{A^{*}_{20}})$ $\mathtt{alloc}(\mathtt{x}) \wedge \mathtt{alloc}(\mathtt{y}) \wedge \mathtt{x} \neq \mathtt{y} \Rightarrow \mathtt{size} \geq 2$
$*$-**Intro**: $\dfrac{\varphi \Rightarrow \chi}{\varphi * \psi \Rightarrow \chi * \psi}$	where $a \dot{-} b = a - b$ if $a \geq b$, 0 otherwise.

(System 3) Axioms and inference rules for the separating implication

$(\mathbf{A^{-\!*}_{21}})$ $(\mathtt{size} = 1 \wedge \bigwedge_{\mathtt{x}\in\mathtt{X}} \neg\mathtt{alloc}(\mathtt{x})) \mathbin{-\!\circledast} \top$ $\blacktriangleleft$X $\subseteq_{\mathsf{fin}}$ VAR]	$*$-**Adj**: $\dfrac{\varphi * \psi \Rightarrow \chi}{\varphi \Rightarrow (\psi \mathbin{-\!*} \chi)}$ $-\!*$-**Adj**: $\dfrac{\varphi \Rightarrow (\psi \mathbin{-\!*} \chi)}{\varphi * \psi \Rightarrow \chi}$
$(\mathbf{A^{-\!*}_{22}})$ $\neg\mathtt{alloc}(\mathtt{x}) \Rightarrow ((\mathtt{x} \hookrightarrow \mathtt{y} \wedge \mathtt{size} = 1) \mathbin{-\!\circledast} \top)$	
$(\mathbf{A^{-\!*}_{23}})$ $\neg\mathtt{alloc}(\mathtt{x}) \Rightarrow ((\mathtt{alloc}(\mathtt{x}) \wedge \mathtt{size} = 1 \wedge \bigwedge_{\mathtt{y}\in\mathtt{X}} \neg\mathtt{x} \hookrightarrow \mathtt{y}) \mathbin{-\!\circledast} \top)$ $\blacktriangleleft$X $\subseteq_{\mathsf{fin}}$ VAR]	

1	$\mathtt{emp} \Rightarrow \neg\mathtt{size} \geq 1$	$(\neg\neg\mathbf{E})$ and def. of $\mathtt{size} \geq 1$
2	$\mathtt{alloc}(\mathtt{x}) \wedge \mathtt{size} = 1 \Rightarrow \neg\mathtt{size} \geq 2$	$(\wedge\mathbf{Er})$
3	$\mathtt{emp} * (\mathtt{alloc}(\mathtt{x}) \wedge \mathtt{size} = 1) \Rightarrow \neg\mathtt{size} \geq 1 * \neg\mathtt{size} \geq 2$	$*$-**Ilr**, 1, 2
4	$\neg\mathtt{size} \geq 1 * \neg\mathtt{size} \geq 2 \Rightarrow \neg\mathtt{size} \geq 2$	$(\mathbf{A^{*}_{19}})$
5	$\big(\mathtt{emp} * (\mathtt{alloc}(\mathtt{x}) \wedge \mathtt{size} = 1)\big) \Rightarrow \neg\mathtt{size} \geq 2$	$\Rightarrow$-**Tr**, 3, 4
6	$\mathtt{emp} \Rightarrow \big((\mathtt{alloc}(\mathtt{x}) \wedge \mathtt{size} = 1) \mathbin{-\!*} \neg\mathtt{size} \geq 2\big)$	$*$-**Adj** rule, 5

Figure 1 Proof of $\mathtt{emp} \Rightarrow \big((\mathtt{alloc}(\mathtt{x}) \wedge \mathtt{size} = 1) \mathbin{-\!*} \neg\mathtt{size} \geq 2\big)$.

core formulae [29]. However, as every core formula is an $\mathsf{SL}(*, -\!*)$ formula, we stay in the original language and we can derive an axiomatisation of $\mathsf{SL}(*, -\!*)$ by extending the axiom system of propositional calculus with three sets of axioms and inference rules: the axioms and inference rules of the propositional logic of core formulae (System 1), the axioms and inference rules witnessing that every formula of the form $\varphi_1 * \varphi_2$, where φ_1, φ_2 are Boolean combinations of core formulae is logically equivalent to a Boolean combination of core formulae (System 2), and the axioms and inference rules to eliminate formulae whose outermost connective is the separating implication $-\!*$ (System 3). The *core formulae* are expressions of the form $\mathtt{x} = \mathtt{y}$, $\mathtt{alloc}(\mathtt{x})$, $\mathtt{x} \hookrightarrow \mathtt{y}$ and $\mathtt{size} \geq \beta$, where $\mathtt{x}, \mathtt{y} \in \mathsf{VAR}$ and $\beta \in \mathbb{N}$. As previously shown, these formulae are from $\mathsf{SL}(*, -\!*)$ and are used in the axiom system as abbreviations. Given $\mathtt{X} \subseteq_{\mathsf{fin}} \mathsf{VAR}$ and $\alpha \in \mathbb{N}$, we define $\mathsf{Core}(\mathtt{X}, \alpha)$ as the set $\{\mathtt{x} = \mathtt{y},\ \mathtt{alloc}(\mathtt{x}),\ \mathtt{x} \hookrightarrow \mathtt{y},\ \mathtt{size} \geq \beta \mid \mathtt{x}, \mathtt{y} \in \mathtt{X},\ \beta \in [0, \alpha]\}$. $\mathsf{Bool}(\mathsf{Core}(\mathtt{X}, \alpha))$ is the set of Boolean combinations of formulae from $\mathsf{Core}(\mathtt{X}, \alpha)$, whereas $\mathsf{Conj}(\mathsf{Core}(\mathtt{X}, \alpha))$ is the set of conjunctions of literals built upon $\mathsf{Core}(\mathtt{X}, \alpha)$ (a literal being a core formula or its negation). Given $\varphi = L_1 \wedge \cdots \wedge L_n \in \mathsf{Conj}(\mathsf{Core}(\mathtt{X}, \alpha))$, every L_i being a literal, $\mathsf{Lt}(\varphi) \stackrel{\text{def}}{=} \{L_1, \ldots, L_n\}$. $\psi \subseteq_{\mathsf{Lt}} \varphi$ stands for $\mathsf{Lt}(\psi) \subseteq \mathsf{Lt}(\varphi)$. We write $\chi \subseteq_{\mathsf{Lt}} \{\varphi \mid \psi\}$, $\{\varphi \mid \psi\} \subseteq_{\mathsf{Lt}} \chi$ and $\chi \subseteq_{\mathsf{Lt}} \{\varphi \,;\, \psi\}$ for "$\chi \subseteq_{\mathsf{Lt}} \varphi$ or $\chi \subseteq_{\mathsf{Lt}} \psi$", "$\varphi \subseteq_{\mathsf{Lt}} \chi$ or $\psi \subseteq_{\mathsf{Lt}} \chi$", and "$\chi \subseteq_{\mathsf{Lt}} \varphi$ and $\chi \subseteq_{\mathsf{Lt}} \psi$", respectively.

Example. To show the flavour of the axioms and the rules, Figure 1 displays a proof in $\mathcal{H}_{\mathsf{C}}(*, -\!*)$. In the proof, a line "$j \mid \chi\ A, i_1, \ldots, i_k$" states that χ is a theorem denoted by the index j and derivable by the axiom or the rule A. If A is a rule, the indices $i_1, \ldots, i_k{<}j$ denote the theorems used as premises in order to derive χ. The example uses the rule $*$**-Adj**, which together with $\mathrel{-\!\!*}$**-Adj** states that the $*$ and $\mathrel{-\!\!*}$ are adjoint operators, and the axiom $(\mathbf{A^*_{19}})$, stating that $\mathrm{card}(\mathrm{dom}(h)) \leq \beta_1{+}\beta_2$ holds whenever a heap h can be split into two subheaps that have less than $\beta_1{+}1$ and $\beta_2{+}1$ memory cells, respectively. We also use the following theorems and rules, which can be shown derivable/admissible in the forthcoming calculus:

$$(\wedge\mathbf{Er})\ \ \psi \wedge \varphi \Rightarrow \varphi \qquad (\neg\neg\mathbf{E})\ \ \neg\neg\varphi \Rightarrow \varphi \qquad *\text{-}\mathbf{Ilr}\text{: } \frac{\varphi \Rightarrow \varphi' \quad \psi \Rightarrow \psi'}{\varphi * \psi \Rightarrow \varphi' * \psi'} \qquad \Rightarrow\text{-}\mathbf{Tr}\text{: } \frac{\varphi \Rightarrow \chi \quad \chi \Rightarrow \psi}{\varphi \Rightarrow \psi}$$

3.1 A simple calculus for the core formulae

To axiomatise $\mathsf{SL}(*, \mathrel{-\!\!*})$, we start by introducing the proof system $\mathcal{H}_{\mathsf{C}}$ (presented in System 1) dedicated to Boolean combinations of core formulae. $\mathcal{H}_{\mathsf{C}}$ and all the subsequent proof systems contain the axiom schemata and modus ponens for the propositional calculus. The axioms $I_i^?$ in System n are necessary for the fragment the System n governs, but are admissible when the axioms/rules from the System $n{+}1$ are present. In $(\mathbf{A^{C}_2})$, $\varphi[\mathtt{y}{\leftarrow}\mathtt{x}]$ is the formula obtained from φ by replacing with $\mathtt{x}$ every occurrence of $\mathtt{y}$. Let (s, h) be a memory state. The axioms state that $=$ is an equivalence relation (first two axioms), $h(s(\mathtt{x})){=}s(\mathtt{y})$ implies $s(\mathtt{x}) \in \mathrm{dom}(h)$ (axiom $(\mathbf{A^{C}_3})$) and that h is a (partial) function (axiom $(\mathbf{A^{C}_4})$). Furthermore, there are two intermediate axioms about size formulae: $(\mathbf{I^{C}_5})$ states that if $\mathrm{dom}(h)$ has at least $\beta{+}1$ elements, then it has at least β elements, whereas $(\mathbf{I^{C}_6})$ states that if there are β distinct memory cells corresponding to program variables, then indeed $\mathrm{dom}(h) \geq \beta$. It is easy to check that $\mathcal{H}_{\mathsf{C}}$ is sound (right-to-left direction of Theorem 2, below). In order to establish its completeness with respect to $\mathsf{Bool}(\mathsf{Core}(\mathtt{X}, \alpha))$, we first establish that $\mathcal{H}_{\mathsf{C}}$ is complete for a fragment of $\mathsf{Bool}(\mathsf{Core}(\mathtt{X}, \alpha))$, made of *core types*. Let $\mathtt{X} {\subseteq_{\mathsf{fin}}} \mathsf{VAR}$, $\alpha \in \mathbb{N}^+$ and $\widehat{\alpha}{=}\alpha{+}\mathrm{card}(\mathtt{X})$. We write $\mathsf{CoreTypes}(\mathtt{X}, \alpha)$ to denote the set of *core types* defined by $\{\varphi \in \mathsf{Conj}(\mathsf{Core}(\mathtt{X}, \widehat{\alpha})) \mid \forall\psi \in \mathsf{Core}(\mathtt{X}, \widehat{\alpha}),\ \{\psi \mid \neg\psi\} \subseteq_{\mathtt{Lt}} \varphi,\ \text{and } (\psi \wedge \neg\psi) \not\subseteq_{\mathtt{Lt}} \varphi\}$. Every formula in this set is a conjunction having exactly one literal built upon ψ for every $\psi \in \mathsf{Core}(\mathtt{X}, \widehat{\alpha})$.

▶ **Lemma 1.** *Let $\varphi \in \mathsf{CoreTypes}(\mathtt{X}, \alpha)$. We have $\neg\varphi$ is valid iff $\vdash_{\mathcal{H}_{\mathsf{C}}} \neg\varphi$.*

By classical reasoning, one can show that every $\varphi \in \mathsf{Bool}(\mathsf{Core}(\mathtt{X}, \alpha))$ is provably equivalent to a disjunction of core types. Together with Lemma 1, this implies that $\mathcal{H}_{\mathsf{C}}$ is complete.

▶ **Theorem 2.** (Adequacy) *A Boolean combination of core formulae φ is valid iff $\vdash_{\mathcal{H}_{\mathsf{C}}} \varphi$.*

3.2 A constructive elimination of $*$ to axiomatise $\mathsf{SL}(*, \mathtt{alloc})$

We enrich $\mathcal{H}_{\mathsf{C}}$ by adding axioms and inference rule that handle $*$ (System 2). The axioms deal with the commutative monoid properties of $(*, \mathtt{emp})$ and its distributivity over $\vee$ (as for Boolean BI, see e.g. [21]). In $(\mathbf{A^*_{14}})$, the notation $\varphi \blacktriangleleft\!\![\mathcal{B}]$ refers to the axiom schema φ assuming that the Boolean condition $\mathcal{B}$ holds. The rule $*$**-Intro** states that logical equivalence is a congruence for $*$. This allows us to remove the intermediate axioms $(\mathbf{I^{C}_5})$ and $(\mathbf{I^{C}_6})$ from the proof system. Hence, we call $\mathcal{H}_{\mathsf{C}}(*)$ the proof system obtained from $\mathcal{H}_{\mathsf{C}}$ by adding all schemata from System 2 and removing $(\mathbf{I^{C}_5})$ and $(\mathbf{I^{C}_6})$. It is easy to check that $\mathcal{H}_{\mathsf{C}}(*)$ is sound. More importantly, $\mathcal{H}_{\mathsf{C}}(*)$ enjoys the $*$ elimination property with respect to core types.

▶ **Lemma 3.** *Let φ and ψ in $\mathsf{CoreTypes}(\mathtt{X}, \alpha)$. There is a conjunction of core formulae literals $\chi \in \mathsf{Conj}(\mathsf{Core}(\mathtt{X}, 2\alpha))$ such that $\vdash_{\mathcal{H}_{\mathsf{C}}(*)} \varphi * \psi \Leftrightarrow \chi$.*

Proof. (sketch) Let $\varphi, \psi {\in} \mathsf{CoreTypes}(\mathtt{X}, \alpha)$. If φ is unsatisfiable, then $\vdash_{\mathcal{H}_{\mathsf{C}}} \varphi \Rightarrow \bot$, by Lemma 1. By the rule $*$**-Intro** and the axiom $(\mathbf{I^*_{10}})$, we get $\vdash_{\mathcal{H}_{\mathsf{C}}(*)} \varphi * \psi \Rightarrow \bot$ and we take $\chi = \bot$.

Assume now both φ and ψ to be satisfiable. Then $\varphi * \psi$ can be shown provably equivalent to:

$$
\begin{array}{llll}
 & \bigwedge \left\{\mathtt{x} \sim \mathtt{y} \subseteq_{\mathsf{Lt}} \{\varphi \mid \psi\} \,\middle|\, {\sim} \in \{=,\neq\}\right\} & \land & \bigwedge \{\mathtt{alloc}(\mathtt{x}) \subseteq_{\mathsf{Lt}} \{\varphi \mid \psi\}\} \\
\land & \bigwedge \{\mathtt{x} \hookrightarrow \mathtt{y} \subseteq_{\mathsf{Lt}} \{\varphi \mid \psi\}\} & \land & \bigwedge \{\neg\mathtt{alloc}(\mathtt{x}) \subseteq_{\mathsf{Lt}} \{\varphi \,;\, \psi\}\} \\
\land & \bigwedge \left\{\bot \,\middle|\, \mathtt{alloc}(\mathtt{x}) \subseteq_{\mathsf{Lt}} \{\varphi \,;\, \psi\}\right\} & \land & \bigwedge \left\{\neg\mathtt{x} \hookrightarrow \mathtt{y} \,\middle|\, \mathtt{alloc}(\mathtt{x}) \land \neg\mathtt{x} \hookrightarrow \mathtt{y} \subseteq_{\mathsf{Lt}} \{\varphi \mid \psi\}\right\} \\
\land & \bigwedge \left\{\mathtt{size} \geq \beta_1 + \beta_2 \,\middle|\, \begin{array}{r} \mathtt{size} \geq \beta_1 \subseteq_{\mathsf{Lt}} \varphi \\ \mathtt{size} \geq \beta_2 \subseteq_{\mathsf{Lt}} \psi \end{array}\right\} & \land & \bigwedge \left\{\neg\mathtt{size} \geq \beta_1 + \beta_2 \dot{-} 1 \,\middle|\, \begin{array}{r} \neg\mathtt{size} \geq \beta_1 \subseteq_{\mathsf{Lt}} \varphi \\ \neg\mathtt{size} \geq \beta_2 \subseteq_{\mathsf{Lt}} \psi \end{array}\right\}
\end{array}
$$

This equivalence is reminiscent to the one in [20, Lemma 3] that is proved semantically. In a way, because $\mathcal{H}_{\mathsf{C}}(*)$ will reveal to be complete, the restriction of the proof of [20, Lemma 3] to $\mathsf{SL}(*, \mathtt{alloc})$ can actually be replayed completely syntactically within $\mathcal{H}_{\mathsf{C}}(*)$. ◀

By the distributivity axiom $(\mathbf{I}^*_{\mathbf{9}})$, this result is extended from core types to arbitrary Boolean combinations of core formulae. $\mathcal{H}_{\mathsf{C}}(*)$ is therefore complete for $\mathsf{SL}(*, \mathtt{alloc})$, i.e. the logic obtained from $\mathsf{SL}(*, \mathrel{-\!\!*})$ by removing $\mathrel{-\!\!*}$ and adding the formulae $\mathtt{alloc}(\mathtt{x})$ (only core formulae requiring $\mathrel{-\!\!*}$). Then, to prove that a formula $\varphi \in \mathsf{SL}(*, \mathtt{alloc})$ is valid, we repeatedly apply the $*$ elimination bottom-up obtaining a Boolean combination of core formulae ψ that is equivalent to φ. We rely on the completeness of $\mathcal{H}_{\mathsf{C}}$ (Theorem 2) to prove that ψ is valid.

▶ **Theorem 4.** *A formula φ in $\mathsf{SL}(*, \mathtt{alloc})$ is valid iff $\vdash_{\mathcal{H}_{\mathsf{C}}(*)} \varphi$.*

3.3 A constructive elimination of $\mathrel{-\!\!*}$ to axiomatise $\mathsf{SL}(*, \mathrel{-\!\!*})$

The proof system $\mathcal{H}_{\mathsf{C}}(*, \mathrel{-\!\!*})$ is defined as $\mathcal{H}_{\mathsf{C}}(*)$ augmented with the axioms and inference rules from System 3 dedicated to separating implication. The axioms involving $\mathrel{-\!\!\circledast}$ (kind of dual of $\mathrel{-\!\!*}$ introduced in Section 2) express that it is always possible to extend a given heap with an extra cell, and that the address and the content of this cell can be fixed arbitrarily (provided it is not already allocated). The adjunction rules are from the Hilbert-style axiomatisation of Boolean BI [21, Section 2]. One can observe that the axioms $(\mathbf{I}^*_{\mathbf{9}})$, $(\mathbf{I}^*_{\mathbf{10}})$, $(\mathbf{I}^*_{\mathbf{12}})$ and $(\mathbf{I}^*_{\mathbf{13}})$ are derivable in $\mathcal{H}_{\mathsf{C}}(*, \mathrel{-\!\!*})$. It is easy to check that $\mathcal{H}_{\mathsf{C}}(*, \mathrel{-\!\!*})$ is sound. Analogously, $\mathcal{H}_{\mathsf{C}}(*, \mathrel{-\!\!*})$ enjoys the $\mathrel{-\!\!*}$ elimination property, stated below by means of $\mathrel{-\!\!\circledast}$.

▶ **Lemma 5.** *Let φ and ψ in $\mathsf{CoreTypes}(\mathtt{X}, \alpha)$. There is a conjunction of core formulae literals $\chi \in \mathsf{Conj}(\mathsf{Core}(\mathtt{X}, \alpha))$ such that $\vdash_{\mathcal{H}_{\mathsf{C}}(*, \mathrel{-\!\!*})} (\varphi \mathrel{-\!\!\circledast} \psi) \Leftrightarrow \chi$.*

Proof. (sketch) If either φ or ψ is unsatisfiable, then one can show that $\vdash_{\mathcal{H}_{\mathsf{C}}(*, \mathrel{-\!\!*})} \varphi \mathrel{-\!\!\circledast} \psi \Rightarrow \bot$. Otherwise, $\varphi \mathrel{-\!\!\circledast} \psi$ can be shown provably equivalent to

$$
\begin{array}{llllll}
 & \bigwedge \left\{\mathtt{x} \sim \mathtt{y} \subseteq_{\mathsf{Lt}} \{\varphi \mid \psi\} \,\middle|\, {\sim} \in \{=,\neq\}\right\} & \land & \bigwedge \{\neg\mathtt{alloc}(\mathtt{x}) \subseteq_{\mathsf{Lt}} \psi\} & \land & \bigwedge \{\neg\mathtt{x} \hookrightarrow \mathtt{y} \subseteq_{\mathsf{Lt}} \psi\} \\
\land & \bigwedge \left\{\mathtt{alloc}(\mathtt{x}) \,\middle|\, \begin{array}{r} \neg\mathtt{alloc}(\mathtt{x}) \subseteq_{\mathsf{Lt}} \varphi \\ \mathtt{alloc}(\mathtt{x}) \subseteq_{\mathsf{Lt}} \psi \end{array}\right\} & \land & \bigwedge \left\{\mathtt{x} \hookrightarrow \mathtt{y} \,\middle|\, \begin{array}{r} \neg\mathtt{alloc}(\mathtt{x}) \subseteq_{\mathsf{Lt}} \varphi \\ \mathtt{x} \hookrightarrow \mathtt{y} \subseteq_{\mathsf{Lt}} \psi \end{array}\right\} & \land & \bigwedge \left\{\neg\mathtt{alloc}(\mathtt{x}) \,\middle|\, \mathtt{alloc}(\mathtt{x}) \subseteq_{\mathsf{Lt}} \varphi\right\} \\
\land & \bigwedge \left\{\mathtt{size} \geq \beta_2 + 1 \dot{-} \beta_1 \,\middle|\, \begin{array}{r} \neg\mathtt{size} \geq \beta_1 \subseteq_{\mathsf{Lt}} \varphi \\ \mathtt{size} \geq \beta_2 \subseteq_{\mathsf{Lt}} \psi \end{array}\right\} & \land & \bigwedge \left\{\bot \,\middle|\, \begin{array}{r} \mathtt{x} \hookrightarrow \mathtt{y} \subseteq_{\mathsf{Lt}} \varphi \\ \neg\mathtt{x} \hookrightarrow \mathtt{y} \subseteq_{\mathsf{Lt}} \psi \end{array}\right\} & \land & \bigwedge \left\{\bot \,\middle|\, \begin{array}{r} \mathtt{alloc}(\mathtt{x}) \land \neg\mathtt{x} \hookrightarrow \mathtt{y} \subseteq_{\mathsf{Lt}} \varphi \\ \mathtt{x} \hookrightarrow \mathtt{y} \subseteq_{\mathsf{Lt}} \psi \end{array}\right\} \\
\land & \bigwedge \left\{\neg\mathtt{size} \geq \beta_2 \dot{-} \beta_1 \,\middle|\, \begin{array}{r} \mathtt{size} \geq \beta_1 \subseteq_{\mathsf{Lt}} \varphi \\ \neg\mathtt{size} \geq \beta_2 \subseteq_{\mathsf{Lt}} \psi \end{array}\right\} & \land & \bigwedge \left\{\bot \,\middle|\, \begin{array}{r} \mathtt{alloc}(\mathtt{x}) \subseteq_{\mathsf{Lt}} \varphi \\ \neg\mathtt{alloc}(\mathtt{x}) \subseteq_{\mathsf{Lt}} \psi \end{array}\right\} & &
\end{array}
$$

where $a \dot{-} b$ stands for $a - b$ if $a \geq b$, 0 otherwise. Again, this equivalence is reminiscent to the one in [20, Lemma 4] proved semantically. Herein, the proof is completely syntactical. ◀

Again, this result for core types can be extended to arbitrary Boolean combinations of core formulae, as we show that the distributivity of $\mathrel{-\!\!\circledast}$ over disjunctions is provable in $\mathcal{H}_{\mathsf{C}}(*, \mathrel{-\!\!*})$. As a consequence of this development, we achieve one of the main results of the paper.

▶ **Theorem 6.** *$\mathcal{H}_{\mathsf{C}}(*, \mathrel{-\!\!*})$ is sound and complete for $\mathsf{SL}(*, \mathrel{-\!\!*})$.*

What's next? To provide further evidence that our method is robust, we shall apply it to axiomatise other separation logics, for instance by adding the list segment predicate $\texttt{ls}$ [2] (or inductive predicates in general) or first-order quantification. Of course, the set of valid formulae must be r.e., which discards any attempt with $\mathsf{SL}(*, \mathrel{-\!\!*}, \texttt{ls})$ or with the first-order version of $\mathsf{SL}(*, \mathrel{-\!\!*})$ [15, 4]. In Section 4, we introduce an extension of $\mathsf{SL}(*, \texttt{ls})$ and we axiomatise it with our method, whose main ingredients are recalled below.

3.4 Ingredients of the method

The Hilbert-style axiomatisation of $\mathsf{SL}(*, \mathrel{-\!\!*})$ has culminated with Theorem 6 that states the adequateness of $\mathcal{H}_{\mathsf{C}}(*, \mathrel{-\!\!*})$. Below, we would like to recapitulate the key ingredients of the proposed method, not only to provide a vade-mecum for axiomatising other separation logics (which we illustrate on the newly introduced logic $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ in Section 4), but also to identify the essential features and where variations are still possible.

Core formulae. To axiomatise $\mathsf{SL}(*, \mathrel{-\!\!*})$ internally, the core formulae have played an essential role. The main properties of these formulae is that their Boolean combinations capture the full logic $\mathsf{SL}(*, \mathrel{-\!\!*})$ [29] and all the core formulae can be expressed in $\mathsf{SL}(*, \mathrel{-\!\!*})$. Generally speaking, our axiom system naturally leads to a form of constructive completeness, as advocated in [19, 31]: the axiomatisation provides proof-theoretical means to transform any formula into an equivalent Boolean combination of core formulae, and it contains also a part dedicated to the derivation of valid Boolean combinations of core formulae (understood as a syntactical fragment of $\mathsf{SL}(*, \mathrel{-\!\!*})$). What is specific to each logic is the design of the set of core formulae and in the case of $\mathsf{SL}(*, \mathrel{-\!\!*})$, this was already known since [29].

Big-step vs. small-step axiom schemas. $\mathcal{H}_{\mathsf{C}}(*, \mathrel{-\!\!*})$ simulates the bottom-up elimination of separating connectives (see Lemmata 3 and 5) when the arguments are two Boolean combinations of core formulae. To do so, $\mathcal{H}_{\mathsf{C}}(*, \mathrel{-\!\!*})$ contains axiom schemas that perform such an elimination in multiple "small-step" derivations, e.g. by deriving a single $\texttt{alloc}(\texttt{x})$ predicate from $\texttt{alloc}(\texttt{x}) * \top$ (axiom $(\mathbf{I}^*_{\mathbf{12}})$). Alternatively, it would have been possible to include "big-step" axiom schemas that, given the two Boolean combinations of core formulae, derive the equivalent formula in one single derivation step. Instances of this are given in the proof sketch of Lemma 3, and later in Section 4 (axiom $(*_{\mathbf{48}})$). The main difference is that small-step axioms provide a simpler understanding of the key properties of the logic.

4 How to axiomatise internally the separation logic $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$

Though core formulae are handful for several existing separation logics, see e.g. recently [15, 32, 20], we would like to test our method with first-order quantification and reachability predicates, standard features in specifications. However, $\mathsf{SL}(*, \mathrel{-\!\!*}, \texttt{ls})$ is already known to be non-finitely axiomatisable, see the developments in Section 2. So, we need to downgrade our ambitions and we suggest to consider a new logic with guarded quantification and $\texttt{ls}$ and this is $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ presented below. Note that the idea of having guarded quantification with second-order features is not new, see e.g. in [24] extensions of the guarded fragment of first-order logic with fixed points, but herein, this is done in the framework of separation logics and their axiomatisation. In short, we introduce the new separation logic $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ that admits the connective $*$, the list segment predicate $\texttt{ls}$ (implicitly) and a guarded form of first-order quantification involving $\texttt{ls}$. It contains the symbolic heap fragment [2, 11] but

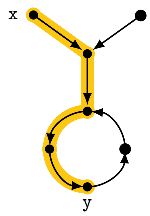

Figure 2 Path quantifier.

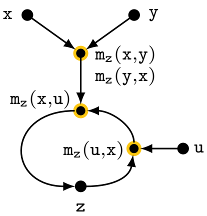

Figure 3 Meet points.

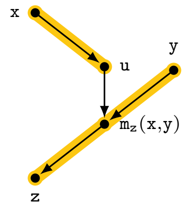

Figure 4 Sees predicates.

also richer logics such as $\mathsf{SL}(*, \mathtt{reach}^+)$ (see e.g. [15]). As a by-product of our completeness proof, we are able to characterise the complexity of the satisfiability problem for $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$.

4.1 A guarded logic with ls: $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$

Formulae of $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ are defined according to the grammar below (where $\mathtt{x}, \mathtt{y}, \mathtt{z} \in \mathsf{VAR}$):

$$\varphi := \mathtt{x} = \mathtt{y} \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \mathtt{emp} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi * \varphi \mid \exists \mathtt{z}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle\varphi$$

All the syntactic ingredients are standard except the quantifier (denoted with $\exists{:}\rightsquigarrow$). Intuitively (the formal definition is provided below), $\exists \mathtt{z}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle\varphi$ is a guarded form of quantification that is intended to hold true whenever $\mathtt{y}$ is reachable from $\mathtt{x}$ in at least one step, and there is a location ℓ along the minimal path between $\mathtt{x}$ and $\mathtt{y}$ so that the formula φ holds whenever ℓ is assigned to $\mathtt{z}$. Figure 2 highlights the possible assignments of $\mathtt{z}$ (arrows represent the heap). Given a heap h and $\ell_1, \ell_2 \in \mathsf{LOC}$, we define $h[\ell_1, \ell_2[$ as the set of locations in the shortest path from ℓ_1 to ℓ_2 (ℓ_2 possibly excluded). Formally:

$$h[\ell_1, \ell_2[\stackrel{\text{def}}{=} \left\{ \ell \in \mathsf{LOC} \;\middle|\; \begin{array}{l} \text{there are } \delta_1 \geq 0 \text{ and } \delta_2 \geq 1 \text{ such that } h^{\delta_1}(\ell_1) = \ell, \\ h^{\delta_2}(\ell) = \ell_2 \text{ and, for every } \delta \in [1, \delta_1{+}\delta_2{-}1], h^{\delta}(\ell_1) \neq \ell_2 \end{array} \right\}$$

For example, $h[\ell, \ell[= \emptyset$ holds iff ℓ is not in a cycle. Otherwise, $h[\ell, \ell[$ contains all the locations in the cycle containing ℓ. By definition, the minimal paths are preserved when considering heap extensions. Then, the satisfaction relation $\models$ is completed with

$$(s, h) \models \exists \mathtt{z}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle\ \varphi \stackrel{\text{def}}{\Leftrightarrow} h[s(\mathtt{x}), s(\mathtt{y})[\neq \emptyset \text{ and } \exists \ell \in h[s(\mathtt{x}), s(\mathtt{y})[\cup \{s(\mathtt{y})\} \text{ s.t. } (s[\mathtt{z} \leftarrow \ell], h) \models \varphi.$$

We define $\forall \mathtt{z}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle\varphi \stackrel{\text{def}}{=} \neg\exists \mathtt{z}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle\neg\varphi$. In a separation logic *lingua* admitting first-order quantification of program variables over the set of locations LOC, and a predicate $\mathtt{reach}^+(\mathtt{x}, \mathtt{y})$ (reachability in at least one step, as in [15]), the formula $\exists \mathtt{z}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle\varphi$ is equivalent to

$$\mathtt{reach}^+(\mathtt{x}, \mathtt{y}) \wedge \exists\, \mathtt{z}\ \varphi \wedge (\mathtt{z} = \mathtt{x} \vee \mathtt{z} = \mathtt{y} \vee ((\mathtt{reach}^+(\mathtt{x}, \mathtt{z}) \wedge \neg\mathtt{reach}^+(\mathtt{x}, \mathtt{y})) * \mathtt{reach}^+(\mathtt{z}, \mathtt{y}))).$$

Obviously, $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ does not allow unrestricted first-order quantification but it can faithfully define the reachability predicates classically studied in separation logic [15, 38]. $\mathtt{reach}^+(\mathtt{x}, \mathtt{y})$ is definable as $\exists \mathtt{z}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle\top$, and allows us to define $\mathtt{ls}(\mathtt{x}, \mathtt{y})$ and $\mathtt{reach}(\mathtt{x}, \mathtt{y})$ as shown in [15]: $\mathtt{ls}(\mathtt{x}, \mathtt{y}) \stackrel{\text{def}}{=} (\mathtt{x} = \mathtt{y} \wedge \mathtt{emp}) \vee (\mathtt{x} \neq \mathtt{y} \wedge \mathtt{reach}^+(\mathtt{x}, \mathtt{y}) \wedge \neg(\neg\mathtt{emp} * \mathtt{reach}^+(\mathtt{x}, \mathtt{y})))$, whereas $\mathtt{reach}(\mathtt{x}, \mathtt{y}) \stackrel{\text{def}}{=} \mathtt{x} = \mathtt{y} \vee \mathtt{reach}^+(\mathtt{x}, \mathtt{y})$. There are two features of $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$, we would like to emphasize. First, it is possible to enforce a heap domain of exponential size. Indeed, we define the formula $\mathrm{R}^n(\mathtt{x}, \mathtt{y})$ of size linear in n, but enforcing the existence of a path of length at least 2^n between two distinct locations corresponding to $\mathtt{x}$ and $\mathtt{y}$, respectively. $\mathrm{R}^0(\mathtt{x}, \mathtt{y}) \stackrel{\text{def}}{=} \mathtt{x} \neq \mathtt{y} \wedge \exists \mathtt{z}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle\top$, whereas for $n \geq 0$, $\mathrm{R}^{n+1}(\mathtt{x}, \mathtt{y})$ is defined as

$$\mathtt{x}{\neq}\mathtt{y}{\wedge}\exists \mathtt{z}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle\, \forall \mathtt{z}'{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle\, \forall \mathtt{z}''{:}\langle \mathtt{x} \rightsquigarrow \mathtt{y}\rangle\, ((\mathtt{z}' = \mathtt{x} \wedge \mathtt{z}'' = \mathtt{z}) \vee (\mathtt{z}' = \mathtt{z} \wedge \mathtt{z}'' = \mathtt{y}) \Rightarrow \mathrm{R}^n(\mathtt{z}', \mathtt{z}''))\,.$$

Nevertheless, in Section 4.6 we show how the satisfiability and validity problems for $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ are in PSpace. Another interesting feature of $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ is illustrated by its ability to state that from two locations corresponding to program variables (say $\mathtt{x}, \mathtt{y}$), it is possible to reach a different location, which in turn reaches another location corresponding to a program variable (say $\mathtt{z}$). This can be done with the formula $\exists \mathtt{w}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{z}\rangle(\mathtt{reach}^{+}(\mathtt{y},\mathtt{w}) \wedge \bigwedge_{\mathtt{v}\in\{\mathtt{x},\mathtt{y},\mathtt{z}\}} \mathtt{w} \neq \mathtt{v})$. Thus, the logic is able to express that two paths meet at a specific location. This naturally leads to the notion of meet-points, introduced next in order to define the core formulae for $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$.

4.2 Core formulae are back!

In order to axiomatise internally $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ with our method, we need to possess a set of core formulae that captures $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$. Below, we design such core formulae and establish its appropriateness. They make intensive use of meet-point terms, a concept introduced in [15] but that will play a crucial role herein. Informally, given a memory state (s,h), a meet-point between $s(\mathtt{x})$ and $s(\mathtt{y})$ leading to $s(\mathtt{z})$ is a location ℓ such that (I) ℓ reaches $s(\mathtt{z})$, (II) both locations $s(\mathtt{x})$ and $s(\mathtt{y})$ reach ℓ, and (III) there is no location ℓ' satisfying these properties and reachable from $s(\mathtt{x})$ in strictly fewer steps. A meet-point term of the form $\mathtt{m}_{\mathtt{z}}(\mathtt{x},\mathtt{y})$, where $\mathtt{x},\mathtt{y},\mathtt{z} \in \mathsf{VAR}$, is then an expression that, given a memory state (s,h), is intended to be interpreted by a meet-point between $s(\mathtt{x})$ and $s(\mathtt{y})$ leading to $s(\mathtt{z})$ (if it exists). Figure 3 shows some of the meet-points between $\mathtt{x}$ and other program variables, highlighting their distribution in a memory state. In particular, notice how in the figure, $\mathtt{m}_{\mathtt{z}}(\mathtt{x},\mathtt{u})$ is different from $\mathtt{m}_{\mathtt{z}}(\mathtt{u},\mathtt{x})$, which happens because of the condition (III) and as the two corresponding locations are in a cycle. We call this type of meet-points *asymmetric*. We now formalise these concepts. Given $\mathtt{X} \subseteq \mathsf{VAR}$, we write $\mathbb{T}(\mathtt{X})$ to denote the set $\mathtt{X} \cup \{\mathtt{m}_{\mathtt{z}}(\mathtt{x},\mathtt{y}) \mid \mathtt{x},\mathtt{y},\mathtt{z} \in \mathtt{X}\}$. Elements of $\mathbb{T}(\mathsf{VAR})$ are called *terms*. Expressions $\mathtt{m}_{\mathtt{z}}(\mathtt{x},\mathtt{y})$ are syntactic constructs called *meet-point terms*. Terms are denoted with $\mathtt{t},\mathtt{t}_1,\mathtt{t}_2,\ldots$, when we do not need to distinguish between variables and meet-point terms. To give a semantics to these objects, we interpret the terms by means of the interpretation function $[\![.]\!]_{s,h} : \mathbb{T}(\mathsf{VAR}) \to \mathsf{LOC}$ s.t. $[\![\mathtt{x}]\!]_{s,h} \stackrel{\text{def}}{=} s(\mathtt{x})$ for $\mathtt{x} \in \mathsf{VAR}$, and $[\![\mathtt{m}_{\mathtt{z}}(\mathtt{x},\mathtt{y})]\!]_{s,h}$ is defined and takes the value ℓ iff there are $\delta_1,\delta_2 \geq 0$ s.t.

- $h^{\delta_1}(s(\mathtt{x})) = h^{\delta_2}(s(\mathtt{y})) = \ell$ and there is $\delta \geq 0$ such that $h^{\delta}(\ell) = s(\mathtt{z})$;
- for every $\delta_1' \in [0,\delta_1-1]$ and $\delta_2' \geq 0$, $h^{\delta_1'}(s(\mathtt{x})) \neq h^{\delta_2'}(s(\mathtt{y}))$.

One last object is needed in order to define the core formulae. Given a memory state (s,h) and a finite set of pairs of terms $\mathfrak{P} \subseteq_{\mathsf{fin}} \mathbb{T}(\mathsf{VAR}) \times \mathbb{T}(\mathsf{VAR})$, we write $\mathsf{Rem}^{\mathfrak{P}}_{s,h}$ to denote the subset of $\mathrm{dom}(h)$ made of the locations that are not in the path between two locations corresponding to terms in a pair of $\mathfrak{P}$. Formally: $\mathsf{Rem}^{\mathfrak{P}}_{s,h} \stackrel{\text{def}}{=} \mathrm{dom}(h) \setminus \big(\bigcup_{(\mathtt{t}_1,\mathtt{t}_2)\in\mathfrak{P}} h[[\![\mathtt{t}_1]\!]_{s,h}, [\![\mathtt{t}_2]\!]_{s,h}[\big)$.

The *core formulae* are expressions of the form: $\mathtt{t}_1 = \mathtt{t}_2$, $\mathtt{sees}_{\mathfrak{T}}(\mathtt{t}_1,\mathtt{t}_2) {\geq} \beta{+}1$ and $\mathtt{rem}_{\mathfrak{P}} {\geq} \beta$, where $\mathtt{t}_1,\mathtt{t}_2 \in \mathbb{T}(\mathsf{VAR})$, $\mathfrak{T} \subseteq_{\mathsf{fin}} \mathbb{T}(\mathsf{VAR})$, $\mathfrak{P} \subseteq_{\mathsf{fin}} \mathbb{T}(\mathsf{VAR}) \times \mathbb{T}(\mathsf{VAR})$ and $\beta \in \mathbb{N}$. We write $\mathtt{sees}_{\mathfrak{T}}(\mathtt{t}_1,\mathtt{t}_2)$ for $\mathtt{sees}_{\mathfrak{T}}(\mathtt{t}_1,\mathtt{t}_2){\geq}1$. The satisfaction relation $\models$ is extended to core formulae:

- $(s,h) \models \mathtt{t}_1 = \mathtt{t}_2 \stackrel{\text{def}}{\Leftrightarrow} [\![\mathtt{t}_1]\!]_{s,h} = [\![\mathtt{t}_2]\!]_{s,h}$;
- $(s,h) \models \mathtt{rem}_{\mathfrak{P}} {\geq} \beta \stackrel{\text{def}}{\Leftrightarrow} \mathrm{card}(\mathsf{Rem}^{\mathfrak{P}}_{s,h}) \geq \beta$;
- $(s,h) \models \mathtt{sees}_{\mathfrak{T}}(\mathtt{t}_1,\mathtt{t}_2) {\geq} \beta \stackrel{\text{def}}{\Leftrightarrow}$ there is $\delta \geq \beta$ such that $h^{\delta}([\![\mathtt{t}_1]\!]_{s,h}) = [\![\mathtt{t}_2]\!]_{s,h}$ and for all $\delta' \in [1,\delta-1]$, $h^{\delta'}([\![\mathtt{t}_1]\!]_{s,h}) \notin \{[\![\mathtt{t}_2]\!]_{s,h}\} \cup \{[\![\mathtt{t}]\!]_{s,h} \mid \mathtt{t} \in \mathfrak{T}\}$.

As earlier in Section 3, we write $\mathsf{Core}(\mathtt{X},\alpha)$ to denote the set of core formulae restricted to terms from $\mathbb{T}(\mathtt{X})$, where $\mathtt{X} \subseteq_{\mathsf{fin}} \mathsf{VAR}$ and β is bounded above by α. In order to become more familiar with these core formulae, let us consider the memory state (s,h) outlined in Figure 4. Since both $s(\mathtt{x})$ and $s(\mathtt{y})$ reach $s(\mathtt{z})$, $[\![\mathtt{m}_{\mathtt{z}}(\mathtt{x},\mathtt{y})]\!]_{s,h}$ is defined, or alternatively $(s,h) \models \mathtt{m}_{\mathtt{z}}(\mathtt{x},\mathtt{y}) = \mathtt{m}_{\mathtt{z}}(\mathtt{x},\mathtt{y})$. Therefore, we have that $(s,h) \models \mathtt{sees}_{\emptyset}(\mathtt{x},\mathtt{m}_{\mathtt{z}}(\mathtt{x},\mathtt{y}))$. We also note that $s(\mathtt{u})$ is a location in the minimal path from $s(\mathtt{x})$ to $[\![\mathtt{m}_{\mathtt{z}}(\mathtt{x},\mathtt{y})]\!]_{s,h}$. However, as $s(\mathtt{u})$ is distinct from these two locations, we conclude that $(s,h) \models \neg\mathtt{sees}_{\{\mathtt{u}\}}(\mathtt{x},\mathtt{m}_{\mathtt{z}}(\mathtt{x},\mathtt{y}))$. Lastly, let us take for example the sets of locations corresponding to the two paths highlighted in yellow: $h[s(\mathtt{x}),s(\mathtt{u})[$ and $h[s(\mathtt{y}),s(\mathtt{z})[$. The location $s(\mathtt{u})$ does not belong to any of these sets.

As it is in dom(h), we conclude that $(s,h) \models \mathtt{rem}_{\{(\mathtt{x},\mathtt{u}),(\mathtt{y},\mathtt{z})\}} \geq 1$.

Expressing core formulae in $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$. A crucial point for axiomatising $\mathsf{SL}(*, -\!\!*)$ is that every core formula is a mere abbreviation for a formula of the logic. This is the property that leads to an *internal* axiomatisation. The same holds for $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ as one can show that every core formula can be defined in $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ and, in the forthcoming axiomatisation, should be considered as an abbreviation. For example, the formula $\mathtt{sees}_\emptyset(\mathtt{x},\mathtt{y}) \geq \beta$ can be shown equivalent to $(\mathtt{strict}(\mathtt{reach}^+(\mathtt{x},\mathtt{y})) \wedge \mathtt{size} \geq \beta) * \top$, where $\mathtt{strict}(\varphi)$ is a shortcut for $\varphi \wedge \neg(\neg\mathtt{emp} * \varphi)$ and states that φ holds in the current model, say (s,h) but does not hold in any submodel (i.e. in (s,h') where $h' \sqsubset h$). Similarly, $\mathtt{x} = \mathtt{m_u}(\mathtt{y},\mathtt{z})$ is equivalent to

$$\mathtt{reach}(\mathtt{x},\mathtt{u}) \wedge (\mathtt{reach}(\mathtt{y},\mathtt{x}) * \mathtt{reach}(\mathtt{z},\mathtt{x})) \wedge (\mathtt{reach}^+(\mathtt{x},\mathtt{x}) \Rightarrow (\mathtt{reach}(\mathtt{y},\mathtt{x}) * \mathtt{reach}^+(\mathtt{x},\mathtt{x}))),$$

whereas $\mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_w}(\mathtt{u},\mathtt{v})$ is $\exists \mathtt{j}{:}\langle \mathtt{x} \rightsquigarrow \mathtt{z}\rangle(\mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{j} \wedge \mathtt{j} = \mathtt{m_w}(\mathtt{u},\mathtt{v}))$, where $\mathtt{j} \notin \{\mathtt{x},\mathtt{y},\mathtt{z},\mathtt{u},\mathtt{v},\mathtt{w}\}$.

▶ **Lemma 7.** *Every core formula is logically equivalent to a formula of* $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$.

4.3 Axiomatisation of the logic of core formulae

As done in Section 3, to axiomatise $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ we start by extending the axiom system for the propositional calculus in order to obtain the proof system $\mathcal{H}_{\mathsf{C}}$ dedicated to Boolean combinations of core formulae. The axioms, presented in System 4, are divided into axioms for equalities between terms, whose name is of the form $=_i^{\mathsf{C}}$; axioms essentially about the predicates $\mathtt{sees}$, whose name is of the form $\mathtt{s}_i^{\mathsf{C}}$; and axioms essentially about the predicates $\mathtt{rem}$, whose name is of the form $\mathtt{r}_i^{\mathsf{C}}$. In order to obtain this axiom system, the two main difficulties (which lead to very technical formulae) are given by the distribution of meet-points within the memory state and the axiomatisation of the predicates $\mathtt{sees}$. For the former, it is important to distinguish between symmetric and asymmetric meet-points. For this reason, System 4 uses the formulae $\mathtt{def}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \stackrel{\text{def}}{=} \mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_z}(\mathtt{x},\mathtt{y})$, which checks if a meet-point is defined, $\mathtt{sym}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \stackrel{\text{def}}{=} \mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_z}(\mathtt{y},\mathtt{x})$ for symmetric meet-points, and $\mathtt{asym}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \stackrel{\text{def}}{=} \mathtt{def}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \wedge \neg\mathtt{sym}(\mathtt{m_z}(\mathtt{x},\mathtt{y}))$ for asymmetric ones. The definition of these formulae, as well as the ones below, is extended on a variable $\mathtt{x} \in \mathsf{VAR}$ simply by replacement with the meet-point $\mathtt{m_x}(\mathtt{x},\mathtt{x})$ (the two terms are always equivalent, see the axiom $(=_1^{\mathsf{C}})$). So, for example $\mathtt{def}(\mathtt{x})$ is defined as $\mathtt{def}(\mathtt{m_x}(\mathtt{x},\mathtt{x}))$. For $\mathtt{sees}$ predicates, an important distinction is given by terms corresponding to different locations in the same tree (no cycle is involved) and terms that correspond to different locations in the same cycle. Hence, we define the abbreviations $\mathtt{before}(\mathtt{t}_1,\mathtt{t}_2)$ and $\mathtt{samecycle}(\mathtt{t}_1,\mathtt{t}_2)$ with the following meanings:

$(s,h) \models \mathtt{before}(\mathtt{t}_1,\mathtt{t}_2)$ iff $[\![\mathtt{t}_1]\!]_{s,h} \neq [\![\mathtt{t}_2]\!]_{s,h}$ and, there is a path from $[\![\mathtt{t}_1]\!]_{s,h}$ to $[\![\mathtt{t}_2]\!]_{s,h}$ s.t. the only location on the path that may belong to a cycle is $[\![\mathtt{t}_2]\!]_{s,h}$.

$(s,h) \models \mathtt{samecycle}(\mathtt{t}_1,\mathtt{t}_2)$ iff $[\![\mathtt{t}_1]\!]_{s,h} \neq [\![\mathtt{t}_2]\!]_{s,h}$ and there is a cycle with both $[\![\mathtt{t}_1]\!]_{s,h}$ and $[\![\mathtt{t}_2]\!]_{s,h}$.

They are defined as follows for meet-points (and extended for $\mathtt{x} \in \mathsf{VAR}$ as shown for $\mathtt{def}(\mathtt{x})$)

- The formulae $\mathtt{before}(\mathtt{m_z}(\mathtt{x},\mathtt{y}),\mathtt{m_v}(\mathtt{x},\mathtt{u}))$ and $\mathtt{before}(\mathtt{m_z}(\mathtt{y},\mathtt{x}),\mathtt{m_v}(\mathtt{x},\mathtt{u}))$ are both defined as $\mathtt{sym}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \wedge \mathtt{def}(\mathtt{m_v}(\mathtt{x},\mathtt{y})) \wedge \mathtt{def}(\mathtt{m_v}(\mathtt{x},\mathtt{u})) \wedge \mathtt{m_z}(\mathtt{x},\mathtt{y}) \neq \mathtt{m_v}(\mathtt{x},\mathtt{u}) \wedge \mathtt{m_z}(\mathtt{x},\mathtt{y}) \neq \mathtt{m_v}(\mathtt{y},\mathtt{u})$;
- $\mathtt{before}(\mathtt{m_z}(\mathtt{x},\mathtt{y}),\mathtt{m_w}(\mathtt{u},\mathtt{v})) \stackrel{\text{def}}{=} \bigvee_{a \in \{\mathtt{u},\mathtt{v}\}} \mathtt{before}(\mathtt{m_z}(\mathtt{x},\mathtt{y}),\mathtt{m_w}(\mathtt{x},a)) \wedge \mathtt{m_w}(\mathtt{x},a) = \mathtt{m_w}(\mathtt{u},\mathtt{v})$;
- $\mathtt{samecycle}(\mathtt{m_z}(\mathtt{x},\mathtt{y}),\mathtt{m_w}(\mathtt{u},\mathtt{v})) \stackrel{\text{def}}{=} \mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_w}(\mathtt{x},\mathtt{u}) \wedge \mathtt{m_w}(\mathtt{u},\mathtt{v}) = \mathtt{m_z}(\mathtt{u},\mathtt{x}) \wedge \mathtt{asym}(\mathtt{m_z}(\mathtt{x},\mathtt{u}))$.

We write $\mathtt{t} \in \mathfrak{T}$ (finite set of terms $\mathfrak{T}$) to denote $\bigvee_{\mathtt{t}_2 \in \mathfrak{T}} \mathtt{t} = \mathtt{t}_2$. Like the axiom $(\mathbf{A}_2^{\mathsf{C}})$, the axiom $(=_3^{\mathsf{C}})$ performs a substitution of every occurrence of $\mathtt{t}_1$ with $\mathtt{t}_2$. We have to be careful here: when substituting a variable $\mathtt{x}$ with a meet-point $\mathtt{m_u}(\mathtt{y},\mathtt{z})$, we only substitute the occurrences of $\mathtt{x}$ that are not inside meet-point terms. For example, $\mathtt{sees}_{\{\mathtt{x},\mathtt{m_x}(\mathtt{x},\mathtt{x})\}}(\mathtt{x},\mathtt{m_x}(\mathtt{x},\mathtt{x}))[\mathtt{x} \leftarrow \mathtt{m_u}(\mathtt{y},\mathtt{z})]$

is equal to $\mathtt{sees}_{\{\mathtt{m_u(y,z)},\mathtt{m_x(x,x)}\}}(\mathtt{m_u}(\mathtt{y},\mathtt{z}),\mathtt{m_x}(\mathtt{x},\mathtt{x}))$. By way of example, let us explain why all the instances of the axiom $(=^{\mathsf{C}}_{\mathbf{6}})$ are valid. Suppose $(s,h) \models \mathtt{def}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \wedge \mathtt{def}(\mathtt{m_u}(\mathtt{x},\mathtt{y}))$. Since $[\![\mathtt{m_z}(\mathtt{x},\mathtt{y})]\!]_{s,h}$ is defined (say equal to ℓ), there are $\delta_1, \delta_2 \geq 0$ such that

- $h^{\delta_1}(s(\mathtt{x})) = h^{\delta_2}(s(\mathtt{y})) = \ell$ and there is $\delta \geq 0$ such that $h^{\delta}(\ell) = s(\mathtt{z})$;
- for every $\delta_1' \in [0, \delta_1 - 1]$ and $\delta_2' \geq 0$, $h^{\delta_1'}(s(\mathtt{x})) \neq h^{\delta_2'}(s(\mathtt{y}))$.

Similarly, as $[\![\mathtt{m_u}(\mathtt{x},\mathtt{y})]\!]_{s,h}$ is also defined (say equal to ℓ') , there are also $\gamma_1, \gamma_2 \geq 0$ such that

- $h^{\gamma_1}(s(\mathtt{x})) = h^{\gamma_2}(s(\mathtt{y})) = \ell'$ and there is $\delta' \geq 0$ such that $h^{\delta'}(\ell') = s(\mathtt{u})$;
- for every $\gamma_1' \in [0, \gamma_1 - 1]$ and $\gamma_2' \geq 0$, $h^{\gamma_1'}(s(\mathtt{x})) \neq h^{\gamma_2'}(s(\mathtt{y}))$.

Combining the two types of inequality constraints, we can conclude that $\delta_1 = \gamma_1$ and therefore $\ell = \ell'$, i.e. $(s,h) \models \mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_u}(\mathtt{x},\mathtt{y})$. Soundness of $\mathcal{H}_{\mathsf{C}}$ is certainly not immediate but this can be done similarly to the above developments for the axiom $(=^{\mathsf{C}}_{\mathbf{6}})$.

▶ **Lemma 8.** *$\mathcal{H}_{\mathsf{C}}$ is sound.*

As done in Section 3, in order to establish that $\mathcal{H}_{\mathsf{C}}$ is complete, we first show its completeness with respect to *core types*, where $\mathsf{CoreTypes}(\mathtt{X},\alpha)$ is here defined as the set of formulae $\{\varphi \in \mathsf{Conj}(\mathsf{Core}(\mathtt{X},\alpha)) \mid \forall\psi{\in}\mathsf{Core}(\mathtt{X},\alpha), \{\psi \mid \neg\psi\} \subseteq_{\mathsf{Lt}} \varphi,\ \text{and}\ (\psi \wedge \neg\psi) \not\subseteq_{\mathsf{Lt}} \varphi\}$.

▶ **Lemma 9.** *Let $\varphi \in \mathsf{CoreTypes}(\mathtt{X},\alpha)$. We have $\neg\varphi$ is valid iff $\vdash_{\mathcal{H}_{\mathsf{C}}} \neg\varphi$. If $\vdash_{\mathcal{H}_{\mathsf{C}}} \neg\varphi$ is provable then it has a proof where all derivation steps only have formulae from $\mathsf{Bool}(\mathsf{Core}(\mathtt{X},\alpha))$.*

Then, the proof of completeness of $\mathcal{H}_{\mathsf{C}}$ follows with the same arguments used for Theorem 2.

▶ **Theorem 10.** *A Boolean combination of core formulae φ is valid iff $\vdash_{\mathcal{H}_{\mathsf{C}}} \varphi$.*

4.4 Constructive elimination of $\exists{:}\rightsquigarrow$

We write $\mathcal{H}_{\mathsf{C}}(\exists{:}\rightsquigarrow)$ to denote the system $\mathcal{H}_{\mathsf{C}}$ augmented by the axioms and the inference rule from System 5. In System 5, given an arbitrary object $\mathtt{O}$ (this can be a term, a set of terms, a formula etc.), we write $\mathsf{var}(\mathtt{O})$ to denote the set of program variables occurring in $\mathtt{O}$. For instance, $\mathsf{var}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) = \{\mathtt{x},\mathtt{y},\mathtt{z}\}$. Axioms from $(\exists_{\mathbf{40}})$ to $(\exists_{\mathbf{42}})$ and the introduction rule are classical tautologies of first-order quantification, whereas the other axioms characterise the peculiar semantics of $\exists{:}\rightsquigarrow$. By way of example, let us explain why the axiom $(\exists_{\mathbf{45}})$, equal to $\mathtt{sees}_{\emptyset}(\mathtt{x},\mathtt{y}) \wedge \mathtt{sees}_{\{\mathtt{y}\}}(\mathtt{x},\mathtt{t}_1) \Rightarrow \exists\mathtt{z}{:}\langle\mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle\ \mathtt{z} = \mathtt{t}_1\ (\mathtt{z} \notin \mathsf{var}(\{\mathtt{x},\mathtt{y},\mathtt{t}_1\}))$ is sound. Suppose $(s,h) \models \mathtt{sees}_{\emptyset}(\mathtt{x},\mathtt{y}) \wedge \mathtt{sees}_{\{\mathtt{y}\}}(\mathtt{x},\mathtt{t}_1)$. By the semantics of core formulae, we have $\emptyset \neq h[s(\mathtt{x}), [\![\mathtt{t}_1]\!]_{s,h}[\subseteq h[s(\mathtt{x}), s(\mathtt{y})[$ and therefore $[\![\mathtt{t}_1]\!]_{s,h}$ is defined. Given $\mathtt{z} \notin \mathsf{var}(\{\mathtt{x},\mathtt{y},\mathtt{t}_1\})$, we have $(s[\mathtt{z} \leftarrow [\![\mathtt{t}_1]\!]_{s,h}], h) \models \mathtt{z} = \mathtt{t}_1$. This holds because $\mathtt{z} \notin \mathsf{var}(\mathtt{t}_1)$ as we want to guarantee $[\![\mathtt{t}_1]\!]_{s,h} = [\![\mathtt{t}_1]\!]_{s[\mathtt{z}\leftarrow[\![\mathtt{t}_1]\!]_{s,h}],h}$. From $\emptyset \neq h[s(\mathtt{x}), [\![\mathtt{t}_1]\!]_{s,h}[\subseteq h[s(\mathtt{x}), s(\mathtt{y})[$, we conclude that $h[s(\mathtt{x}), s(\mathtt{y})[\neq \emptyset$ and $[\![\mathtt{t}_1]\!]_{s,h} \in h[s(\mathtt{x}), s(\mathtt{y})[\cup \{s(\mathtt{y})\}$. Therefore, $(s,h) \models \exists\mathtt{z}{:}\langle\mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle\ \mathtt{z} = \mathtt{t}_1$. As done in Section 3 for $*$ and $-\!\!*$, given a formula $\exists\mathtt{z}{:}\langle\mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle\varphi$, where φ is in $\mathsf{CoreTypes}(\mathtt{X},\alpha)$, we can show within $\mathcal{H}_{\mathsf{C}}(\exists{:}\rightsquigarrow)$ that there is a conjunction χ from $\mathsf{Conj}(\mathsf{Core}(\mathtt{X}, 2\alpha))$ equivalent to it. By the axiom $(\exists_{\mathbf{42}})$, this applies when φ is a Boolean combination of core formulae.

▶ **Lemma 11.** *Let $\varphi \in \mathsf{Bool}(\mathsf{Core}(\mathtt{X} \cup \{\mathtt{z}\}, \alpha))$ with $\mathtt{z} \notin \mathtt{X} \supseteq \{\mathtt{x},\mathtt{y}\}$. There is a Boolean combination of core formulae $\chi \in \mathsf{Bool}(\mathsf{Core}(\mathtt{X}, 2\alpha))$ such that $\vdash_{\mathcal{H}_{\mathsf{C}}(\exists:\rightsquigarrow)} \exists\mathtt{z}{:}\langle\mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle\varphi \Leftrightarrow \chi$.*

4.5 Eliminating $*$ with a big-step axiom

The proof system $\mathcal{H}_{\mathsf{C}}(*, \exists{:}\rightsquigarrow)$ for $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ is defined as $\mathcal{H}_{\mathsf{C}}(\exists{:}\rightsquigarrow)$ augmented by the axioms and the rule from System 6. Its main ingredient is given by the axiom $(*_{\mathbf{48}})$ which, following the description in Section 3.4, is clearly a big-step axiom. Indeed, as much as we would like to give a set of small-step axioms as we did for $\mathsf{SL}(*, -\!\!*)$, we argue that producing

(System 4) $\mathcal{H}_{\mathsf{C}}$: Axioms for Boolean combinations of core formulae

$(=^{\mathsf{C}}_{1})$ $\mathtt{x} = \mathtt{m_x}(\mathtt{x},\mathtt{x})$

$(=^{\mathsf{C}}_{2})$ $\mathtt{t_1} = \mathtt{t_2} \Rightarrow \mathtt{t_2} = \mathtt{t_1}$

$(=^{\mathsf{C}}_{3})$ $\varphi \wedge \mathtt{t_1} = \mathtt{t_2} \Rightarrow \varphi[\mathtt{t_1}\leftarrow\mathtt{t_2}]$

$(=^{\mathsf{C}}_{4})$ $\mathtt{def}(\mathtt{m_x}(\mathtt{x},\mathtt{y})) \Rightarrow \mathtt{x} = \mathtt{m_x}(\mathtt{x},\mathtt{y})$

$(=^{\mathsf{C}}_{5})$ $\mathtt{def}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \Rightarrow \mathtt{def}(\mathtt{m_z}(\mathtt{y},\mathtt{x}))$

$(=^{\mathsf{C}}_{6})$ $\mathtt{def}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \wedge \mathtt{def}(\mathtt{m_u}(\mathtt{x},\mathtt{y})) \Rightarrow \mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_u}(\mathtt{x},\mathtt{y})$

$(=^{\mathsf{C}}_{7})$ $\mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_w}(\mathtt{u},\mathtt{v}) \Rightarrow \mathtt{def}(\mathtt{m_w}(\mathtt{x},\mathtt{y}))$

$(=^{\mathsf{C}}_{8})$ $\mathtt{def}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \wedge \mathtt{def}(\mathtt{m_v}(\mathtt{z},\mathtt{z})) \Rightarrow \mathtt{def}(\mathtt{m_v}(\mathtt{x},\mathtt{y}))$

$(=^{\mathsf{C}}_{9})$ $\mathtt{def}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \wedge \mathtt{def}(\mathtt{m_v}(\mathtt{x},\mathtt{u})) \Rightarrow \mathtt{def}(\mathtt{m_v}(\mathtt{z},\mathtt{z})) \vee \mathtt{def}(\mathtt{m_z}(\mathtt{v},\mathtt{v}))$

$(=^{\mathsf{C}}_{10})$ $\mathtt{def}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \wedge \mathtt{def}(\mathtt{m_z}(\mathtt{u},\mathtt{v})) \Rightarrow \mathtt{def}(\mathtt{m_z}(\mathtt{x},\mathtt{u}))$

$(=^{\mathsf{C}}_{11})$ $\mathtt{sym}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \wedge \mathtt{def}(\mathtt{m_z}(\mathtt{x},\mathtt{u})) \wedge \mathtt{m_z}(\mathtt{x},\mathtt{u}) \neq \mathtt{m_z}(\mathtt{y},\mathtt{u}) \Rightarrow (\mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_z}(\mathtt{x},\mathtt{u}) \vee \mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_z}(\mathtt{y},\mathtt{u}))$

$(=^{\mathsf{C}}_{12})$ $\mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_z}(\mathtt{u},\mathtt{v}) \Rightarrow \mathtt{sym}(\mathtt{m_z}(\mathtt{x},\mathtt{u})) \wedge (\mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_z}(\mathtt{x},\mathtt{u}) \vee \mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_z}(\mathtt{x},\mathtt{v}))$

$(=^{\mathsf{C}}_{13})$ $\mathtt{sym}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \wedge \mathtt{asym}(\mathtt{m_v}(\mathtt{x},\mathtt{u})) \Rightarrow \mathtt{m_v}(\mathtt{y},\mathtt{u}) = \mathtt{m_v}(\mathtt{x},\mathtt{u}) \wedge \mathtt{m_v}(\mathtt{u},\mathtt{y}) = \mathtt{m_v}(\mathtt{u},\mathtt{x})$

$(=^{\mathsf{C}}_{14})$ $\mathtt{asym}(\mathtt{m_z}(\mathtt{x},\mathtt{y})) \wedge \mathtt{asym}(\mathtt{m_v}(\mathtt{x},\mathtt{u})) \Rightarrow \mathtt{m_z}(\mathtt{x},\mathtt{y}) = \mathtt{m_v}(\mathtt{x},\mathtt{u})$

$(\mathtt{s}^{\mathsf{C}}_{15})$ $\mathtt{t}{=}\mathtt{t}' \wedge \mathtt{sees}_{\{\mathtt{t}\}\cup\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) \Rightarrow \mathtt{sees}_{\{\mathtt{t},\mathtt{t}'\}\cup\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2})$

$(\mathtt{s}^{\mathsf{C}}_{16})$ $\mathtt{sees}_{\mathfrak{T}\cup\{\mathtt{t}\}}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta \Rightarrow \mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta$

$(\mathtt{s}^{\mathsf{C}}_{17})$ $\mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta \Rightarrow \mathtt{sees}_{\mathfrak{T}\cup\{\mathtt{t_1},\mathtt{t_2}\}}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta$

$(\mathtt{s}^{\mathsf{C}}_{18})$ $\mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta{+}2 \Rightarrow \mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta{+}1$

$(\mathtt{s}^{\mathsf{C}}_{19})$ $\mathtt{sees}_{\{\mathtt{t_3}\}}(\mathtt{t_1},\mathtt{t_2}) \wedge \mathtt{sees}_{\{\mathtt{t_2}\}}(\mathtt{t_1},\mathtt{t_3}) \Rightarrow \mathtt{t_2} = \mathtt{t_3}$

$(\mathtt{s}^{\mathsf{C}}_{20})$ $\mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) \Rightarrow \mathtt{def}(\mathtt{t_1}) \wedge \mathtt{def}(\mathtt{t_2})$

$(\mathtt{s}^{\mathsf{C}}_{21})$ $\mathtt{sees}_{\emptyset}(\mathtt{t_1},\mathtt{t_1}) {\wedge} \neg \mathtt{sees}_{\{\mathtt{t_2}\}}(\mathtt{t_1},\mathtt{t_1}) \Leftrightarrow \mathtt{samecycle}(\mathtt{t_1},\mathtt{t_2})$

$(\mathtt{s}^{\mathsf{C}}_{22})$ $\mathtt{before}(\mathtt{t_1},\mathtt{t_2}) \Rightarrow \mathtt{sees}_{\emptyset}(\mathtt{t_1},\mathtt{t_2})$

$(\mathtt{s}^{\mathsf{C}}_{23})$ $\mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta \wedge \mathtt{sees}_{\mathfrak{T}'}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta' \Rightarrow \mathtt{sees}_{\mathfrak{T}\cup\mathfrak{T}'}(\mathtt{t_1},\mathtt{t_2}) {\geq} \max(\beta,\beta')$

$(\mathtt{s}^{\mathsf{C}}_{24})$ $\mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta_1 {\wedge}\, \mathtt{sees}_{\mathfrak{T}}(\mathtt{t_2},\mathtt{t_3}) {\geq} \beta_2 \wedge \mathtt{t_2} {\notin} \mathfrak{T} \wedge \mathtt{t_3} {\in} \mathfrak{T} \Rightarrow \mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_3}) {\geq} \beta_1{+}\beta_2 \wedge \neg \mathtt{sees}_{\{\mathtt{t_2}\}}(\mathtt{t_1},\mathtt{t_3})$

$(\mathtt{s}^{\mathsf{C}}_{25})$ $\mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_3}) {\geq} \beta \wedge \neg \mathtt{sees}_{\{\mathtt{t_2}\}}(\mathtt{t_1},\mathtt{t_3}) \Rightarrow \bigvee_{\beta_1+\beta_2=\max(2,\beta)-2} (\mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta_1{+}1 \wedge \mathtt{sees}_{\mathfrak{T}}(\mathtt{t_2},\mathtt{t_3}) {\geq} \beta_2{+}1)$

$(\mathtt{s}^{\mathsf{C}}_{26})$ $\mathtt{sees}_{\mathfrak{T}}(\mathtt{m_z}(\mathtt{x},\mathtt{y}),\mathtt{m_w}(\mathtt{u},\mathtt{v})) \Rightarrow \mathtt{def}(\mathtt{m_w}(\mathtt{x},\mathtt{y}))$

$(\mathtt{s}^{\mathsf{C}}_{27})$ $\mathtt{sees}_{\mathfrak{T}}(\mathtt{m_z}(\mathtt{x},\mathtt{y}),\mathtt{m_w}(\mathtt{u},\mathtt{v})) \wedge \mathtt{asym}(\mathtt{m_w}(\mathtt{x},\mathtt{u})) \Rightarrow \mathtt{m_w}(\mathtt{u},\mathtt{v}) = \mathtt{m_w}(\mathtt{u},\mathtt{x})$

$(\mathtt{s}^{\mathsf{C}}_{28})$ $\mathtt{sees}_{\mathfrak{T}}(\mathtt{m_z}(\mathtt{x},\mathtt{y}),\mathtt{m_w}(\mathtt{u},\mathtt{v})) \wedge \mathtt{sym}(\mathtt{m_w}(\mathtt{x},\mathtt{u})) \wedge \mathtt{m_z}(\mathtt{x},\mathtt{y}) \neq \mathtt{m_w}(\mathtt{u},\mathtt{v}) \Rightarrow \mathtt{before}(\mathtt{m_z}(\mathtt{x},\mathtt{y}),\mathtt{m_w}(\mathtt{u},\mathtt{v}))$

$(\mathtt{s}^{\mathsf{C}}_{29})$ $\mathtt{before}(\mathtt{t_1},\mathtt{t_2}) \wedge \neg \mathtt{sees}_{\{\mathtt{t_3}\}}(\mathtt{t_1},\mathtt{t_2}) \Rightarrow \neg \mathtt{sees}_{\emptyset}(\mathtt{t_2},\mathtt{t_3}) \wedge \mathtt{before}(\mathtt{t_1},\mathtt{t_3})$

$(\mathtt{s}^{\mathsf{C}}_{30})$ $\mathtt{samecycle}(\mathtt{t_1},\mathtt{t_2}) \wedge \mathtt{samecycle}(\mathtt{t_2},\mathtt{t_3}) \wedge \mathtt{t_1} \neq \mathtt{t_3} \Rightarrow (\mathtt{sees}_{\{\mathtt{t_2}\}}(\mathtt{t_1},\mathtt{t_3}) \Leftrightarrow \neg \mathtt{sees}_{\{\mathtt{t_2}\}}(\mathtt{t_3},\mathtt{t_1}))$

$(\mathtt{r}^{\mathsf{C}}_{31})$ $\mathtt{rem}_{\mathfrak{P}} {\geq} 0$

$(\mathtt{r}^{\mathsf{C}}_{32})$ $\mathtt{rem}_{\mathfrak{P}} {\geq} \beta + 1 \Rightarrow \mathtt{rem}_{\mathfrak{P}} {\geq} \beta$

$(\mathtt{r}^{\mathsf{C}}_{33})$ $\mathtt{rem}_{\{(\mathtt{t_1},\mathtt{t_2})\}\cup\mathfrak{P}} {\geq} \beta \Rightarrow \mathtt{rem}_{\mathfrak{P}} {\geq} \beta$

$(\mathtt{r}^{\mathsf{C}}_{34})$ $\mathtt{t_1} = \mathtt{t_2} \wedge \mathtt{rem}_{\{(\mathtt{t_1},\mathtt{t_3})\}\cup\mathfrak{P}} {\geq} \beta \Rightarrow \mathtt{rem}_{\{(\mathtt{t_1},\mathtt{t_3}),(\mathtt{t_2},\mathtt{t_3})\}\cup\mathfrak{P}} {\geq} \beta$

$(\mathtt{r}^{\mathsf{C}}_{35})$ $\mathtt{t_1} = \mathtt{t_2} \wedge \mathtt{rem}_{\{(\mathtt{t_3},\mathtt{t_1})\}\cup\mathfrak{P}} {\geq} \beta \Rightarrow \mathtt{rem}_{\{(\mathtt{t_3},\mathtt{t_1}),(\mathtt{t_3},\mathtt{t_2})\}\cup\mathfrak{P}} {\geq} \beta$

$(\mathtt{r}^{\mathsf{C}}_{36})$ $\neg \mathtt{sees}_{\emptyset}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta_2{+}1 \wedge \mathtt{rem}_{\mathfrak{P}} {\geq} \beta_1 \Rightarrow \mathtt{rem}_{\mathfrak{P}\cup\{(\mathtt{t_1},\mathtt{t_2})\}} {\geq} \beta_1 \dot{-} \beta_2$

$(\mathtt{r}^{\mathsf{C}}_{37})$ $\mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) \wedge \neg \mathtt{sees}_{\{\mathtt{t_3}\}}(\mathtt{t_1},\mathtt{t_2}) \wedge \mathtt{rem}_{\{(\mathtt{t_1},\mathtt{t_3}),(\mathtt{t_3},\mathtt{t_2})\}\cup\mathfrak{P}} {\geq} \beta \Rightarrow \mathtt{rem}_{\{(\mathtt{t_1},\mathtt{t_2})\}\cup\mathfrak{P}} {\geq} \beta$

$(\mathtt{r}^{\mathsf{C}}_{38})$ $\mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) \wedge \neg \mathtt{sees}_{\{\mathtt{t_3}\}}(\mathtt{t_1},\mathtt{t_2}) \wedge \mathtt{rem}_{\{(\mathtt{t_1},\mathtt{t_2})\}\cup\mathfrak{P}} {\geq} \beta \Rightarrow \mathtt{rem}_{\{(\mathtt{t_1},\mathtt{t_2}),(\mathtt{t_1},\mathtt{t_3}),(\mathtt{t_3},\mathtt{t_2})\}\cup\mathfrak{P}} {\geq} \beta$

$(\mathtt{r}^{\mathsf{C}}_{39})$ $\big(\mathtt{sees}_{\emptyset}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta_2 \wedge \bigwedge_{(\mathtt{t_3},\mathtt{t_4})\in\mathfrak{P}} (\mathtt{sees}_{\emptyset}(\mathtt{t_3},\mathtt{t_4}) \Rightarrow \mathtt{sees}_{\{\mathtt{t_3},\mathtt{t_4}\}}(\mathtt{t_1},\mathtt{t_2}) \wedge \mathtt{sees}_{\{\mathtt{t_1},\mathtt{t_2}\}}(\mathtt{t_3},\mathtt{t_4}) \wedge \mathtt{t_3} {\neq} \mathtt{t_1})$
$\wedge\, \mathtt{rem}_{\mathfrak{P}\cup\{(\mathtt{t_1},\mathtt{t_2})\}} {\geq} \beta_1\big) \Rightarrow \mathtt{rem}_{\mathfrak{P}} {\geq} \beta_1{+}\beta_2$

(System 5) Axioms and inference rule for the guarded quantification $\exists{:}\rightsquigarrow$

$(\exists_{40})$ $\exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle \varphi \Rightarrow \exists \mathtt{u}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle (\varphi[\mathtt{z}\leftarrow\mathtt{u}])$ $\blacktriangleleft[\mathtt{u} \notin \mathsf{var}(\varphi)]$

$(\exists_{41})$ $\exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle (\varphi \wedge \psi) \Leftrightarrow (\exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle \varphi) \wedge \psi$ $\blacktriangleleft[\mathtt{z} \notin \mathsf{var}(\psi)]$

$(\exists_{42})$ $\exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle (\varphi_1 \vee \varphi_2) \Leftrightarrow (\exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle \varphi_1) \vee (\exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle \varphi_2)$

$\exists$-Intro: $\dfrac{\varphi \Rightarrow \psi}{\exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle \varphi \Rightarrow \exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle \psi}$

$(\exists_{43})$ $\mathtt{sees}_{\emptyset}(\mathtt{x},\mathtt{y}) \Rightarrow \exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle\ \mathtt{z} = \mathtt{x}$ $\blacktriangleleft[\mathtt{z} \notin \{\mathtt{x},\mathtt{y}\}]$

$(\exists_{44})$ $\neg \exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle\ \bot$

$(\exists_{45})$ $\mathtt{sees}_{\emptyset}(\mathtt{x},\mathtt{y}) \wedge \mathtt{sees}_{\{\mathtt{y}\}}(\mathtt{x},\mathtt{t_1}) \Rightarrow \exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle\ \mathtt{z} = \mathtt{t_1}$ $\blacktriangleleft[\mathtt{z} \notin \mathsf{var}(\{\mathtt{x},\mathtt{y},\mathtt{t_1}\})]$

$(\exists_{46})$ $(\mathtt{x}{=}\mathtt{t_1} \vee \mathtt{sees}_{\mathfrak{T}'}(\mathtt{x},\mathtt{t_1})) \wedge \mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{t_2}) {\geq} \beta_1{+}\beta_2 \wedge (\mathtt{t_2}{=}\mathtt{y} \vee \mathtt{sees}_{\mathfrak{T}''}(\mathtt{t_2},\mathtt{y})) \wedge (\mathtt{y}{=}\mathtt{t_1} \Rightarrow \mathtt{x}{=}\mathtt{y})$
$\Rightarrow \exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle (\mathtt{sees}_{\mathfrak{T}}(\mathtt{t_1},\mathtt{z}) \sim_1 \beta_1 \wedge \mathtt{sees}_{\mathfrak{T}}(\mathtt{z},\mathtt{t_2}) \sim_2 \beta_2 \wedge \mathtt{z} \notin \{\mathtt{t_1},\mathtt{t_2}\})$
$\blacktriangleleft\big[\{\mathtt{x},\mathtt{y},\mathtt{t_1},\mathtt{t_2}\} \subseteq \mathfrak{T},\mathfrak{T}',\mathfrak{T}'',\ \mathtt{z} \notin \mathsf{var}(\mathfrak{T}),\ \beta_1,\beta_2 \in \mathbb{N}^+,\ \geq\, \in \{\sim_1,\sim_2\} \subseteq \{\geq,=\}\big]$

$(\exists_{47})$ $\neg \exists \mathtt{z}{:}\langle \mathtt{x}{\rightsquigarrow}\mathtt{y}\rangle ((\mathtt{x} \neq \mathtt{z} \wedge \mathtt{y} \neq \mathtt{z} \wedge \mathtt{sees}_{\{\mathtt{x},\mathtt{z},\mathtt{y}\}}(\mathtt{x},\mathtt{y})) \vee \neg \mathtt{sees}_{\emptyset}(\mathtt{x},\mathtt{y}))$ $\blacktriangleleft[\mathtt{z} \notin \{\mathtt{x},\mathtt{y}\}]$

(System 6) Axioms and inference rule for the separating conjunction

$(*_{48})$ $\Gamma_{\mathsf{sms}}(\mathfrak{S}_1) * \Gamma_{\mathsf{sms}}(\mathfrak{S}_2) \Leftrightarrow \bigvee_{\mathfrak{S} \text{ s.t. } +^{\mathsf{s}}(\mathfrak{S}_1,\mathfrak{S}_2,\mathfrak{S})} \Gamma_{\mathsf{sms}}(\mathfrak{S})$
$\blacktriangleleft[\mathfrak{S}_1,\mathfrak{S}_2$ resp. over $(\mathtt{X},\alpha_1)$ and $(\mathtt{X},\alpha_2)]$

$*$-Intro: $\dfrac{\varphi \Rightarrow \chi}{\varphi * \psi \Rightarrow \chi * \psi}$

$(*_{49})$ $(\varphi \vee \psi) * \chi \Rightarrow (\varphi * \chi) \vee (\psi * \chi)$

$(*_{50})$ $(\varphi * \psi) \Leftrightarrow (\psi * \varphi)$

$(*_{51})$ $(\bot * \varphi) \Leftrightarrow \bot$

such an axiomatisation for $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$ is unfeasible. In the proof system for $\mathsf{SL}(*, \mathrel{-\!\!*})$, we found out that given two core types φ and ψ, $\varphi * \psi$ is equivalent to a conjunction of core formulae literals (see the proof sketch of Lemma 3). Similar results hold for the separating implication $\mathrel{-\!\!*}$ (Lemma 5) and the $\exists{:}\rightsquigarrow$ quantifier. This property of being equivalent to a simple conjunction of core formulae literals facilitates the design of small-step axioms. This is not the case for $*$ within $\mathsf{SL}(*, \exists{:}\rightsquigarrow)$: given two core types φ and ψ, the formula $\varphi * \psi$ is equivalent to a non-trivial disjunction of possibly exponentially many conjunctions. Because of this, small-step axioms are hard to obtain and some technical developments are needed in order to produce an adequate axiom system. These developments are centered around the notions of symbolic memory states and characteristic formulae. A *symbolic memory state* is an abstraction on the memory state (s, h) that is guided by the definition of core formulae, essentially highlighting the properties of (s, h) that are expressible through these formulae, while removing the ones that are not expressible. Given $\mathtt{X} \subseteq_{\mathsf{fin}} \mathsf{VAR}$ and $\alpha \in \mathbb{N}^+$, a *symbolic memory states* $\mathfrak{S}$ over $(\mathtt{X}, \alpha)$ is defined as a finite structure $(\mathfrak{D}, \mathfrak{f}, \mathfrak{r})$ such that

- $\mathfrak{D}$ is a partition of a subset of $\mathbb{T}(\mathtt{X})$, encoding (dis)equalities. We introduce the partial function $[\ .\ ]_{\mathfrak{D}} : \mathbb{T}(\mathtt{X}) \to \mathfrak{D}$ such that given $\mathtt{t} \in \mathbb{T}(\mathtt{X})$ returns $\mathsf{T} \in \mathfrak{D}$ and $\mathtt{t} \in \mathsf{T}$, if it exists;
- $\mathfrak{f} : \mathfrak{D} \to \mathfrak{D} \times [1, \alpha]$ is a partial function encoding paths between terms and their length;
- $\mathfrak{r} \in [0, \alpha]$, encoding the number of memory cells (up to α) not in paths between terms.

We denote with $\mathsf{SMS}^{\mathtt{X}}_{\alpha}$ the set of these structures. The *abstraction* $\mathtt{Symb}^{\mathtt{X}}_{\alpha}(s, h)$ of a memory state (s, h) is defined as the symbolic memory state $(\mathfrak{D}, \mathfrak{f}, \mathfrak{r})$ over $(\mathtt{X}, \alpha)$ such that

- $\mathfrak{D} \stackrel{\text{def}}{=} \{\{\mathtt{t}_1 \in \mathbb{T}(\mathtt{X}) \mid (s, h) \models \mathtt{t}_1 = \mathtt{t}_2\} \mid \mathtt{t}_2 \in \mathbb{T}(\mathtt{X})\}$;
- $\mathfrak{f}(\mathsf{T}) = (\mathsf{T}', \beta) \ \stackrel{\text{def}}{\Leftrightarrow}$ there are $\mathtt{t}_1{\in}\mathsf{T}$ and $\mathtt{t}_2{\in}\mathsf{T}'$ such that $(s, h) \models \mathtt{sees}_{\mathbb{T}(\mathtt{X})}(\mathtt{t}_1, \mathtt{t}_2){\geq}\beta$ and if $\beta < \alpha$ then $(s, h) \models \neg\mathtt{sees}_{\mathbb{T}(\mathtt{X})}(\mathtt{t}_1, \mathtt{t}_2){\geq}\beta{+}1$;
- $\mathfrak{r} = \beta \ \stackrel{\text{def}}{\Leftrightarrow} \ (s, h) \models \mathtt{rem}_{\mathbb{T}(\mathtt{X})\times\mathbb{T}(\mathtt{X})}{\geq}\beta$ and if $\beta < \alpha$ then $(s, h) \models \neg\mathtt{rem}_{\mathbb{T}(\mathtt{X})\times\mathbb{T}(\mathtt{X})}{\geq}\beta{+}1$.

Thus, a symbolic memory state $(\mathfrak{D}, \mathfrak{f}, \mathfrak{r})$ over $(\mathtt{X}, \alpha)$ simply stores the truth values for equalities, `sees` and `rem` predicates with respect to a memory state. Its semantics is best given through the *characteristic formula* $\Gamma_{\mathtt{sms}}(\mathfrak{D}, \mathfrak{f}, \mathfrak{r})$ defined below (sets understood as conjunctions):

$$\begin{aligned}
&\big\{\mathtt{rem}_{\mathbb{T}(\mathtt{X})\times\mathbb{T}(\mathtt{X})}{\sim}\mathfrak{r} \,\big|\, \text{if } \mathfrak{r}{\neq}\alpha \text{ then } (\sim \text{ is } =) \text{ else } (\sim \text{ is } \geq)\big\} \wedge \big\{\mathtt{t}_1 \neq \mathtt{t}_2 \,\big|\, [\mathtt{t}_1]_{\mathfrak{D}} \text{ or } [\mathtt{t}_2]_{\mathfrak{D}} \text{ undefined, or } [\mathtt{t}_1]_{\mathfrak{D}} \neq [\mathtt{t}_2]_{\mathfrak{D}}\big\}\\
&\wedge\big\{\mathtt{t}_1 = \mathtt{t}_2 \,\big|\, [\mathtt{t}_1]_{\mathfrak{D}} = [\mathtt{t}_2]_{\mathfrak{D}} \text{ defined}\big\} \wedge \big\{\neg\mathtt{sees}_{\mathbb{T}(\mathtt{X})}(\mathtt{t}_1, \mathtt{t}_2) \,\big|\, [\mathtt{t}_1]_{\mathfrak{D}} \text{ undefined or } \forall\beta \in [1, \alpha] : \mathfrak{f}([\mathtt{t}_1]_{\mathfrak{D}}) \neq ([\mathtt{t}_2]_{\mathfrak{D}}, \beta)\big\}\\
&\wedge\big\{\mathtt{sees}_{\mathbb{T}(\mathtt{X})}(\mathtt{t}_1, \mathtt{t}_2){=}\beta \,\big|\, \mathfrak{f}([\mathtt{t}_1]_{\mathfrak{D}}) = ([\mathtt{t}_2]_{\mathfrak{D}}, \beta) \text{ and } \beta < \alpha\big\} \wedge \big\{\mathtt{sees}_{\mathbb{T}(\mathtt{X})}(\mathtt{t}_1, \mathtt{t}_2){\geq}\beta \,\big|\, \mathfrak{f}([\mathtt{t}_1]_{\mathfrak{D}}) = ([\mathtt{t}_2]_{\mathfrak{D}}, \beta) \text{ and } \beta = \alpha\big\}
\end{aligned}$$

From the definitions of $\Gamma_{\mathtt{sms}}(\mathfrak{S})$ and $\mathtt{Symb}^{\mathtt{X}}_{\alpha}(s, h)$, we can easily prove the following result.

▶ **Lemma 12.** *For every (s, h) and every $\mathfrak{S} \in \mathsf{SMS}^{\mathtt{X}}_{\alpha}$, $(s, h) \models \Gamma_{\mathtt{sms}}(\mathfrak{S})$ iff $\mathfrak{S} = \mathtt{Symb}^{\mathtt{X}}_{\alpha}(s, h)$.*

Thanks to this lemma, it is easy to see that every satisfiable characteristic formula $\Gamma_{\mathtt{sms}}(\mathfrak{S})$ of a symbolic memory state $\mathfrak{S}$ over $(\mathtt{X}, \alpha)$ is equivalent to exactly one core type in $\mathsf{CoreTypes}(\mathtt{X}, \alpha)$. Indeed, by definition of core types, the conjunction $\varphi \wedge \psi$ of two core types φ and ψ that are not syntactically equivalent up to associativity and commutativity of $\wedge$ is unsatisfiable. Hence, by Lemma 12, if a core type $\varphi \in \mathsf{CoreTypes}(\mathtt{X}, \alpha)$ is satisfied by a memory state (s, h), it must be equivalent to $\Gamma_{\mathtt{sms}}(\mathtt{Symb}^{\mathtt{X}}_{\alpha}(s, h))$. By Theorem 10 this equivalence is provable in $\mathcal{H}_{\mathsf{C}}$.

The fundamental reason for taking symbolic memory states over memory states is that, given $\mathtt{X}$ and α, there are finitely many symbolic memory states in $\mathsf{SMS}^{\mathtt{X}}_{\alpha}$. This leads to the definition of the axiom $(*_{\mathbf{48}})$, which given two characteristic formulae φ and ψ computes a finite disjunction of characteristic formulae that is equivalent to $\varphi * \psi$. This disjunction is defined over a new composition operator $+^{\mathsf{S}}$ on symbolic memory states that mimicks the disjoint union $+$ on memory states. More precisely, the following property shall be satisfied.

For all (s, h) and all $\mathfrak{S}_1, \mathfrak{S}_2$ resp. over $(\mathtt{X}, \alpha_1)$ and $(\mathtt{X}, \alpha_2)$, $+^{\mathsf{S}}(\mathfrak{S}_1, \mathfrak{S}_2, \mathtt{Symb}^{\mathtt{X}}_{\alpha_1+\alpha_2}(s, h))$ iff there are h_1 and h_2 such that $h_1 + h_2 = h$, $\mathfrak{S}_1 = \mathtt{Symb}^{\mathtt{X}}_{\alpha_1}(s, h_1)$ and $\mathfrak{S}_2 = \mathtt{Symb}^{\mathtt{X}}_{\alpha_2}(s, h_2)$, where $+^{\mathsf{S}} \subseteq \sum_{\mathtt{X},\alpha_1,\alpha_2} \mathsf{SMS}^{\mathtt{X}}_{\alpha_1} \times \mathsf{SMS}^{\mathtt{X}}_{\alpha_2} \times \mathsf{SMS}^{\mathtt{X}}_{\alpha_1+\alpha_2}$, and $\mathfrak{S}_1$, $\mathfrak{S}_2$ have satisfiable characteristic formulae. Defining $+^{\mathsf{S}}$ is clearly challenging. Unlike the disjoint union of memory states, $+^{\mathsf{S}}$ is not functional on its first two components. For instance, let $\mathfrak{S} = (\{\mathtt{x}, \mathtt{m}_{\mathtt{x}}(\mathtt{x},\mathtt{x})\}, \emptyset, 1)$ and let us determine for which $\mathfrak{S}'$, we have $+^{\mathsf{S}}(\mathfrak{S},\mathfrak{S},\mathfrak{S}')$:

1. As $\mathfrak{S}$ is the abstraction of the memory states $(s, \{\ell_1 \mapsto \ell_2\})$ and $(s, \{\ell_2 \mapsto \ell_1)\})$ where $s(\mathtt{x}) = \ell_1 \neq \ell_2$, the abstraction of $(s, \{\ell_1 \mapsto \ell_2, \ell_2 \mapsto \ell_1\})$ must be a solution for $\mathfrak{S}'$. More precisely, this abstraction is $(\mathsf{T}, \{\mathsf{T} \mapsto (\mathsf{T}, 2)\}, 0)$ where $\mathsf{T} = \{\mathtt{x}, \mathtt{m}_{\mathtt{x}}(\mathtt{x},\mathtt{x})\}$.
2. $\mathfrak{S}$ is however also the abstraction of $(s, \{\ell_1 \mapsto \ell_2\})$ and $(s, \{\ell_3 \mapsto \ell_4)\})$ such that $s(\mathtt{x}) \notin \{\ell_1, \ell_3\}$. Then, the abstraction $(\{\mathtt{x}, \mathtt{m}_{\mathtt{x}}(\mathtt{x},\mathtt{x})\}, \emptyset, 2)$ must also be a solution for $\mathfrak{S}'$.

The main challenge for defining $+^{\mathsf{S}}$ is the composition of the two "garbage": memory cells that are abstracted with $\mathfrak{r}_1$ and $\mathfrak{r}_2$ in $\mathtt{Symb}^{\mathtt{X}_1}_{\alpha_1}(s, h_1)$ and $\mathtt{Symb}^{\mathtt{X}_2}_{\alpha_2}(s, h_2)$ may generate new paths between program variables in $h_1 + h_2$. This possibility was depicted in the first case above. The definition of $+^{\mathsf{S}}$ can be found in [17] and is too long to be presented herein. Roughly speaking, for $((\mathfrak{D}, \mathfrak{f}_1, \mathfrak{r}_1), (\mathfrak{D}, \mathfrak{f}_2, \mathfrak{r}_2), (\mathfrak{D}, \mathfrak{f}, \mathfrak{r}))$ being in $+^{\mathsf{S}}$, one needs to witness two graph homomorphisms from the graphs $(\mathfrak{D}_1, \mathfrak{f}_1)$ and $(\mathfrak{D}_2, \mathfrak{f}_2)$ to $(\mathfrak{D}, \mathfrak{f})$, together with the existence of a partition that guarantees that paths that do not belong to the homomorphisms can be generated using the memory cells from the garbage (abstracted by $\mathfrak{r}_1$ and $\mathfrak{r}_2$).

Together with the other axioms in System 6, which essentially allows to rewrite every formula into a disjunction of $\varphi * \psi$ where φ and ψ are characteristic formulae, the axiom $(*_{\mathbf{48}})$ allows us to eliminate $*$, as done in Lemma 3 for $\mathsf{SL}(*, -\!\!*)$.

▶ **Lemma 13.** *Let $\varphi \in \mathsf{Bool}(\mathsf{Core}(\mathtt{X}, \alpha_1))$ and $\psi \in \mathsf{Bool}(\mathsf{Core}(\mathtt{X}, \alpha_2))$. There is a Boolean combination of core formulae $\chi \in \mathsf{Bool}(\mathsf{Core}(\mathtt{X}, \alpha_1 + \alpha_2))$ such that $\vdash_{\mathcal{H}_{\mathsf{C}}(*,\exists:\leadsto)} \varphi * \psi \Leftrightarrow \chi$.*

The adequacy of $\mathcal{H}_{\mathsf{C}}(*, \exists:\leadsto)$ then stems from Theorem 10 and Lemmata 11 and 13.

▶ **Theorem 14.** *$\mathcal{H}_{\mathsf{C}}(*, \exists:\leadsto)$ is sound and complete for $\mathsf{SL}(*, \exists:\leadsto)$.*

4.6 A PSpace upper bound for checking SL($*$, $\exists$:$\leadsto$) satisfiability

In this short section, we explain why the satisfiability problem for $\mathsf{SL}(*, \exists:\leadsto)$ is in PSPACE. The *memory size* of a formula φ, written $|\varphi|_{\mathtt{m}}$, is defined inductively as: $|\mathtt{x} = \mathtt{y}|_{\mathtt{m}} \stackrel{\text{def}}{=} |\mathtt{emp}|_{\mathtt{m}} = 1$, $|\mathtt{x} \hookrightarrow \mathtt{y}|_{\mathtt{m}} \stackrel{\text{def}}{=} 2$, $|\exists \mathtt{z}{:}\langle \mathtt{x} \leadsto \mathtt{y}\rangle \varphi|_{\mathtt{m}} \stackrel{\text{def}}{=} 2 \times |\varphi|_{\mathtt{m}}$, $|\neg\psi|_{\mathtt{m}} \stackrel{\text{def}}{=} |\psi|_{\mathtt{m}}$, $|\psi_1 * \psi_2|_{\mathtt{m}} \stackrel{\text{def}}{=} |\psi_1|_{\mathtt{m}} + |\psi_2|_{\mathtt{m}}$ and $|\psi_1 \wedge \psi_2|_{\mathtt{m}} \stackrel{\text{def}}{=} \max(|\psi_1|_{\mathtt{m}}, |\psi_2|_{\mathtt{m}})$. Given φ with tree height δ, $|\varphi|_{\mathtt{m}} \leq 2^{\delta+1}$. Intuitively, $|\varphi|_{\mathtt{m}}$ provides an upper bound on the path length between terms and on the size of the garbage on models for φ (above $|\varphi|_{\mathtt{m}}$, φ cannot see the difference). As a consequence of the proofs for the elimination of the connectives $\exists:\leadsto$ and $*$ in the calculus, for each φ in $\mathsf{SL}(*, \exists:\leadsto)$, there is a Boolean combination of core formulae from $\mathsf{Core}(\mathsf{var}(\varphi), |\varphi|_{\mathtt{m}})$ logically equivalent to φ.

$\mathsf{SL}(*, \exists:\leadsto)$ may require small memory states whose heap has an exponential amount of memory cells, as shown in Section 4.1 with the formula $\mathrm{R}^n(\mathtt{x}, \mathtt{y})$. So, to establish a PSPACE bound, we cannot rely on an algorithm that guesses a polynomial-size memory state and performs model-checking on it without further refinements. Nevertheless, polynomial-size symbolic memory states are able to abstract a garbage of exponential size or a path between terms of exponential length by encoding these quantities in binary, which leads to PSPACE.

▶ **Theorem 15.** *The satisfiability problem for $\mathsf{SL}(*, \exists:\leadsto)$ is* PSPACE*-complete.*

PSPACE-hardness is from [10]. To establish PSPACE-easiness, there is a nondeterministic polynomial-space algorithm that guesses a satisfiable $\mathfrak{S} \in \mathsf{SMS}^{\mathsf{var}(\varphi)}_{|\varphi|_{\mathtt{m}}}$ and that performs a symbolic model-checking on $\mathfrak{S}$ against φ. This works fine as $*$ and $\exists:\leadsto$ have symbolic counterparts that can be decided in polynomial space.

5 Conclusion

We presented a method to axiomatise internally separation logics based on the axiomatisation of Boolean combinations of core formulae. We designed the first proof system for $\mathsf{SL}(*, -\!\!*)$ that is completely internal and highlights the essential ingredients of the heaplet semantics. To further illustrate our method, we provided an internal Hilbert-style axiomatisation for the new separation logic $\mathsf{SL}(*, \exists{:}\!\leadsto)$. It contains the "list quantifier" $\exists \mathtt{z}{:}\langle \mathtt{x}\!\leadsto\!\mathtt{y}\rangle$ that, we believe, is of interest for its own sake as it allows to quantify over elements of a list. The completeness proof, following our general pattern, still reveals to be very complex as not only we had to invent the adequate family of core formulae but their axiomatisation was challenging. As far as we know, this is the first axiomatisation of a separation logic having `ls` and a guarded form of quantification. Moreover, through a small model property derived from its proof system, we proved that $\mathsf{SL}(*, \exists{:}\!\leadsto)$ has a PSpace-complete satisfiability problem. Obviously, our proof systems for separation logics are of theoretical interest, for instance to grasp the essential features of the logics. It is open whether it can help for designing decision procedures, e.g. to feed provers with axiom instances to speed-up the proof search.

References

1. C. Areces, P. Blackburn, and M. Marx. Hybrid logics: characterization, interpolation and complexity. *The Journal of Symbolic Logic*, 66(3):977–1010, 2001.
2. J. Berdine, C. Calcagno, and P.W. O'Hearn. A decidable fragment of separation logic. In *FST&TCS'04*, volume 3328 of *LNCS*, pages 97–109. Springer, 2004.
3. M. Bozga, R. Iosif, and S. Perarnau. Quantitative Separation Logic and Programs with Lists. *Journal of Automated Reasoning*, 45(2):131–156, 2010.
4. R. Brochenin, S. Demri, and E. Lozes. On the Almighty Wand. *Information and Computation*, 211:106–137, 2012.
5. J. Brotherston. Bunched Logics Displayed. *Studia Logica*, 100(6):1223–1254, 2012. doi:10.1007/s11225-012-9449-0.
6. J. Brotherston and M. Kanovich. Undecidability of Propositional Separation Logic and Its Neighbours. *Journal of the Association for Computing Machinery*, 61(2), 2014.
7. J. Brotherston and M. Kanovich. On the Complexity of Pointer Arithmetic in Separation Logic. In *APLAS'18*, volume 11275 of *LNCS*, pages 329–349. Springer, 2018.
8. J. Brotherston and J. Villard. Parametric completeness for separation theories. In *POPL'14*, pages 453–464. ACM, 2014.
9. C. Calcagno, Ph. Gardner, and M. Hague. From separation logic to first-order logic. In *FoSSaCS'05*, volume 3441 of *LNCS*, pages 395–409. Springer, 2005.
10. C. Calcagno, P.W. O'Hearn, and H. Yang. Computability and Complexity Results for a Spatial Assertion Language for Data Structures. In *FST&TCS'01*, volume 2245 of *LNCS*, pages 108–119. Springer, 2001.
11. B. Cook, C. Haase, J. Ouaknine, M. Parkinson, and J. Worrell. Tractable Reasoning in a Fragment of Separation Logic. In *CONCUR'11*, volume 6901 of *LNCS*, pages 235–249. Springer, 2011.
12. S. Demri and M. Deters. Separation Logics and Modalities: A Survey. *Journal of Applied Non-Classical Logics*, 25(1):50–99, 2015.
13. S. Demri, R. Fervari, and A. Mansutti. Axiomatising Logics with Separating Conjunction and Modalities. In *JELIA'19*, volume 11468 of *LNAI*, pages 692–708. Springer, 2019.
14. S. Demri, D. Galmiche, D. Larchey-Wendling, and D. Mery. Separation Logic with One Quantified Variable. *Theory of Computing Systems*, 61:371–461, 2017.

15 S. Demri, É. Lozes, and A. Mansutti. The Effects of Adding Reachability Predicates in Propositional Separation Logic. In *FoSSaCS*, volume 10803 of *LNCS*, pages 476–493. Springer, 2018.
16 S. Demri, E. Lozes, and A. Mansutti. The Effects of Adding Reachability Predicates in Propositional Separation Logic. arXiv:1810.05410, October 2018. 44 pages. Long version of [15].
17 S. Demri, E. Lozes, and A. Mansutti. Internal Calculi for Separation logics. arXiv:1910.05016, October 2019.
18 S. Docherty and D. Pym. Modular Tableaux Calculi for Separation Theories. In *FoSSaCS'18*, volume 10803 of *LNCS*, pages 441–458. Springer, 2018.
19 A. Doumane. Constructive completeness for the linear-time μ-calculus. In *LICS'17*, pages 1–12. IEEE Computer Society, 2017.
20 M. Echenim, R. Iosif, and N. Peltier. The Bernays-Schönfinkel-Ramsey Class of Separation Logic on Arbitrary Domains. In *FoSSaCS'19*, volume 11425 of *LNCS*, pages 242–259. Springer, 2019.
21 D. Galmiche and D. Larchey-Wending. Expressivity properties of Boolean BI through relational models. In *FST&TCS'06*, volume 4337 of *LNCS*, pages 358–369. Springer, 2006.
22 D. Galmiche and D. Méry. Tableaux and Resource Graphs for Separation Logic. *Journal of Logic and Computation*, 20(1):189–231, 2010.
23 V. Goranko and G. van Drimmelen. Complete axiomatization and decidability of Alternating-time temporal logic. *Theoretical Computer Science*, 353(1-3):93–117, 2006.
24 E. Grädel and I. Walukiewicz. Guarded Fixed Point Logic. In *LICS'99*, pages 45–54, 1999.
25 Z. Hou, R. Clouston, R. Goré, and A. Tiu. Modular labelled sequent calculi for abstract separation logics. *ACM Transactions on Computational Logic*, 19(2):13:1–13:35, 2018.
26 Z. Hou and A. Tiu. Completeness for a First-Order Abstract Separation Logic. In *APLAS'16*, volume 10017 of *LNCS*, pages 444–463. Springer, 2016.
27 S. Ishtiaq and P.W. O'Hearn. BI as an assertion language for mutable data structures. In *POPL'01*, pages 14–26. ACM, 2001.
28 R. Kaivola. Axiomatising linear time mu-calculus. In *CONCUR'95*, volume 962 of *LNCS*, pages 423–437. Springer, 1995.
29 E. Lozes. *Expressivité des Logiques Spatiales*. PhD thesis, ENS Lyon, 2004.
30 E. Lozes. Separation Logic preserves the expressive power of classical logic. In *SPACE'04*, 2004.
31 M. Lück. Axiomatizations of team logics. *Annals of Pure and Applied Logic*, 169(9):928–969, 2018.
32 A. Mansutti. Extending Propositional Separation Logic for Robustness Properties. In *FST&TCS'18*, volume 122 of *LIPIcs*, pages 42:1–42:23. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
33 P.W. O'Hearn. A Primer on Separation Logic. In *Software Safety and Security: Tools for Analysis and Verification*, volume 33 of *NATO Science for Peace and Security Series*, pages 286–318, 2012.
34 P.W. O'Hearn and D. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999.
35 R. Piskac, Th. Wies, and D. Zufferey. Automating Separation Logic using SMT. In *CAV'13*, volume 8044 of *LNCS*, pages 773–789. Springer, 2013.
36 D. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*, volume 26 of *Applied Logic*. Kluwer Academic Publishers, 2002.
37 D. Pym, J. Spring, and P.W. O'Hearn. Why Separation Logic Works. *Philosophy & Technology*, pages 1–34, 2018.
38 J.C. Reynolds. Separation logic: a logic for shared mutable data structures. In *LICS'02*, pages 55–74. IEEE, 2002.

39 I. Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. *Information and Computation*, 157(1–2):142–182, 2000.

40 Y. Wang and Q. Cao. On axiomatizations of public announcement logic. *Synthese*, 190(Supplement-1):103–134, 2013.

41 H. Yang. *Local Reasoning for Stateful Programs*. PhD thesis, University of Illinois, Urbana-Champaign, 2001.