# An EXPTIME-complete entailment problem in separation logic ⋆

Nicolas Peltier

# An EXPTIME-complete entailment problem in separation logic ⋆

Nicolas Peltier

Univ. Grenoble Alpes, CNRS, LIG, F-38000 Grenoble France

**Abstract.** Separation logic (SL) is extensively employed in verification to analyze programs that manipulate dynamically allocated memory. The entailment problem, when dealing with inductively defined predicates or data constraints, is undecidable for SL formulas. Our focus is on addressing a specific fragment of this issue, wherein the consequent is restricted to clauses of some particular form, devoid of inductively defined predicates. We present an algorithm designed to determine the validity of such entailments and demonstrate that the problem is decidable and ExpTime complete under some conditions on the data theory. This algorithm serves the purpose of verifying that the data structures outlined by a given SL formula (the antecedent) adhere to certain shape constraints expressed by the consequent.

## 1 Introduction

Separation logic (SL) [19] is a variant of bunched logic [18] introduced in program verification for reasoning about programs that manipulate dynamically allocated memory. This logic employs a specific connective, called the separating conjunction and denoted by $*$, to assert that two formulas hold on disjoint sections of memory, facilitating more concise specifications. The main advantage of SL is that it supports *local reasoning*, meaning that program properties can be asserted and proven by referencing only the portion of memory affected by the program, without considering the global state of the system. The expressive power of the logic can be strongly augmented by utilizing *inductively defined predicates*, enabling the definition of recursive data structures of unbounded sizes, such as lists or trees. For example, the following rules define a predicate $\texttt{lseg}(x, y)$ denoting a list segment from $x$ to $y$ (with no data): $\{\texttt{lseg}(x, y) \Leftarrow \texttt{emp} \wedge x \simeq y,\ \texttt{lseg}(x, y) \Leftarrow \exists z.(x \mapsto (z) * \texttt{lseg}(z, y))\}$. Informally, $x, y, z$ represent locations (i.e., memory addresses), $\texttt{emp}$ asserts that the heap is empty, $x \mapsto (z)$ asserts that location $x$ is allocated and points to a tuple only containing location $z$, and the separating conjunction $x \mapsto (z) * \texttt{lseg}(z, y)$ indicates that the heap contains a list segment $\texttt{lseg}(z, y)$ along with an additional memory cell $x$ pointing to $z$ (implicitly ensuring that $x$ is distinct from all memory locations allocated in the list segment from $z$ to $y$). The first rule corresponds to the case where the list segment is empty (in which case we must have $x = y$). These predicates can be either hard-coded or defined by the user to handle custom data structures. In the fragment of separation logic known as *symbolic*

⋆ This work has been partially funded by the the French National Research Agency project ANR-21-CE48-0011.

*heaps* (formally defined later), satisfiability is decidable for formulas with inductively defined predicates [2], but entailment is undecidable (entailment cannot be reduced to satisfiability since the fragment does not include negations).

In this present paper, our focus lies on a specific fragment of the entailment problem in SL, and we show that this fragment is decidable and ExpTime-complete (under some particular conditions on the data theory). These entailments exhibit dissymmetry: the antecedent and consequent belong to distinct classes of formulas. Formal definitions will be provided subsequently, but in essence, the antecedent is a symbolic heap describing data structures of unbounded size (with constraints on the data contained within those structures) whereas the consequent is a clause that articulates shape constraints over these structures, and assert conditions on the allocated and referenced locations and on the data stored in the structures. Such clauses may be used to assert that a structure contains – or does not contain – some specific patterns. For example, the following (valid) entailment problem asserts that, in all structures satisfying $\texttt{lseg}(x, y)$, all referenced locations are allocated, except for the one associated with $y$ ($\top$ denotes the atom true and $\texttt{alloc}(z')$ states that $z'$ is allocated): $\texttt{lseg}(x, y) \models \forall z \forall z' (\neg(z \mapsto (z') * \top) \vee z' \simeq y \vee \texttt{alloc}(z'))$. Another instance expresses that each location is referenced only once: $\texttt{lseg}(x, y) \models \forall z \forall z' \forall z'' \neg(z \mapsto (z'') * z' \mapsto (z'') * \top)$. The latter is deemed invalid as $\texttt{lseg}(x, y)$ possesses models representing cyclic lists (where $y$ is allocated, and referenced twice). Now, consider a predicate $\texttt{dll}(x, y, z)$ denoting a doubly linked list segment (with data) from $x$ to $y$ ($z$ denotes the element preceding $x$): $\{\texttt{dll}(x, y, z) \Leftarrow \texttt{emp} \wedge (x \simeq y),\ \texttt{dll}(x, y, z) \Leftarrow \exists x' \exists w (x \mapsto (z, w, x') * \texttt{dll}(x', y, x))\}$. Here, each location refers to a tuple $(x', w, x'')$ where $x'$ and $x''$ are pointers to the previous and next elements, respectively, and $w$ is the data stored in the current cell. The (valid) problem $\texttt{dll}(x, y, z) \models \phi$, with $\phi$ is the formula: $\forall x_1 \forall x_2 \forall x_3 \forall x_1' \forall x_2' \forall w_1 \forall w_2 (\neg(x_1 \mapsto (x_1', w_1, x_2) * x_2 \mapsto (x_2', w_2, x_3)) \vee x_2' \simeq x_1)$ ensures that the "previous" pointer of the successor of any location $x_1$ is indeed $x_1$. The (non valid) problem $\texttt{dll}(x, y, z) \models \psi$ where $\psi$ is $\forall x_1 \forall x_2 \forall x_3 \forall x_1' \forall x_2' \forall w_1 \forall w_2 (\neg(x_1 \mapsto (x_1', w_1, x_2) * x_2 \mapsto (x_2', w_2, x_3)) \vee w_1 < w_2)$ asserts that the list is sorted in strict ascending order.

*Related Work* A significant portion of research within automated reasoning in SL is directed toward decidable fragments of entailment problems. As the entailment problem is undecidable when dealing with formulas with inductively defined predicates, substantial effort has been expended to pinpoint decidable fragments and design corresponding proof procedures (see, e.g., [1,3,6,13,12,7]). Most works focus on formulas of some particular form, called *symbolic heaps* (defined as existentially quantified conjunctions and separated conjunctions of atoms). A very general class of decidable entailment problems (specifically for formulas involving no theory beyond equality) is devised in [14], obtained by restricting the considered inductive definitions to so-called PCE (for Progressing, Connected and Established) rules. The decidability result rests upon the decidability of the satisfiability problem for monadic second-order logic over graphs with bounded treewidth. A more recent advancement is the proposal of a 2-ExpTime algorithm for such entailments [17]. In [8] the optimality of this bound is established, and in [9,10], novel algorithms are introduced, capable of handling more expressive classes of inductive definitions. More recently, a proof procedure has been devised [11] for the PCE fragment. In the pursuit of computational efficiency, less ex-

pressive fragments have been explored, leading to the development of more efficient algorithms. For instance, in [15], a strict subclass of PCE entailments is identified, with an ExpTime complexity derived through a reduction to the language inclusion problem for tree automata [5]. Another example is found in [12], where an algorithm is devised to handle various types of (potentially nested) singly linked lists, relying on a reduction to the membership problem for tree automata. A polynomial proof procedure was also proposed for the specific case of singly linked lists [6]. The tractability result is extended to more expressive fragments [4], incorporating formulas defined on a unique nonlinear compositional inductive predicate with distinguished source, destination, and static parameters. Recently [16], a polynomial-time cyclic proof procedure has been introduced to efficiently solve the entailment problem, subject to certain conditions on the inductive rules.

Our contribution diverges from previous work in two key aspects: on one hand, we impose no syntactic restrictions on the considered inductive definitions[1], which may in particular contain data constraints (with some conditions on the data theory). On the other hand, we concentrate on entailment problems where the consequents are simpler than their corresponding antecedents, in the sense that they contain no inductive predicates. It is important to note that, even if the considered inductive definitions satisfy the PCE conditions of [14], the entailment problems we consider still do not fall in the scope of the previously cited results, because the consequents are not symbolic heaps. They cannot be expressed either as guarded formulas (in the sense of [17]).

## 2 Separation logic with inductive definitions

We define the syntax and semantics of SL with inductively defined predicates and data constraints. Most definitions in this section are standard.

*Basic notations.* For every finite set $S$, $card(S)$ denotes the number of elements in $S$. For every partial function $f$, $dom(f)$ denotes the domain of $f$. Partial functions will be taken as relations, e.g., if $x_1, \ldots, x_n$ are pairwise distinct, then $\{(x_i, y_i) \mid i \in \{1, \ldots, n\}\}$ denotes the function $f$ of domain $\{x_1, \ldots, x_n\}$ such that $f(x_i) = y_i$ for all $i \in \{1, \ldots, n\}$ ($\emptyset$ is the function with an empty domain). An *extension* of a partial function $f$ to a set $S$, is a function $g$ such that $dom(g) = dom(f) \cup S$ coinciding with $f$ on $dom(f)$ (then $f$ is called a *restriction* of $g$ to $dom(f)$). We sometimes identify tuples with sets when the order and number of repetitions is unimportant, i.e., we may write $\boldsymbol{x} \subseteq S$ to state that every component of the tuple $\boldsymbol{x}$ occurs in the set $S$, or $y \in \boldsymbol{x}$ to state that $y$ occurs in $\boldsymbol{x}$.

*Syntax.* Let $\mathcal{S} = \{\texttt{loc}, \texttt{data}\}$ be the set of *sort symbols*, where `loc` denotes locations (i.e., memory addresses) and `data` denotes data. Let $\mathcal{V}$ be a countably infinite set of *variables* and let $\mathcal{C}$ be a finite set of *constant symbols*. Let $\mathcal{T} = \mathcal{V} \cup \mathcal{C}$ be the set of *terms*. Every term $x$ is associated with a unique sort $\texttt{sort}(x) \in \mathcal{S}$, and $\mathcal{T}(\texttt{s})$ denotes the set of terms of sort `s`. Let $\mathcal{P}_s$ and $\mathcal{P}_d$ be two finite sets of predicate symbols, denoting *spatial predicates* and *data predicates*, respectively. Each symbol in $p \in \mathcal{P}_s$ (resp. $p \in \mathcal{P}_d$) is associated with a unique *profile* $\texttt{pr}(p) \in \mathcal{S}^*$ (resp. $\texttt{pr}(p) \in \{\texttt{data}\}^*$).

[1] except from the fact that the right-hand side of the inductive rules must be a symbolic heap.

**Definition 2.1 (Syntax of SL).** *A* separation logic formula $\phi$ *(or simply formula) is built inductively as follows:*

$$\texttt{emp} \mid \ \mid p(x_1, \ldots, x_n) \mid u \mapsto (v_1, \ldots, v_k) \mid \texttt{alloc}(u) \mid \texttt{ref}(u) \mid \top \mid u \simeq v \mid$$

$$(\phi_1 * \phi_2) \mid (\phi_1 \vee \phi_2) \mid (\phi_1 \wedge \phi_2) \mid \neg\psi \mid \exists x\, \psi$$

*where* $k, n \in \mathbb{N}$*,* $p$ *is a predicate in* $\mathcal{P}_s \cup \mathcal{P}_d$*, of some profile* $(\mathtt{s}_1, \ldots, \mathtt{s}_n)$*,* $x_i \in \mathcal{T}(\mathtt{s}_i)$*,* $u, v \in \mathcal{T}(\texttt{loc})$*,* $v_i \in \mathcal{T}$*,* $x \in \mathcal{V}$ *and* $\phi_1, \phi_2, \psi$ *are formulas.*

The separating implication is not considered in our framework. Formal definitions will be provided later, but the atoms $\texttt{alloc}(x)$ and $\texttt{ref}(x)$ are intended to state that $x$ is allocated and referenced in the heap, respectively. A *data atom* (resp. a spatial predicate atom) is of the form $p(x_1, \ldots, x_n)$ where $p \in \mathcal{P}_d$ (resp. $p \in \mathcal{P}_s$). A *data literal* is either a data atom or its negation. A formula of the form $u \mapsto (v_1, \ldots, v_k)$ is a *points-to atom*. A *spatial atom* is either a points-to atom or a spatial predicate atom. A *pure atom* is either a data atom or an equational atom $x \simeq y$ or $\top$. A *pure literal* is either a pure atom or the negation of a pure atom. We denote by $fv(\phi)$ the set of variables freely occurring in $\phi$. We assume (by $\alpha$-renaming) that distinct quantifiers bind distinct variables and that the set of free and bound variables are disjoint. A *substitution* is a partial function mapping every variable $x$ to a term of sort $\texttt{sort}(x)$. For every substitution $\sigma$ and expression (term, tuple of terms or formula) $\phi$ we denote by $\phi\sigma$ the expression obtained from $\phi$ by replacing every free occurrence of a variable $x \in dom(\sigma)$ by $\sigma(x)$.

*Inductive rules.* The semantics of the spatial predicates is defined using inductive rules of some particular form:

**Definition 2.2 (Symbolic heaps and inductive rules).** *We recall that a* symbolic heap *is a formula of the form* $\exists \boldsymbol{x}[(\alpha_1 * \cdots * \alpha_n) \wedge \bigwedge_{j=1}^{m} \beta_j]$*, where* $\alpha_i$ *(for all* $i \in \{1, \ldots, n\}$*) is a spatial atom and* $\beta_j$ *(for all* $j \in \{1, \ldots, m\}$*) is a pure literal. A* disjunctive symbolic heap *(DSH) is a formula of the form* $\bigvee_{i=1}^{n} \phi_i$ *where* $\phi_i$ *(for all* $i \in \{1, \ldots, n\}$*) is a symbolic heap. An* inductive rule *is an expression of the form* $p(x_1, \ldots, x_n) \Leftarrow \xi$ *where* $p$ *is a spatial predicate of profile* $(\mathtt{s}_1, \ldots, \mathtt{s}_n)$*,* $x_1, \ldots, x_n$ *are pairwise distinct variables of sort* $\mathtt{s}_1, \ldots, \mathtt{s}_n$ *(respectively) and* $\xi$ *is a DSH with* $fv(\xi) \subseteq \{x_1, \ldots, x_n\}$*.*

Let $\mathcal{R}$ be a finite set of inductive rules. We write $\phi \leadsto_{\mathcal{R}} \psi$ if $\psi$ is obtained from the formula $\phi$ by replacing one occurrence of a spatial atom $p(x_1, \ldots, x_n)$ by a formula $\xi\sigma$ where $p(y_1, \ldots, y_n) \Leftarrow \xi$ is a rule in $\mathcal{R}$ and $\sigma = \{(y_i, x_i) \mid i \in \{1, \ldots, n\}\}$. Then $\leadsto_{\mathcal{R}}^*$ denotes the reflexive and transitive closure of $\leadsto_{\mathcal{R}}$. We assume that the existential variables in $\xi$ are renamed to avoid any collision with variables already occurring in $\phi$.

*Semantics.* Let $\mathcal{L}$ be a countably infinite set of *locations* and let $\mathcal{D}$ be an arbitrary set of *data*. The domain of a sort $\mathtt{s}$ is $\mathcal{D}$ if $\mathtt{s} = \texttt{data}$ and $\mathcal{L}$ if $\mathtt{s} = \texttt{loc}$. We assume that every predicate $p \in \mathcal{P}_d$ of profile $\texttt{data}^n$ is mapped to a subset $\langle p \rangle$ of $\mathcal{D}^n$, and that every constant $c$ of sort $\texttt{data}$ is associated with an element $\langle c \rangle \in \mathcal{D}$. Note that the interpretation of data constants and data predicates is fixed in our setting.

**Definition 2.3 (SL Structures).** *A* store $\mathfrak{s}$ *is a partial function mapping every term* $x$ *to an element of* $\mathcal{L}$ *if* $x$ *is of sort* `loc` *or to an element of* $\mathcal{D}$ *if* $x$ *is of sort* `data`*, where* $C \subseteq dom(\mathfrak{s})$ *and* $\mathfrak{s}(c) = \langle c \rangle$ *if* $c \in C$ *and* $\mathtt{sort}(c) = \mathtt{data}$*. A* heap $\mathfrak{h}$ *is a partial function mapping locations in* $\mathcal{L}$ *to tuples* $(\ell_1, \ldots, \ell_n)$ *with* $n \geq 0$ *and* $\ell_i \in \mathcal{L} \cup \mathcal{D}$ *(for all* $i \in \{1, \ldots, n\}$*) such that* $dom(\mathfrak{h})$ *is finite. A* structure *is a pair* $(\mathfrak{s}, \mathfrak{h})$ *where* $\mathfrak{s}$ *is a store and* $\mathfrak{h}$ *is a heap.*

A location $\ell$ is *allocated* in $\mathfrak{h}$ if $\ell \in dom(\mathfrak{h})$ and an element $e \in \mathcal{L} \cup \mathcal{D}$ is *referenced* in $\mathfrak{h}$ if there exists $\ell \in dom(\mathfrak{h})$ such that $e \in \mathfrak{h}(\ell)$. A variable $x$ is *allocated* (resp. *referenced*) in a structure $(\mathfrak{s}, \mathfrak{h})$ if $\mathfrak{s}(x)$ is allocated (resp. referenced) in $\mathfrak{h}$. Two heaps $\mathfrak{h}_1, \mathfrak{h}_2$ are *disjoint* if $dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2) = \emptyset$, in which case $\mathfrak{h}_1 \cup \mathfrak{h}_2$ denotes the heap of domain $dom(\mathfrak{h}_1) \cup dom(\mathfrak{h}_2)$, coinciding with $\mathfrak{h}_i$ on $dom(\mathfrak{h}_i)$ (for all $i \in \{1, 2\}$).

**Definition 2.4 (Semantics of SL).** *Let* $\mathcal{R}$ *be a finite set of inductive rules. Let* $(\mathfrak{s}, \mathfrak{h})$ *be a structure. For every formula* $\phi$ *with* $fv(\phi) \subseteq dom(\mathfrak{s})$*, we write* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ *(and say that* $(\mathfrak{s}, \mathfrak{h})$ *is an* $\mathcal{R}$*-model of* $\phi$*) if one of the following conditions holds:*

- $\phi$ *is* $\top$*; or* $\phi$ *is* `emp` *and* $\mathfrak{h} = \emptyset$*; or* $\phi$ *is* $x \simeq y$ *and* $\mathfrak{s}(x) = \mathfrak{s}(y)$*;*
- $\phi = x \mapsto (y_1, \ldots, y_n)$ *and* $\mathfrak{h} = \{(\mathfrak{s}(x), \mathfrak{s}(y_1), \ldots, \mathfrak{s}(y))\}$*;*
- $\phi = p(x_1, \ldots, x_n)$*,* $p \in \mathcal{P}_d$ *and* $(\mathfrak{s}(x_1), \ldots, \mathfrak{s}(x_n)) \in \langle p \rangle$*;*
- *or* $\phi = \neg\psi$ *and* $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathcal{R}} \psi$*;*
- $\phi = \phi_1 \vee \phi_2$ *(resp.* $\phi_1 \wedge \phi_2$*) and* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi_i$*, for some* $i \in \{1, 2\}$ *(resp. for all* $i \in \{1, 2\}$*);*
- $\phi = \phi_1 * \phi_2$ *and there exist disjoint heaps* $\mathfrak{h}_1, \mathfrak{h}_2$ *with* $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$*, and for all* $i \in \{1, 2\}$*,* $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathcal{R}} \phi_i$*;*
- $\phi = \exists x \psi$ *and there exists an extension* $\mathfrak{s}'$ *of* $\mathfrak{s}$ *with* $dom(\mathfrak{s}') = dom(\mathfrak{s}) \cup \{x\}$ *and* $(\mathfrak{s}', \mathfrak{h}) \models_{\mathcal{R}} \psi$*;*
- $\phi = \mathtt{alloc}(x)$ *and* $\mathfrak{s}(x) \in dom(\mathfrak{h})$*; or* $\phi = \mathtt{ref}(x)$ *and* $\exists \ell \in dom(\mathfrak{h})$ *s.t.* $\mathfrak{s}(x) \in \mathfrak{h}(\ell)$*;*
- $\phi = p(x_1, \ldots, x_n)$*,* $p \in \mathcal{P}_s$ *and there is a formula* $\psi$ *containing no spatial predicate symbol such that* $\phi \rightsquigarrow^*_{\mathcal{R}} \psi$ *and* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \psi$*.*

*If* $S$ *is a set of formulas, then* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} S$ *holds iff* $\forall \phi \in S\ (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$*. If* $\phi, \psi$ *are formulas, we write* $\phi \models_{\mathcal{R}} \psi$ *if the implication* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \implies (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \psi$ *holds for every structure* $(\mathfrak{s}, \mathfrak{h})$ *with* $dom(\mathfrak{s}) \supseteq fv(\phi) \cup fv(\psi)$*.*

Formulas are taken modulo associativity and commutativity of $*$ and $\wedge, \vee$, modulo neutrality of `emp` for $*$, modulo commutativity of $\exists$ (i.e. $\exists x \exists y \phi$ is equivalent to $\exists y \exists x \phi$) modulo contraction for $\wedge$ and $\vee$ and modulo contraction of pure atoms for $*$ (i.e., $\phi * \phi \equiv \phi$ if $\phi$ is pure, note that the contraction does not hold for spatial atoms). If $\boldsymbol{x} = (x_1, \ldots, x_n)$ (with $n \geq 0$), we may write $\exists \boldsymbol{x}\, \phi$ for $\exists x_1 \ldots \exists x_n\, \phi$ (if $\boldsymbol{x}$ is empty then $\exists \boldsymbol{x}\, \phi$ is $\phi$). For any formula $\phi$, $|\phi|$ denotes the size of $\phi$.

*Example 2.5 (Sorted Lists).* The rules below define an atom $\mathtt{sls}(x, y)$ describing sorted lists starting at $x$, ending at a constant `nil`, and containing only elements bigger or equal to $y$:

$$\mathtt{sls}(x, y) \Leftarrow x \mapsto (\mathtt{nil}) \qquad \mathtt{sls}(x, y) \Leftarrow \exists x' \exists y' [x \mapsto (x', y') * \mathtt{sls}(x', y') \wedge y' \geq y]$$

Each location (other than the last one) in the list points to a pair $(\ell, v)$ where $\ell$ denotes the next element and $v$ is the stored value. The profile of `sls` is $(\mathtt{loc}, \mathtt{data})$. The variables $x, x'$ are of sort `loc` and the variables $y, y'$ are of sort `data`. The formula $\exists y, y' (\mathtt{sls}(x, y) * \mathtt{sls}(x', y') \wedge y \geq y') \vee \mathtt{emp}$ is a DSH.

We make the following assumptions on the data theory, which are all essential to the soundness of the results.

1. The set of data predicates $\mathcal{P}_d$ contains in particular the equality predicate $\approx$ interpreted as usual.
2. The satisfiability problem is decidable for conjunctions of data literals.
3. Every formula of the form $\exists x\, \phi$ where $\phi$ is a conjunction of data literals is equivalent to a formula $\bigvee_{i=1}^{n} \phi_i$, where $\phi_i$ (for all $i \in \{1, \ldots, n\}$) is a conjunction of data literals, with $fv(\phi_i) \subseteq fv(\exists x\, \phi)$.

Assumptions (1) and (2) are natural, but Assumption (3) is rather strong, as the set of constants and the set of data predicates are both finite (of course, this cannot be overcome since we want to get a decidable entailment problem with fixpoint computations). The conditions are satisfied for instance by the theory of reals (or rational numbers) with predicates $\approx, <, \leq$.

## 3 The Entailment problem ENT(DSH,HC)

We now define the entailment problem we are considering in the present paper. As explained in the introduction, this entailment problem is dissymmetric: the antecedent and consequent belong to different fragments of SL. The antecedent is a disjunctive symbolic heap, and the consequent is a universally quantified disjunction of literals (with no inductive predicates) of some specific form:

**Definition 3.1** **(𝔥-clauses).** *A* 𝔥-atom *is either a data atom, or a formula of the form* $\texttt{alloc}(x)$, $\texttt{ref}(x)$, $x \bowtie y$ *(with* $\bowtie \in \{\simeq, \not\simeq\}$*), or a separating conjunction* $\alpha_1 * \cdots * \alpha_n$ *where* $\alpha_i$ *is either a points-to atoms or* $\top$ *(spatial predicate atoms are not allowed). Formulas of the latter form are called* spatial 𝔥-atoms*. A* 𝔥-literal *is either a* 𝔥*-atom (positive* 𝔥*-literal) or the negation of a* 𝔥*-atom (negative* 𝔥*-literal). The* 𝔥*-literal* complementary to some 𝔥-literal $\phi$ *is either* $\neg\phi$ *if* $\phi$ *is positive or* $\psi$ *if* $\phi = \neg\psi$*. A* 𝔥-clause *(HC) is of the form* $\forall x_1 \ldots \forall x_n\, \phi$*, where* $\phi$ *is a disjunction of* 𝔥*-literals (*$\phi$ *may contain variables not occurring in* $\{x_1, \ldots, x_n\}$*).*

*Example 3.2.* Consider the following 𝔥-clauses:

$$\begin{array}{ll} \xi_1: & \forall x \forall x' \forall y'\, (\neg(x \mapsto (x', y') * \top) \vee y' \geq y) \\ \xi_2: & \forall x_1 \forall x_2 \forall y_1 \forall y_2 \forall x'\, \neg(x_1 \mapsto (x', y_1) * x_2 \mapsto (x', y_2) * \top) \\ \xi_3: & \forall x_1 \forall x_2 \forall x_1' \forall x_2' \forall y\, \neg(x_1 \mapsto (x_1', y) * x_2 \mapsto (x_2', y) * \top) \end{array}$$

With the conventions of Ex. 2.5, $\xi_1$ asserts that the list contains no element strictly lower than $y$, $\xi_2$ states that every location is referenced at most once and $\xi_3$ states that all elements are pairwise distinct.

**Definition 3.3** **(ENT(DSH,HC)).** *The entailment problem* ENT*(DSH,HC) consists in determining, given a finite set of inductive rules* $\mathcal{R}$*, a disjunctive symbolic heap* $\xi$ *and a* 𝔥*-clause* $\gamma$*, whether the entailment* $\xi \models_{\mathcal{R}} \gamma$ *holds.*

*Example 3.4.* With the definitions of Examples 2.5 and 3.2, $p(x, y) \models_{\mathcal{R}} \xi_i$ is an instance of ENT(DSH,HC) for all $i \in \{1, 2, 3\}$. It is valid if $i \in \{1, 2\}$ and not valid if $i = 3$. Additional instances of ENT(DSH,HC) can be found in the introduction. One specific instance of ENT(DSH,HC) consists to test whether a given a symbolic heap $\xi$ is *established* (in the sense of [14]), i.e., that all the locations occurring in the heap of the models of $\xi$ are either allocated or equal to some free variable: $\xi \models_{\mathcal{R}} \forall x[\mathtt{alloc}(x) \vee \neg\mathtt{ref}(x) \vee x \simeq y_1 \vee \cdots \vee x \simeq y_n)]$, with $\{y_1, \ldots, y_n\} = \{y \in fv(\phi) \mid \mathtt{sort}(y) = \mathtt{loc}\}$.

## 4 Abstracting structures

The decision procedure operates by computing abstractions of the models of the antecedent. Intuitively, these abstractions will encompass information pertaining to the key characteristics of these models which is adequate for determining whether these models satisfy the consequent. We firstly define a notion of an *E-relation*, denoting an equality relation between terms:

**Definition 4.1 (*E*-Relation).** *An E-relation $\sim$ (for a set of variables $V \subseteq \mathcal{V}$) is an equivalence relation on $V \cup C$ satisfying the following conditions: (i) if $x \sim y$ then* $\mathtt{sort}(x) = \mathtt{sort}(y)$*; and (ii) if $x, y$ are constants of sort* `data` *then $x \sim y \iff \langle x \rangle = \langle y \rangle$. For every term $x$, we denote by $x{\downarrow_\sim}$ a unique representative of the equivalence class of $x$, e.g., the minimal term $y$ (w.r.t. to some arbitrary but fixed order on terms) such that $x \sim y$. This notation is extended to any formula $\phi$, where $\phi{\downarrow_\sim}$ denotes the formula obtained form $\phi$ by replacing every term $x$ by $x{\downarrow_\sim}$. A formula $\phi$ is* $\sim$-normalized *if $\phi = \phi{\downarrow_\sim}$. We denote by $\mathcal{E}(V)$ the set of E-relations for $V$.*

If $V$ is clear from the context, then we shall denote an $E$-relation $\sim$ by a set of equations $S$, with the convention that $\sim$ is the smallest $E$-relation for $V$ such that $x \sim y$ holds for all $(x \simeq y) \in S$. We then define the notion of a *consistent set*, which denotes the set of data atoms that are true in some specific structure.

**Definition 4.2 (Consistency).** *For every set of variables $V$ and E-relation $\sim$, we denote by $\mathcal{A}^D_\sim(V)$ the set of $\sim$-normalized data atoms $\phi$ such that $fv(\phi) \subseteq V$. A subset $S$ of $\mathcal{A}^D_\sim(V)$ is* consistent *(w.r.t. $\sim$ and $V$) if there exists an injective store $\mathfrak{s}$ such that for all $\sim$-normalized data atoms $\alpha$ in $\mathcal{A}^D_\sim(V)$: $(\mathfrak{s}, \emptyset) \models_{\mathcal{R}} \alpha \iff \alpha \in S$.*

**Definition 4.3 (Abstraction).** *An* abstraction *is a tuple $(V, \sim, A, R, \Phi, \Delta)$ where $V$ is a set of variables, $\sim \in \mathcal{E}(V)$, $A$ and $R$ are sets of $\sim$-normalized terms, $\Phi$ is a $\sim$-normalized spatial $\mathfrak{h}$-atom and $\Delta$ is a consistent subset of $\mathcal{A}^D_\sim(V)$.*

Intuitively, $(V, \sim, A, , R, \Phi, \Delta)$ encapsulates crucial information about a structure. The set $V$ denotes the variables in the domain of the store. The relation $\sim$ is the equality relation between terms. The sets $A$ and $R$ respectively contain the sets of allocated and referenced terms. The formula $\Phi$ is a spatial $\mathfrak{h}$-atom describing the part of the heap that corresponds to locations and data associated with variables in $V$ and abstracting away the remainder of the heap. Finally, $\Delta$ is the set of data atoms that are true in the considered model. The formalization of the relationship between structures and abstractions is provided by Def. 4.4 and 4.6.

**Definition 4.4.** *For every structure* $(\mathfrak{s},\mathfrak{h})$*, for every set of variables* $V \subseteq dom(\mathfrak{s})$ *and for every E-relation* $\sim$*, we denote by* $\Phi_{\sim}(\mathfrak{s},\mathfrak{h})$ *the* $\mathfrak{h}$*-atom* $\Phi^1_{\sim}(\mathfrak{s},\mathfrak{h}) * \Phi^2_{\sim}(\mathfrak{s},\mathfrak{h})$*, where:*

- $\Phi^1_{\sim}(\mathfrak{s},\mathfrak{h}) = \alpha_1 * \cdots * \alpha_n$*, where* $\{\alpha_i \mid i \in \{1,\ldots,n\}\}$ *is the set of atoms of the form* $x_0 \mapsto (x_1,\ldots,x_k)$ *such that* $x_0,\ldots,x_k$ *are* $\sim$*-normalized terms in* $dom(\mathfrak{s})$*,* $\mathfrak{s}(x_0) \in dom(\mathfrak{h})$ *and* $\mathfrak{h}(\mathfrak{s}(x_0)) = (\mathfrak{s}(x_1),\ldots,\mathfrak{s}(x_k))$ *(if this set is empty then* $n = 0$ *and* $\Phi^1_{\sim}(\mathfrak{s},\mathfrak{h}) = \mathtt{emp}$*).*
- $\Phi^2_{\sim}(\mathfrak{s},\mathfrak{h})$ *is* $\mathtt{emp}$ *if* $card(dom(\mathfrak{h})) = n$ *and* $\top$ *otherwise.*

*In particular, if* $\mathfrak{h} = \emptyset$*, then* $\Phi_{\sim}(\mathfrak{s},\mathfrak{h}) = \mathtt{emp} * \mathtt{emp} = \mathtt{emp}$*, and if* $dom(\mathfrak{s}) = \emptyset$ *and* $\mathfrak{h} \neq \emptyset$ *then* $\Phi_{\sim}(\mathfrak{s},\mathfrak{h}) = \mathtt{emp} * \top = \top$*.*

*Example 4.5.* Let $\mathcal{L} = \mathbb{N}$, $\mathcal{D} = \mathbb{Q}$ and let $\mathfrak{h}$ be the heap $\{(2,1,0.5),(1,0,-0.5)\}$. Let $\mathfrak{s} = \{(x,2),(y,1),(z_1,0.5),(z_2,-0.5),(\mathtt{nil},0)\}$ and $\mathfrak{s}' = \{(u,1),(v,1),(w,-0.5),(\mathtt{nil},0)\}$.

Then $\Phi_{\sim}(\mathfrak{s},\mathfrak{h})$ is $x \mapsto (y,z_1) * y \mapsto (\mathtt{nil},z_2)$ and $\Phi_{\sim}(\mathfrak{s}',\mathfrak{h})$ is $u \mapsto (\mathtt{nil},w) * \top$ (assuming that the representative of the class of $\{u,v\}$ is $u$).

**Definition 4.6 (Abstraction of a structure).** *Let* $k \in \mathbb{N}$*,* $\mathcal{W} \subseteq \mathcal{V}$*, Let* $(\mathfrak{s},\mathfrak{h})$ *be a structure, and let* $\mathcal{A} = (V,\sim,A,R,\Phi,\Delta)$ *be an abstraction. The abstraction* $\mathcal{A}$ *is called an* abstraction of $(\mathfrak{s},\mathfrak{h})$ *(written* $\mathcal{A} \triangleright (\mathfrak{s},\mathfrak{h})$*) if all the following conditions hold. (1)* $dom(\mathfrak{s}) = V \cup C$*; (2) For all terms* $x,y \in V \cup C$*,* $x \sim y \iff \mathfrak{s}(x) = \mathfrak{s}(y)$*; (3)* $A$ *is the set of* $\sim$*-normalized terms in* $V \cup C$ *such that* $\mathfrak{s}(x) \in dom(\mathfrak{h})$*; (4)* $R$ *is the set of* $\sim$*-normalized terms of sort* $\mathtt{loc}$ *in* $V \cup C$ *such that there exists* $\ell \in dom(\mathfrak{h})$ *with* $\mathfrak{s}(x) \in \mathfrak{h}(\ell)$*; (5)* $\Phi = \Phi_{\sim}(\mathfrak{s},\mathfrak{h})$*; (6)* $\Delta$ *is the set of data atoms in* $\mathcal{A}^D_{\sim}(V)$*, such that* $(\mathfrak{s},\emptyset) \models_R \phi$*.*

Note that, when the representative of each term equivalence class is fixed, each structure has a unique abstraction, where the components $V,\sim,A,R,\Phi$ and $\Delta$ are defined in accordance with conditions (1)-(6).

*Example 4.7.* With the definitions of Ex. 4.5, and with $\mathcal{P}_d = \{\approx,\geq\}$, the abstractions of $(\mathfrak{s},\mathfrak{h})$ and $(\mathfrak{s}',\mathfrak{h})$ are $\mathcal{A}$ and $\mathcal{A}'$, respectively, with:

$$\mathcal{A} = (\{x,y,z_1,z_2\},\emptyset,\{x,y\},\{y,\mathtt{nil}\},x \mapsto (y,z_1) * y \mapsto (\mathtt{nil},z_2),\Delta)$$
$$\Delta = \{z_1 \geq z_2, z_1 \approx z_1, z_1 \geq z_1, z_2 \approx z_2, z_2 \geq z_2\}$$
$$\mathcal{A}' = (\{u,v,w\},\{u \simeq v\},\{u\},\{\mathtt{nil}\},u \mapsto (\mathtt{nil},w) * \top,\{w \geq w, w \approx w\})$$

We show (see Lem. 4.10) that the truth value of a $\mathfrak{h}$-literal in a structure depends solely on the structure's abstraction. This allows for testing the validity of entailments in Ent(DSH,HC) by solely considering abstractions (Prop. 4.11). For this purpose, Def. 4.8 offers a simple criterion for determining whether an abstracted structure satisfies a particular $\mathfrak{h}$-literal or set of $\mathfrak{h}$-literals.

**Definition 4.8.** *For all spatial* $\mathfrak{h}$*-atoms* $\phi,\psi$*, we write* $\phi \prec \psi$ *if* $\phi$ *and* $\psi$ *are respectively of the form (up to AC)* $\phi_1 * \top$ *and* $\phi_1 * \phi_2$ *(with possibly* $\phi_2 = \mathtt{emp}$*). For all abstractions* $\mathcal{A} = (V,\sim,A,R,\Phi,\Delta)$ *and* $\mathfrak{h}$*-literal* $\phi$*,* $\mathcal{A} \models \phi$ *iff* $fv(\phi) \subseteq V$ *and one of the following conditions holds: (1)* $\phi = (x \simeq y)$ *and* $x \sim y$*; (2)* $\phi = \mathtt{alloc}(x)$ *and* $x\!\downarrow_{\sim} \in A$*; (3)* $\phi = \mathtt{ref}(x)$ *and* $x\!\downarrow_{\sim} \in R$*; (4)* $\phi\!\downarrow_{\sim} \preceq \Phi$*; (5)* $\phi\!\downarrow_{\sim} \in \Delta$*; or (6)* $\phi = \neg\psi$ *and* $\mathcal{A} \not\models \psi$*. For any set of* $\mathfrak{h}$*-literals* $S$*,* $\mathcal{A} \models S$ *iff* $\forall \phi \in S\ \mathcal{A} \models \phi$*.*

*Example 4.9.* With the definitions of Ex. 4.7, we have (for instance):

$$\mathcal{A} \models \{\mathtt{alloc}(y),\ \mathtt{ref}(y),\ x \neq y,\ x \mapsto (y, z_1) * \top, z_2 \not\simeq z_1,\ \neg(x \mapsto (y, z_1))\}$$
$$\mathcal{A}' \models \{\mathtt{alloc}(v),\ \neg\mathtt{ref}(u),\ u \simeq v,\ u \mapsto (\mathtt{nil}, w) * \top,\ \neg\mathtt{emp}\}$$

**Lemma 4.10.** *Let $\phi$ be a $\mathfrak{h}$-literal. If $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$ and $dom(\mathfrak{s}) \supseteq fv(\phi)$ then $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ iff $\mathcal{A} \models \phi$.*

*Proof.* Let $\mathcal{A} = (V, \sim, A, R, \Phi, \Delta)$. Note that, due to $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$, $x \sim y$ holds iff $\mathfrak{s}(x) = \mathfrak{s}(y)$ (for all $x, y \in V \cup C$). For conciseness, we focus on the case where $\phi$ is a spatial $\mathfrak{h}$-atom (the other cases are covered in App. A). By the previous remark, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \iff (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi{\downarrow_\sim}$. By definition of $\mathfrak{h}$-atoms, $\phi{\downarrow_\sim}$ can be written on the form $\alpha_1 * \cdots * \alpha_n * \beta$ where $\alpha_i$ (for all $i \in \{1, \ldots, n\}$) is a points-to atom $x_i \mapsto (\boldsymbol{y}_i)$, and $\beta$ is either $\top$ or $\mathtt{emp}$. Moreover, by Def. 4.6 (5), we have $\Delta = \Phi_\sim(\mathfrak{s}, \mathfrak{h})$ , thus, by Def. 4.4, $\Delta$ is of the form $\alpha'_1 * \cdots * \alpha'_m * \beta'$, where: $\{\alpha'_i \mid i \in \{1, \ldots, m\}\}$ is the set of atoms of the form $x'_i \mapsto (\boldsymbol{y}'_i)$ such that $x'_i, \boldsymbol{y}'_i$ only contain $\sim$-normalized terms in $V \cup C$, $\mathfrak{s}(x'_i) \in dom(\mathfrak{h})$, $\mathfrak{h}(\mathfrak{s}(x'_i)) = (\mathfrak{s}(\boldsymbol{y}'_i))$, and $\beta'$ is $\mathtt{emp}$ if $card(dom(\mathfrak{h})) = m$ and $\top$ otherwise. Note that by definition the atoms $\alpha'_i$ for $i \in \{1, \ldots, m\}$ are pairwise distinct. We establish the double implication.

$\Rightarrow$ Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ (hence $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi{\downarrow_\sim}$). Then $\mathfrak{h} = \bigcup_{i=0}^{n} \mathfrak{h}_i$, where $\mathfrak{h}_0, \ldots, \mathfrak{h}_n$ are pairwise disjoint heaps such that $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathcal{R}} \alpha_i$ (for all $i \in \{1, \ldots, n\}$) and $(\mathfrak{s}, \mathfrak{h}_0) \models_{\mathcal{R}} \beta$. This entails that $\mathfrak{h}_i = \{(\mathfrak{s}(x_i), \mathfrak{s}(\boldsymbol{y}_i))\}$ (for all $i \in \{1, \ldots, n\}$), so that $\mathfrak{s}(x_i) \in dom(\mathfrak{h})$ and $\mathfrak{h}(\mathfrak{s}(x_i)) = (\mathfrak{s}(\boldsymbol{y}_i))$. As $\alpha_i$ is normalized, this entails that each atom $\alpha_i$ necessarily occurs in $\{\alpha'_1, \ldots, \alpha'_m\}$. Assume by symmetry that $\alpha_i = \alpha'_i$ for all $i \in \{1, \ldots, n\}$ (with $n \leq m$). If $\beta = \top$ then we get $\phi{\downarrow_\sim} = (\alpha_1 * \cdots * \alpha_n) * \top$ and $\Delta = (\alpha_1 * \cdots * \alpha_n) * (\alpha'_{n+1} * \ldots \alpha'_m * \beta')$, so that $\phi{\downarrow_\sim} \prec \Delta$, whence $\mathcal{A} \models \phi$ by Def. 4.8 (4). Otherwise (i.e., if $\beta = \mathtt{emp}$), we must have $card(dom(\mathfrak{h})) = n$, which entails that $m = n = card(dom(\mathfrak{h}))$ (as $card(dom(\mathfrak{h})) \geq m$ by definition of $\{\alpha'_1, \ldots, \alpha'_m\}$). Therefore, $\beta' = \mathtt{emp}$ and $\phi{\downarrow_\sim} = \Delta$, so that $\mathcal{A} \models \phi$ by Def. 4.8 (4).

$\Leftarrow$ Assume that $\mathcal{A} \models \phi$, i.e., by Def. 4.8, $\phi{\downarrow_\sim} \preceq \Delta$. We may assume by symmetry that $\alpha_i = \alpha'_i$ for all $i \in \{1, \ldots, n\}$ (with $n \leq m$), and $\beta$ is either $\beta'$ (with $n = m$) or $\top$. By definition of the set $\{\alpha'_1, \ldots, \alpha'_m\}$, we have $\mathfrak{h}(\mathfrak{s}(x'_i)) = \mathfrak{s}(\boldsymbol{y}'_i)$. As the atoms $\alpha'_i$ are pairwise distinct and $\sim$-normalized, this entails that the locations $\mathfrak{s}(x'_i)$ are pairwise distinct: indeed, if $\mathfrak{s}(x'_i) = \mathfrak{s}(x'_j)$ with $i \neq j$, then $x'_i = x'_j$ as $x'_i, x'_j$ are $\sim$ -normalized, thus $\mathfrak{s}(\boldsymbol{y}'_i) = \mathfrak{s}(\boldsymbol{y}'_j)$ hence $\boldsymbol{y}'_i = \boldsymbol{y}'_j$ (as $\boldsymbol{y}'_i, \boldsymbol{y}'_j$ are $\sim$-normalized), hence $\alpha'_i = \alpha'_j$, which contradicts the fact that the atoms $\alpha'_i$ are pairwise distinct. Consequently there exist disjoint subheaps $\mathfrak{h}_i = \{(\mathfrak{s}(\boldsymbol{x}'_i), \mathfrak{s}(\boldsymbol{y}'_i))\}$ of $\mathfrak{h}$ (for all $i \in \{1, \ldots, m\}$) such that $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathcal{R}} \alpha'_i$. If $\beta = \top$ then we get $(\mathfrak{s}, \mathfrak{h}_1 \cup \cdots \cup \mathfrak{h}_n) \models_{\mathcal{R}} \alpha'_1 * \cdots * \alpha'_n = \alpha_1 * \cdots * \alpha_n$ hence $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \alpha_1 * \cdots * \alpha_n * \top$ (as $\mathfrak{h}_1 \cup \cdots \cup \mathfrak{h}_n \subseteq \mathfrak{h}$) and the proof is completed. Otherwise we must have $\beta = \beta'$ and $n = m$ so that $(\mathfrak{s}, \mathfrak{h}_1 \cup \cdots \cup \mathfrak{h}_m) \models_{\mathcal{R}} \alpha'_1 * \ldots \alpha'_m = \alpha_1 * \ldots \alpha_n = \phi{\downarrow_\sim}$.

For every formula $\phi$ and for every set of variables $\mathcal{W}$, we denote by $\mathcal{A}_{\mathcal{R}}(\phi, \mathcal{W})$ the set of abstractions of the models $(\mathfrak{s}, \mathfrak{h})$ that interpret exactly the variables in $\mathcal{W}$, i.e., $\mathcal{A}_{\mathcal{R}}(\phi, \mathcal{W}) = \{\mathcal{A} \mid (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi,\ dom(\mathfrak{s}) = \mathcal{W} \cup C,\ \mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})\}$. The following lemma stems from Lem. 4.10 and from the definitions of $\Phi_\sim(\mathfrak{s}, \mathfrak{h})$ and $\mathcal{A}_{\mathcal{R}}(\xi, \mathcal{W})$ (see App. B):

**Lemma 4.11.** *For every $\mathfrak{h}$-clause $\gamma$, we denote by $\overline{\gamma}$ the set of $\mathfrak{h}$-literals complementary to those occurring in $\gamma$. Let $\xi \models_{\mathcal{R}} \gamma$ be an instance of* Ent*(DSH,HC), with $fv(\xi) \cup fv(\gamma) \subseteq V$. The entailment $\xi \models_{\mathcal{R}} \gamma$ is valid iff for all $\mathcal{A} \in \mathcal{A}_{\mathcal{R}}(\xi, V)$, $\mathcal{A} \not\models \overline{\gamma}$.*

## 5 Computing abstractions

We now show how to compute abstractions. To achieve this, we introduce two basic operations on abstractions. The first one consists in removing a variable from $V$. If $\mathcal{A} = (V, \sim, A, R, \Phi, \Delta)$ and $x \in V$ then we denote by $\mathcal{A} \setminus \{x\}$ the abstraction of the form $(V \setminus \{x\}, \sim', A', R', \Phi', \Delta')$ where $\sim' = \{(u, v) \mid u \sim v, u \neq x, v \neq x\}$, and:

- If there exists a term $y \neq x$ such that $x \sim y$ then $A'$, $R'$, $\Phi'$ and $\Delta'$ are obtained from $A$, $R$, $\Phi$ and $\Delta$ respectively by replacing all occurrences of $x$ by $y{\downarrow_\sim}$.
- Otherwise, $A' = A \setminus \{x\}$, $R' = R \setminus \{x\}$, $\Phi'$ is obtained from $\Phi$ by replacing every atom containing $x$ by $\top$ and $\Delta'$ is the set of formulas in $\Delta$ that do not contain $x$.

*Example 5.1.* With the definitions of Ex. 4.7 we have:

$$\mathcal{A} \setminus \{x\} = (\{y, z_1, z_2\}, \emptyset, \{y\}, \{y, \mathtt{nil}\}, \top * y \mapsto (\mathtt{nil}, z_2), \Delta)$$
$$\mathcal{A}' \setminus \{u\} = (\{v, w\}, \emptyset, \{v\}, \{w, \mathtt{nil}\}, v \mapsto (\mathtt{nil}, w) * \top, \{w \geq w, w \approx w\})$$
$$\mathcal{A}' \setminus \{w\} = (\{u, v\}, \{u \simeq v\}, \{u\}, \{\mathtt{nil}\}, \top, \emptyset)$$

The second operation consists in computing the disjoint union of two abstractions. Let $\mathcal{A}_i = (V_i, \sim_i, A_i, R_i, \Phi_i, \Delta_i)$ be two abstractions (with $i \in \{1, 2\}$). The abstraction $\mathcal{A}_1 * \mathcal{A}_2$ is defined if $V_1 = V_2$, $\sim_1 = \sim_2$, $A_1 \cap A_2 = \emptyset$ and $\Delta_1 = \Delta_2$, and in this case $\mathcal{A}_1 * \mathcal{A}_2$ is $(V_1, \sim_1, A_1 \cup A_2, R_1 \cup R_2, \Phi_1 * \Phi_2, \Delta_1)$.

*Example 5.2.* Consider the abstraction $\mathcal{A}$ of Ex. 4.7 together with:

$$\mathcal{A}_1 = (\{x, x', y, z_1, z_2\}, \emptyset, \emptyset, \emptyset, \top, \Delta) \qquad \mathcal{A}_2 = (\{x, y, z_1, z_2\}, \emptyset, \{x\}, \emptyset, \top, \Delta)$$
$$\mathcal{A}_3 = (\{x, y, z_1, z_2\}, \emptyset, \{\mathtt{nil}\}, \{x\}, \mathtt{nil} \mapsto (x), \Delta)$$

Then $\mathcal{A} * \mathcal{A}_i$ is undefined if $i = 1$ (as $\mathcal{A}_1$ has a variable $x'$ that is not in $\mathcal{A}$) or if $i = 2$ (as $\mathcal{A}$ and $\mathcal{A}_2$ both allocate $x$), and $\mathcal{A} * \mathcal{A}_3$ is:

$$(\{x, y, z_1, z_2\}, \emptyset, \{x, y, \mathtt{nil}\}, \{x, y, \mathtt{nil}\}, x \mapsto (y, z_1) * y \mapsto (\mathtt{nil}, z_2) * \mathtt{nil} \mapsto (x), \Delta)$$

Lemma 5.3 relates these operators to the corresponding operations on the abstracted structures (the proof is given in App. C).

**Lemma 5.3.** *The two following assertions hold:*

1. *If $\mathcal{A}_i \triangleright (\mathfrak{s}, \mathfrak{h}_i)$, for all $i \in \{1, 2\}$, $dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2) = \emptyset$ and $\mathcal{A}_1 * \mathcal{A}_2$ is defined, then $\mathcal{A}_1 * \mathcal{A}_2 \triangleright (\mathfrak{s}, \mathfrak{h}_1 \cup \mathfrak{h}_2)$.*
2. *If $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$, and $\mathfrak{s}'$ is the restriction of $\mathfrak{s}$ to the variables distinct from $x$, then $\mathcal{A} \setminus \{x\} \triangleright (\mathfrak{s}', \mathfrak{h})$.*

Using the above operations on abstraction, we define a set of rules (Fig. 1) that inductively compute the set of abstractions $\mathcal{A}^*_{\mathcal{R}}(\phi, \mathcal{W})$, for all DSH $\phi$. The first two rules correspond to base cases, where $\phi$ is atomic. The first rule tackles the case where $\phi = \mathtt{emp}$. In this case, both $A$ and $R$ are empty (as the heap is empty) and $\Phi$ is $\mathtt{emp}$. The second rule handles the case where $\phi$ is a points-to atom $x_0 \mapsto (x_1, \ldots, x_n)$. Then $A$ contains the representative of $x_0$ (as it is the only allocated location), $R$ contains the representatives of the location terms in $x_1, \ldots, x_n$, and $\Phi$ is simply the $\sim$-normalized

form of $\phi$. The next four rules cover conjunctions of the form $\phi \wedge \alpha$ where $\phi$ is a symbolic heap and $\alpha$ is either an equational literal or a data literal. In each case, one only has to compute abstractions of $\phi$ and check whether $\alpha$ is satisfied. Note that it is impossible to compute abstractions of $\alpha$ as the latter formula is not a symbolic heap. Abstractions of existential quantifications $\exists x\, \phi$ are computed by removing the variable $x$ from the abstractions of $\phi$ using the operation $\mathcal{A} \setminus \{x\}$ defined above, and abstractions of separating conjunctions $\phi_1 * \phi_2$ are computed by combining abstractions of $\phi_1$ and $\phi_2$ using the operator $*$. The last rules handle the case of disjunctions and inductive definitions, respectively (the computation is straightforward in both cases).

*Remark 5.4.* Note that abstractions may contain variables not occurring in the considered formula, and are defined to associate a truth value with every equation and data atom (this is why we may assume, when combining abstractions $(V_i, \sim_i, A_i, R_i, \Phi_i, \Delta_i)$ using the operator $*$, that $V_1 = V_2$, $\sim_1 = \sim_2$ and $\Delta_1 = \Delta_2$). This design choice was made for the sake of readability and conciseness, but it leads to some computational overhead, as not all of these variables and atoms are necessarily relevant for evaluating the formulas at hand. In practice, variables and constraints should be added on demand, when they become necessarily to check that the premises of the rules in Fig. 1 are satisfied.

$$\frac{\Delta \subseteq \mathcal{A}^D_\sim(V) \quad \sim \in \mathcal{E}(V)}{(V, \sim, \emptyset, \emptyset, \mathtt{emp}, \Delta) \in \mathcal{A}^*_\mathcal{R}(\mathtt{emp}, V)}$$

$$\frac{\sim \in \mathcal{E}(V) \quad \Delta \subseteq \mathcal{A}^D_\sim(V) \quad \Phi = \{x_0{\downarrow_\sim} \mapsto (x_1{\downarrow_\sim}, \ldots, x_n{\downarrow_\sim}\}}{(V, \sim, \{x_0{\downarrow_\sim}\}, \{x_i{\downarrow_\sim} \mid i \in \{1, \ldots, n\}, \mathtt{sort}(x_i) = \mathtt{loc}\}, \Phi, \Delta) \in \mathcal{A}^*_\mathcal{R}(x_0 \mapsto (x_1, \ldots, x_n), V)}$$

$$\frac{x \sim y \quad (V, \sim, A, R, \Phi, \Delta) \in \mathcal{A}^*_\mathcal{R}(\phi, V)}{(V, \sim, A, R, \Phi, \Delta) \in \mathcal{A}^*_\mathcal{R}(\phi \wedge (x \simeq y), V)} \qquad \frac{x \not\sim y \quad (V, \sim, A, R, \Phi, \Delta) \in \mathcal{A}^*_\mathcal{R}(\phi, V)}{(V, \sim, A, R, \Phi, \Delta) \in \mathcal{A}^*_\mathcal{R}(\phi \wedge (x \not\simeq y), V)}$$

$$\frac{\alpha \in \Delta \quad (V, \sim, A, R, \Phi, \Delta) \in \mathcal{A}^*_\mathcal{R}(\phi, V)}{(V, \sim, A, R, \Phi, \Delta) \in \mathcal{A}^*_\mathcal{R}(\phi \wedge \alpha, V)} \qquad \frac{\alpha \notin \Delta \quad (V, \sim, A, R, \Phi, \Delta) \in \mathcal{A}^*_\mathcal{R}(\phi, V)}{(V, \sim, A, R, \Phi, \Delta) \in \mathcal{A}^*_\mathcal{R}(\phi \wedge \neg\alpha, V)}$$

$$\frac{\mathcal{A} \in \mathcal{A}^*_\mathcal{R}(\phi, V)}{\mathcal{A} \setminus \{x\} \in \mathcal{A}^*_\mathcal{R}(\exists x\, \phi, V \setminus \{x\})} \qquad \frac{\forall i \in \{1, 2\}\, \mathcal{A}_i \in \mathcal{A}^*_\mathcal{R}(\phi_i, V) \quad \mathcal{A}_1 * \mathcal{A}_2 \text{ is defined}}{\mathcal{A}_1 * \mathcal{A}_2 \in \mathcal{A}^*_\mathcal{R}(\phi_1 * \phi_2, V)}$$

$$\frac{\mathcal{A} \in \mathcal{A}_\mathcal{R}(\phi_i, V) \quad i \in \{1, 2\} \quad \mathit{fv}(\phi_1 \vee \phi_2) \subseteq V}{\mathcal{A} \in \mathcal{A}_\mathcal{R}(\phi_1 \vee \phi_2, V)}$$

$$\frac{\mathcal{A} \in \mathcal{A}^*_\mathcal{R}(\phi, V) \quad p(\boldsymbol{x}) \rightsquigarrow_\mathcal{R} \phi \quad p \in \mathcal{P}_s}{\mathcal{A} \in \mathcal{A}^*_\mathcal{R}(p(\boldsymbol{x}), V)}$$

The formula $\alpha$ denotes a data atom. For all rules, the following additional condition applies: $(V, \sim, A, \Phi, \Delta) \in \mathcal{A}_\mathcal{R}(\phi, V)$ only when $V \supseteq \mathit{fv}(\phi)$ and $\Delta$ is consistent w.r.t. $\sim$ and $V$ (this property is decidable by Assumptions 1 and 2).

**Fig. 1.** Inductive rules for computing abstractions of a formula

Lemma 5.5 asserts that the rules for computing abstractions are correct and complete, i.e., that the computed set of abstractions $\mathcal{A}^*_\mathcal{R}(\phi, \mathcal{W})$ is indeed the set of all ab-

stractions of the models of $\phi$. The result crucially relies on Assumption 3 (see App. D).

**Lemma 5.5.** *For all DSH $\phi$ and all $\mathcal{W} \supseteq fv(\phi)$, $\mathcal{A}^*_{\mathcal{R}}(\phi, \mathcal{W}) = \mathcal{A}_{\mathcal{R}}(\phi, \mathcal{W})$.*

Lemmata 4.11 and 5.5 yield an algorithm for testing the validity of entailment $\xi \models_{\mathcal{R}} \gamma$ in Ent(DSH,HC), described in the proof of Th. 5.6.

**Theorem 5.6.** *The problem* Ent*(DSH,HC) is decidable, and it is* ExpTime*-complete if the satisfiability test is in* ExpTime *for data constraints.*

*Proof.* The lower bound stems from the ExpTime-hardness of the satisfiability problem for symbolic heaps [2] (with $\mathcal{P}_d = \{\approx\}$). The decision procedure runs as follows. Consider an instance $\xi \models_{\mathcal{R}} \gamma$ of Ent(DSH,HC). We first compute the set $\mathcal{A}^*_{\mathcal{R}}(\xi, V)$ using the rules in Fig. 1, where $V = fv(\xi) \cup fv(\gamma)$. To this purpose, we only have to compute the sets $\mathcal{A}^*_{\mathcal{R}}(\phi, \mathcal{W})$ for formulas $\phi$ occurring either in $\xi$ or in some instance of a rule in $\mathcal{R}$, where $\mathcal{W}$ contains all variables occurring in $\xi, \gamma$ or $\mathcal{R}$. By Lem. 5.5, we have $\mathcal{A}^*_{\mathcal{R}}(\phi, V) = \mathcal{A}_{\mathcal{R}}(\phi, V)$. Then, by Lem. 4.11, to test whether $\xi \models_{\mathcal{R}} \gamma$, it suffices to test whether $\mathcal{A} \not\models \overline{\gamma}$ holds for all $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\xi, V)$.

We now briefly analyze the complexity of the procedure. We first observe that the number of formulas $\phi$ to consider is simply exponential w.r.t. $|\xi| + |\mathcal{R}|$ (up to a renaming of variables). The test $\mathcal{A} \models \overline{\gamma}$ is decidable in polynomial time (see Def. 4.8). Furthermore, the size of the abstractions is polynomial in $|\xi| + |\gamma| + |\mathcal{R}|$. Indeed, it is possible to prove that it is sufficient to compute abstractions with variables occurring in $\mathcal{W}$, so that the number of terms is bounded by $card(\mathcal{W} \cup C) \leq |\xi| + |\gamma| + |\mathcal{R}|$ thus the $E$-relation $\sim$ is of quadratic size, the sets $A$ and $R$ are of linear size and the set of data atoms $\Delta$ is of size $O(card(\mathcal{P}_d) \times (|\xi| + |\mathcal{R}|)^n)$, where $n$ denotes the maximal arity of the predicates in $\mathcal{P}_d$ (which is fixed in the context). Finally, the size of points-to atoms is bounded by $|\xi| + |\gamma| + |\mathcal{R}|$, and the maximal number of points-to atom occurring in a spatial $\mathfrak{h}$-atom is bounded by $card(\mathcal{W} \cup C)$ (as the same term cannot be allocated twice, otherwise the $\mathfrak{h}$-atom is trivially unsatisfiable). This entails that there is exponentially many such abstractions (assuming, w.l.o.g., that every constant and inductive predicate occurs in $\xi$, $\gamma$ or $\mathcal{R}$). Finally, it is straightforward to check that each rule in Fig. 1 can be applied in exponential time w.r.t. the size of the abstractions (assuming the satisfiability problem is in ExpTime for data constraints). Thus the algorithm runs in exponential time.

## 6 Conclusion

A new decidable fragment of the entailment problem in SL has been identified. It is ExpTime-complete (if the satisfiability problem for data constraints is in ExpTime) and incomparable with previously known decidable fragments. A natural follow-up involves implementing the decision procedure and enhancing its efficiency, as outlined in Rem. 5.4. Although the theoretical complexity of the procedure remains unaffected, this approach has the potential to significantly improve the practical efficiency of the procedure by reducing the number of abstractions to be considered. One could also try to relax the assumptions on the data theory, by imposing additional syntactic restrictions on data literals.

## References


1. J. Berdine, C. Calcagno, and P. W. O'Hearn. A decidable fragment of separation logic. In *FSTTCS'04*, volume 3328 of *LNCS*. Springer, 2004.
2. J. Brotherston, C. Fuhs, J. A. N. Pérez, and N. Gorogiannis. A decision procedure for satisfiability in separation logic with inductive predicates. In *CSL'14*, pages 25:1–25:10. ACM, 2014.
3. C. Calcagno, H. Yang, and P. W. O'hearn. Computability and complexity results for a spatial assertion language for data structures. In *FSTTCS 2001*, pages 108–119. Springer, 2001.
4. T. Chen, F. Song, and Z. Wu. Tractability of separation logic with inductive definitions: Beyond lists. In R. Meyer and U. Nestmann, editors, *CONCUR 2017*, volume 85 of *LIPIcs*, pages 37:1–37:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
5. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 2007. release October, 12th 2007.
6. B. Cook, C. Haase, J. Ouaknine, M. J. Parkinson, and J. Worrell. Tractable reasoning in a fragment of separation logic. In *CONCUR'11*, volume 6901 of *LNCS*. Springer, 2011.
7. S. Demri, D. Galmiche, D. Larchey-Wendling, and D. Méry. Separation logic with one quantified variable. In *CSR'14*, volume 8476 of *LNCS*, pages 125–138. Springer, 2014.
8. M. Echenim, R. Iosif, and N. Peltier. Entailment checking in separation logic with inductive definitions is 2-exptime hard. In *LPAR 2020*, volume 73 of *EPiC Series in Computing*, pages 191–211. EasyChair, 2020.
9. M. Echenim, R. Iosif, and N. Peltier. Decidable entailments in separation logic with inductive definitions: Beyond establishment. In *CSL 2021*, EPiC Series in Computing. EasyChair, 2021.
10. M. Echenim, R. Iosif, and N. Peltier. Unifying decidable entailments in separation logic with inductive definitions. In A. Platzer and G. Sutcliffe, editors, *CADE 28*, volume 12699 of *LNCS*, pages 183–199. Springer, 2021.
11. M. Echenim and N. Peltier. A proof procedure for separation logic with inductive definitions and data. *J. Autom. Reason.*, 67(3):30, 2023.
12. C. Enea, O. Lengál, M. Sighireanu, and T. Vojnar. Compositional entailment checking for a fragment of separation logic. *Formal Methods Syst. Des.*, 51(3):575–607, 2017.
13. C. Enea, M. Sighireanu, and Z. Wu. On automated lemma generation for separation logic with inductive definitions. In *ATVA 2015*, pages 80–96, 2015.
14. R. Iosif, A. Rogalewicz, and J. Simacek. The tree width of separation logic with recursive definitions. In *CADE-24*, volume 7898 of *LNCS*, 2013.
15. R. Iosif, A. Rogalewicz, and T. Vojnar. Deciding entailments in inductive separation logic with tree automata. In F. Cassez and J. Raskin, editors, *ATVA 2014*, volume 8837 of *LNCS*, pages 201–218. Springer, 2014.
16. Q. L. Le and X. D. Le. An efficient cyclic entailment procedure in a fragment of separation logic. In O. Kupferman and P. Sobocinski, editors, *FoSSaCS 2023*, volume 13992 of *LNCS*, pages 477–497. Springer, 2023.
17. C. Matheja, J. Pagel, and F. Zuleger. A decision procedure for guarded separation logic complete entailment checking for separation logic with inductive definitions. *ACM Trans. Comput. Log.*, 24(1):1:1–1:76, 2023.
18. P. W. O'Hearn and D. J. Pym. The logic of bunched implications. *Bull. Symb. Log.*, 5(2):215–244, 1999.
19. J. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *LICS'02*, 2002.

## A Proof of Lemma 4.10

We handle the cases that have not been already covered in the body of the paper:

- If $\phi = (x \simeq y)$, then $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff \mathfrak{s}(x) = \mathfrak{s}(y)$ (Def. 2.4). By Def. 4.6 (2), $\mathfrak{s}(x) = \mathfrak{s}(y) \iff x \sim y$, and by Def. 4.8 (1) $x \sim y \iff \mathcal{A} \models \phi$.
- $\phi = \texttt{alloc}(x)$, then $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff \mathfrak{s}(x) \in dom(\mathfrak{h})$ (Def. 2.4). As $\mathfrak{s}(x) = \mathfrak{s}(x\downarrow_\sim)$, we get $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff \mathfrak{s}(x\downarrow_\sim) \in dom(\mathfrak{h})$. By Def. 4.6 (3), $\mathfrak{s}(x\downarrow_\sim) \in dom(\mathfrak{h}) \iff x\downarrow_\sim \in A$ (as $x\downarrow_\sim$ is $\sim$-normalized by definition). By Def. 4.8 (2) we have $x\downarrow_\sim \in A \iff \mathcal{A} \models \texttt{alloc}(x)$. Thus $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff \mathcal{A} \models \phi$.
- $\phi = \texttt{ref}(x)$, then $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff \exists \ell \in dom(\mathfrak{h})$ s.t. $\mathfrak{s}(x) \in \mathfrak{h}(\ell)$ (Def. 2.4). As $\mathfrak{s}(x) = \mathfrak{s}(x\downarrow_\sim)$, we get $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff \exists \ell \in dom(\mathfrak{h})$ s.t. $\mathfrak{s}(x\downarrow_\sim) \in \mathfrak{h}(\ell)$. By Def. 4.6 (4), we deduce $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff x\downarrow_\sim \in R$ (as $x\downarrow_\sim$ is $\sim$-normalized). By Def. 4.8 (3) we get $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff \mathcal{A} \models \phi$.
- If $\phi$ is a data atom, then $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff (\mathfrak{s},\emptyset) \models_{\mathcal{R}} \phi\downarrow_\sim$ (as the truth value of $\phi$ does not depend on the heap). By Def. 4.6 (6), we get $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff \phi\downarrow_\sim \in \Delta$ (since $\phi\downarrow_\sim \in \mathcal{A}^D_\sim(V)$, by definition, as $\phi\downarrow_\sim$ is $\sim$-normalized and $fv(\phi\downarrow_\sim) \subseteq V$), thus by Def. 4.8 (5) $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff \mathcal{A} \models \phi$.
- If $\phi = \neg\psi$, then by the previous items we get $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \psi \iff \mathcal{A} \models \psi$. Moreover $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff (\mathfrak{s},\mathfrak{h}) \not\models_{\mathcal{R}} \psi$ (Def. 2.4) and $\mathcal{A} \models \phi \iff \mathcal{A} \not\models \psi$ by Def. 4.8 (6) so that $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \iff \mathcal{A} \models \phi$.

## B Proof of Lemma 4.11

By definition, $\xi \models_{\mathcal{R}} \gamma$ iff $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \xi \implies (\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \gamma$ holds for all structures $(\mathfrak{s},\mathfrak{h})$ where $dom(\mathfrak{s}) \supseteq fv(\xi) \cup fv(\gamma)$. By definition of $\overline{\gamma}$, $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \gamma$ holds iff $(\hat{\mathfrak{s}},\mathfrak{h}) \not\models_{\mathcal{R}} \overline{\gamma}$, for all extensions $\hat{\mathfrak{s}}$ of $\mathfrak{s}$ to $fv(\gamma) \setminus dom(\mathfrak{s})$ (moreover, as we assume that the variables that are bound in $\gamma$ do not occur in $fv(\xi)$, we have $(\hat{\mathfrak{s}},\mathfrak{h}) \models_{\mathcal{R}} \xi \iff (\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \xi$). Thus $\xi \models_{\mathcal{R}} \gamma$ holds iff $(\hat{\mathfrak{s}},\mathfrak{h}) \models_{\mathcal{R}} \xi \implies (\hat{\mathfrak{s}},\mathfrak{h}) \not\models_{\mathcal{R}} \overline{\gamma}$ holds for all structures $(\hat{\mathfrak{s}},\mathfrak{h})$ where $dom(\hat{\mathfrak{s}}) = V$. By definition of $\mathcal{A}_{\mathcal{R}}(\xi, fv(\gamma))$, $(\hat{\mathfrak{s}},\mathfrak{h}) \models_{\mathcal{R}} \xi$ with $dom(\hat{\mathfrak{s}}) = V$ iff there exists an abstraction $\mathcal{A} \in \mathcal{A}_{\mathcal{R}}(\xi, V)$ such that $\mathcal{A} \triangleright (\hat{\mathfrak{s}},\mathfrak{h})$. By Lem. 4.10, for all such structures $(\hat{\mathfrak{s}},\mathfrak{h})$ and abstractions $\mathcal{A}$, $(\hat{\mathfrak{s}},\mathfrak{h}) \not\models_{\mathcal{R}} \overline{\gamma}$ iff $\mathcal{A} \not\models \overline{\gamma}$. Consequently, $\xi \models_{\mathcal{R}} \gamma$ holds iff $\mathcal{A} \not\models \overline{\gamma}$ for all abstractions $\mathcal{A} \in \mathcal{A}_{\mathcal{R}}(\xi, V)$.

## C Proof of Lemma 5.3

*Assertion 1.* Let $\mathcal{A}_i = (V_i, \sim_i, A_i, R_i, \Phi_i, \Delta_i)$, $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, and $\mathcal{A} = (V, \sim, A, R, \Phi, \Delta)$, with (since $\mathcal{A}_1 * \mathcal{A}_2 = \mathcal{A}$) $V_1 = V_2 = V$, $\sim_1 = \sim_2 = \sim$, $A = A_1 \cup A_2$, $A_1 \cap A_2 = \emptyset$, $R = R_1 \cup R_2$, $\Phi = \Phi_1 * \Phi_2$, $\Delta_1 = \Delta_2 = \Delta$. We have $\{x\downarrow_\sim \mid \mathfrak{s}(x) \in dom(\mathfrak{h})\} = \bigcup_{i=1}^2 \{x\downarrow_\sim \mid \mathfrak{s}(x) \in dom(\mathfrak{h}_i)\} = A_1 \cup A_2$ (by Cond. 3 in Def. 4.6). Thus $\{x\downarrow_\sim \mid \mathfrak{s}(x) \in dom(\mathfrak{h})\} = A$. Similarly, $\{x\downarrow_\sim \mid \exists \ell \in dom(\mathfrak{h})$ s.t. $\mathfrak{s}(x) \in \mathfrak{h}(\ell)\} = \bigcup_{i=1}^2 \{x\downarrow_\sim \mid \exists \ell \in dom(\mathfrak{h}_i)$ s.t. $\mathfrak{s}(x) \in \mathfrak{h}_i(\ell)\} = R_1 \cup R_2 = R$. Finally, $\Phi_\sim(\mathfrak{s},\mathfrak{h}) = \alpha_1 * \cdots * \alpha_n * \Phi^2_\sim(\mathfrak{s},\mathfrak{h})$, where $\{\alpha_1, \ldots, \alpha_n\}$ is the set of $\sim$-normalized atoms $v_0 \mapsto (v_1, \ldots, v_k)$ such that $\mathfrak{h}(\mathfrak{s}(v_0)) = (\mathfrak{s}(v_1), \ldots, \mathfrak{s}(v_n))$ and $\Phi^2_\sim(\mathfrak{s},\mathfrak{h}) = \texttt{emp}$ if $card(dom(\mathfrak{h})) = n$ and $\top$ otherwise. Similarly, $\Phi_\sim(\mathfrak{s},\mathfrak{h}_i) = \alpha^i_1 * \cdots * \alpha^i_{n_i} * \Phi^2_\sim(\mathfrak{s},\mathfrak{h}_i)$, where $\{\alpha^i_1, \ldots, \alpha^i_{n_i}\}$ is the set of $\sim$-normalized atoms $v_0 \mapsto (v_1, \ldots, v_k)$ such that $\mathfrak{h}_i(\mathfrak{s}(v_0)) =$

$(\mathsf{s}(v_1), \ldots, \mathsf{s}(v_n))$ and $\Phi^2_\sim(\mathsf{s}, \mathfrak{h}_i) = \mathtt{emp}$ if $card(dom(\mathfrak{h}_i)) = n_i$ and $\top$ otherwise. As $\mathfrak{h}$ is the disjoint union of $\mathfrak{h}_1$ and $\mathfrak{h}_2$, $\{\alpha_1, \ldots, \alpha_n\} = \bigcup_{i=1}^{2} \{\alpha^i_1, \ldots, \alpha^i_{n_i}\}$ with $n = n_1 + n_2$, thus $\alpha_1 * \cdots * \alpha_n = \alpha^1_1 * \cdots * \alpha^1_{n_1} * \alpha^2_1 * \cdots * \alpha^2_{n_2}$. Moreover, $\Phi^2_\sim(\mathsf{s}, \mathfrak{h}) = \top$ iff $card(dom(\mathfrak{h})) > n$, i.e., iff $card(dom(\mathfrak{h}_i)) > n_i$ for some $i \in \{1, 2\}$, thus $\Phi^2_\sim(\mathsf{s}, \mathfrak{h}_i) = \Phi^2_\sim(\mathsf{s}, \mathfrak{h}_1) * \Phi^2_\sim(\mathsf{s}, \mathfrak{h}_2)$ (up of neutrality of $\mathtt{emp}$ and contraction of $\top$). Consequently, $\Phi_\sim(\mathsf{s}, \mathfrak{h}) = \Phi_\sim(\mathsf{s}, \mathfrak{h}_1) * \Phi_\sim(\mathsf{s}, \mathfrak{h}_2) = \Phi_1 * \Phi_2 = \Phi$ and $\mathcal{A} \triangleright (\mathsf{s}, \mathfrak{h})$.

*Assertion 2.* Let $\mathcal{A} = (V, \sim, A, R, \Phi, \Delta)$ and $\mathcal{A} \setminus \{x\} = (V \setminus \{x\}, \sim', A', R', \Phi', \Delta')$.

- If $\mathsf{s}(x) = \mathsf{s}(y)$ for some $y \in (V \setminus \{x\} \cup C$ then $A'$, $R'$, $\Phi'$ and $\Delta'$ are identical to $A$, $R$, $\Phi$ and $\Delta$, up to the replacement of the variable $x$ by $y\downarrow_\sim$, and it is straightforward to check that all the conditions of Def. 4.6 hold. Thus $\mathcal{A} \setminus \{x\} \triangleright (\mathsf{s}', \mathfrak{h})$.
- Now assume that $\mathsf{s}(x) \notin \mathsf{s}((V \setminus \{x\}) \cup C)$. This entails that $u\downarrow_\sim = u\downarrow_{\sim'}$ for all terms $u \in (V \setminus \{x\}) \cup C$. We show that all the conditions of Def. 4.6 hold.
  1. As $\mathcal{A} \triangleright (\mathsf{s}, \mathfrak{h})$, we have $dom(\mathsf{s}) = V \cup C$. By definition $dom(\mathsf{s}') = dom(\mathsf{s}) \setminus \{x\} = (V \cup C) \setminus \{x\} = (V \setminus \{x\}) \cup C$.
  2. For all $u, v \in (V \setminus \{x\}) \cup C$, we have $u \sim v \iff \mathsf{s}(u) = \mathsf{s}(v)$, (as $\mathcal{A} \triangleright (\mathsf{s}, \mathfrak{h})$). Moreover $u \sim v \iff u \sim' v$ (as $\sim'$ is the restriction of $\sim$ to pairs not containing $x$), and, since $\mathsf{s}'$ is the restriction of $\mathsf{s}$ to variables other than $x$, $\mathsf{s}(u) = \mathsf{s}'(u)$ and $\mathsf{s}(v) = \mathsf{s}'(v)$. Thus $u \sim' v \iff \mathsf{s}'(u) = \mathsf{s}'(v)$.
  3. Let $u$ be a $\sim'$-normalized term in $(V \setminus \{x\}) \cup C$. Then $u \in V \cup C$ and $u$ is $\sim$-normalized (since $\mathsf{s}(x) \notin (V \setminus \{x\}) \cup C$), hence $u \in A \iff \mathsf{s}(u) \in dom(\mathfrak{h})$. As $u \neq x$, $u \in A' \iff u \in A$ and $\mathsf{s}'(u) = \mathsf{s}(u)$. Consequently $u \in A' \iff \mathsf{s}'(u) \in dom(\mathfrak{h})$.
  4. We prove in the same way that for all $\sim'$-normalized terms of sort $\mathtt{loc}$ $u \in (V \setminus \{x\}) \cup C$, $u \in R' \iff \exists \ell \in dom(\mathfrak{h})$ s.t. $\mathsf{s}'(u) \in \mathfrak{h}(\ell)$.
  5. It is clear that all atoms in $\Phi^1_\sim(\mathsf{s}', \mathfrak{h})$ also occur in $\Phi^1_\sim(\mathsf{s}, \mathfrak{h})$, since $\mathsf{s}'$ is a restriction of $\mathsf{s}$ and $u\downarrow_\sim = u\downarrow_{\sim'}$ for all terms in $(V \setminus \{x\}) \cup C$. If $\Phi^1_\sim(\mathsf{s}', \mathfrak{h}) = \Phi^1_\sim(\mathsf{s}, \mathfrak{h})$, then by Def. 4.4 we also have $\Phi^2_\sim(\mathsf{s}', \mathfrak{h}) = \Phi^2_\sim(\mathsf{s}, \mathfrak{h})$, so that $\Phi_\sim(\mathsf{s}', \mathfrak{h}) = \Phi_\sim(\mathsf{s}, \mathfrak{h})$. Otherwise, we must have $\Phi^2_\sim(\mathsf{s}', \mathfrak{h}) = \top$, so that $\Phi_\sim(\mathsf{s}', \mathfrak{h})$ may be obtained from $\Phi_\sim(\mathsf{s}, \mathfrak{h})$ (up to contraction $\top * \top = \top$) by replacing all atoms containing $x$ by $\top$.
  6. Let $\phi \in \mathcal{A}^D_\sim(V \setminus \{x\})$. Then we have $\phi \in \mathcal{A}^D_\sim(V)$, thus $\phi \in \Delta \iff (\mathsf{s}, \emptyset) \models_R \phi$. As $x \notin fv(\phi)$, we get $(\mathsf{s}, \emptyset) \models_R \phi \iff (\mathsf{s}', \emptyset) \models_R \phi$ and $\phi \in \Delta \iff \phi \in \Delta'$, so that $\phi \in \Delta' \iff (\mathsf{s}', \emptyset) \models_R \phi$.

## D Proof of Lemma 5.5

### $\mathcal{A}^*_R(\phi, \mathcal{W}) \subseteq \mathcal{A}_R(\phi, \mathcal{W})$

We need to establish a stronger inductive lemma, formalized as follows.

**Lemma D.1.** *A store $\mathsf{s}$ is* compatible *with a set of variables $V$, an E-relation $\sim$ and a set of $\sim$-normalized data atoms $\Delta$ if the three following conditions hold: (i) $dom(\mathsf{s}) = V \cup C$; (ii) for all $x, y \in V \cup C$, $x \sim y \iff \mathsf{s}(x) = \mathsf{s}(y)$; and (iii) for all $\psi \in \mathcal{A}^D_\sim(V)$, $(\mathsf{s}, \emptyset) \models_R \psi \iff \psi \in \Delta$.*

*For every abstraction $\mathcal{A} = (V, \sim, A, R, \Phi, \Delta) \in \mathcal{A}^*_R(\phi, V)$ and every store $\mathsf{s}$ compatible with $V, \sim$ and $\Delta$, there exists a heap $\mathfrak{h}$ such that $(\mathsf{s}, \mathfrak{h}) \models_R \phi$ and $\mathcal{A} \triangleright (\mathsf{s}, \mathfrak{h})$.*

Lem. D.1 entails that $\mathcal{A}^*_{\mathcal{R}}(\phi, \mathcal{W}) \subseteq \mathcal{A}_{\mathcal{R}}(\phi, \mathcal{W})$. Indeed, if an abstraction $\mathcal{A} = (V, \sim, A, , R, \Phi, \Delta)$ is in $\mathcal{A}^*_{\mathcal{R}}(\phi, \mathcal{W})$ then necessarily $V = \mathcal{W}$ and $\Delta$ is consistent w.r.t. $\sim$ and $V$, thus there exists a store $\mathfrak{s}$ such that for all $\alpha \in \mathcal{A}^D_{\sim}(V)$: $(\mathfrak{s}, \emptyset) \models_{\mathcal{R}} \alpha \iff \alpha \in \Delta$. As $\Delta$ is $\sim$-normalized, this store may be transformed into a store $\hat{\mathfrak{s}}$ compatible with $\mathcal{W}$, $\sim$ and $\Delta$ by letting: $\hat{\mathfrak{s}}(x) = \mathfrak{s}(x{\downarrow_\sim})$, for all terms $x \in \mathcal{W} \cup \mathcal{C}$. By Lem. D.1, we deduce that there exists a heap $\mathfrak{h}$ such that $(\hat{\mathfrak{s}}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ and $\mathcal{A} \triangleright (\hat{\mathfrak{s}}, \mathfrak{h})$, so that $\mathcal{A} \in \mathcal{A}_{\mathcal{R}}(\phi, \mathcal{W})$ by definition.

*Proof (of Lemma D.1).* The proof is by induction on $\mathcal{A}^*_{\mathcal{R}}(\phi, V)$. We distinguish several cases, according to the rule in Fig. 1 used to derive the conclusion $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi, V)$.

- Assume that $\phi = \texttt{emp}$, $A = R = \emptyset$, $\Phi = \texttt{emp}$, with $\Delta \subseteq \mathcal{A}^D_{\sim}(V)$ and $\sim \in \mathcal{E}(V)$. Taking $\mathfrak{h} = \emptyset$, we get $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$. Moreover, by Def. 4.4 $\Phi_\sim(\mathfrak{s}, \mathfrak{h}) = \texttt{emp}$, so that $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$ by Def. 4.6 (as $dom(\mathfrak{h}) = \emptyset$).
- Assume that $\phi = \psi \wedge (x \simeq y)$ and $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\psi, V)$, with $\Delta \subseteq \mathcal{A}^D_{\sim}(V)$, $\sim \in \mathcal{E}(V)$ and $x \sim y$. By the induction hypothesis, there exists a heap $\mathfrak{h}$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \psi$ and $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$. As $\sim \in \mathcal{E}(V)$ and $\mathfrak{s}$ is compatible with $\sim$, we also have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} x \simeq y$, so that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$.
- The proof is similar if $\phi$ is of the form $\psi \wedge x \not\simeq y$ or $\psi \wedge \alpha$ where $\alpha$ is a data literal.
- Assume that $\phi = x_0 \mapsto (x_1, \ldots, x_n)$, $A = \{x_0{\downarrow_\sim}\}$, $R = \{x_i{\downarrow_\sim} \mid i \in \{1, \ldots, n\}, \texttt{sort}(x_i) = \texttt{loc}\}$ and $\Phi = x_0 {\downarrow_\sim} \mapsto (x_1 {\downarrow_\sim}, \ldots, x_n {\downarrow_\sim})$. Let $\mathfrak{h} = \{(\mathfrak{s}(x_0), \ldots, \mathfrak{s}(x_n))\}$. By definition, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$. Moreover, by Def. 4.4 we have $\Phi_\sim(\mathfrak{s}, \mathfrak{h}) = \mathfrak{s}(x_0 {\downarrow_\sim}) \mapsto (\mathfrak{s}(x_1 {\downarrow_\sim}, \ldots, \mathfrak{s}(x_n {\downarrow_\sim})) = \phi{\downarrow_\sim}$, $dom(\mathfrak{h}) = \{\mathfrak{s}(x_0)\} = \mathfrak{s}(A)$, and $\{x \in V{\downarrow_\sim} \mid \exists \ell \in dom(\mathfrak{h}) \text{ s.t. } \mathfrak{s}(x) \in \mathfrak{h}(\ell)\} = \mathfrak{s}(R)$. Thus $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$.
- Assume that $\phi = \exists x\, \phi'$, $\mathcal{A} = \mathcal{A}' \setminus \{x\}$, with $\mathcal{A}' = (V', \sim', A', R', \Phi', \Delta') \in \mathcal{A}^*_{\mathcal{R}}(\psi, V \cup \{x\})$. We first construct an extension $\hat{\mathfrak{s}}$ of $\mathfrak{s}$ to $\{x\}$ that is compatible with $\sim'$ and $\Delta'$. This is straightforward if $x$ is of sort $\texttt{loc}$, as $\mathcal{L}$ is infinite: if $x \sim y$ for some $y \neq x$ then we let $\hat{\mathfrak{s}}(x) = \mathfrak{s}(y)$, otherwise, $\hat{\mathfrak{s}}(x)$ is some arbitrary chosen location not occurring in the image of $\mathfrak{s}$. If $x$ is of sort $\texttt{data}$, we assume that $\hat{\mathfrak{s}}$ does not exist and we derive a contradiction. The assumption entails that $(\mathfrak{s}, \emptyset)$ falsifies the formula $\psi = \exists x \bigwedge_{\rho \in \Delta'} \rho \wedge \bigwedge_{\rho \in \mathcal{A}^D_{\sim}(V') \setminus \Delta'} \neg\rho$. As $\mathcal{A}'$ is an abstraction, $\Delta$ is necessarily consistent w.r.t. $V'$, $\sim'$, hence $\psi$ admits a model $(\mathfrak{s}', \emptyset)$. By Assumption 3, there exists a formula $\psi'$ that is equivalent to $\psi$ and contains no quantifier, with $fv(\psi') \subseteq fv(\psi)$. By definition, $\psi'$ only contains atoms in $\mathcal{A}^D_{\sim}(V)$. Since $\mathfrak{s}$ is compatible with $V, \sim, \Delta$, $(\mathfrak{s}, \emptyset)$ and $(\mathfrak{s}', \emptyset)$ coincide on all atoms in $\mathcal{A}^D_{\sim}(V)$, which entails that $(\mathfrak{s}, \emptyset) \models_{\mathcal{R}} \psi'$, i.e., $(\mathfrak{s}, \emptyset) \models_{\mathcal{R}} \psi$, which contradicts our assumption. Thus $\hat{\mathfrak{s}}$ necessarily exists. By the induction hypothesis, this entails that there exists a heap $\mathfrak{h}$ such that $(\hat{\mathfrak{s}}, \mathfrak{h}) \models_{\mathcal{R}} \phi'$ and $\mathcal{A}' \triangleright (\hat{\mathfrak{s}}, \mathfrak{h})$. Then $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$, and by Lem. 5.3 (2), $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$.
- Assume that $\phi = \phi_1 * \phi_2$, with $\mathcal{A}_i = (V_i, \sim_i, A_i, R_i, \Phi_i, \Delta_i) \in \mathcal{A}^*_{\mathcal{R}}(\phi_i, V)$ (for all $i \in \{1, 2\}$) and $\mathcal{A} = \mathcal{A}_1 * \mathcal{A}_2$. Note that since $\mathcal{A}_1 * \mathcal{A}_2$ is defined we must have $V_i = V$, $\sim_i = \sim$, $\Delta_i = \Delta$ and $A_1 \cap A_2 = \emptyset$. By the induction hypothesis, there exist heaps $\mathfrak{h}_i$ (for all $i \in \{1, 2\}$) such that $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathcal{R}} \phi_i$, and $\mathcal{A}_i \triangleright (\mathfrak{s}, \mathfrak{h}_i)$. By renaming locations if needed, we may assume that $\mathfrak{h}_1$ and $\mathfrak{h}_2$ share no location, other than those in the image of $\mathfrak{s}$. As $A_1 \cap A_2 = \emptyset$, and $A_i = \{x{\downarrow_\sim} \mid \mathfrak{s}(x) \in dom(\mathfrak{h}_i)\}$ (by Cond. 3 in Def. 4.6)), this entails that $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are disjoint. By Lem. 5.3 (1) we get $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h}_1 \cup \mathfrak{h}_2)$.
- Assume that $\phi = \phi_1 \vee \phi_2$, with $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi_i, V)$ (for some $i \in \{1, 2\}$). By the induction hypothesis there exists a heap $\mathfrak{h}$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi_i$ and $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$. This entails that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ hence the proof is completed.

– Assume that $\phi$ is a spatial predicate atom, $\phi \rightsquigarrow_{\mathcal{R}} \psi$, and $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\psi, V)$. Then there exists a heap $\mathfrak{h}$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \psi$ and $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$. This entails that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ thus the proof is completed.

### $\mathcal{A}_{\mathcal{R}}(\phi, W) \subseteq \mathcal{A}^*_{\mathcal{R}}(\phi, W)$

Let $\mathcal{A} = (V, \sim, A, , R, \Phi, \Delta) \in \mathcal{A}_{\mathcal{R}}(\phi, V)$. By definition, there is a structure $(\mathfrak{s}, \mathfrak{h})$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$, $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$ and $dom(\mathfrak{s}) = V \cup C$. By definition of the semantics of inductive predicates, there is a (minimal) natural number $\kappa(\phi)$ such that $\phi \rightsquigarrow^{\kappa(\phi)}_{\mathcal{R}} \phi'$, $\phi'$ contains no spatial predicates and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi'$. We show that $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi, V)$ by induction on the pair $(\kappa(\phi), |\phi|)$. We distinguish several cases according to the form of $\phi$.

– If $\phi = \texttt{emp}$, then by definition $\mathfrak{h} = \emptyset$. As $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$, we get, by Def. 4.6, $A = R = \emptyset$, and $\Phi = \Phi_{\sim}(\mathfrak{s}, \emptyset) = \texttt{emp}$. Then $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi, V)$ using the first rule in Fig. 1.
– If $\phi$ is a points-to atom $x_0 \mapsto (x_1, \ldots, x_n)$, then by definition $\mathfrak{h} = (\mathfrak{s}(x_0), \ldots, \mathfrak{s}(x_n))$. As $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$, $\mathfrak{s}$ is compatible with $\sim$, hence $\mathfrak{s}(x_i \downarrow_{\sim}) = \mathfrak{s}(x_i)$ (for all $i \in \{0, \ldots, n\}$). Thus $\mathfrak{h} = (\mathfrak{s}(x_0 \downarrow_{\sim}), \ldots, \mathfrak{s}(x_n \downarrow_{\sim}))$. Moreover, $A = \{x_0 \downarrow_{\sim}\}$ and $R = \{x_i \downarrow_{\sim} | \ i \in \{1, \ldots, n\}, \texttt{sort}(x_i) = \texttt{loc}\}$. By Def. 4.4, we have $\Phi_{\sim}(\mathfrak{s}, \mathfrak{h}) = x_0 \downarrow_{\sim} \mapsto (x_1 \downarrow_{\sim}, \ldots, x_n \downarrow_{\sim})$, so that $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi, W)$ using the second rule in Fig. 1.
– if $\phi$ is of the form $\psi \wedge x \simeq y$, then $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \psi$ (with $\kappa(\psi) = \kappa(\phi)$ as no inductive rule applies on $x \simeq y$), $\{x, y\} \subseteq dom(\mathfrak{s})$ and $\mathfrak{s}(x) = \mathfrak{s}(y)$. Since $\mathcal{A} \triangleright (\mathfrak{s}, \mathfrak{h})$, this entails that (by Def. 4.6) that $x \sim y$. By the induction hypothesis, we deduce that $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi, W)$ so that $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi, W)$ using the third rule in Fig. 1.
– The proof is similar if $\phi = \psi \wedge \alpha$ and $\alpha$ is a disequation (using the fourth rule in Fig. 1) or a data literal (using the fifth or sixth rule depending on the sign of $\alpha$).
– Assume that $\phi = \exists x \psi$. There exists a formula $\phi'$ with no inductive predicate such that $\phi \rightsquigarrow^{\kappa(\phi)}_{\mathcal{R}} \phi'$ and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$. This entails that $\psi \rightsquigarrow^{\kappa(\phi)}_{\mathcal{R}} \psi'$ with $\phi' = \exists x \psi'$, and there exists an extension $\hat{\mathfrak{s}}$ of $\mathfrak{s}$ to $\{x\}$ such that $(\hat{\mathfrak{s}}, \mathfrak{h}) \models_{\mathcal{R}} \psi'$, hence $(\hat{\mathfrak{s}}, \mathfrak{h}) \models_{\mathcal{R}} \psi$. Let $\mathcal{A}'$ be the (necessarily unique, if the representative of each variable is fixed) abstraction such that $\mathcal{A}' \triangleright (\hat{\mathfrak{s}}, \mathfrak{h})$. By the induction hypothesis, we have $\mathcal{A}' \in \mathcal{A}^*_{\mathcal{R}}(\psi, W \cup \{x\})$, so that $\mathcal{A}' \setminus \{x\} \in \mathcal{A}^*_{\mathcal{R}}(\phi, W)$ using the seventh rule in Fig. 1. By Lem. 5.3 (2) we get $\mathcal{A}' \setminus \{x\} \triangleright (\mathfrak{s}, \mathfrak{h})$, and using the unicity of the abstraction of a structure, we deduce that $\mathcal{A} = \mathcal{A}' \setminus \{x\}$, hence $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi, W)$.
– Assume that $\phi = \phi_1 * \phi_2$. Then there exist heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathcal{R}} \phi_i$ (for all $i \in \{1, 2\}$) and $\mathfrak{h} = \mathfrak{h}_1 * \mathfrak{h}_2$. It is clear that $\kappa(\phi) = \kappa(\phi_1) + \kappa(\phi_2)$, so that $\kappa(\phi_i) \leq \kappa(\phi)$ for all $i \in \{1, 2\}$. Let $\mathcal{A}_i$ be be the abstraction of $(\mathfrak{s}, \mathfrak{h}_i)$. By the induction hypothesis, we have $\mathcal{A}_i \in \mathcal{A}^*_{\mathcal{R}}(\phi_i, W)$, so that $\mathcal{A}_1 * \mathcal{A}_2 \in \mathcal{A}^*_{\mathcal{R}}(\phi, W)$ using the eighth rule in Fig. 1. Using Lem 5.3 (1) and the unicity of the abstraction of $(\mathfrak{s}, \mathfrak{h})$, we deduce that $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi, W)$.
– Assume that $\phi = \phi_1 \vee \phi_2$. Then $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathcal{R}} \phi_i$, for some $i \in \{1, 2\}$. We get $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi_i, W)$, so that $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi, W)$ by the penultimate rule in Fig. 1.
– Assume that $\phi$ is a spatial predicate atom. Then there exists a formula $\psi$ such that $\phi \rightsquigarrow_{\mathcal{R}} \psi$, with $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \psi$ and $\kappa(\psi) = \kappa(\phi) - 1$. This entails, by the induction hypothesis, that $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\psi, W)$, so that $\mathcal{A} \in \mathcal{A}^*_{\mathcal{R}}(\phi, W)$, using the last rule in Fig. 1.