On the Entailment Problem in Dynamic Separation Logic with Inductive Definitions

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

Abstract

Separation Logic (SL) is a well-established framework for reasoning about programs that manipulate dynamic memory. To express and verify properties of custom recursive data structures, SL is extended with spatial predicates defined by user-specified inductive rules. Many verification problems reduce to deciding entailments between formulas involving these predicates. While the general entailment problem is undecidable, a broad class of inductive rules – known as PCE (Progressing, Connected, and Established) – has been identified for which entailment is decidable. In this work, we extend the study of the entailment problem to Dynamic Separation Logic (DSL), an extension of SL that includes dynamic modalities for reasoning about actions on the heap and store. We show that entailment in DSL remains decidable for PCE rules by proving that dynamic modalities can be automatically eliminated.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Automated reasoning

Keywords and phrases Separation logic, Dynamic logic, Entailment problem

Digital Object Identifier 10.4230/LIPIcs.CSL.2026.16

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

Acknowledgements The author wishes to thank the anonymous referees for their valuable comments.

1 Introduction

Separation Logic (SL) has emerged as a central formalism for reasoning about mutable data structures in program verification [14, 17]. Its core contribution lies in enabling local reasoning: the ability to specify and verify properties of program fragments by focusing only on the memory cells they access, without requiring global knowledge of the entire heap. This locality principle has proved useful in scaling formal verification to realistic programs, particularly those involving pointer manipulation and dynamic memory allocation. At the heart of SL is the *separating conjunction*, a connective that expresses spatial separation between heap fragments. Combined with inductive definitions with a fixpoint semantics, this allows for elegant and concise specifications of custom linked data structures such as lists or trees. This gives rise to rich logical systems that support expressive specifications. These extensions introduce significant challenges for automation: while satisfiability is decidable [2], the entailment problem – deciding whether one SL formula is a logical consequence of another – is undecidable in general. Consequently, a substantial body of work has been devoted to identifying fragments of Separation Logic with recursive definitions for which the entailment problem is decidable (see, e.g., [1, 3, 4, 11, 10, 6, 13, 18]). Among these, a particularly expressive and well-structured class of inductive definitions, known as PCE rules (for Progressing, Connected, and Established), was introduced in [12]. For this class, entailment in SL was shown to be 2-ExpTime complete [15]. The original proof is based on a reduction to monadic second-order logic (MSO) over graphs of bounded tree width, a highly general approach that, however, does not yield efficient procedures in practice. More

recently, optimal (doubly exponential) decision procedures for the PCE fragment have been proposed (e.g., [15, 9]), significantly improving the theoretical and practical feasibility of automated reasoning.

Despite these advances, traditional forms of SL remain insufficient to express and verify certain dynamic behaviors of programs, particularly those involving changes in control flow or temporal evolution of the heap. To address these limitations, Dynamic Separation Logic (DSL) has been proposed [5] as an extension of SL with dynamic modalities, inspired by dynamic logic. These modalities enable reasoning not only about static heap configurations but also about how these configurations evolve during program execution. By incorporating modalities for program actions, DSL enables to express the result of actions that operate on data structures directly within the logic's syntax (using construction of the form $[\mathtt{a}]\,\phi$, meaning that ϕ holds after action $\mathtt{a}$ is applied). This extension increases the expressiveness of SL, making it well-suited for modeling and verifying complex program behaviors, including dynamic allocation patterns. This approach provides a compelling alternative to traditional methods based on Hoare logic and weakest precondition calculi [16] for reasoning about sequential heap manipulations. It avoids the use of the separating implication, which, although essential in the weakest precondition calculus (as it enables to express extensions of the heap), poses challenges for automated reasoning.

This paper builds on these foundational insights to explore the entailment problem in Dynamic Separation Logic (DSL), focusing on inductive definitions that satisfy the PCE conditions established in [12]. We prove that entailment is decidable for a specific fragment – namely, formulas built on the connectives $\vee, \wedge, \star$ together with existential quantifications and dynamic modalities – by showing that the dynamic modalities introduced in DSL can be systematically and automatically eliminated (for PCE rules). Specifically, we present a transformation procedure that rewrites both the formulas and the inductive rules defining spatial predicates, while preserving equivalence. This reduction reduces the entailment problem in DSL to one in standard SL and enables the application of existing proof techniques and decision procedures. Remarkably, one of the crucial properties that enables this reduction is the same one that guarantees the decidability of the entailment problem [8]: the *establishment* property of [12]. Intuitively, this property ensures that the considered structures contain only a bounded number of "pending edges" (see Proposition 7).

The rest of the paper is structured as follows. Section 2 defines the syntax and semantics of the fragment of dynamic separation logic considered in this work. Section 3 recalls the definition of the properties of inductive rules that ensure the decidability of entailment. Section 4 introduces several useful transformations on formulas and rules, and shows that these transformations preserve equivalence under certain conditions. Section 5, the heart of the paper, presents a systematic and automatic procedure for eliminating dynamic modalities from formulas that meet the conditions defined in Section 3. Finally, Section 6 concludes the paper by outlining some interesting directions for future work.

2 Dynamic Separation Logic

We define the syntax and semantics of Dynamic Separation Logic as a natural extension of the definitions in [12], which address basic separation logic with recursive spatial predicates, and [5], which introduces dynamic modalities.

Basic notations

We begin by reviewing some basic notations. For any partial function f, $\mathsf{dom}(f)$ denotes the domain of f, $img(f)$ denotes its image (i.e., $img(f) = \{f(x) \mid x \in \mathsf{dom}(f)\}$) and $f|_D$ denotes the restriction of f to D. Two partial functions f and g are *disjoint* if their domains are disjoint, in which case $f \cup g$ denotes the function h of domain $\mathsf{dom}(f) \cup \mathsf{dom}(g)$ defined as follows:

$$h(x) = \begin{cases} f(x) & \text{if } x \in \mathsf{dom}(f) \\ g(x) & \text{if } x \in \mathsf{dom}(g) \\ \textit{undefined} & \text{otherwise} \end{cases}$$

For every function f and elements x, y, we denote by $f[x \leftarrow y]$ the function such that $\mathsf{dom}(f[x \leftarrow y]) = \mathsf{dom}(f) \cup \{x\}$, $f[x \leftarrow y](x) = y$ and $f[x \leftarrow y](x') = f(x')$ if $x' \in \mathsf{dom}(f) \setminus \{x\}$. For any binary relation $\rightarrow$, $\rightarrow^+$ (resp. $\rightarrow^*$) denotes the transitive closure (resp. the reflexive and transitive closure) of $\rightarrow$.

Syntax

We now define the syntax of DSL. We begin by defining the notion of *actions*, which are atomic operations on structures, including assignments $x \leftarrow y$, look-ups $x \leftarrow z.i$, redirections $x.i \leftarrow y$, disallocations $\texttt{dispose(x)}$ and allocations $x \leftarrow \texttt{cons}(z_1, \ldots, z_\kappa)$ (x is allocated with fields pointing to $z_1, \ldots, z_\kappa$). The semantics of actions is intuitive and will be formally defined in a later section.

▶ **Definition 1** (Actions). *Let $\mathcal{V}$ be a countably infinite set of* variables *and let $\kappa \in \mathbb{N}$ (denoting a number of record fields). The set of* actions *$\mathcal{A}$ is the set of expressions of the following forms:*

$$x \leftarrow y \quad | \quad x \leftarrow z.i \quad | \quad x.i \leftarrow y \quad | \quad \texttt{dispose(x)} \quad | \quad x \leftarrow \texttt{cons}(z_1, \ldots, z_\kappa)$$

with $x, y, z \in \mathcal{V}$, $x \neq z$, $z_1, \ldots, z_\kappa$ is a tuple of variables not containing x and $i \in \{1, \ldots, \kappa\}$.

The conditions $x \neq z$ and $x \notin \{z_1, \ldots, z_\kappa\}$ are meant to simplify forthcoming definitions (they are not restrictive; for instance, an action $z \leftarrow z.1$ could be replaced by a sequence of two actions $z' \leftarrow z.1$ and $z \leftarrow z'$, where z' is a fresh variable).

The syntax of DSL formulas includes the standard logical connectives $\vee, \wedge$ and quantifier $\exists$, as well as the special connective $\star$ of separation logic and a modality $[\mathtt{a}]\,\psi$, which asserts that formula ψ holds after action $\mathtt{a}$ is applied. Atomic formulas include points-to atoms $y \mapsto (x_1, \ldots, x_\kappa)$, stating that y is the only allocated locations and refers to the tuple $(x_1, \ldots, x_\kappa)$, as well a spatial predicate atoms, which are intended to denote recursive data structures (e.g., lists or trees). As usual in SL, negations are not allowed.

▶ **Definition 2** (Syntax of Dynamic Separation Logic). *Let $\mathcal{P}$ be a set of* predicate symbols*, where each symbol in $p \in \mathcal{P}$ is associated with a unique arity $\#(p)$. A* DSL formula *ϕ (or simply formula) is built inductively as follows:*

$$p(x_1, \ldots, x_n) \quad | \quad y \mapsto (x_1, \ldots, x_\kappa) \quad | \quad x \simeq y \quad | \quad x \not\simeq y \quad |$$

$$(\phi_1 \star \phi_2) \quad | \quad (\phi_1 \vee \phi_2) \quad | \quad (\phi_1 \wedge \phi_2) \quad | \quad \exists x\, \psi \quad | \quad [\mathtt{a}]\,\psi$$

where $n \in \mathbb{N}$, p is a predicate in $\mathcal{P}$ of arity n, x, x_i and y are variables, $\mathtt{a}$ is an action and ϕ_1, ϕ_2, ψ are formulas. The false formula $\bot$ is a shorthand for $x \not\simeq x$ (where x is an arbitrary variable) and $\texttt{emp}$ is a shorthand for $x \simeq x$.

We emphasize that $x \not\simeq y$ it not exactly the negation of $x \simeq y$, as both atoms require the heap to be empty (see Definition 5). For simplicity, the syntax excludes constants (e.g., `nil`). This is not a limitation, as constants can be readily represented by variables passed as parameters to every predicate symbol. Formulas of the form $y \mapsto (x_1, \ldots, x_\kappa)$ (resp. $p(x_1, \ldots, x_n)$) are called $\mapsto$-*atoms* (resp. $\mathcal{P}$-*atoms*). We denote by $|\phi|$ the size of the formula ϕ (i.e., the number of occurrences of symbols in it). If $S = \{\phi_1, \ldots, \phi_n\}$ is a finite set of formulas, then $\bigvee S$ denotes the formula $\bigvee_{i=1}^{n} \phi_i$. For every formula ϕ we denote by $\mathsf{fv}(\phi)$ the set of variables freely occurring in ϕ.

A formula is *static* if it contains no subformula of the form $[\mathtt{a}]\,\psi$. A *symbolic heap* is a static formula that contains no occurrence of $\vee$ or $\wedge$. If $\vec{x} = (x_1, \ldots, x_n)$ and $\vec{y} = (y_1, \ldots, y_n)$ are tuples of variables, then $\vec{x} \simeq \vec{y}$ is a shorthand for $\bigstar_{i=1}^{n}(x_i \simeq y_i)$ (we use $\star$ instead of $\wedge$ to obtain a symbolic heap).

We now introduce the inductive rules that define the semantics of the predicates in $\mathcal{P}$. A *set of inductive rules* (SID) is a set $\mathcal{R}$ of rules $p(x_1, \ldots, x_n) \Leftarrow \phi$, where p is a predicate of arity n in $\mathcal{P}$ and ϕ is a symbolic heap such that $\mathsf{fv}(\phi) \subseteq \{x_1, \ldots, x_n\}$. We assume that for any given predicate p there are finitely many rules of the above form (but the sets $\mathcal{P}$ and $\mathcal{R}$ may be infinite). Examples of inductive definitions will be presented in a later section. Given two predicate symbols $p, q \in \mathcal{P}$, we write $p \succ_{\mathcal{R}} q$ if $\mathcal{R}$ contains a rule of the form $p(x_1, \ldots, x_n) \Leftarrow \phi$ where q occurs in ϕ. Note that actions are not allowed inside inductive rules (such rules are used to denote recursive data structures of unbounded size, not sequences of operations).

A *substitution* is a mapping from variables to variables. If $x_1, \ldots, x_n$ are pairwise distinct, then the set $\{x_i \leftarrow y_i \mid 1 \leq i \leq n\}$ denotes a substitution σ such that $\sigma(x_i) = y_i$ for all $i \in \{1, \ldots, n\}$ and $\sigma(z) = z$ if $z \notin \{x_1, \ldots, x_n\}$. For any expression (formula, variable or tuples of variables) ϕ and substitution σ, $\phi\sigma$ denotes the expression obtained from ϕ by replacing every free occurrence of every variable x by $\sigma(x)$ (using α-renaming to avoid variable capture).

If $\mathcal{R}$ is a SID, we write $\psi \rightarrow_{\mathcal{R}} \psi'$ (and say that "ψ unfolds to ψ'") if $\mathcal{R}$ contains a rule $p(x_1, \ldots, x_n) \Leftarrow \phi$ and ψ' is obtained from ψ by replacing one occurrence of a formula $p(y_1, \ldots, y_n)$ by $\phi\{x_i \leftarrow y_i \mid 1 \leq i \leq n\}$ (i.e., the formal parameters x_i are replaced by the actual parameters y_i, the existential variables in ϕ are renamed if needed to avoid collisions with variables occurring in ψ).

Semantics

We are now in the position to define the semantics of DSL. The formulas are interpreted in structures defined as follows.

▶ **Definition 3** (Structures). *Let $\mathcal{L}$ be a countably infinite set of* locations. *A* store *is a function mapping every variable to a location. A* heap *is a partial function mapping some locations to κ-tuples of locations, with a finite domain. A* structure *is a pair $(\mathfrak{s}, \mathfrak{h})$, where $\mathfrak{s}$ is a store and $\mathfrak{h}$ is a heap.*
A location ℓ (resp. a variable x) is allocated *in a heap $\mathfrak{h}$ (resp. in a structure $(\mathfrak{s}, \mathfrak{h})$) if $\ell \in \mathsf{dom}(\mathfrak{h})$ (resp. if $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$).*

A heap $\mathfrak{h}$ of domain $\ell_1, \ldots, \ell_n$ with $\mathfrak{h}(\ell_i) = \vec{t_i}$ for all $i \in \{1, \ldots, n\}$ is denoted by $\{\ell_1 \mapsto \vec{t_1}, \ldots, \ell_n \mapsto \vec{t_n}\}$. We denote by $|\mathfrak{h}|$ the size of $\mathfrak{h}$, i.e., the cardinality of $\mathsf{dom}(\mathfrak{h})$ (which is finite by hypothesis) and by $\mathit{locs}(\mathfrak{h})$ the set of locations occurring in $\mathfrak{h}$, i.e.: $\mathit{locs}(\mathfrak{h}) = \{\ell_i \mid \ell_0 \in \mathsf{dom}(\mathfrak{h}), \mathfrak{h}(\ell_0) = (\ell_1, \ldots, \ell_\kappa), 0 \leq i \leq \kappa\}$.

The relation $\rightsquigarrow_{\mathtt{a}}$ formalizes the effect of applying an action on a structure:

▶ **Definition 4** (Applying Actions on Structures). *Let* $(\mathfrak{s},\mathfrak{h})$ *and* $(\mathfrak{s}',\mathfrak{h}')$ *be two structures and let* $\mathtt{a}\in\mathcal{A}$. *We write* $(\mathfrak{s},\mathfrak{h})\rightsquigarrow_{\mathtt{a}}(\mathfrak{s}',\mathfrak{h}')$ *if one of the following assertions holds:*

- $\mathtt{a}=x\leftarrow y$, $\mathfrak{h}'=\mathfrak{h}$ *and* $\mathfrak{s}'=\mathfrak{s}[x\leftarrow\mathfrak{s}(y)]$
- $\mathtt{a}=x.i\leftarrow y$, $\mathfrak{s}'=\mathfrak{s}$, $\mathfrak{h}(\mathfrak{s}(x))=(\ell_1,\ldots,\ell_\kappa)$ *and* $\mathfrak{h}'$ *is the following heap:* $\mathfrak{h}'=\mathfrak{h}[\mathfrak{s}(x)\leftarrow(\ell_1,\ldots,\ell_{i-1},\mathfrak{s}(y),\ell_{i+1},\ldots,\ell_\kappa)]$.
- $\mathtt{a}=x\leftarrow y.i$, $\mathfrak{h}'=\mathfrak{h}$, $\mathfrak{h}(\mathfrak{s}(y))=(\ell_1,\ldots,\ell_\kappa)$ *and* $\mathfrak{s}'=\mathfrak{s}[x\leftarrow\ell_i]$.
- $\mathtt{a}=\mathtt{dispose}(\mathtt{x})$, $\mathfrak{s}'=\mathfrak{s}$, $\mathfrak{s}(x)\in\mathsf{dom}(\mathfrak{h})$ *and* $\mathfrak{h}'$ *is the restriction of* $\mathfrak{h}$ *such that* $\mathsf{dom}(\mathfrak{h}')=\mathsf{dom}(\mathfrak{h})\setminus\{\mathfrak{s}(x)\}$.
- $\mathtt{a}=x\leftarrow\mathtt{cons}(y_1,\ldots,y_\kappa)$, $\ell\in\mathcal{L}\setminus\mathsf{dom}(\mathfrak{h})$, $\mathfrak{s}'=\mathfrak{s}[x\leftarrow\ell]$, $\mathfrak{h}'=\mathfrak{h}[\ell\leftarrow(\mathfrak{s}(y_1),\ldots,\mathfrak{s}(y_\kappa))]$.

An action $\mathtt{a}$ fails on $(\mathfrak{s},\mathfrak{h})$ *if there is no structure* $(\mathfrak{s}',\mathfrak{h}')$ *such that* $(\mathfrak{s},\mathfrak{h})\rightsquigarrow_{\mathtt{a}}(\mathfrak{s}',\mathfrak{h}')$.

The satisfiability relations $\models^n_{\mathcal{R}}$ and $\models_{\mathcal{R}}$ are defined as follows (the exponent n denotes the maximal number of nested applications of inductive rules in $\mathcal{R}$).

▶ **Definition 5** (Satisfiability Relation). *We write* $(\mathfrak{s},\mathfrak{h})\models^n_{\mathcal{R}}\phi$ *if one of the following conditions holds:*

- ϕ *is of the form* $(x\simeq y)$ *(resp.* $(x\not\simeq y)$*),* $\mathfrak{h}=\emptyset$ *and* $\mathfrak{s}(x)=\mathfrak{s}(y)$ *(resp.* $(\mathfrak{s}(x)\neq\mathfrak{s}(y))$*).*
- ϕ *is of the form* $x\mapsto(y_1,\ldots,y_\kappa)$ *and* $\mathfrak{h}=\{\mathfrak{s}(x)\mapsto(\mathfrak{s}(y_1),\ldots,\mathfrak{s}(y_\kappa))\}$.
- ϕ *is of the form* $\phi_1\vee\phi_2$ *(resp.* $\phi_1\wedge\phi_2$*) and* $(\mathfrak{s},\mathfrak{h})\models^n_{\mathcal{R}}\phi_i$ *holds for some* $i\in\{1,2\}$ *(resp. for all* $i\in\{1,2\}$*).*
- ϕ *is of the form* $\phi_1\star\phi_2$ *and there exist disjoint heaps* $\mathfrak{h}_1,\mathfrak{h}_2$ *such that* $\mathfrak{h}=\mathfrak{h}_1\cup\mathfrak{h}_2$ *and* $(\mathfrak{s},\mathfrak{h}_i)\models^n_{\mathcal{R}}\phi_i$*, for all* $i\in\{1,2\}$.
- ϕ *is of the form* $\exists x\,\psi$ *and there exists* $\ell\in\mathcal{L}$ *such that* $(\mathfrak{s}[x\leftarrow\ell],\mathfrak{h})\models^n_{\mathcal{R}}\psi$.
- ϕ *is of the form* $p(x_1,\ldots,x_n)$, $\phi\rightarrow_{\mathcal{R}}\psi$, $n>0$ *and* $(\mathfrak{s},\mathfrak{h})\models^{n-1}_{\mathcal{R}}\psi$.
- ϕ *is of the form* $[\mathtt{a}]\,\psi$, $\mathtt{a}$ *does not fail on* $(\mathfrak{s},\mathfrak{h})$ *and* $(\mathfrak{s}',\mathfrak{h}')\models^n_{\mathcal{R}}\psi$ *for all* $(\mathfrak{s}',\mathfrak{h}')$ *such that* $(\mathfrak{s},\mathfrak{h})\rightsquigarrow_{\mathtt{a}}(\mathfrak{s}',\mathfrak{h}')$.

We write $(\mathfrak{s},\mathfrak{h})\models_{\mathcal{R}}\phi$ *if* $(\mathfrak{s},\mathfrak{h})\models^n_{\mathcal{R}}\phi$ *for some* $n\in\mathbb{N}$*, and in this case* $(\mathfrak{s},\mathfrak{h})$ *is called an* $\mathcal{R}$-model *of* ϕ. *We write* $\phi\models_{\mathcal{R}}\psi$ *if the entailment* $(\mathfrak{s},\mathfrak{h})\models_{\mathcal{R}}\phi\implies(\mathfrak{s},\mathfrak{h})\models_{\mathcal{R}}\psi$ *holds for all structures* $(\mathfrak{s},\mathfrak{h})$*, and* $\phi\equiv_{\mathcal{R}}\psi$ *if we have both* $\phi\models_{\mathcal{R}}\psi$ *and* $\psi\models_{\mathcal{R}}\phi$.

It is straightforward to verify that $\models^n_{\mathcal{R}}\subseteq\models^{n+1}_{\mathcal{R}}$ for all $n\in\mathbb{N}$. Moreover, if ϕ contains no predicate symbols, then the satisfiability relation does not depend on $\mathcal{R}$, and we may write $(\mathfrak{s},\mathfrak{h})\models\phi$ instead of $(\mathfrak{s},\mathfrak{h})\models_{\mathcal{R}}\phi$. Finally, if $(\mathfrak{s},\mathfrak{h})\models_{\mathcal{R}}\phi$, then there exists a formula ψ containing no symbols from $\mathcal{P}$ such that $\phi\rightarrow^*_{\mathcal{R}}\psi$ and $(\mathfrak{s},\mathfrak{h})\models_{\mathcal{R}}\psi$.

Note that equalities and disequalities are satisfied only by structures with an empty heap. This convention simplifies the notation by allowing us to omit standard conjunctions in certain cases, and it is not restrictive in our setting. Also note that there is no formula that is satisfied by all structures – this property is essential for the decidability of entailment in standard SL, if standard conjunctions are allowed [15].

3 PCE Rules

We now recall several definitions from [12], introducing conditions on the inductive rules that, in standard Separation Logic, ensure the decidability of the entailment problem. Intuitively, these conditions ensure that each rule allocates exactly one location, that the set of allocated locations has a tree-shaped structure, and that every location is either allocated or associated with a free variable.

▶ **Definition 6** (Progress, Connectedness and Establishment (PCE)). *A rule* $p(x_1, \ldots, x_n) \leftarrow \exists \vec{y}\, \phi$ *is:*

- progressing *if* ϕ *is of the form* $x_1 \mapsto (z_1, \ldots, z_\kappa) \star \phi'$*, where* ϕ' *contains no* $\mapsto$*-atom (i.e., the rule allocates exactly one location* x_1*);*
- connected *if, moreover, every predicate atom in* ϕ' *is of the form* $q(z, \vec{v})$ *with* $z \in \{z_1, \ldots, z_\kappa\}$ *(i.e., the locations allocated by the called predicates are successors of* x_1*).*

A SID $\mathcal{R}$ *is progressing (resp. connected) if all the rules in* $\mathcal{R}$ *are progressing (resp. connected). It is* established *if for every atom* $p(x_1, \ldots, x_n)$ *and for every formula* ϕ *containing no predicate symbol, if* $p(x_1, \ldots, x_n) \rightarrow^*_{\mathcal{R}} \phi$ *and* x *is existentially quantified in* ϕ *then* ϕ *contains atoms* $y_i \simeq y_{i+1}$ *(for* $i = 0, \ldots, n$*, with* $n \geq 0$*) such that* $x = y_{n+1}$ *and either* ϕ *contains a* $\mapsto$*-atom of the form* $y_0 \mapsto (\vec{z})$ *or* $y_0 \in \{x_1, \ldots, x_n\}$ *(i.e., every existentially quantified variable either is equal to a free variable or is eventually allocated).*

The following proposition recalls a key property of PCE rules: in every model of a quantifier-free formula, all locations that do not correspond to free variables are allocated.

▶ **Proposition 7.** *Let* ϕ *be a static formula containing no quantifier. Let* $(\mathfrak{s}, \mathfrak{h})$ *be an* $\mathcal{R}$*-model of* ϕ*. If* $\mathcal{R}$ *is PCE, then* $locs(\mathfrak{h}) \subseteq \mathsf{dom}(\mathfrak{h}) \cup \{\mathfrak{s}(x) \mid x \in \mathsf{fv}(\phi)\}$.

▶ **Example 8.** The following rules, defining a non empty list segment from x to y are PCE:

$$\mathtt{lseg}(x, y) \Leftarrow x \mapsto (y) \qquad \mathtt{lseg}(x, y) \Leftarrow \exists z \left(x \mapsto (z) \star \mathtt{lseg}(z, y)\right)$$

In contrast, the following rules, defining the same structures, are *not* PCE, because the second rule contains no $\mapsto$-atom (hence is not progressing):

$$\mathtt{lseg}'(x, y) \Leftarrow x \mapsto (y) \qquad \mathtt{lseg}'(x, y) \Leftarrow \exists z \left(\mathtt{lseg}'(x, z) \star \mathtt{lseg}'(z, y)\right)$$

The following rules, defining a list segment with last cell z, are not connected.

$$\mathtt{lseg}''(z, x, y) \Leftarrow z \mapsto (y) \star z \simeq x \qquad \mathtt{lseg}''(z, x, y) \Leftarrow \exists u \left(z \mapsto (y) \star \mathtt{lseg}(u, x, z)\right)$$

The following PCE rules define a tree rooted at x, where all nodes have an additional pointer to their parent node y and the leaves point to z:

$$\begin{array}{lcl} \mathtt{tree}(x, y, z) & \Leftarrow & \exists x_1, x_2 \left(x \mapsto (x_1, x_2, y) \star \mathtt{tree}(x_1, x, z) \star \mathtt{tree}(x_2, x, z)\right) \\ \mathtt{tree}(x, y, z) & \Leftarrow & x \mapsto (z, z, y) \end{array}$$

4 Some Basic Transformations on Inductive Predicates

In this section, we introduce a set of transformations that apply to formulas and the inductive rules defining the predicates they contain. These transformations preserve equivalence (under certain conditions) and will serve as the foundation for the algorithm that eliminates modalities, presented in the next section. We begin with the notion of *context predicates* (borrowed and adapted from [18, 9]), which enable the extraction of some predicate calls q from the rules of a predicate symbol p, lifting them to the initial formula.

▶ **Definition 9** (Context Predicates). *For every pair of predicates* p, q *with* $\#(p) = n$ *and* $\#(q) = m$*, we define a predicate* $(q \multimap p)$ *of arity* $n + m$ *associated with the following rules, for each rule of the form* $p(u_1, \ldots, u_n) \Leftarrow \exists \vec{w}\, (u_1 \mapsto (\vec{y}) \star p'(\vec{z}) \star \psi)$ *in* $\mathcal{R}$ *(modulo AC):*

$$(q \multimap p)(u_1, \ldots, u_n, v_1, \ldots, v_m) \Leftarrow \exists \vec{w}\, (u_1 \mapsto (\vec{y}) \star (q \multimap p')(\vec{z}, v_1, \ldots, v_m) \star \psi)$$

$$(q \multimap p)(u_1, \ldots, u_n, v_1, \ldots, v_m) \Leftarrow \exists \vec{w}\, (u_1 \mapsto (\vec{y}) \star \vec{z} \simeq (v_1, \ldots, v_m) \star \psi) \quad \text{if } q = p'$$

The rules of $(q \mathrel{-\!\!\bullet} p)(x_1, \ldots, x_n, y_1, \ldots, y_m)$ are obtained from those of $p(x_1, \ldots, x_n)$ by removing exactly one call to the atom $q(y_1, \ldots, y_m)$ (occurring at some non-root position in the derivation tree). Consequently, $(q \mathrel{-\!\!\bullet} p)(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is satisfied by all (non-empty) structures that will satisfy $p(x_1, \ldots, x_n)$ after a disjoint heap satisfying $q(y_1, \ldots, y_m)$ is added to the current heap. It is easy to check that the rules of $(q \mathrel{-\!\!\bullet} p)$ fulfill the conditions of Definition 6.

▶ **Example 10.** Let p, q be predicates associated with the following rules:

$$\begin{array}{lcl} p(x) & \Leftarrow & \exists y, z \left(x \mapsto (y, z) \star p(y) \star q(z, y)\right) \\ q(z, y) & \Leftarrow & \exists z' \left(z \mapsto (z', z') \star q(z', z)\right) \\ q(z, y) & \Leftarrow & z \mapsto (y, y) \end{array}$$

The predicate $(q \mathrel{-\!\!\bullet} p)$ is defined as follows:

$$\begin{array}{lcl} (q \mathrel{-\!\!\bullet} p)(x, u, v) & \Leftarrow & \exists y, z \left(x \mapsto (y, z) \star (q \mathrel{-\!\!\bullet} p)(y, u, v) \star q(z, y)\right) \\ (q \mathrel{-\!\!\bullet} p)(x, u, v) & \Leftarrow & \exists y, z \left(x \mapsto (y, z) \star p(y) \star (q \mathrel{-\!\!\bullet} q)(z, y, u, v)\right) \\ (q \mathrel{-\!\!\bullet} p)(x, u, v) & \Leftarrow & \exists y, z \left(x \mapsto (y, z) \star p(y) \star z \simeq u \star y \simeq v\right) \\ (q \mathrel{-\!\!\bullet} q)(z, y, u, v) & \Leftarrow & \exists z' \left(z \mapsto (z', z') \star (q \mathrel{-\!\!\bullet} q)(z', z, u, v)\right) \\ (q \mathrel{-\!\!\bullet} q)(z, y, u, v) & \Leftarrow & \exists z' \left(z \mapsto (z', z') \star z' \simeq u \star z \simeq v\right) \end{array}$$

The following lemma states that $q \mathrel{-\!\!\bullet} p$ denotes the structures obtained by removing the subheap satisfying q from a structure that satisfies p.

▶ **Lemma 11.** *Assume that $\mathcal{R}$ is PCE. Let $(\mathfrak{s}, \mathfrak{h})$ be a structure, let x be a variable and let $p(y_1, \ldots, y_n)$ be a formula. If $\mathfrak{s}(x) \in dom(\mathfrak{h})$ and $\mathfrak{s}(x) \neq \mathfrak{s}(y_1)$ then the two following assertions are equivalent:*

- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} p(y_1, \ldots, y_n)$.
- *There exists a predicate q of arity m, pairwise distinct variables $z_2, \ldots, z_m$, distinct from $x, y_1, \ldots, y_n$ and locations $\ell_2, \ldots, \ell_n$ such that $p \succ^+_{\mathcal{R}} q$ and:*

$$(\hat{\mathfrak{s}}, \mathfrak{h}) \models_{\mathcal{R}} (q \mathrel{-\!\!\bullet} p)(y_1, \ldots, y_n, x, z_2, \ldots, z_m) \star q(x, z_2, \ldots, z_m)$$

with $\hat{\mathfrak{s}} = \mathfrak{s}[z_i \leftarrow \ell_i \mid i \in \{2, \ldots, m\}]$.

The following result follows immediately from Lemma 11:

▶ **Corollary 12.** *Assume that $\mathcal{R}$ is PCE. Let $(\mathfrak{s}, \mathfrak{h})$ be a structure such that $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$. Let $(z_i)_{i \in \mathbb{N}}$ be a sequence of pairwise distinct variables, all distinct from $x, y_1, \ldots, y_n$. Then the following equivalence holds:*

$$(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} y_1 \not\simeq x \star p(y_1, \ldots, y_n) \iff (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} y_1 \not\simeq x \star \bigvee \{\phi_q \mid p \succ^+_{\mathcal{R}} q\}$$

with $\phi_q = \exists z_2, \ldots, z_m \left((q \mathrel{-\!\!\bullet} p)(y_1, \ldots, y_n, x, z_2, \ldots, z_m) \star q(x, z_2, \ldots, z_m)\right)$ and $m = \#(q)$.

The following definition associates each pair (ϕ, x), consisting of a static formula ϕ and a variable x, with a formula $\langle \phi \rangle^{\mathtt{N}(x)}$. This transformation enriches ϕ by explicitly adding assertions that ensure x is not allocated (see Lemma 15). It also implicitly extends the SID $\mathcal{R}$ by introducing new predicate symbols and rules, which enforce that x is not allocated within the data structures described by the predicates occurring in ϕ.

▶ **Definition 13.** *Let ϕ be a static formula and let x be a variable. We denote by $\langle \phi \rangle^{\mathtt{N}(x)}$ the formula inductively defined as follows.*

- *If ϕ is of the form $y \simeq z$ or $y \not\simeq z$ then $\langle \phi \rangle^{\mathtt{N}(x)} = \phi$.*

- *If ϕ is of the form $x' \mapsto (y_1, \ldots, y_\kappa)$ then $\langle\phi\rangle^{\mathtt{N}(x)} = \phi \star (x \not\simeq x')$.*
- *If $\phi = \phi_1 \wedge \phi_2$ then $\langle\phi\rangle^{\mathtt{N}(x)} = \langle\phi_1\rangle^{\mathtt{N}(x)} \wedge \phi_2$.*
- *If $\phi = \phi_1 \bullet \phi_2$ (with $\bullet \in \{\vee, \star\}$) then $\langle\phi\rangle^{\mathtt{N}(x)} = \langle\phi\rangle^{\mathtt{N}(x)} \bullet \langle\phi_2\rangle^{\mathtt{N}(x)}$.*
- *If $\phi = \exists y\, \psi$ then $\langle\phi\rangle^{\mathtt{N}(x)} = \exists y\, \langle\psi\rangle^{\mathtt{N}(x)}$ (we assume by α-renaming that $x \neq y$).*
- *If $\phi = p(x_1, \ldots, x_n)$ then $\langle\phi\rangle^{\mathtt{N}(x)} = p'(x_1, \ldots, x_n, x)$ if p' is a predicate symbol of arity $n+1$, associated with rules of the form $p'(y_1, \ldots, y_n, y) \Leftarrow \langle\phi\rangle^{\mathtt{N}(y)}$, for all rules of the form $p(y_1, \ldots, y_n) \Leftarrow \phi$ occurring in $\mathcal{R}$.*

Note that, alternatively, one could define $\langle\phi_1 \wedge \phi_2\rangle^{\mathtt{N}(x)}$ as $\langle\phi\rangle^{\mathtt{N}(x)} \wedge \langle\phi_2\rangle^{\mathtt{N}(x)}$; however, the chosen definition is more flexible and tends to yield simpler formulas. Due to the commutativity of $\wedge$, the transformation can be applied to either conjunct without loss of generality. Since the transformation can be applied recursively, it may generate predicates of the form p', p'', and so on. Consequently, both the set of rules $\mathcal{R}$ and the set of predicates $\mathcal{P}$ are infinite. This is not problematic in practice, as new predicates and rules are generated on demand.

▶ **Example 14.** Let ϕ be the formula $y \mapsto (u, v) \star (p(u) \vee p(v))$ where p is defined as in Example 10. Then $\langle\phi\rangle^{\mathtt{N}(x)}$ is $y \mapsto (u, v) \star y \not\simeq x \star (p'(u, x) \vee p'(v, x))$, where p' is defined as follows (x is renamed into x' in the first rule to avoid conflict):

$$\begin{array}{lcl} p'(x, x') & \Leftarrow & \exists y, z \left(x \mapsto (y, z) \star x' \not\simeq x \star p'(y, x') \star q'(z, y, x')\right) \\ q'(z, y, x) & \Leftarrow & \exists z' \left(z \mapsto (z', z') \star x \not\simeq z \star q'(z', z, x)\right) \\ q'(z, y, x) & \Leftarrow & z \mapsto (y, y) \star z \not\simeq x \end{array}$$

Note that in every model of ϕ, x is not allocated, because each $\mapsto$-atom $u \mapsto (\ldots)$ is associated with a disequation $x \not\simeq u$.

We now establish the soundness of the transformation. Lemma 15 shows that $\langle\phi\rangle^{\mathtt{N}(x)}$ indeed captures the intended properties, i.e., that ϕ is true and x is not allocated.

▶ **Lemma 15.** *For every structure $(\mathfrak{s}, \mathfrak{h})$, for every static formula and for every variable x, the following equivalence holds:*

$$(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)} \iff (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$$

We now introduce a similar but slightly more complex transformation. It associates every formula ϕ and variables $x, y_1, \ldots, y_\kappa$ with a formula $\langle\phi\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)}$ which is intended to denote the structures in which ϕ holds after a mapping $x \mapsto (y_1, \ldots, y_\kappa)$ is added to the heap (see Lemma 18). In other words $\langle\phi\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)}$ is equivalent to the formula $(x \mapsto (y_1, \ldots, y_\kappa)) \mathrel{-\!\!*} \phi$, where $\mathrel{-\!\!*}$ is the usual separating implication (a.k.a. "magic wand") of SL (as defined for instance in [17]).

▶ **Definition 16.** *Assume that $\mathcal{R}$ is PCE. Let ϕ be a static formula and let $x, y_1, \ldots, y_\kappa$ be variables. We denote by $\langle\phi\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)}$ the formula inductively defined as follows.*

- *If ϕ is of the form $y \simeq z$ or $y \not\simeq z$ then $\langle\phi\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)} = \bot$.*
- *If ϕ is of the form $x' \mapsto (y'_1, \ldots, y'_\kappa)$ then $\langle\phi\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)} = (x \simeq x') \star \bigstar_{i=1}^{\kappa} (y_i \simeq y'_i)$.*
- *If $\phi = \phi_1 \bullet \phi_2$ (with $\bullet \in \{\vee, \wedge\}$) then $\langle\phi\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)} = \langle\phi_1\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)} \bullet \langle\phi_2\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)}$.*
- *If $\phi = \phi_1 \star \phi_2$ then $\langle\phi\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)}$ is the formula: $(\langle\phi_1\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)} \star \phi_2) \vee (\phi_1 \star \langle\phi_2\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)})$.*
- *If $\phi = \exists z\, \psi$ then $\langle\phi\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)} = \exists z\, \langle\psi\rangle|_{x \mapsto (y_1, \ldots, y_\kappa)}$ (we assume by α-renaming that $z \notin \{x, y_1, \ldots, y_\kappa\}$).*

- *If* $\phi = p(z_1, \ldots, z_n)$ *and* $z_1 = x$ *then* $\langle p(z_1, \ldots, z_n)\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}$ *is:*

$$\bigvee\{\exists\vec{u}\,(\psi \star \bigstar_{j=1}^{\kappa}(y_i \simeq y_i')) \mid p(z_1, \ldots, z_n) \to_{\mathcal{R}} \exists\vec{u}\,(z_1 \mapsto (y_1', \ldots, y_\kappa') \star \psi)\}$$

  Note that the above set is finite (up to a renaming of variables) as there are finitely many rules associated with any given predicate p*.*
- *If* $\phi = p(z_1, \ldots, z_n)$ *and* $z_1 \neq x$ *then* $\langle\phi\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}$ *is:*

$$\begin{aligned}&(\langle p(x, z_2, \ldots, z_n)\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)} \star (z_1 \simeq x))\\ &\vee \bigvee_{p\succ_{\mathcal{R}}^{+}q,\#(q)=m} (\exists u_2, \ldots, u_m\, \langle q(x, u_2, \ldots, u_m)\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}\\ &\qquad\qquad \star (q \mathrel{-\!\!\bullet} p)(z_1, \ldots, z_n, x, u_2, \ldots, u_m) \star (z_1 \not\simeq x))\end{aligned}$$

It is straightforward to check that the computation of $\langle\phi\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}$ always terminates. Indeed, in every item except the last one, the size of ϕ decreases strictly at each recursive call, whereas in the last item, all the calls are of the form $\langle r(x, \vec{z})\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}$ (for some $r \in \mathcal{P}$) and thus terminate immediately as there is no recursive call for the formulas of this form (see penultimate item).

▶ **Example 17.** Let ϕ be the formula $x \mapsto (y) \star \mathtt{lseg}(y, z)$, where `lseg` is defined as in Example 8. Then $\langle\phi\rangle|_{u\mapsto(v)}$ is given by the following disjunction:

$$\begin{aligned}&x \simeq u \star y \simeq v \star \mathtt{lseg}(y, z)\\ \vee\ &x \mapsto (y) \star \exists y' \left(\mathtt{lseg}(y', z) \star y' \simeq v\right) \star y \simeq u\\ \vee\ &x \mapsto (y) \star \exists z' \left(z' \simeq v \star (\mathtt{lseg} \mathrel{-\!\!\bullet} \mathtt{lseg})(y, z, u, z')\right) \star y \not\simeq u\\ \vee\ &x \mapsto (y) \star \exists z', x' \left(x' \simeq v \star \mathtt{lseg}(x', z') \star (\mathtt{lseg} \mathrel{-\!\!\bullet} \mathtt{lseg})(y, z, u, z')\right) \star y \not\simeq u\end{aligned}$$

The structures described by $\langle\phi\rangle|_{u\mapsto(v)}$ are obtained by removing the mapping $u \mapsto (v)$ from the heap described by ϕ. The first disjunct corresponds to the case where the removed mapping is exactly $x \mapsto (y)$. The second disjunct handles the case where the removed mapping is the first one in the list segment $\mathtt{lseg}(y, z)$ (which entails that $y = u$). The third disjunct covers the case where the removed mapping is the last one in the list segment $\mathtt{lseg}(y, z)$. The fourth disjunct accounts for the case where the removed mapping is part of the structure generated by $\mathtt{lseg}(y, z)$, but is not its first or last element. The formula can be further simplified by replacing z' with z (since the second argument of `lseg` remains unchanged through recursive calls) and x', y' by v (based on the equations $x' \simeq v$ and $y' \simeq v$).

▶ **Lemma 18.** *Assume that* $\mathcal{R}$ *is PCE. Let* $(\mathfrak{s}, \mathfrak{h})$ *be a structure, let* ϕ *be a static formula and let* $x, y_1, \ldots, y_\kappa$ *be variables. If* $\mathfrak{h} = \mathfrak{h}' \cup \{\mathfrak{s}(x) \mapsto (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_\kappa))\}$ *(with* $\mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h}')$*), then:*

$$(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \iff (\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \langle\phi\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}$$

For any static formula ϕ and variable x, we denote by $\langle\phi\rangle^{\mathtt{A}(x)}$ the formula defined as follows: $\exists z_1, \ldots, z_\kappa \left(x \mapsto (z_1, \ldots, z_\kappa) \star \langle\phi\rangle|_{x\mapsto(z_1,\ldots,z_\kappa)}\right)$. Intuively, $\langle\phi\rangle^{\mathtt{A}(x)}$ asserts that ϕ holds and that x is allocated. The following two lemmata follow from Lemma 18.

▶ **Lemma 19.** *Assume that* $\mathcal{R}$ *is PCE. For every structure* $(\mathfrak{s}, \mathfrak{h})$*, for every static formula and variable* x *the following equivalence holds:*

$$(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \langle\phi\rangle^{\mathtt{A}(x)} \iff (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \wedge \mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$$

▶ **Lemma 20.** *For every structure* $(\mathfrak{s}, \mathfrak{h})$*, for every static formula and variable* x *the following two assertions are equivalent:*

- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} x \mapsto (y_1, \ldots, y_\kappa) \star \langle \phi \rangle|_{x \mapsto (y_1, \ldots, y_\kappa)}$.
- $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \wedge \mathfrak{s}(x) = (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_\kappa))$.

Proof. The proof is similar to that of Lemma 19. ◀

5 Elimination of Modalities

Building on the results in Section 4, we now show how dynamic modalities can be eliminated (if the rules satisfy the PCE conditions), thereby proving that any formula can be transformed into an equivalent static formula. We distinguish several cases, depending on the actions considered. In essence, eliminating modalities reduces to computing the weakest precondition of the corresponding modal formulas. The case of assignments can be handled using the following standard property:

▶ **Proposition 21.** *For all static formulas* ϕ *and all variables* x, y*:* $[x \leftarrow y]\, \phi \equiv_{\mathcal{R}} \phi\{x \leftarrow y\}$

Desallocations $[\texttt{dispose(x)}]\, \phi$ are also straightforward to handle: it is sufficient to assert that the desallocated variable x is initially allocated and that the remainder of the heap satisfies ϕ:

▶ **Proposition 22.** *For all static formulas* ϕ *for all variables* x*, and for all pairwise distinct variables* $y_1, \ldots, y_\kappa$ *not occurring in* ϕ*:*

$$[\texttt{dispose(x)}]\, \phi \;\equiv_{\mathcal{R}}\; \exists y_1, \ldots, y_\kappa \left(x \mapsto (y_1, \ldots, y_\kappa) \star \phi\right)$$

▶ **Example 23.** Consider the formula $\phi = [y \leftarrow x]\, [\texttt{dispose(y)}]\, \texttt{lseg}(x, x)$. We get:

$$\begin{array}{rcll} \phi & \equiv & [y \leftarrow x]\, \exists z \left(y \mapsto (z) \star \texttt{lseg}(x, x)\right) & \text{(Proposition 22)} \\ & \equiv & \exists z \left(x \mapsto (z) \star \texttt{lseg}(x, x)\right) & \text{(Proposition 21)} \\ & \equiv & \bot & \end{array}$$

While the previous reductions (for assignments and deallocations) apply to any formula and set of rules, the remaining actions are more challenging to address. Their handling relies on the transformations described in Section 4, which crucially depend on the rules being PCE. We begin by presenting results that aim to enforce additional properties on the formulas appearing under the considered modality, based on the semantics of actions and their post-conditions. These properties will later simplify the elimination of modalities. We first consider redirections and look-ups:

▶ **Lemma 24.** *Assume that* $\mathcal{R}$ *is PCE. Let* ϕ *be a static formula. If* $\texttt{a}$ *is an action of the form* $x.i \leftarrow y$ *or* $y \leftarrow x.i$*, then:* $[\texttt{a}]\, \phi \equiv_{\mathcal{R}} [\texttt{a}]\, \langle \phi \rangle^{\mathtt{A}(x)}$

Allocations are addressed in a similar way, using Lemma 20:

▶ **Lemma 25.** *Assume that* $\mathcal{R}$ *is PCE. Let* ϕ *be a static formula. If* $\texttt{a}$ *is an action of the form* $x \leftarrow \texttt{cons}(y_1, \ldots, y_\kappa)$*, then:* $[\texttt{a}]\, \phi \equiv_{\mathcal{R}} [\texttt{a}]\, (x \mapsto (y_1, \ldots, y_\kappa) \star \langle \phi \rangle|_{x \mapsto (y_1, \ldots, y_\kappa)})$

Next, Lemma 26 can be applied to eliminate redirection modalities, provided that the considered formula has a specific form, which will be ensured by applying Lemma 24.

▶ **Lemma 26.** *For all formulas ϕ, for all variables $x, y_1, \ldots, y_n, y$, for all $i \in \{1, \ldots, \kappa\}$, for all sequences of variables $\vec{z}$ (not containing x or y) and for all variables u not occurring in $x, y_1, \ldots, y_n, y$, $\vec{z}$ or ϕ, the following equivalence holds:*

$$[x.i \leftarrow y]\, \exists \vec{z} \big(x \mapsto (y_1, \ldots, y_\kappa) \star \phi)\big) \equiv_{\mathcal{R}} \exists \vec{z} \exists u \big(x \mapsto (y_1, \ldots, y_{i-1}, u, y_{i+1}, y_\kappa) \star \phi \star y_i \simeq y\big) \quad (1)$$

Proof. Let $(\mathfrak{s}, \mathfrak{h})$ be a structure. By Definition 4, $x.i \leftarrow y$ fails on $(\mathfrak{s}, \mathfrak{h})$ if $\mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$. Moreover, $(\mathfrak{s}, \mathfrak{h}) \rightsquigarrow_{x.i \leftarrow y} (\mathfrak{s}', \mathfrak{h}')$ exactly when $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$, $\mathfrak{s}' = \mathfrak{s}$ and $\mathfrak{h}' = \mathfrak{h}[\mathfrak{s}(x) \leftarrow (\ell_1, \ldots, \ell_{i-1}, \mathfrak{s}(y), \ell_{i+1}, \ldots, \ell_\kappa)]$ with $\mathfrak{h}(\mathfrak{s}(x)) = (\ell_1, \ldots, \ell_\kappa)$.

- Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} [x.i \leftarrow y]\, \exists \vec{z} \big(x \mapsto (y_1, \ldots, y_\kappa) \star \phi)\big)$. By Definition 5, we deduce that $x.i \leftarrow y$ does not fail on $(\mathfrak{s}, \mathfrak{h})$ (i.e., $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$), and that $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} (\exists \vec{z} (x \mapsto (y_1, \ldots, y_\kappa) \star \phi)$, with $\mathfrak{h}' = \mathfrak{h}[\mathfrak{s}(x) \leftarrow (\ell_1, \ldots, \ell_{i-1}, \mathfrak{s}(y), \ell_{i+1}, \ldots, \ell_\kappa)]$ and $\mathfrak{h}(\mathfrak{s}(x)) = (\ell_1, \ldots, \ell_\kappa)$. Consequently, there exist a vector of locations $\vec{m}$ and disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $\mathfrak{h}_1 \cup \mathfrak{h}_2 = \mathfrak{h}'$, $(\mathfrak{s}[\vec{z} \leftarrow \vec{m}], \mathfrak{h}_1) \models_{\mathcal{R}} x \mapsto (y_1, \ldots, y_\kappa)$ and $(\mathfrak{s}[\vec{z} \leftarrow \vec{m}], \mathfrak{h}_2) \models_{\mathcal{R}} \phi$. This entails that $\mathsf{dom}(\mathfrak{h}_1) = \{\mathfrak{s}(x)\}$ (as x does not occur in $\vec{z}$), $\mathsf{dom}(\mathfrak{h}_2) = \mathsf{dom}(\mathfrak{h}) \setminus \{\mathfrak{s}(x)\}$, $\mathfrak{s}[\vec{z} \leftarrow \vec{m}](y_j) = \ell_j$, for all $j \in \{1, \ldots, i-1, i+1, \ldots, \kappa\}$ and $\mathfrak{s}[\vec{z} \leftarrow \vec{m}](y_i) = \mathfrak{s}(y) = \mathfrak{s}[\vec{z} \leftarrow \vec{m}](y)$ (as y does not occur in $\vec{z}$). This implies that $\mathfrak{h} = \{\mathfrak{s}(x) \mapsto (\ell_1, \ldots, \ell_\kappa)\} \cup \mathfrak{h}_2$, and that $(\mathfrak{s}[\vec{z} \leftarrow \vec{m}], \{\mathfrak{s}(x) \mapsto (\ell_1, \ldots, \ell_\kappa)\}) \models_{\mathcal{R}} \exists u\, x \mapsto (y_1, \ldots, y_{i-1}, u, y_{i+1}, \ldots, y_\kappa)$. Consequently, $(\mathfrak{s}[\vec{z} \leftarrow \vec{m}], \mathfrak{h}) \models_{\mathcal{R}} \exists u \big(x \mapsto (y_1, \ldots, y_{i-1}, u, y_{i+1}, \ldots, y_\kappa) \star \phi\big)$. Since $\mathfrak{s}[\vec{z} \leftarrow \vec{m}](y_i) = \mathfrak{s}[\vec{z} \leftarrow \vec{m}](y)$, we also have $(\mathfrak{s}[\vec{z} \leftarrow \vec{m}], \mathfrak{h}) \models_{\mathcal{R}} \exists u \big(x \mapsto (y_1, \ldots, y_\kappa) \star \phi \star y \simeq y_i\big)$, so that $(\mathfrak{s}, \mathfrak{h})$ is an $\mathcal{R}$-model of $\exists \vec{z} \exists u \big(x \mapsto (y_1, \ldots, y_{i-1}, u, y_{i+1}, y_\kappa) \star \phi \star y_i \simeq y\big)$.
- Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \exists \vec{z} \exists u \big(x \mapsto (y_1, \ldots, y_{i-1}, u, y_{i+1}, y_\kappa) \star \phi \star y_i \simeq y\big)$. This means that there exist a sequence of locations $\vec{m}, \ell$ and disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, $(\mathfrak{s}[\vec{z} \leftarrow \vec{m}, u \leftarrow \ell], \mathfrak{h}_1) \models_{\mathcal{R}} x \mapsto (y_1, \ldots, y_{i-1}, u, y_{i+1}, y_\kappa)$, $(\mathfrak{s}[\vec{z} \leftarrow \vec{m}, \mathfrak{h}_2) \models_{\mathcal{R}} \phi$ and $\mathfrak{s}[\vec{z} \leftarrow \vec{m}](y_i) = \mathfrak{s}[\vec{z} \leftarrow \vec{m}](y) = \mathfrak{s}(y)$ (as $u \neq y_i$ and y does not occur in $\vec{z}$). Therefore $\mathsf{dom}(\mathfrak{h}_1) = \{\mathfrak{s}[\vec{z} \leftarrow \vec{m}, u \leftarrow \ell](x)\} = \{\mathfrak{s}(x)\}$ (as x does not occur in $\vec{z}, u$), $\mathsf{dom}(\mathfrak{h}_2) = \mathsf{dom}(\mathfrak{h}) \setminus \{\mathfrak{s}(x)\}$, and $\mathfrak{h}(\mathfrak{s}(x)) = \mathfrak{h}_1(\mathfrak{s}(x)) = (\ell_1, \ldots, \ell_\kappa)$, with $\ell_i = \mathfrak{s}[\vec{z} \leftarrow \vec{m}, u \leftarrow \ell](u) = \ell$ and $\ell_j = \mathfrak{s}[\vec{z} \leftarrow \vec{m}, u \leftarrow \ell](y_i) = \mathfrak{s}[\vec{z} \leftarrow \vec{m}](y_i)$ for all $j \in \{1, \ldots, i-1, i+1, \ldots, \kappa\}$. Let $\mathfrak{h}_1'$ be the heap $\{\mathfrak{s}(x) \mapsto (\ell_1, \ldots, \ell_{i-1}, \mathfrak{s}(y), \ell_{i+1}, \ldots, \ell_\kappa)\}$. From the previous assertions we derive that $(\mathfrak{s}[\vec{z} \leftarrow \vec{m}], \mathfrak{h}_1') \models_{\mathcal{R}} x \mapsto (y_1, \ldots, y_\kappa)$. As $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$, $x.i \leftarrow y$ does not fail on $(\mathfrak{s}, \mathfrak{h})$. Let $\mathfrak{h}' = \mathfrak{h}[\mathfrak{s}(x) \leftarrow (\ell_1, \ldots, \ell_{i-1}, \mathfrak{s}(y), \ell_{i+1}, \ldots, \ell_\kappa)]$. It is clear that $\mathfrak{h}' = \mathfrak{h}_1' \cup \mathfrak{h}_2$, so that $(\mathfrak{s}[\vec{z} \leftarrow \vec{m}], \mathfrak{h}') \models_{\mathcal{R}} x \mapsto (y_1, \ldots, y_\kappa) \star \phi$, and thus $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \exists \vec{z} (x \mapsto (y_1, \ldots, y_\kappa) \star \phi)$. It follows that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} [x.i \leftarrow y]\, \exists \vec{z} \big(x \mapsto (y_1, \ldots, y_\kappa) \star \phi)\big)$. ◀

▶ **Example 27.** Let $\phi = [x.1 \leftarrow y]\, \mathtt{lseg}(x, x)$. We get:

$$\begin{array}{lll}
\phi & \equiv & [x.1 \leftarrow y]\, \langle \mathtt{lseg}(x,x) \rangle^{\mathtt{A}(x)} \qquad \text{(Lemma 24)} \\
& \equiv & [x.1 \leftarrow y]\, \exists z \big(x \mapsto (z) \star \langle \mathtt{lseg}(x,x) \rangle_{|x \mapsto (z)}\big) \\
& \equiv & [x.1 \leftarrow y]\, \exists z, z' \big(x \mapsto (z) \star z \simeq z' \star \mathtt{lseg}(z', x)\big) \qquad \text{(Definition 16)} \\
& \equiv & \exists z, z', u \big(x \mapsto (u) \star z \simeq z' \star \mathtt{lseg}(z', x) \star z \simeq y\big) \qquad \text{(Lemma 26)} \\
& \equiv & \exists u \big(x \mapsto (u) \star \mathtt{lseg}(y, x)\big)
\end{array}$$

A similar result holds for look-ups (we recall that actions $y \leftarrow x.i$ with $y = x$ are not allowed in the syntax):

▶ **Lemma 28.** *For all formulas ϕ, for all variables $x, y_1, \ldots, y_n, y$ (with $x \neq y$), for all $i \in \{1, \ldots, \kappa\}$, and for all sequences of variables $\vec{z}$ (not containing x or y), the following equivalence holds:* $[y \leftarrow x.i]\, \exists \vec{z} \big(x \mapsto (y_1, \ldots, y_\kappa) \star \phi)\big) \equiv_{\mathcal{R}} \exists \vec{z} \big(x \mapsto (y_1, \ldots, y_\kappa)) \star \phi\big)\{y \leftarrow y_i\}\big)$

It remains to consider the case of allocation actions $[x \leftarrow \mathtt{cons}(y_1, \ldots, y_\kappa)]\,\phi$. This introduces new challenges, because the action associates the variable x with a fresh, unallocated location ℓ. We need to ensure that the formula ϕ holds for all possible choices of ℓ (provided ℓ is unallocated). The most natural way to express this would be through universal quantification or separating implication. However, this is not feasible within our fragment – these constructs are not supported by existing entailment procedures for separation logic with inductive definitions. To address this, we establish Lemma 30 below. Intuitively, it states that the truth value of a static formula within a structure remains unchanged when some locations are replaced, as long as the replacement preserves variable aliasing and does not affect the heap. This will allow us to consider only a finite number of possible choices for the location ℓ as all locations outside the image of the store behave identically. More formally, we introduce the following equivalence relation on structures:

▶ **Definition 29.** *Let $(\mathfrak{s}_i, \mathfrak{h}_i)$ (for $i \in \{1,2\}$) be two structures and let V be a finite set of variables. We write $(\mathfrak{s}_1, \mathfrak{h}_1) \sim_V (\mathfrak{s}_2, \mathfrak{h}_2)$ if the three following conditions are fulfilled:*

1. $\mathfrak{h}_1 = \mathfrak{h}_2$.
2. *The equivalence $\mathfrak{s}_1(x) = \mathfrak{s}_1(y) \iff \mathfrak{s}_2(x) = \mathfrak{s}_2(y)$ holds for all variables $x, y \in V$.*
3. *For all variables $x \in V$, if $\mathfrak{s}_1(x) \in \mathit{locs}(\mathfrak{h}_1)$ or $\mathfrak{s}_2(x) \in \mathit{locs}(\mathfrak{h}_1)$ then $\mathfrak{s}_1(x) = \mathfrak{s}_2(x)$.*

▶ **Lemma 30.** *Let $(\mathfrak{s}_i, \mathfrak{h}_i)$ (for $i \in \{1,2\}$) be two structures with $(\mathfrak{s}_1, \mathfrak{h}_1) \sim_{\mathsf{fv}(\phi)} (\mathfrak{s}_2, \mathfrak{h}_2)$. Let ϕ be a static formula. If $(\mathfrak{s}_1, \mathfrak{h}_1) \models^n_{\mathcal{R}} \phi$ then $(\mathfrak{s}_2, \mathfrak{h}_2) \models^n_{\mathcal{R}} \phi$.*

Proof. The result is established by induction on $(n, |\phi|)$. By Definition 29 (1) $\mathfrak{h}_1 = \mathfrak{h}_2$.

- If $\phi = (x \simeq y)$ then, as $(\mathfrak{s}_1, \mathfrak{h}_1) \models_{\mathcal{R}} \phi$, we must have $\mathfrak{h}_1 = \emptyset$ and $\mathfrak{s}_1(x) = \mathfrak{s}_1(y)$, thus $\mathfrak{s}_2(x) = \mathfrak{s}_2(y)$ by Definition 29 (2), and $(\mathfrak{s}_2, \mathfrak{h}_1) \models_{\mathcal{R}} \phi$. The proof is similar if $\phi = (x \not\simeq y)$.
- If $\phi = y_0 \mapsto (y_1, \ldots, y_\kappa)$ then $\mathfrak{h}_1 = \{\mathfrak{s}_1(y_0) \mapsto (\mathfrak{s}_1(y_1), \ldots, \mathfrak{s}_1(y_\kappa))\}$ thus $\mathfrak{s}_1(y_i) \in \mathit{locs}(\mathfrak{h}_1)$ for all $i \in \{0, \ldots, \kappa\}$. Consequently $\mathfrak{s}_1(y_i) = \mathfrak{s}_2(y_i)$ by Definition 29 (3) and $\mathfrak{h}_1 = \{\mathfrak{s}_2(y_0) \mapsto (\mathfrak{s}_2(y_1), \ldots, \mathfrak{s}_2(y_\kappa))\}$. Hence $(\mathfrak{s}_2, \mathfrak{h}_1) \models^n_{\mathcal{R}} \phi$.
- If $\phi = \phi_1 \vee \phi_2$ then $(\mathfrak{s}_1, \mathfrak{h}_1) \models^n_{\mathcal{R}} \phi_i$ for some $i \in \{1,2\}$. By the induction hypothesis we get $(\mathfrak{s}_2, \mathfrak{h}_2) \models^n_{\mathcal{R}} \phi_i$ and $(\mathfrak{s}_2, \mathfrak{h}_1) \models^n_{\mathcal{R}} \phi$.
- The proof is similar if $\phi = \phi_1 \wedge \phi_2$.
- If $\phi = \phi_1 \star \phi_2$ then there exist heaps $\mathfrak{h}^1_1, \mathfrak{h}^2_1$ such that $\mathfrak{h}_1 = \mathfrak{h}^1_1 \cup \mathfrak{h}^2_1$ and $(\mathfrak{s}_1, \mathfrak{h}^i_1) \models^n_{\mathcal{R}} \phi_i$ for all $i \in \{1,2\}$. It is straightforward to verify that $(\mathfrak{s}_1, \mathfrak{h}^i_1) \sim_{\mathsf{fv}(\phi)} (\mathfrak{s}_2, \mathfrak{h}^i_1)$, as $\mathit{locs}(\mathfrak{h}^i_1) \subseteq \mathit{locs}(\mathfrak{h}_1)$. By the induction hypothesis, we get $(\mathfrak{s}_2, \mathfrak{h}^i_1) \models^n_{\mathcal{R}} \phi_i$ (for all $i \in \{1,2\}$) thus $(\mathfrak{s}_2, \mathfrak{h}_1) \models^n_{\mathcal{R}} \phi$.
- If $\phi = \exists x\, \psi$ then there exists $\ell_1 \in \mathcal{L}$ such that $(\mathfrak{s}_1[x \leftarrow \ell], \mathfrak{h}_1) \models_{\mathcal{R}} \psi$. We distinguish several cases. In every case we show that there exists a location ℓ_2 such that $(\mathfrak{s}_1[x \leftarrow \ell_1], \mathfrak{h}_1) \sim_{\mathsf{fv}(\psi)} (\mathfrak{s}_2[x \leftarrow \ell_2], \mathfrak{h}_1)$, i.e., such that Conditions 2 and 3 in Definition 29 both hold.
  - If $\ell_1 \in \mathit{locs}(\mathfrak{h}_1)$ then we take $\ell_2 = \ell_1$. It is easy to check that Condition 3 holds, as it trivially holds for x and it also holds for all $y \in \mathsf{fv}(\psi) \setminus \{x\}$ since $\mathfrak{s}_1 \sim_{\mathsf{fv}(\phi)} \mathfrak{s}_2$ by hypothesis and $\mathfrak{s}_i[x \leftarrow \ell_i]$ coincides with $\mathfrak{s}_i$ on all $y \neq x$. Now consider two variables $u, v \in \mathsf{fv}(\psi)$. We show that $\mathfrak{s}_1[x \leftarrow \ell_1](u) = \mathfrak{s}_1[x \leftarrow \ell_1](v) \iff \mathfrak{s}_2[x \leftarrow \ell_2](u) = \mathfrak{s}_2[x \leftarrow \ell_2](v)$. The proof is trivial if $u = v$, thus we assume that $u \neq v$.
    - If u, v are both distinct from x then $\mathfrak{s}_1[x \leftarrow \ell_1](u) = \mathfrak{s}_1[x \leftarrow \ell_1](v) \iff \mathfrak{s}_1(u) = \mathfrak{s}_1(v) \iff \mathfrak{s}_2(u) = \mathfrak{s}_2(v)$ (as $\mathfrak{s}_1 \sim_{\mathsf{fv}(\phi)} \mathfrak{s}_2$ and $u, v \in \mathsf{fv}(\phi)$). Thus the above equivalence holds.
    - If one of the variables, say u is x, and if, for some $i \in \{1,2\}$, $\mathfrak{s}_i[x \leftarrow \ell_i](u) = \mathfrak{s}_i[x \leftarrow \ell_i](v)$, then, as $\ell_1 \in \mathit{locs}(\mathfrak{h}_1)$ and $\ell_2 = \ell_1$, we get $\ell_1 = \mathfrak{s}_i[x \leftarrow \ell_i](v) = \mathfrak{s}_i(v) \in \mathit{locs}(\mathfrak{h}_1)$ and by Condition 3, $\mathfrak{s}_i(v) = \mathfrak{s}_{3-i}(v)$, thus $\mathfrak{s}_{3-i}[x \leftarrow \ell_{3-i}](u) = \mathfrak{s}_{3-i}[x \leftarrow \ell_{3-i}](v)$.

- If $\ell_1 \notin \mathit{locs}(\mathfrak{h}_1)$ and $\ell_1 = \mathfrak{s}_1(y)$ for some variable $y \in \mathsf{fv}(\phi)$ then we take $\ell_2 = \mathfrak{s}_2(y)$. We show that Condition 3 holds. Assume that $\mathfrak{s}_i[x \leftarrow \ell_i](y) \in \mathit{locs}(\mathfrak{h}_1)$ for some $i \in \{1,2\}$. Since $y \in \mathsf{fv}(\phi)$, $y \neq x$, which entails that $\mathfrak{s}_i(y) \in \mathit{locs}(\mathfrak{h}_1)$, and thus $\mathfrak{s}_i(y) = \mathfrak{s}_{3-i}(y)$ by Condition 3. Then we show that Condition 2 holds. Consider two variables $u, v \in \mathsf{fv}(\psi)$ and assume that $\mathfrak{s}_i[x \leftarrow \ell_i](u) = \mathfrak{s}_i[x \leftarrow \ell_i](v)$ for some $i \in \{1,2\}$. We show that $\mathfrak{s}_{3-i}[x \leftarrow \ell_{3-i}](u) = \mathfrak{s}_{3-i}[x \leftarrow \ell_{3-i}](v)$. We may assume, w.l.o.g., that $u \neq v$. If one of the variables, say u, is x, then we get $\mathfrak{s}_i(y) = \ell_i = \mathfrak{s}_i[x \leftarrow \ell_i](v) = \mathfrak{s}_i(v)$, hence $\ell_{3-i} = \mathfrak{s}_{3-i}(y) = \mathfrak{s}_{3-i}(v)$ by Condition 2, as $(\mathfrak{s}_1, \mathfrak{h}_1) \sim_{\mathsf{fv}(\phi)} (\mathfrak{s}_2, \mathfrak{h}_1)$. Thus $\mathfrak{s}_{3-i}[x \leftarrow \ell_{3-i}](u) = \mathfrak{s}_{3-i}[x \leftarrow \ell_{3-i}](v)$. If $x \notin \{u, v\}$ then $\{u, v\} \subseteq \mathsf{fv}(\phi)$ and the proof follows from the fact that $(\mathfrak{s}_1, \mathfrak{h}_1) \sim_{\mathsf{fv}(\phi)} (\mathfrak{s}_2, \mathfrak{h}_1)$.
- Otherwise (i.e., if $\ell_1 \notin \mathfrak{s}_1(\mathsf{fv}(\phi)) \cup \mathit{locs}(\mathfrak{h}_1))$, ℓ_2 is any location not occurring in $\mathfrak{s}_2(\mathsf{fv}(\phi)) \cup \mathit{locs}(\mathfrak{h}_2)$ (such a location exists as $\mathcal{L}$ is infinite). Condition 3 trivially holds for x as $\{\ell_1, \ell_2\} \cap \mathit{locs}(\mathfrak{h}_1) = \emptyset$, and also holds for all variables y distinct from x since $\mathfrak{s}_i[x \leftarrow \ell_i]$ coincides with $\mathfrak{s}_i$ on all $y \neq x$. Now consider two variables $u, v \in \mathsf{fv}(\psi)$ and assume that $\mathfrak{s}_i[x \leftarrow \ell_i](u) = \mathfrak{s}_i[x \leftarrow \ell_i](v)$ for some $i \in \{1,2\}$. We show that $\mathfrak{s}_{3-i}[x \leftarrow \ell_{3-i}](u) = \mathfrak{s}_{3-i}[x \leftarrow \ell_{3-i}](v)$. The proof is immediate if $u = v$, thus we assume that $u \neq v$. This implies that $x \notin \{u, v\}$, as $\ell_i \notin \mathfrak{s}_i(\mathsf{fv}(\phi))$ (due to the conditions above on ℓ_1 and to the choice of ℓ_2), so that $\{u, v\} \subseteq \mathsf{fv}(\phi)$. Then the proof follows from the fact that $(\mathfrak{s}_1, \mathfrak{h}_1) \sim_{\mathsf{fv}(\phi)} (\mathfrak{s}_2, \mathfrak{h}_1)$, as $\mathfrak{s}_i[x \leftarrow \ell_i](y) = \mathfrak{s}_i(y)$ for all $y \neq x$.

  Consequently, we get in all cases $(\mathfrak{s}_2[x \leftarrow \ell_2], \mathfrak{h}_1) \models^n_{\mathcal{R}} \psi$ by the induction hypothesis, which implies that $(\mathfrak{s}_2, \mathfrak{h}_1) \models^n_{\mathcal{R}} \phi$.
- Assume that ϕ is of the form $p(x_1, \ldots, x_n)$. By Definition 5, there exists a formula ψ such that $\phi \rightarrow_{\mathcal{R}} \psi$ and $(\mathfrak{s}_1, \mathfrak{h}_1) \models^{n-1}_{\mathcal{R}} \psi$. By the induction hypothesis, we get $(\mathfrak{s}_2, \mathfrak{h}_2) \models^{n-1}_{\mathcal{R}} \psi$, thus $(\mathfrak{s}_2, \mathfrak{h}_2) \models^{n-1}_{\mathcal{R}} \phi$. ◀

Based on the previous result and on the key proposition 7, we now establish the equivalence that permits the elimination of allocations.

▶ **Lemma 31.** *Assume that $\mathcal{R}$ is PCE. Let $\phi = \exists v_1, \ldots, v_m\, \chi$ be a static formula in prenex form (where χ is quantifier-free) and let $x, x', y_1, \ldots, y_\kappa$ be variables, where $x' \notin \mathsf{fv}(\chi)$. Let $\{z_1, \ldots, z_n\} = (\mathsf{fv}(\chi) \setminus \{x\}) \cup \{x'\}$. The following equivalence holds:*

$$[x \leftarrow \mathtt{cons}(y_1, \ldots, y_\kappa)]\left(x \mapsto (y_1, \ldots, y_\kappa) \star \phi\right) \equiv \exists v_1, \ldots, v_m, x'\left(\psi_1 \wedge \psi_2 \wedge \psi_3\right)$$

where:

- $\psi_1 = \langle \chi\{x \leftarrow x'\}\rangle^{\mathtt{N}(x')}$;
- $\psi_2 = \exists x\left(\langle\phi\rangle^{\mathtt{N}(x)} \star \bigstar_{i=1}^{n} x \not\approx z_i\right)$;
- $\psi_3 = \bigwedge_{i=1}^{n}(\langle\psi_1\rangle^{\mathtt{A}(z_i)} \vee \phi\{x \leftarrow z_i\})$

Intuitively, the formula ψ_1 (which states that $\chi\{x \leftarrow x'\}$ holds and that x' is not allocated) is obtained by applying the allocation $x \leftarrow \mathtt{cons}(y_1, \ldots, y_\kappa)$ using any (arbitrary chosen) unallocated location x'. As the resulting structure satisfies $x \mapsto (y_1, \ldots, y_\kappa) \star \phi$, there must exist $v_1, \ldots, v_m$ such that $\chi\{x \leftarrow x'\}$ holds, and since x' is not allocated, this implies that $\langle\chi\{x \leftarrow x'\}\rangle^{\mathtt{N}(x')}$ also holds. Afterwards, the action is applied again, this time mapping x to another unallocated location distinct from the values of all (free or existential) variables occurring in the previous formula ψ_1, yielding the formula ψ_2. Finally, the formula ψ_3 is obtained by mapping x to each (unallocated) location z_i. The disjunct $\langle\psi_1\rangle^{\mathtt{A}(z_i)}$ is added to exclude allocated locations. To establish the converse, we observe that the locations that are unallocated and distinct from the free variables in ψ_1 cannot occur in the heap (by Proposition 7), and are therefore indiscernible (by Lemma 30). Consequently, the variable x in ψ_2 serves as a single representative for all such locations. More formally:

Proof. Let $(\mathfrak{s},\mathfrak{h})$ be a structure. By Definition 4, $x \leftarrow \mathtt{cons}(y_1,\ldots,y_\kappa)$ never fails on $(\mathfrak{s},\mathfrak{h})$ (as $\mathfrak{h}$ is finite and $\mathcal{L}$ is infinite) and $(\mathfrak{s},\mathfrak{h}) \rightsquigarrow_{x\leftarrow\mathtt{cons}(y_1,\ldots,y_\kappa)} (\mathfrak{s}',\mathfrak{h}')$ iff $\mathfrak{s}' = \mathfrak{s}[x \leftarrow \ell]$ and $\mathfrak{h}' = \mathfrak{h} \cup \{\ell \mapsto (\mathfrak{s}(y_1),\ldots,\mathfrak{s}(y_\kappa))\}$, for some $\ell \notin \mathsf{dom}(\mathfrak{h})$.

- Assume that $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} [x \leftarrow \mathtt{cons}(y_1,\ldots,y_\kappa)]\big(x \mapsto (y_1,\ldots,y_\kappa) \star \phi\big)$. Let ℓ be any location not occurring in $\mathsf{dom}(\mathfrak{h})$. By definition of the semantics, there exist disjoint heaps $\mathfrak{h}_1,\mathfrak{h}_2$ such that $\mathfrak{h} \cup \{\ell \mapsto (\mathfrak{s}(y_1),\ldots,\mathfrak{s}(y_\kappa))\} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, $(\mathfrak{s}[x \leftarrow \ell],\mathfrak{h}_1) \models_{\mathcal{R}} x \mapsto (y_1,\ldots,y_\kappa)$ and $(\mathfrak{s}[x \leftarrow \ell],\mathfrak{h}_2) \models_{\mathcal{R}} \phi$. This implies that $\mathfrak{h}_1 = \{\ell \mapsto (\mathfrak{s}(y_1),\ldots,\mathfrak{s}(y_\kappa))\}$ and $\mathfrak{h}_2 = \mathfrak{h}$ thus $(\mathfrak{s}[x \leftarrow \ell],\mathfrak{h}) \models_{\mathcal{R}} \phi$, and (as $\phi = \exists v_1,\ldots,v_m\,\chi$), there exist locations $\ell_1,\ldots,\ell_m$ such that $(\mathfrak{s}[x \leftarrow \ell, v_i \leftarrow \ell_i \mid 1 \leq i \leq n],\mathfrak{h}) \models_{\mathcal{R}} \chi$, thus $(\mathfrak{s}[x' \leftarrow \ell, v_i \leftarrow \ell_i \mid 1 \leq i \leq n],\mathfrak{h}) \models_{\mathcal{R}} \chi\{x \leftarrow x'\}$. By Lemma 15, since $\ell \notin \mathsf{dom}(\mathfrak{h})$, we get $(\mathfrak{s}[x' \leftarrow \ell, v_i \leftarrow \ell_i \mid 1 \leq i \leq n],\mathfrak{h}) \models_{\mathcal{R}} \langle\chi\{x \leftarrow x'\}\rangle^{\mathtt{N}(x')}$. Hence $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \psi_1$, with $\mathfrak{s}' = \mathfrak{s}[x' \leftarrow \ell, v_i \leftarrow \ell_i \mid 1 \leq i \leq n]$. We now prove that $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \psi_2$. Let ℓ' be any location not occurring in $\{\mathfrak{s}'(z_i) \mid 1 \leq i \leq n\} \cup \mathsf{dom}(\mathfrak{h})$. Using the same reasoning as before, we show (by definition of the semantics of allocations), that $(\mathfrak{s}[x \leftarrow \ell'],\mathfrak{h}) \models_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$ and therefore $(\mathfrak{s}'[x \leftarrow \ell'],\mathfrak{h}) \models_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$ (since the variables $v_1,\ldots,v_m$ are not free in ϕ, thus in $\langle\phi\rangle^{\mathtt{N}(x)}$). As $\ell' \neq \ell_i = \mathfrak{s}'(z_i)$ we get $(\mathfrak{s}'[x \leftarrow \ell'],\mathfrak{h}) \models_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)} \star \bigstar_{i=1}^{n} x \not\simeq z_i$ which implies that $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \exists x\,\big(\langle\phi\rangle^{\mathtt{N}(x)} \star \bigstar_{i=1}^{n} x \not\simeq z_i\big) = \psi_2$. Finally, we prove that $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \psi_3$. Consider any $i \in \{1,\ldots,n\}$. If $\mathfrak{s}'(z_i) \in \mathsf{dom}(\mathfrak{h})$ then by Lemma 19, $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \langle\psi_1\rangle^{\mathtt{A}(z_i)}$ (since $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \psi_1$). Otherwise, by definition of the semantics, $(\mathfrak{s}'[x \leftarrow \mathfrak{s}'(z_i)],\mathfrak{h}) \models_{\mathcal{R}} \phi$, and therefore $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \phi\{x \leftarrow z_i\}$. Therefore, we have $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \psi_1 \wedge \psi_2 \wedge \psi_3$ which implies that $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \exists v_1,\ldots,v_m\,\big(\psi_1 \wedge \psi_2 \wedge \psi_3\big)$.
- Assume that $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \exists v_1,\ldots,v_m\,\big(\psi_1 \wedge \psi_2 \wedge \psi_3\big)$. Let ℓ be any location not occurring in $\mathsf{dom}(\mathfrak{h})$, we need to show that $(\mathfrak{s}[x \leftarrow \ell],\mathfrak{h} \cup \{\ell \mapsto (\mathfrak{s}(y_1),\ldots,\mathfrak{s}(y_\kappa))\}) \models_{\mathcal{R}} x \mapsto (y_1,\ldots,y_\kappa) \star \phi$. Since $(\mathfrak{s}[x \leftarrow \ell],\{\ell \mapsto (\mathfrak{s}(y_1),\ldots,\mathfrak{s}(y_\kappa))\}) \models_{\mathcal{R}} x \mapsto (y_1,\ldots,y_\kappa)$ (as x does not occur in $y_1,\ldots,y_\kappa$, by Definition 1), we only have to prove that $(\mathfrak{s}[x \leftarrow \ell],\mathfrak{h}) \models_{\mathcal{R}} \phi$. By definition, there exists a store $\mathfrak{s}'$, coinciding with $\mathfrak{s}$ on all variables distinct from $v_1,\ldots,v_m,x'$, such that $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \psi_1 \wedge \psi_2 \wedge \psi_3$. We distinguish two cases.
  - Assume that $\ell = \mathfrak{s}'(z_i)$, for some $i \in \{1,\ldots,n\}$. Then, by Lemma 19, we cannot have $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \langle\psi_1\rangle^{\mathtt{A}(x)}$ (otherwise ℓ would occur in $\mathsf{dom}(\mathfrak{h})$), thus we must have $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \phi\{x \leftarrow z_i\}$, hence $(\mathfrak{s}'[x \leftarrow \ell],\mathfrak{h}) \models_{\mathcal{R}} \phi$. Since $v_1,\ldots,v_m,x'$ do not freely occur in ϕ, this implies that $(\mathfrak{s}[x \leftarrow \ell],\mathfrak{h}) \models_{\mathcal{R}} \phi$.
  - Assume that $\ell \neq \mathfrak{s}'(z_i)$, for all $i \in \{1,\ldots,n\}$. In this case, as $(\mathfrak{s}',\mathfrak{h}) \models_{\mathcal{R}} \psi_2$, there exists ℓ' such that $(\mathfrak{s}'[x \leftarrow \ell'],\mathfrak{h}) \models_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$ and $\ell' \neq \mathfrak{s}(z_i)$, for all $i \in \{1,\ldots,m\}$. By Proposition 7, since ψ_1 is quantifier-free, $\mathit{locs}(\mathfrak{h}) \subseteq \mathsf{dom}(\mathfrak{h}) \cup \{\mathfrak{s}'(u) \mid u \in \mathsf{fv}(\psi_1)\} \subseteq \mathsf{dom}(\mathfrak{h}) \cup \{\mathfrak{s}'(z_i) \mid 1 \leq i \leq n\}$. Since $\ell,\ell' \notin \mathsf{dom}(\mathfrak{h}) \cup \{\mathfrak{s}'(z_i) \mid 1 \leq i \leq n\}$, we deduce that $\ell,\ell' \notin \mathit{locs}(\mathfrak{h})$, which implies that $(\mathfrak{s}'[x \leftarrow \ell],\mathfrak{h}) \sim_{\{z_1,\ldots,z_n,x\}} (\mathfrak{s}'[x \leftarrow \ell'],\mathfrak{h})$. By Lemma 30, this entails that $(\mathfrak{s}'[x \leftarrow \ell],\mathfrak{h}) \models_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$, thus (by Lemma 18) $(\mathfrak{s}[x \leftarrow \ell],\mathfrak{h}) \models_{\mathcal{R}} \phi$ (since $v_1,\ldots,v_m,x'$ are not free in ϕ). ◀

▶ **Lemma 32.** *There exists an algorithm transforming every formula (with a PCE set of rules) of the form* $[\mathtt{a}]\,\gamma$*, where* γ *is static, into an equivalent static formula* δ*.*

Proof. If $\mathtt{a}$ is of the form $x \leftarrow y$, then by Proposition 21 it suffices to take $\delta = \phi\{x \leftarrow y\}$. If γ is of the form $\exists\vec{z}\,(x \mapsto (y_1,\ldots,y_\kappa) \star \phi)$ and $\mathtt{a}$ is of the form $x.i \leftarrow y$ or $y \leftarrow i.x$ then by Lemmata 26 or 28 it suffices to take $\delta = \exists\vec{z}\,(\exists u\,(x \mapsto (y_1,\ldots,y_{i-1},u,y_{i+1},y_\kappa)) \star \phi \star y_i \simeq y)$ or $\delta = \exists\vec{z}\,((x \mapsto (y_1,\ldots,y_\kappa)) \star \phi')\{y \leftarrow y_i\})$ respectively. If $\mathtt{a}$ is of the form $x.i \leftarrow y$ or $y \leftarrow i.x$ but γ is not of the form $\exists\vec{z}\,(x \mapsto (y_1,\ldots,y_\kappa) \star \phi)$, then it suffices to apply Lemma 24, to transform γ into a formula of the latter form and get back to the previous case. If γ is of the form $\exists\vec{z}\,(x \mapsto (y_1,\ldots,y_\kappa) \star \phi)$ and $\mathtt{a}$ is of the form $x \leftarrow \mathtt{cons}(y_1,\ldots,_\kappa)$ then we take

the formula $\delta = \exists v_1, \ldots, v_m, x' \left(\psi_1 \wedge \psi_2 \wedge \psi_3\right)$, as defined in Lemma 31. If $\mathtt{a}$ is of the form $x \leftarrow \mathtt{cons}(y_1, \ldots, y_\kappa)$ but γ is not of the form $\exists \vec{z}(x \mapsto (y_1, \ldots, y_\kappa) \star \phi)$, then it suffices to apply Lemma 25 to get an equivalent formula of the previous form and to get back to the previous case. ◀

We derive the following theorem, which constitutes the main result of the paper:

▶ **Theorem 33.** *There exists an algorithm transforming every formula (with a PCE set of rules) into an equivalent static formula.*

Proof. It suffices to apply Lemma 32 recursively, on every subformula of the form $[\mathtt{a}]\,\phi$ occurring innermost in the fomula (so that ϕ is static), until no such subformula exists (exactly one dynamic modality is eliminated at each application of the lemma). ◀

In particular, the entailment problem for DSL formulas (with PCE rules) can thus be reduced to an entailment problem in SL, which is decidable [12].

6 Future Work

We devised an algorithm for eliminating dynamic modalities from Dynamic Separation Logic formulas satisfying the PCE conditions in [12], while preserving semantic equivalence. The reduction exploits key properties of the structures generated by PCE rules. In particular, when addressing allocation actions, the fact that the structures contain a bounded number of unallocated referenced locations is essential. The result implies that entailment remains decidable for DSL PCE formulas, thereby extending the applicability of SL-based reasoning to dynamic program behaviors while retaining automation benefits.

Future work includes implementing the proposed transformation and integrating it into existing verification frameworks. Since entailment testing is doubly exponential in the worst case [7], an important research direction is to reduce the computational overhead introduced by the transformation. To this end, one could refine the transformation process to minimize the size of the resulting formulas and the number of additional inductive rules. It could also be interesting to identify subclasses of formulas and inductive definitions for which the transformation is tractable, thereby enhancing the practical usability of the presented reduction.

Another possibility would be to enrich the base SL fragment to make the transformation more efficient, by introducing new constructs that express the necessary constraints more concisely and efficiently. For example, atomic constraints stating that a variable is unallocated (or does not occur in the heap) could be added to the logic, using formulas with negations, such as: $\forall y_1, \ldots, y_\kappa \neg\big(x \mapsto (y_1, \ldots, y_\kappa)\big)$ or $\forall y_0\, y_1, \ldots, y_\kappa \left(y_0 \mapsto (y_1, \ldots, y_\kappa) \Rightarrow \bigwedge_{i=0}^{\kappa}(x \not\approx y_i)\right)$. This would eliminate the need to introduce additional inductive rules to enforce these properties. Naturally, such negations must be carefully restricted to preserve the decidability of entailment, and the entailment procedure must be extended to deal with such constructs.

Extending the proposed transformation to handle rules that do not satisfy the PCE conditions could also be explored. It should be noted that inductive rules do not involve dynamic modalities. Allowing recursion in the actions would be interesting, as it would enable the expression of unbounded sequences of transformations.

References

1 J. Berdine, C. Calcagno, and P. W. O'Hearn. A decidable fragment of separation logic. In *Proc. of FSTTCS'04*, volume 3328 of *LNCS*. Springer, 2004. doi:10.1007/978-3-540-30538-5_9.

2 James Brotherston, Carsten Fuhs, Juan Antonio Navarro Pérez, and Nikos Gorogiannis. A decision procedure for satisfiability in separation logic with inductive predicates. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 25:1–25:10. ACM, 2014. doi:10.1145/2603088.2603091.

3 Cristiano Calcagno, Hongseok Yang, and Peter W O'hearn. Computability and complexity results for a spatial assertion language for data structures. In *FST TCS 2001, Proceedings*, pages 108–119. Springer, 2001. doi:10.1007/3-540-45294-X_10.

4 B. Cook, C. Haase, J. Ouaknine, M. J. Parkinson, and J. Worrell. Tractable reasoning in a fragment of separation logic. In *Proc. of CONCUR'11*, volume 6901 of *LNCS*. Springer, 2011. doi:10.1007/978-3-642-23217-6_16.

5 Frank S. de Boer, Hans-Dieter A. Hiep, and Stijn de Gouw. Dynamic separation logic. In Marie Kerjean and Paul Blain Levy, editors, *Proceedings of the 39th Conference on the Mathematical Foundations of Programming Semantics, MFPS XXXIX, Indiana University, Bloomington, IN, USA, June 21-23, 2023*, volume 3 of *EPTICS*. EpiSciences, 2023. doi:10.46298/ENTICS.12297.

6 Stéphane Demri, Didier Galmiche, Dominique Larchey-Wendling, and Daniel Méry. Separation logic with one quantified variable. In *CSR'14*, volume 8476 of *LNCS*, pages 125–138. Springer, 2014. doi:10.1007/978-3-319-06686-8_10.

7 Mnacho Echenim, Radu Iosif, and Nicolas Peltier. Entailment checking in separation logic with inductive definitions is 2-exptime hard. In Elvira Albert and Laura Kovács, editors, *LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Alicante, Spain, May 22-27, 2020*, volume 73 of *EPiC Series in Computing*, pages 191–211. EasyChair, 2020. doi:10.29007/F5WH.

8 Mnacho Echenim, Radu Iosif, and Nicolas Peltier. Entailment is undecidable for symbolic heap separation logic formulæ with non-established inductive rules. *Inf. Process. Lett.*, 173:106169, 2022. doi:10.1016/j.ipl.2021.106169.

9 Mnacho Echenim and Nicolas Peltier. A proof procedure for separation logic with inductive definitions and data. *J. Autom. Reason.*, 67(3):30, 2023. doi:10.1007/S10817-023-09680-4.

10 Constantin Enea, Ondrej Lengál, Mihaela Sighireanu, and Tomás Vojnar. Compositional entailment checking for a fragment of separation logic. *Formal Methods Syst. Des.*, 51(3):575–607, 2017. doi:10.1007/S10703-017-0289-4.

11 Constantin Enea, Mihaela Sighireanu, and Zhilin Wu. On automated lemma generation for separation logic with inductive definitions. In *ATVA 2015, Proceedings*, pages 80–96, 2015. doi:10.1007/978-3-319-24953-7_7.

12 Radu Iosif, Adam Rogalewicz, and Jiri Simacek. The tree width of separation logic with recursive definitions. In *Proc. of CADE-24*, volume 7898 of *LNCS*, 2013.

13 Radu Iosif, Adam Rogalewicz, and Tomás Vojnar. Deciding entailments in inductive separation logic with tree automata. In Franck Cassez and Jean-François Raskin, editors, *ATVA 2014, Proceedings*, volume 8837 of *LNCS*, pages 201–218. Springer, 2014. doi:10.1007/978-3-319-11936-6_15.

14 Samin S Ishtiaq and Peter W O'Hearn. Bi as an assertion language for mutable data structures. In *ACM SIGPLAN Notices*, volume 36, pages 14–26, 2001. doi:10.1145/360204.375719.

15 Christoph Matheja, Jens Pagel, and Florian 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. doi:10.1145/3534927.

16 Peter W. O'Hearn, John C. Reynolds, and Hongseok Yang. Local reasoning about programs that alter data structures. In Laurent Fribourg, editor, *Computer Science Logic, 15th International Workshop, CSL 2001. 10th Annual Conference of the EACSL, Paris, France, September 10-13, 2001, Proceedings*, volume 2142 of *LNCS*, pages 1–19. Springer, 2001. doi:10.1007/3-540-44802-0_1.

17 John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *17th IEEE Symposium on Logic in Computer Science (LICS 2002), 22-25 July 2002, Copenhagen, Denmark, Proceedings*, pages 55–74. IEEE Computer Society, 2002. doi:10.1109/LICS.2002.1029817.

18 Makoto Tatsuta, Koji Nakazawa, and Daisuke Kimura. Completeness of cyclic proofs for symbolic heaps with inductive definitions. In Anthony Widjaja Lin, editor, *Programming Languages and Systems - 17th Asian Symposium, APLAS 2019, Nusa Dua, Bali, Indonesia, December 1-4, 2019, Proceedings*, volume 11893 of *LNCS*, pages 367–387. Springer, 2019. doi:10.1007/978-3-030-34175-6_19.

A Proof of Lemma 15

We use the following proposition, which is immediate to prove:

▶ **Proposition 34.** *For every structure* $(\mathfrak{s},\mathfrak{h})$*, static formula* ϕ*, natural number* n*, and substitution* σ*:* $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi\sigma \iff (\mathfrak{s}',\mathfrak{h}) \models^n_{\mathcal{R}} \phi$*, where* $\mathfrak{s}'(x) = \mathfrak{s}(\sigma(x))$*, for all variables* x*.*

We show that $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)} \iff (\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$ by induction on the pair $(n,|\phi|)$, ordered by lexicographic extension of the usual order on natural numbers (the result follows immediately).

- If ϕ is of the form $y \bowtie z$ (with $\bowtie \in \{\simeq,\not\simeq\}$) then $\langle\phi\rangle^{\mathtt{N}(x)} = \phi$ and $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi$ only if $\mathfrak{h} = \emptyset$, thus the proof is immediate.
- If ϕ is of the form $x' \mapsto (y_1,\ldots,y_\kappa)$ then $\langle\phi\rangle^{\mathtt{N}(x)} = \phi \star (x \not\simeq x')$. Moreover, $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi$ only if $\mathsf{dom}(\mathfrak{h}) = \{\mathfrak{s}(x')\}$, hence the equivalence $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h}) \iff \mathfrak{s}(x) = \mathfrak{s}(x')$ holds for all $\mathcal{R}$-models $(\mathfrak{s},\mathfrak{h})$ of ϕ. Consequently, $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi \star (x \not\simeq x') \iff (\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$.
- If ϕ is of the form $\phi_1 \wedge \phi_2$ then $\langle\phi\rangle^{\mathtt{N}(x)} = \langle\phi_1\rangle^{\mathtt{N}(x)} \wedge \phi_2$. By Definition 5, $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$ iff $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi_1\rangle^{\mathtt{N}(x)}$ and $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi_2$. By the induction hypothesis, $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi_1\rangle^{\mathtt{N}(x)} \iff (\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi_1 \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$, hence $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$ is equivalent to: $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi_i$ (for all $i \in \{1,2\}$) and $\mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$. Thus $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)} \iff (\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$.
- If ϕ is of the form $\phi_1 \vee \phi_2$ then $\langle\phi\rangle^{\mathtt{N}(x)} = \langle\phi_1\rangle^{\mathtt{N}(x)} \vee \langle\phi_2\rangle^{\mathtt{N}(x)}$. By definition, $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$ iff $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi_i\rangle^{\mathtt{N}(x)}$ (for some $i \in \{1,2\}$). By the induction hypothesis, $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi_i\rangle^{\mathtt{N}(x)} \iff (\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi_i \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$. Therefore the assertion $\exists i \in \{1,2\}$ s.t. $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi_i\rangle^{\mathtt{N}(x)}$ holds iff $\exists i \in \{1,2\}$ s.t. $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi_i$ and $\mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$. Thus $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$ iff $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$.
- If ϕ is of the form $\phi_1 \star \phi_2$ then $\langle\phi\rangle^{\mathtt{N}(x)} = \langle\phi_1\rangle^{\mathtt{N}(x)} \star \langle\phi_2\rangle^{\mathtt{N}(x)}$. By Definition 5, $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$ iff there exist disjoint heaps $\mathfrak{h}_1,\mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$ and $(\mathfrak{s},\mathfrak{h}_i) \models^n_{\mathcal{R}} \langle\phi_i\rangle^{\mathtt{N}(x)}$ (for all $i \in \{1,2\}$). By the induction hypothesis, $(\mathfrak{s},\mathfrak{h}_i) \models^n_{\mathcal{R}} \langle\phi_i\rangle^{\mathtt{N}(x)} \iff (\mathfrak{s},\mathfrak{h}_i) \models^n_{\mathcal{R}} \phi_i \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h}_i)$, so that $(\mathfrak{s},\mathfrak{h}_i) \models^n_{\mathcal{R}} \langle\phi_i\rangle^{\mathtt{N}(x)}$ (for all $i \in \{1,2\}$) iff $(\mathfrak{s},\mathfrak{h}_i) \models^n_{\mathcal{R}} \phi_i$ (for all $i \in \{1,2\}$) and $\mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h}_1) \cup \mathsf{dom}(\mathfrak{h}_2)$. Thus $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$ iff there exist disjoint heaps $\mathfrak{h}_1,\mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$ and $(\mathfrak{s},\mathfrak{h}_i) \models^n_{\mathcal{R}} \phi_i$ (for all $i \in \{1,2\}$) and $x \notin \mathsf{dom}(\mathfrak{h})$, i.e., $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)} \iff (\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$.
- If $\phi = \exists y\,\psi$ (with $x \neq y$), then $\langle\phi\rangle^{\mathtt{N}(x)} = \exists y\,\langle\phi\rangle^{\mathtt{N}(x)}$. By Definition 5, $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$ iff there exists $\ell \in \mathcal{L}$ such that $(\mathfrak{s}[y \leftarrow \ell],\mathfrak{h}) \models^n_{\mathcal{R}} \langle\psi\rangle^{\mathtt{N}(x)}$. By the induction hypothesis $(\mathfrak{s}[y \leftarrow \ell],\mathfrak{h}) \models^n_{\mathcal{R}} \langle\psi\rangle^{\mathtt{N}(x)}$ holds iff $(\mathfrak{s}[y \leftarrow \ell],\mathfrak{h}) \models^n_{\mathcal{R}} \psi \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$. Thus $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)} \iff (\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \phi \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$.
- Now assume that $\phi = p(x_1,\ldots,x_n)$. Then $\langle\phi\rangle^{\mathtt{N}(x)} = p'(x_1,\ldots,x_n,x)$. By definition of the rules of $\langle\phi\rangle^{\mathtt{N}(x)}$, $(\mathfrak{s},\mathfrak{h}) \models^n_{\mathcal{R}} \langle\phi\rangle^{\mathtt{N}(x)}$ iff there exists a rule $p(y_1,\ldots,y_n) \Leftarrow \psi$ such that $(\mathfrak{s},\mathfrak{h}) \models^{n-1}_{\mathcal{R}} \langle\psi\rangle^{\mathtt{N}(y)}\sigma$, with $\sigma = \{y \leftarrow x, y_i \leftarrow x_i \mid i \in \{1,\ldots,n\}\}$. By Proposition 34, this is equivalent to $(\mathfrak{s}',\mathfrak{h}) \models^{n-1}_{\mathcal{R}} \langle\psi\rangle^{\mathtt{N}(y)}$, with $\mathfrak{s}'(y) = \mathfrak{s}(x)$ and $\mathfrak{s}'(y_i) = \mathfrak{s}(x_i)$ for all

$i \in \{1, \ldots, n\}$. By the induction hypothesis, $(\mathfrak{s}', \mathfrak{h}) \models_{\mathcal{R}}^{n-1} \langle \psi \rangle^{\mathtt{N}(y)}$ iff $(\mathfrak{s}', \mathfrak{h}) \models_{\mathcal{R}}^{n-1} \psi$ and $\mathfrak{s}'(y) \notin \mathsf{dom}(\mathfrak{h})$, i.e., iff $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{n-1} \psi\sigma$ and $\mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$. Thus we obtain the equivalence $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{n} \langle \phi \rangle^{\mathtt{N}(x)} \iff (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{n} \phi \wedge \mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$.

B Proof of Lemma 18

Let $\phi' = \langle \phi \rangle|_{x \mapsto (y_1, \ldots, y_\kappa)}$. We establish the equivalence $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \iff (\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \phi'$ by induction on $|\phi|$.

- If ϕ is of the form $y \bowtie z$ (with $\bowtie \in \{\simeq, \not\simeq\}$) then $\phi' = \bot$ and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathcal{R}} \phi$ (as $\mathfrak{h}$ is not empty while ϕ is satisfied only by structures with an empty heap by Definition 5). Thus $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \iff (\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \phi'$.
- If ϕ is of the form $x' \mapsto (y'_1, \ldots, y'_\kappa)$ then $\phi' = x \simeq x' \star \bigstar_{i=1}^{\kappa} (y_i \simeq y'_i)$. By Definition 5, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ iff $\mathfrak{h} = \{\mathfrak{s}(x') \mapsto (\mathfrak{s}(y'_1), \ldots, \mathfrak{s}(y'_\kappa))\}$, i.e., if $\mathfrak{h}' = \emptyset$, $\mathfrak{s}(x) = \mathfrak{s}(x')$ and $\mathfrak{s}(y_i) = \mathfrak{s}(y'_i)$ (for all $i \in \{1, \ldots, \kappa\}$), since $\mathfrak{h} = \mathfrak{h}' \cup \{\mathfrak{s}(x) \mapsto (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_\kappa))\}$ by the hypothesis of the lemma. Thus $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \iff (\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \phi'$.
- If ϕ is of the form $\phi_1 \wedge \phi_2$ then $\phi' = \phi'_1 \wedge \phi'_2$ with $\phi'_i = \langle \phi_i \rangle|_{x \mapsto (y_1, \ldots, y_\kappa)}$ (for all $i \in \{1, 2\}$). By Definition 5, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ iff $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi_i$ for all $i \in \{1, 2\}$. By the induction hypothesis, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi_i \iff (\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \phi'_i$, so that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ iff $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \phi'_1 \wedge \phi'_2 = \phi'$.
- The proof is similar if $\phi = \phi_1 \vee \phi_2$.
- If ϕ is of the form $\phi_1 \star \phi_2$ then $\phi' = (\phi'_1 \star \phi_2) \vee (\phi_1 \star \phi'_2)$ with $\phi'_i = \langle \phi_i \rangle|_{x \mapsto (y_1, \ldots, y_\kappa)}$ (for all $i \in \{1, 2\}$). By Definition 5, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ iff there exist disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, and $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathcal{R}} \phi_i$ for all $i \in \{1, 2\}$. By definition, as $\mathfrak{h} = \mathfrak{h}' \cup \{\mathfrak{s}(x) \mapsto (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_\kappa))\}$ for all such $\mathfrak{h}_1, \mathfrak{h}_2$, there is $i \in \{1, 2\}$ such that $\mathfrak{h}_i = \mathfrak{h}'_i \cup \{\mathfrak{s}(x) \mapsto (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_\kappa))\}$ and $\mathfrak{h}' = \mathfrak{h}'_i \cup \mathfrak{h}_{3-i}$. By the induction hypothesis, $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathcal{R}} \phi_i \iff (\mathfrak{s}, \mathfrak{h}'_i) \models_{\mathcal{R}} \phi'_i$, so that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ iff there exist $\mathfrak{h}_1, \mathfrak{h}_2$ and $i \in \{1, 2\}$ such that $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, $(\mathfrak{s}, \mathfrak{h}_{3-i}) \models_{\mathcal{R}} \phi_{3-i}$ and $(\mathfrak{s}, \mathfrak{h}'_i) \models_{\mathcal{R}} \phi'_i$ (with $\mathfrak{h}_i = \mathfrak{h}'_i \cup \{\mathfrak{s}(x) \mapsto (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_\kappa))\}$ and $\mathfrak{h}' = \mathfrak{h}'_i \cup \mathfrak{h}_{3-i}$). By Definition 5, this is equivalent to $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} (\phi'_1 \star \phi_2) \vee (\phi_1 \star \phi'_2)$.
- if $\phi = \exists z\, \psi$ then $\phi' = \exists z\, \psi'$, with $\psi' = \langle \psi \rangle|_{x \mapsto (y_1, \ldots, y_\kappa)}$. By Definition 5, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ iff there exists $\ell \in \mathcal{L}$ such that $(\mathfrak{s}[z \leftarrow \ell], \mathfrak{h}) \models_{\mathcal{R}} \psi$. By the induction hypothesis, this is equivalent to: $\exists \ell \in \mathcal{L}$ s.t. $(\mathfrak{s}[z \leftarrow \ell], \mathfrak{h}') \models_{\mathcal{R}} \psi'$, i.e., to $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi'$.
- Assume that $\phi = p(z_1, \ldots, z_n)$ with $z_1 = x$. Let $\{\rho_1, \ldots, \rho_m\}$ be the set of formulas ρ such that $\phi \rightarrow_{\mathcal{R}} \rho$ (this set is finite, up to α-renaming, as there are finitely many rules associated with p). As $\mathcal{R}$ is progressing, each formula ρ_i (for $i \in \{1, \ldots, m\}$) contains exactly one $\mapsto$-atom and the left-hand side of this atom must be z_1 (i.e. x). Thus ρ_i is of the form $\exists \vec{u}_i\, (z_1 \mapsto (y^i_1, \ldots, y^i_\kappa) \star \psi_i)$. By Definition 5, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ iff $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \rho_i$ for some $i \in \{1, \ldots, m\}$, i.e., iff there exist a natural number $i \in \{1, \ldots, m\}$, locations $\vec{\ell}$, and heaps $\mathfrak{h}^1_i, \mathfrak{h}^2_i$ such that $\mathfrak{h} = \mathfrak{h}^1_i \cup \mathfrak{h}^2_i$, $(\mathfrak{s}[\vec{z} \leftarrow \vec{\ell}], \mathfrak{h}^1_i) \models_{\mathcal{R}} z_1 \mapsto (y^i_1, \ldots, y^i_\kappa)$ and $(\mathfrak{s}[\vec{z} \leftarrow \vec{\ell}], \mathfrak{h}^2_i) \models_{\mathcal{R}} \psi_i$. The conditions above hold only if $\mathsf{dom}(\mathfrak{h}^1_i) = \{\mathfrak{s}(z_1)\} = \{\mathfrak{s}(x)\}$, thus only if we have $\mathfrak{h}^1_i = \{\mathfrak{s}(x) \mapsto (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_\kappa))\}$, $\mathfrak{s}(y^i_j) = \mathfrak{s}(y_j)$ for all $j \in \{1, \ldots, \kappa\}$) and $\mathfrak{h}^2_i = \mathfrak{h}'$. Thus $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ iff there exist $i \in \{1, \ldots, m\}$ and $\vec{\ell}$, such that $(\mathfrak{s}[\vec{z} \leftarrow \vec{\ell}], \emptyset) \models_{\mathcal{R}} \bigstar_{j=1}^{\kappa} (y_j \simeq y^i_j)$ and $(\mathfrak{s}[\vec{z} \leftarrow \vec{\ell}], \mathfrak{h}') \models_{\mathcal{R}} \psi_i$. By Definition 5, this is equivalent to $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \bigvee_{i=1}^{m} \exists \vec{u}_i\, (\psi_i \star \bigstar_{j=1}^{\kappa} (y_j \simeq y^i_j)) = \phi'$.
- Finally, assume that $\phi = p(z_1, \ldots, z_n)$ and $z_1 \neq x$. We have $\phi \equiv_{\mathcal{R}} (z_1 \simeq x \star \phi) \vee (z_1 \not\simeq x \star \phi)$. It is clear that $z_1 \simeq x \star \phi$ is equivalent to $z_1 \simeq x \star p(x, z_2, \ldots, z_n)$ and by Corollary 12, $z_1 \not\simeq x \star \phi$ has the same truth value in $(\mathfrak{s}, \mathfrak{h})$ as $z_1 \not\simeq x \star \bigvee_{p \succ^+_{\mathcal{R}} q, \#(q)=m} \exists u_2, \ldots, u_m, \big(q(x, u_2, \ldots, u_m) \star (q \mathrel{-\!\!\bullet} p)(z_1, \ldots, z_n, x, u_2, \ldots, u_m)\big)$. We obtain:

$$\begin{array}{rcl}\langle z_1 \simeq x \star \phi\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)} & = & \langle z_1 \simeq x\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)} \star \phi \\ & & \vee z_1 \simeq x \star \langle\phi\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)} \\ & = & (\bot \star \phi) \vee (z_1 \simeq x \star \langle\phi\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}) \\ & \equiv_{\mathcal{R}} & z_1 \simeq x \star \langle\phi\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}\end{array}$$

Consider the formula $\gamma = z_1 \not\simeq x \star \bigvee_{p \succ^+_{\mathcal{R}} q, \#(q)=m} \exists \vec{u}_m\, (\gamma_{q,m} \star \lambda_{q,m})$ with $\vec{u}_m = (u_2, \ldots, u_m)$, $\gamma_{q,m} = q(x, u_2, \ldots, u_m)$ and $\lambda_{q,m}$ is the formula $(q \multimap p)(z_1, \ldots, z_n, x, u_2, \ldots, u_m)$. In a similar way as for the previous formula, we can show that $\langle\gamma\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}$ is equivalent to the formula: $z_1 \not\simeq x \star \bigvee_{p \succ^+_{\mathcal{R}} q, \#(q)=m} \exists \vec{u}_m \left((\gamma'_{q,m} \star \lambda_{q,m}) \vee (\gamma_{q,m} \star \lambda'_{q,m})\right)$, with $\gamma'_{q,m} = \langle\gamma_{q,m}\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}$ and $\lambda'_{q,m} = \langle\lambda_{q,m}\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}$. Furthermore, as $\mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h}')$, it is clear that $(\mathfrak{s}, \mathfrak{h}') \not\models_{\mathcal{R}} (\gamma_{q,m} \star \lambda'_{q,m})$, since (as $\mathcal{R}$ is PCE) $\gamma_{q,m}$ unfolds to a formula containing a $\mapsto$-atom of the form $x \mapsto (\ldots)$. Consequently, $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \langle\gamma\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}$ iff $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} z_1 \not\simeq x \star \bigvee_{p \succ^+_{\mathcal{R}} q, \#(q)=m} \exists \vec{u}_m\, (\gamma'_{q,m} \star \lambda_{q,m})$. Using the equivalences established in the previous items, we obtain that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ iff $(\mathfrak{s}, \mathfrak{h}')$ is an $\mathcal{R}$-model of $\big(z_1 \simeq x \star \langle\phi\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)}\big) \vee \big(z_1 \not\simeq x \star \bigvee_{p \succ^+_{\mathcal{R}} q, \#(q)=m} \exists \vec{u}_m\, (\gamma'_{q,m} \star \lambda_{q,m})\big)$, which completes the proof.

C Proof of Lemma 19

By definition, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \wedge \mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$ iff there exist locations $\ell_1, \ldots, \ell_\kappa$ such that $\mathfrak{h} = \mathfrak{h}' \cup \{\mathfrak{s}(x) \mapsto (\ell_1, \ldots, \ell_k)\}$ and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$. Let $z_1, \ldots, z_\kappa$ be fresh variables and let $\hat{\mathfrak{s}} = \mathfrak{s}[z_i \leftarrow \ell_i \mid 1 \leq i \leq \kappa]$. By definition, $(\hat{\mathfrak{s}}, \{\mathfrak{s}(x) \mapsto (\ell_1, \ldots, \ell_k)\}) \models_{\mathcal{R}} x \mapsto (z_1, \ldots, z_\kappa)$. By Lemma 18, we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \iff (\hat{\mathfrak{s}}, \mathfrak{h}') \models_{\mathcal{R}} \langle\phi\rangle|_{x\mapsto(z_1,\ldots,z_\kappa)}$, if $\mathfrak{h} = \mathfrak{h}' \cup \{\mathfrak{s}(x) \mapsto (\ell_1, \ldots, \ell_k)\}$. Consequently, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \iff (\hat{\mathfrak{s}}, \mathfrak{h}) \models_{\mathcal{R}} \exists z_1, \ldots, z_\kappa\ (x \mapsto (z_1, \ldots, z_\kappa) \star \langle\phi\rangle|_{x\mapsto(z_1,\ldots,z_\kappa)}) \iff (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \langle\phi\rangle^{\mathtt{A}(x)}$.

D Proof of Proposition 21

Let $(\mathfrak{s}, \mathfrak{h})$ be a structure. By Definition 4, $(\mathfrak{s}, \mathfrak{h}) \rightsquigarrow_{x \leftarrow y} (\mathfrak{s}', \mathfrak{h}') \iff \mathfrak{s}' = \mathfrak{s}[x \leftarrow y] \wedge \mathfrak{h}' = \mathfrak{h}$, thus, by Definition 5, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} [x \leftarrow y]\, \phi \iff (\mathfrak{s}[x \leftarrow y], \mathfrak{h}) \models_{\mathcal{R}} \phi$. Consequently $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} [x \leftarrow y]\, \phi \iff (\mathfrak{s}[x \leftarrow y], \mathfrak{h}) \models_{\mathcal{R}} \phi\{x \leftarrow y\}$ (as $\mathfrak{s}[x \leftarrow y](x) = \mathfrak{s}[x \leftarrow y](y)$), and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} [x \leftarrow y]\, \phi \iff (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi\{x \leftarrow y\}$ (since x does not occur in $\phi\{x \leftarrow y\}$ while $\mathfrak{s}[x \leftarrow y]$ and $\mathfrak{s}$ coincide on all variables other than x).

E Proof of Proposition 22

Let $(\mathfrak{s}, \mathfrak{h})$ be a structure. By Definition 4, `dispose(x)` fails on $(\mathfrak{s}, \mathfrak{h})$ if $\mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$ and $(\mathfrak{s}, \mathfrak{h}) \rightsquigarrow_{\texttt{dispose(x)}} (\mathfrak{s}', \mathfrak{h}')$ iff $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$, $\mathfrak{s}' = \mathfrak{s}$ and $\mathfrak{h}'$ is the restriction of $\mathfrak{h}$ of domain $\mathsf{dom}(\mathfrak{h}) \setminus \{\mathfrak{s}(x)\}$.

- Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} [\texttt{dispose(x)}]\, \phi$. By Definition 5 we deduce that `dispose(x)` does not fail on $(\mathfrak{s}, \mathfrak{h})$ (that i.e., $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$) and that $(\mathfrak{s}, \mathfrak{h}|_{\mathsf{dom}(\mathfrak{h})\setminus\{\mathfrak{s}(x)\}}) \models_{\mathcal{R}} \phi$. As $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$, there exist $\ell_1, \ldots, \ell_\kappa \in \mathcal{L}$ such that $\mathfrak{h}(\mathfrak{s}(x)) = (\ell_1, \ldots, \ell_\kappa)$. Since $y_1, \ldots, y_\kappa$ are pairwise distinct, $\mathfrak{s}[y_i \leftarrow \ell_i \mid 1 \leq i \leq \kappa]$ is well-defined and by definition the structure $(\mathfrak{s}[y_i \leftarrow \ell_i \mid 1 \leq i \leq \kappa], \{\mathfrak{s}(x) \mapsto (\ell_1, \ldots, \ell_\kappa)\})$ is an $\mathcal{R}$-model of $x \mapsto (y_1, \ldots, y_\kappa)$. Thus $(\mathfrak{s}, \{\mathfrak{s}(x) \mapsto (\ell_1, \ldots, \ell_\kappa)\}) \models_{\mathcal{R}} \exists y_1, \ldots, y_\kappa\, x \mapsto (y_1, \ldots, y_\kappa)$. As $\mathfrak{h} = \mathfrak{h}|_{\mathsf{dom}(\mathfrak{h})\setminus\{\mathfrak{s}(x)\}} \cup \{\mathfrak{s}(x) \mapsto (\ell_1, \ldots, \ell_\kappa)\}$, we get $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \exists y_1, \ldots, y_\kappa\, \big(x \mapsto (y_1, \ldots, y_\kappa) \star \phi\big)$.

- Assume that $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \exists y_1,\ldots,y_\kappa\, x \mapsto (y_1,\ldots,y_\kappa) \star \phi$. By definition, (since $y_1,\ldots,y_\kappa$ does not occur in ϕ) there exist disjoint heaps $\mathfrak{h}_1,\mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, $(\mathfrak{s},\mathfrak{h}_1) \models_{\mathcal{R}} \exists y_1,\ldots,y_\kappa\, x \mapsto (y_1,\ldots,y_\kappa)$ and $(\mathfrak{s},\mathfrak{h}_2) \models_{\mathcal{R}} \phi$. This entails that $\mathsf{dom}(\mathfrak{h}_1) = \{\mathfrak{s}(x)\}$, hence $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$, so that $\mathtt{dispose(x)}$ does not fail on $(\mathfrak{s},\mathfrak{h})$. Moreover, $\mathsf{dom}(\mathfrak{h}_2) = \mathsf{dom}(\mathfrak{h}) \setminus \{\mathfrak{s}(x)\}$, so that $(\mathfrak{s},\mathfrak{h}|_{\mathsf{dom}(\mathfrak{h})\setminus\{\mathfrak{s}(x)\}}) \models_{\mathcal{R}} \phi$ and consequently $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} [\mathtt{dispose(x)}]\,\phi$.

F Proof of Lemma 24

Let $(\mathfrak{s},\mathfrak{h})$ be a structure. By definition, $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} [\mathsf{a}]\,\phi$ iff a does not fail on $(\mathfrak{s},\mathfrak{h})$ and $(\mathfrak{s}',\mathfrak{h}') \models_{\mathcal{R}} \phi$, for all $(\mathfrak{s}',\mathfrak{h}')$ such that $(\mathfrak{s},\mathfrak{h}) \rightsquigarrow_{\mathsf{a}} (\mathfrak{s}',\mathfrak{h}')$. As a is of the form $x.i \leftarrow y$ or $y \leftarrow x.i$, a fails iff $\mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$. If a fails, then $[\mathsf{a}]\,\phi$ and $[\mathsf{a}]\,\langle\phi\rangle^{\mathtt{A}(x)}$ are both false in $(\mathfrak{s},\mathfrak{h})$. Otherwise, $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$. Moreover, if $(\mathfrak{s},\mathfrak{h}) \rightsquigarrow_{\mathsf{a}} (\mathfrak{s}',\mathfrak{h}')$ then $\mathsf{dom}(\mathfrak{h}) = \mathsf{dom}(\mathfrak{h}')$, as the actions $x.i \leftarrow y$ or $y \leftarrow x.i$ cannot affect the domain of the heap. Moreover, $\mathfrak{s}(x) = \mathfrak{s}'(x)$, since $x.i \leftarrow y$ does not affect the store and $y \leftarrow x.i$ does not affect the image of x (as $x \neq y$, by the condition in Definition 1). Thus $\mathfrak{s}'(x) \in \mathsf{dom}(\mathfrak{h}')$. By Lemma 19 we deduce that $(\mathfrak{s}',\mathfrak{h}') \models_{\mathcal{R}} \phi$ is equivalent to $(\mathfrak{s}',\mathfrak{h}') \models_{\mathcal{R}} \langle\phi\rangle^{\mathtt{A}(x)}$. Consequently, $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} [\mathsf{a}]\,\phi \iff (\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} [\mathsf{a}]\,\langle\phi\rangle^{\mathtt{A}(x)}$.

G Proof of Lemma 25

The proof is similar to that of Lemma 24. Let $(\mathfrak{s},\mathfrak{h})$ be a structure. If a fails, then $[\mathsf{a}]\,\phi$ and $[\mathsf{a}]\,(x \mapsto (y_1,\ldots,y_\kappa) \star \langle\phi\rangle|_{x\mapsto(y_1,\ldots,y_\kappa)})$ are both false in $(\mathfrak{s},\mathfrak{h})$. Otherwise, by definition of $\rightsquigarrow_{\mathsf{a}}$, if $(\mathfrak{s},\mathfrak{h}) \rightsquigarrow_{\mathsf{a}} (\mathfrak{s}',\mathfrak{h}')$ then $\mathfrak{h}'(\mathfrak{s}'(x)) = (\mathfrak{s}'(y_1),\ldots,\mathfrak{s}'(y_\kappa))$. The proof follows by Lemma 20.

H Proof of Lemma 28

Let $(\mathfrak{s},\mathfrak{h})$ be a structure. By Definition 4, $y \leftarrow x.i$ fails on $(\mathfrak{s},\mathfrak{h})$ if $\mathfrak{s}(x) \notin \mathsf{dom}(\mathfrak{h})$ and otherwise $(\mathfrak{s},\mathfrak{h}) \rightsquigarrow_{y\leftarrow x.i} (\mathfrak{s}',\mathfrak{h}')$ iff $\mathfrak{s}' = \mathfrak{s}\{y \leftarrow \ell_i\}$ and $\mathfrak{h}' = \mathfrak{h}$ with $\mathfrak{h}(\mathfrak{s}(x)) = (\ell_1,\ldots,\ell_\kappa)$.

- Assume that $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} [y \leftarrow x.i]\,\exists\vec{z}\big(x \mapsto (y_1,\ldots,y_\kappa) \star \phi\big)$. Then $y \leftarrow x.i$ does not fail on $(\mathfrak{s},\mathfrak{h})$ (thus $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$), and $(\mathfrak{s}[y \leftarrow \ell_i],\mathfrak{h}) \models_{\mathcal{R}} \exists\vec{z}\big(x \mapsto (y_1,\ldots,y_\kappa) \star \phi\big)$, with $\mathfrak{s}(x) = (\ell_1,\ldots,\ell_\kappa)$. Thus there exists a sequence of locations $\vec{m}$ such that $(\mathfrak{s}[y \leftarrow \ell_i, \vec{z} \leftarrow \vec{m}],\mathfrak{h}) \models_{\mathcal{R}} x \mapsto (y_1,\ldots,y_\kappa) \star \phi$. In particular, (as x or y does not occur in $\vec{z}$ and $x \neq y$, by Definition 1) we must have $\ell_i = \mathfrak{s}[y \leftarrow \ell_i, \vec{z} \leftarrow \vec{m}](y_i)$, hence $\mathfrak{s}[y \leftarrow \ell_i, \vec{z} \leftarrow \vec{m}](y) = \mathfrak{s}[y \leftarrow \ell_i, \vec{z} \leftarrow \vec{m}](y_i)$. Consequently, $(\mathfrak{s}[y \leftarrow \ell_i, \vec{z} \leftarrow \vec{m}],\mathfrak{h}) \models_{\mathcal{R}} \big(x \mapsto (y_1,\ldots,y_\kappa) \star \phi\big)\{y \leftarrow y_i\}$. This implies that $(\mathfrak{s}[\vec{z} \leftarrow \vec{m}],\mathfrak{h}) \models_{\mathcal{R}} (x \mapsto (y_1,\ldots,y_\kappa) \star \phi)\{y \leftarrow y_i\}$ and $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \exists\vec{z}\big(x \mapsto (y_1,\ldots,y_\kappa)) \star \phi\big)\{y \leftarrow y_i\}$.
- Assume that $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \exists\vec{z}\big(x \mapsto (y_1,\ldots,y_\kappa)) \star \phi\big)\{y \leftarrow y_i\}$. This entails that there exist a sequence of locations $\vec{m}$ such that $(\mathfrak{s}[\vec{z} \leftarrow \vec{m}],\mathfrak{h}) \models_{\mathcal{R}} \big(x \mapsto (y_1,\ldots,y_\kappa)) \star \phi\big)\{y \leftarrow y_i\}$, i.e., $(\mathfrak{s}[y \leftarrow \ell_i, \vec{z} \leftarrow \vec{m}],\mathfrak{h}) \models_{\mathcal{R}} x \mapsto (y_1,\ldots,y_\kappa) \star \phi$, with $\ell_i = \mathfrak{s}[\vec{z} \leftarrow \vec{m}](y_i)$ and $\mathfrak{h}(\mathfrak{s}(x)) = (\ell_1,\ldots,\ell_\kappa)$ (as x does not occur in $\vec{z}$). In particular, this entails that $\mathfrak{s}(x) \in \mathsf{dom}(\mathfrak{h})$, thus $y \leftarrow x.i$ does not fail on $(\mathfrak{s},\mathfrak{h})$, and $(\mathfrak{s}[y \leftarrow \ell_i],\mathfrak{h}) \models_{\mathcal{R}} \exists\vec{z}\big(x \mapsto (y_1,\ldots,y_\kappa)) \star \phi\big)$. Thus $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} [y \leftarrow x.i]\,(\exists\vec{z}\,(x \mapsto (y_1,\ldots,y_\kappa)) \star \phi))$.