Testing the Satisfiability of Formulas in Separation Logic with Permissions

Nicolas Peltier

Testing the Satisfiability of Formulas in Separation Logic with Permissions

Nicolas Peltier

Université Grenoble Alpes, LIG, CNRS, Inria, Grenoble INP, F-38000 Grenoble, France

Abstract. We investigate the satisfiability problem for a fragment of Separation Logic (SL) with inductively defined spatial predicates and permissions. We show that the problem is undecidable in general, but decidable under some restrictions on the rules defining the semantics of the spatial predicates. Furthermore, if the satisfiability of permission formulas can be tested in exponential time for the considered permission model then SL satisfiability is EXPTIME complete.

1 Introduction

Separation Logic [14, 22] (SL) is a dialect of bunched logic [18] that is widely used in verification for reasoning on programs manipulating pointer-based data structures. It constitutes the theoretical basis of several industrial scale automated static program analyzers [1, 7, 2]. SL formulas describe *heaps*, with atoms asserting that some location (i.e., a memory address) is allocated and refers to some tuple of locations (i.e., a record), combined with a special connective $*$, called *separating conjunction*, which is used to compose heaps. Custom data structures may be described in this setting by using spatial predicates, the semantics of which is defined using *inductive rules*, similar to those used for defining recursive structures in usual programming languages. Such rules allow one to describe heaps of unbounded size with some particular structure such as lists or trees. In this setting, existing work usually focuses on the fragment of SL called *symbolic heaps* (defined as separating conjunctions of SL atoms).

Usually, SL formulas are interpreted in the *standard heap model*, where heaps are defined as partial finite functions mapping locations to tuples of locations and where the separating conjunction $*$ is interpreted as the disjoint union of heaps. Both the satisfiability and entailment problems have been extensively investigated for this heap model. It was proven that the satisfiability problem is EXPTIME complete [6], whereas the entailment problem is undecidable in general, and 2-EXPTIME complete provided the inductive rules meet some syntactic conditions [13, 11, 12, 15] which are general enough to capture usual data structures used in programming. The combination of spatial reasoning with theory reasoning has also been thoroughly investigated, see for instance [20, 21, 19, 23, 16]).

However, richer models exist (see for instance [8]) accounting for additional features of dynamic memory. The automation of reasoning in these models received little attention. One such model that is of practical relevance is *separation logic with permissions* [3, 5], where allocated locations are associated with so called *permissions* used to model the ownership of a given heap region (e.g., a process may have `read` or `write` permission over some location). The heap composition operator that is used to define the interpretation of the separating conjunction is more complex in this framework than in the above case: non disjoint heaps can be combined if they agree on all the locations on which they are both defined and if the corresponding permissions can be combined (for instance it is natural to assume that `read` permissions can be freely combined but not `write` permissions). The framework is thus parameterized by some *permission model* describing which permissions are available and how they can be combined. In [10] algorithms are provided to decide the satisfiability and entailment problems for SL formulas (symbolic heaps) with permissions in the case of lists, i.e., when all allocated locations refer to a single location (i.e., to a record of size 1) and when there is only one spatial predicate $\texttt{lseg}_p(x, y)$ denoting a list segment from x to y, with permission p. The provided algorithms are generic w.r.t. the permission model, and it is proven that these problems are in NP and co-NP, respectively, assuming that some oracle exists for testing the satisfiability of permission formulas in the considered model.

In the present paper, we investigate the satisfiability problem for SL formulas with permission defined over arbitrary spatial predicates, with user-defined inductive rules. The goal is to allow for more genericity by tackling custom data structures (such as trees, cyclic lists, doubly linked lists etc.) with arbitrary permissions. The addition of permissions makes satisfiability testing much more difficult: we prove that the problem is undecidable in general, and we devise syntactic conditions on the inductive rules for which the problem is EXPTIME-complete. The restrictions are similar – but stronger – to those given in [13] to ensure the decidability of the entailment problem in the standard heap model. In particular, the inductive rules defining the predicate `lseg` mentioned above fulfill these restrictions[1], as well as other usual data structures such as cyclic list, trees etc. (however, doubly linked lists or trees with parent links are not captured). The considered inductive rules use a special connective $\circ$ (different from $*$) that is interpreted as a disjoint union. As we shall see, this is both more natural for defining data structures (see also [5]) and required for deciding satisfiability. Due to lack of space, proofs can be found in the appendix.

2 Definitions

Syntax. We first briefly review some basic notations. If $\boldsymbol{x}$ and $\boldsymbol{y}$ are finite sequences, then we denote by $\boldsymbol{x}.\boldsymbol{y}$ the concatenation of $\boldsymbol{x}$ and $\boldsymbol{y}$. We denote by $|\boldsymbol{x}|$ the length of $\boldsymbol{x}$ and by $\boldsymbol{x}|_i$ its i-th element (if $1 \leq i \leq |\boldsymbol{x}|$). If $E \subseteq \{1, \ldots, |\boldsymbol{x}|\}$ then $\boldsymbol{x}|_E$ denotes the set $\{\boldsymbol{x}|_i \mid i \in E\}$. With a slight abuse of notations, a finite

[1] provided the considered lists are not empty.

sequence $\boldsymbol{x}$ is sometimes identified with the set $\{\boldsymbol{x}|_i \mid i = 1, \ldots, |\boldsymbol{x}|\}$, for instance, we may write $x \in (\boldsymbol{u} \cup \boldsymbol{v}) \setminus \boldsymbol{w}$ to state that x occurs in $\boldsymbol{u}$ or $\boldsymbol{v}$ but not in $\boldsymbol{w}$.

We consider a multisorted framework, with two sorts $\mathtt{l}$ (for locations) and $\mathtt{p}$ (for permissions). Let $\mathcal{V}_\mathtt{l}$ and $\mathcal{V}_\mathtt{p}$ be two countably infinite disjoint sets of *variables* with $\mathcal{V} \stackrel{\text{def}}{=} \mathcal{V}_\mathtt{l} \cup \mathcal{V}_\mathtt{p}$, where $\mathcal{V}_\mathtt{l}$ and $\mathcal{V}_\mathtt{p}$ denote location variables and permission variables, respectively. The set of *permission terms* $\mathcal{T}_\mathtt{p}$ denotes the set of terms built inductively as usual on the set of variables $\mathcal{V}_\mathtt{p}$ and the binary function $\oplus$ (written in infix notation). A *points-to atom* is an expression of the form $x \overset{p}{\mapsto} (y_1, \ldots, y_k)$ with $x, y_1, \ldots, y_k \in \mathcal{V}_\mathtt{l}$ and $p \in \mathcal{T}_\mathtt{p}$. An *equational atom* is an expression of the form $x \simeq y$ or $x \not\simeq y$ with either $x, y \in \mathcal{V}_\mathtt{l}$ or $x, y \in \mathcal{T}_\mathtt{p}$.

We consider two disjoint sets of predicate symbols $\mathcal{P}_\mathtt{p}$ and $\mathcal{P}$. The set $\mathcal{P}_\mathtt{p}$ denotes *permission predicates*, where each predicate $\hat{P} \in \mathcal{P}_\mathtt{p}$ is associated with a unique arity $\#(\hat{P})$. A *permission atom* is an expression of the form $\hat{P}(p_1, \ldots, p_n)$, $\hat{P} \in \mathcal{P}_\mathtt{p}$, $n = \#(\hat{P})$ and $p_1, \ldots, p_n \in \mathcal{T}_\mathtt{p}$. $\mathcal{P}$ is a finite set of *spatial predicate symbols*. Each symbol $P \in \mathcal{P}$ is associated with a *spatial arity* $\#_\mathtt{l}(P) \in \mathbb{N}$ and with an *arity* $\#(P) \in \mathbb{N}$, with $\#(P) > \#_\mathtt{l}(P) > 0$ ($\#_\mathtt{l}(P)$ and $\#(P) - \#_\mathtt{l}(P)$ denote the number of arguments of P that are of sort $\mathtt{l}$ and $\mathtt{p}$, respectively). A *predicate atom* is an expression of the form $P(x_1, \ldots, x_n, p_1, \ldots, p_m)$, with $n = \#_\mathtt{l}(P)$, $n + m = \#(P)$, $x_1, \ldots, x_n \in \mathcal{V}_\mathtt{l}$ and $p_1, \ldots, p_m \in \mathcal{T}_\mathtt{p}$. A *spatial atom* is either a points-to atom or a predicate atom.

The set of *formulas* is built inductively as usual on the logical constants $\mathtt{emp}$, and $\bot$ and on the set of spatial, equational and permission atoms, using the special connectives $*$ and $\circ$ and existential quantification on variables of sort $\mathtt{l}$ only (existential quantification over variables of type $\mathtt{p}$ is not allowed). The connective $*$ is usually called *separating conjunction*, and we call $\circ$ the *disjoint conjunction* (it is intended to capture the disjoint union of heaps[2]). Formulas are taken up to associativity and commutativity of the symbols $*$ and $\circ$, up to the commutativity of $\simeq, \not\simeq$ and up to prenex form. We denote by $|\phi|$ the size of ϕ. For technical convenience, we assume that the symbols $\circ$ and $*$ have weight of 1 and 2, respectively, and that all atoms have size 1. For conciseness, a formula $\exists x_1 \ldots \exists x_n \, \phi$ will often be written $\exists \boldsymbol{x} \, \phi$, with $\boldsymbol{x} = (x_1, \ldots, x_n)$. A *permission formula* is a formula containing no spatial atoms and no equational atom of the form $x \simeq y$ or $x \not\simeq y$ with $x, y \in \mathcal{V}_\mathtt{l}$ (note that $\mathtt{emp}$ is a permission formula). A formula is *spatial* if all the atoms occurring in it are spatial. A *pure formula* is a formula that contains no spatial atom (it is not necessarily a permission formula, as it may contain equations or disequations between locations) A *symbolic heap* is a formula containing no occurrence of $\circ$, and a $\circ$-*formula* is a formula containing no occurrence of $*$.

A variable x is *free* in a formula ϕ if it occurs in ϕ outside of the scope of any quantifier binding x. The set of variables (freely) occurring in a term (or formula) ϕ is denoted by $fv(\phi)$. A *substitution* is a function mapping every variable in $\mathcal{V}_\mathtt{l}$ to a variable in $\mathcal{V}_\mathtt{l}$ and every variable in $\mathcal{V}_\mathtt{p}$ to a term in $\mathcal{T}_\mathtt{p}$. The

[2] The connective $\circ$ is called *strong separating conjunction* in [5] and written $*$ (whereas $*$ is written $\circledast$). Our notations are mostly consistent with those in [10].

domain of a substitution σ (denoted by $dom(\sigma)$) is the set of variables x such that $\sigma(x) \neq x$. A substitution of domain $\{x_1, \ldots, x_n\}$ with $\sigma(x_i) = t_i$ is denoted by $\{x_i \leftarrow t_i \mid i = 1, \ldots, n\}$, or $\{\boldsymbol{x} \leftarrow \boldsymbol{t}\}$, with $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{t} = (t_1, \ldots, t_n)$. For all formulas or terms ϕ, we denote by $\phi\sigma$ the formula or term obtained from ϕ by replacing every free occurrence of a variable x by $\sigma(x)$.

Semantics. Permissions are interpreted in some permission model:

Definition 1 (Adapted from [10]). *A* permission model $\mathfrak{P}$ *is a triple*

$$(\mathcal{P}_{\mathfrak{P}}, \oplus_{\mathfrak{P}}, (\hat{P}_{\mathfrak{P}})_{\hat{P} \in \mathcal{P}_{\mathfrak{p}}})$$

where $\mathcal{P}_{\mathfrak{P}}$ *is a non empty set, called the set of* permissions, $\oplus_{\mathfrak{P}} : \mathcal{P}^2_{\mathfrak{P}} \to \mathcal{P}_{\mathfrak{P}}$ *is a binary partial function that is commutative, associative and cancellative, and* $\hat{P}_{\mathfrak{P}} \subseteq \mathcal{P}^{\#(\hat{P})}_{\mathfrak{P}}$*, for all* $\hat{P} \in \mathcal{P}_{\mathfrak{p}}$*. If* $\pi, \pi' \in \mathcal{P}_{\mathfrak{P}}$*, we write* $\pi \leq_{\mathfrak{P}} \pi'$ *if* $\pi = \pi' \vee (\exists \pi'' \in \mathcal{P}_{\mathfrak{P}}\ \pi' = \pi \oplus \pi'')$*.*

In what follows, $\mathfrak{P}$ always denotes a permission model. If $\pi \in \mathcal{P}_{\mathfrak{P}}$ and $n \in \mathbb{N}$, we denote by π^n the permission $\pi \oplus_{\mathfrak{P}} \ldots \oplus_{\mathfrak{P}} \pi$ (n times), note that π^n is not necessarily defined and implicitly depends on the considered permission model, which will always be clear from the context. In contrast to [10], we do not assume that a maximal "total" permission $1_{\mathfrak{P}}$ exists, we allow instead for arbitrary predicates over permissions (the total permission can be encoded as a unary predicate symbol T, with $T_{\mathfrak{P}} = \{1_{\mathfrak{P}}\}$).

Example 2. Assume that $\mathcal{P}_{\mathfrak{p}} = \emptyset$. A simple example of permission model is $\mathfrak{w} = (\{\texttt{read}, \texttt{write}\}, \oplus_{\mathfrak{w}}, \emptyset)$, with $\texttt{read} \oplus_{\mathfrak{w}} \texttt{read} = \texttt{read}$ and $\texttt{write} \oplus_{\mathfrak{w}} \pi$ is undefined for all $\pi \in \{\texttt{read}, \texttt{write}\}$. Another example (from [4]) is $\mathfrak{f} = (]0, 1], \oplus_{\mathfrak{f}}, \emptyset)$ where $]0, 1]$ denotes the interval of rational numbers, with $\pi \oplus_{\mathfrak{f}} \pi' = \pi + \pi'$ if $\pi + \pi' \leq 1$ and $\pi \oplus_{\mathfrak{f}} \pi'$ is undefined otherwise ($\mathfrak{f}$ stands for *fractional*).

Let $\mathcal{L}$ be a countably infinite set of *locations*. A *store* (for a given permission model $\mathfrak{P}$) is a total mapping associating every variable in $\mathcal{V}_{\mathfrak{l}}$ to an element of $\mathcal{L}$ and every variable in $\mathcal{V}_{\mathfrak{p}}$ to an element of $\mathcal{P}_{\mathfrak{P}}$. A store can be extended into a partial mapping from $\mathcal{T}_{\mathfrak{p}}$ to $\mathcal{P}_{\mathfrak{P}}$ inductively defined as follows: $\mathfrak{s}(p_1 \oplus p_2) \stackrel{\text{def}}{=} \mathfrak{s}(p_1) \oplus_{\mathfrak{P}} \mathfrak{s}(p_2)$. Note that the obtained mapping is partial since $\mathfrak{s}(p_1) \oplus_{\mathfrak{P}} \mathfrak{s}(p_2)$ is not always defined. If $x_1, \ldots, x_n$ are pairwise distinct variables in $\mathcal{V}_{\mathfrak{l}}$ and $\ell_1, \ldots, \ell_n \in \mathcal{L}$, we denote by $\mathfrak{s}\{x_i \leftarrow \ell_i \mid i = 1, \ldots, n\}$ the store $\mathfrak{s}'$ coinciding with $\mathfrak{s}$ on every variable not occurring in $\{x_1, \ldots, x_n\}$ and such that $\mathfrak{s}'(x_i) = \ell_i$ for all $i = 1, \ldots, n$.

A *heap* (for a given permission model $\mathfrak{P}$) is a partial finite function from $\mathcal{L}$ to $\mathcal{L}^* \times \mathcal{P}_{\mathfrak{P}}$. The domain of a heap $\mathfrak{h}$ is denoted by $dom(\mathfrak{h})$, and we denote by $|\mathfrak{h}|$ the finite cardinality of $dom(\mathfrak{h})$. A heap of domain $\ell_1, \ldots, \ell_n$ such that $\mathfrak{h}(\ell_i) = (\ell^i_1, \ldots, \ell^i_{k_i}, \pi_i)$ (for all $i \in \{1, \ldots, n\}$) will be denoted as a set $\{(\ell_i, \ell^i_1, \ldots, \ell^i_{k_i}, \pi_i) \mid i = 1, \ldots, n\}$. For every heap $\mathfrak{h}$ we denote by $loc(\mathfrak{h})$ the set $\{\ell_i \mid \ell_0 \in dom(\mathfrak{h}), \mathfrak{h}(\ell_0) = (\ell_1, \ldots, \ell_k, \pi), 0 \leq i \leq k\}$. A heap may be viewed as a directed (labeled) graph: the locations in $loc(\mathfrak{h})$ are the vertices of the graph

and there is a edge from ℓ to ℓ' if $\mathfrak{h}(\ell) = (\ell_1, \ldots, \ell_n, \pi)$ and $\ell' = \ell_i$ for some $i \in \{1, \ldots, n\}$.

A *subheap* of $\mathfrak{h}$ is any heap $\mathfrak{h}'$ such that $dom(\mathfrak{h}') \subseteq dom(\mathfrak{h})$ and $\mathfrak{h}'(\ell) = \mathfrak{h}(\ell)$ for all $\ell \in dom(\mathfrak{h}')$. A $\mathfrak{p}$-*weakening* of $\mathfrak{h}$ (w.r.t. some permission model $\mathfrak{P}$) is any heap $\mathfrak{h}'$ such that $dom(\mathfrak{h}') = dom(\mathfrak{h})$ and for all $\ell \in dom(\mathfrak{h})$, if $\mathfrak{h}(\ell) = (\ell_1, \ldots, \ell_n, \pi)$ then $\mathfrak{h}'(\ell) = (\ell_1, \ldots, \ell_n, \pi')$ with $\pi' \leq_{\mathfrak{P}} \pi$. We write $\mathfrak{h}' \leq_{\mathfrak{1}} \mathfrak{h}$ (resp. $\mathfrak{h}' \leq_{\mathfrak{p}} \mathfrak{h}$) if $\mathfrak{h}'$ is a subheap (resp. a $\mathfrak{p}$-weakening) of $\mathfrak{h}$. The relation $\leq$ denotes the composition of $\leq_{\mathfrak{1}}$ and $\leq_{\mathfrak{p}}$. We write $\mathfrak{h} \sim \mathfrak{h}'$ if $\mathfrak{h}$ and $\mathfrak{h}'$ only differ by the permissions, i.e., $dom(\mathfrak{h}) = dom(\mathfrak{h}')$ and for all $\ell \in dom(\mathfrak{h})$, if $\mathfrak{h}'(\ell) = (\ell_1, \ldots, \ell_n, \pi')$ then there exists π such that $\mathfrak{h}(\ell) = (\ell_1, \ldots, \ell_n, \pi)$.

Example 3. Consider the permission model $\mathfrak{f}$ defined in Example 2 with $\mathcal{L} = \mathbb{N}$. Then

$$\mathfrak{h}_0 = \{(0,0,1,0.1),(1,0,0,0.2)\}, \qquad \mathfrak{h}_1 = \{(0,0,1,0.1)\},$$
$$\mathfrak{h}_2 = \{(0,0,1,0.1),(1,0,0,0.1)\} \qquad \mathfrak{h}_3 = \{(1,0,0,0.1)\}$$

are heaps, and we have, e.g., $\mathfrak{h}_0(0) = (0, 1, 0.1)$ (meaning that the location 0 is allocated and refers to $(0, 1)$, with permission 0.1), $\mathfrak{h}_1 \leq_{\mathfrak{1}} \mathfrak{h}_0$, $\mathfrak{h}_2 \leq_{\mathfrak{p}} \mathfrak{h}_0$, $\mathfrak{h}_3 \leq_{\mathfrak{1}} \mathfrak{h}_2$, and $\mathfrak{h}_3 \leq \mathfrak{h}_0$. Moreover, $\mathfrak{h}_0 \sim \mathfrak{h}_2$.

Heaps can be composed using the following partial operator. If $\mathfrak{h}_1, \mathfrak{h}_2$ are heaps, then $\mathfrak{h}_1 \sqcup \mathfrak{h}_2$ is defined iff for all $\ell \in dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2)$, we have $\mathfrak{h}_i(\ell) = (\ell^i_1, \ldots, \ell^i_{k_i}, \pi_i)$ (for all $i = 1, 2$) where $k_1 = k_2$, $\ell^1_j = \ell^2_j$ for all $j \in \{1, \ldots, k_1\}$ and $\pi_1 \oplus_{\mathfrak{P}} \pi_2$ is defined. Then $\mathfrak{h}_1 \sqcup \mathfrak{h}_2$ is defined as follows: if $\ell \in dom(\mathfrak{h}_i) \setminus dom(\mathfrak{h}_j)$ with $(i, j) \in \{(1, 2), (2, 1)\}$ then $(\mathfrak{h}_1 \sqcup \mathfrak{h}_2)(\ell) \stackrel{\text{def}}{=} \mathfrak{h}_i(\ell)$, and if $\ell \in dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2)$ then $(\mathfrak{h}_1 \sqcup \mathfrak{h}_2)(\ell) \stackrel{\text{def}}{=} (\ell^1_1, \ldots, \ell^1_{k_1}, \pi_1 \oplus_{\mathfrak{P}} \pi_2)$.

Example 4. Consider the permission model $\mathfrak{f}$ defined in Example 2, with $\mathcal{L} = \mathbb{N}$ and the following heaps:

$$\mathfrak{h}_0 = \{(0,0,0.5),(1,0,0.6)\} \qquad \mathfrak{h}_1 = \{(0,0,0.5),(1,0,0.2),(2,0.1)\}$$
$$\mathfrak{h}_2 = \{(0,0,0.5),(1,0,0.6)\} \qquad \mathfrak{h}_3 = \{(0,0,0.1),(1,0.1)\}$$

Then $\mathfrak{h}_0 \sqcup \mathfrak{h}_1$ is defined, and we have: $\mathfrak{h}_0 \sqcup \mathfrak{h}_1 = \{(0, 0, 1), (1, 0, 0.8), (2, 0.1)\}$. However, neither $\mathfrak{h}_0 \sqcup \mathfrak{h}_2$ nor $\mathfrak{h}_0 \sqcup \mathfrak{h}_3$ is defined (in the former case the permissions of location 1 cannot be combined (as $0.6 + 0.6 > 1$) and in the latter case the location 1 is associated with distinct tuples, (0) and (), respectively.

A *structure* (for a given permission model $\mathfrak{P}$) is a pair $(\mathfrak{s}, \mathfrak{h})$ where $\mathfrak{s}$ is a store and $\mathfrak{h}$ is a heap for $\mathfrak{P}$. It is *injective* if $\mathfrak{s}$ is injective. A location ℓ is *allocated* in a structure $(\mathfrak{s}, \mathfrak{h})$ or in a heap $\mathfrak{h}$ if $\ell \in dom(\mathfrak{h})$, and a variable x is *allocated* in $(\mathfrak{s}, \mathfrak{h})$ if $\mathfrak{s}(x) \in dom(\mathfrak{h})$.

The semantics of spatial predicate is defined by inductive rules. A *set of inductive definitions* (SID) is a set of *rules* of the form $P(x_1, \ldots, x_n, y_1, \ldots, y_m) \Leftarrow \phi$ where $n = \#_{\mathfrak{1}}(P)$, $n + m = \#(P)$, $x_1, \ldots, x_n$ are pairwise distinct variables in $\mathcal{V}_{\mathfrak{1}}$, $y_1, \ldots, y_m$ are pairwise distinct variables in $\mathcal{V}_{\mathfrak{p}}$, and ϕ is a formula such that $fv(\phi) \subseteq \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$. We write $P(z_1, \ldots, z_n, p_1, \ldots, p_m) \Leftarrow_{\mathcal{R}} \psi$ iff $\mathcal{R}$ contains a rule $P(x_1, \ldots, x_n, y_1, \ldots, y_m) \Leftarrow \phi$ with $\psi = \phi\{x_i \leftarrow z_i, y_j \leftarrow p_j \mid i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}\}$.

Definition 5. *(Semantics) For every permission model $\mathfrak{P}$ and SID $\mathcal{R}$, the satisfiability relation $\models^{\mathfrak{P}}_{\mathcal{R}}$ is the smallest relation between structures (for $\mathfrak{P}$) and formulas such that:*

1. $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \mathtt{emp}$ *iff* $\mathfrak{h} = \emptyset$.
2. $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} x \overset{p}{\mapsto} (y_1,\ldots,y_k)$ *if* $\mathfrak{s}(p)$ *is defined and* $\mathfrak{h} = \{(\mathfrak{s}(x),\mathfrak{s}(y_1),\ldots,\mathfrak{s}(y_k),\mathfrak{s}(p))\}$. *Note that this entails that* $dom(\mathfrak{h}) = \{\mathfrak{s}(x)\}$.
3. $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} x \simeq y$ *(resp.* $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} x \not\simeq y$*) if* $\mathfrak{h} = \emptyset$, $\mathfrak{s}(x)$ *and* $\mathfrak{s}(y)$ *are defined and* $\mathfrak{s}(x) = \mathfrak{s}(y)$ *(resp.* $\mathfrak{s}(x) \neq \mathfrak{s}(y)$*).*
4. $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \hat{P}(p_1,\ldots,p_n)$ *with* $\hat{P} \in \mathcal{P}_{\mathtt{p}}$ *if* $\mathfrak{s}(p_i)$ *is defined for all* $i \in \{1,\ldots,n\}$, $(\mathfrak{s}(p_1),\ldots,\mathfrak{s}(p_n)) \in \hat{P}_{\mathfrak{P}}$ *and* $\mathfrak{h} = \emptyset$.
5. $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(x_1,\ldots,x_n,\pi_1,\ldots,\pi_m)$ *with* $P \in \mathcal{P}$ *if there exists* ϕ *such that* $P(x_1,\ldots,x_n,\pi_1,\ldots,\pi_m) \Leftarrow_{\mathcal{R}} \phi$ *and* $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$.
6. $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_1 * \phi_2$ *if there exist heaps* $\mathfrak{h}_1,\mathfrak{h}_2$ *such that* $\mathfrak{h}_1 \sqcup \mathfrak{h}_2$ *is defined,* $\mathfrak{h} = \mathfrak{h}_1 \sqcup \mathfrak{h}_2$ *and* $(\mathfrak{s},\mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$ *for all* $i = 1,2$.
7. $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_1 \circ \phi_2$ *if there exists heaps* $\mathfrak{h}_1,\mathfrak{h}_2$ *such that* $dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2) = \emptyset$, $\mathfrak{h} = \mathfrak{h}_1 \sqcup \mathfrak{h}_2$ *and* $(\mathfrak{s},\mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$ *for all* $i = 1,2$.
8. $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \exists x\, \phi$ *if* $(\mathfrak{s}\{x \leftarrow \ell\},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$ *for some* $\ell \in \mathcal{L}$.

A structure $(\mathfrak{s},\mathfrak{h})$ *such that* $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$ *is an* $(\mathcal{R},\mathfrak{P})$-model *of* ϕ. *A formula admitting an* $(\mathcal{R},\mathfrak{P})$*-model is* $(\mathcal{R},\mathfrak{P})$-satisfiable. *Two formulas are* sat-equivalent *(w.r.t.* $\mathcal{R}$, $\mathfrak{P}$*) if they are both* $(\mathcal{R},\mathfrak{P})$*-satisfiable or both* $(\mathcal{R},\mathfrak{P})$*-unsatisfiable.*

Example 6. The formula $x \overset{u}{\mapsto} (y,z) \circ x \overset{u'}{\mapsto} (y',z')$ is $(\mathcal{R},\mathfrak{P})$-unsatisfiable, as x cannot be allocated in disjoint parts of the heap. $x \overset{u}{\mapsto} (y) * x \overset{u'}{\mapsto} (y') * y \not\simeq y'$ is also $(\mathcal{R},\mathfrak{P})$-unsatisfiable, as x cannot refer to two distinct records, but $x \overset{u}{\mapsto} (y,z) * x \overset{u'}{\mapsto} (y',z')$ admits the model (on the permission model $\mathfrak{f}$) $(\mathfrak{s},\mathfrak{h})$ with $\mathfrak{s}(x) = 0$, $\mathfrak{s}(y) = \mathfrak{s}(y') = 1$, $\mathfrak{s}(z) = \mathfrak{s}(z') = 2$, $\mathfrak{s}(u) = 0.5$, $\mathfrak{s}(u') = 0.2$ and $\mathfrak{h} = \{(0,1,2,0.7)\}$.

Note that there is no logical constant $\top$ (true): no formula can be satisfied on all heaps. The constant $\mathtt{emp}$ is similar to $\top$ but it states that the heap is empty. For all formulas ϕ,ψ, we write $\phi \models^{\mathfrak{P}}_{\mathcal{R}} \psi$ iff the implication $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi \implies (\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$ holds for all structures $(\mathfrak{s},\mathfrak{h})$, and $\phi \equiv^{\mathfrak{P}}_{\mathcal{R}} \psi$ iff we have both $\phi \models^{\mathfrak{P}}_{\mathcal{R}} \psi$ and $\psi \models^{\mathfrak{P}}_{\mathcal{R}} \phi$. If ϕ contains no predicate symbols in $\mathcal{P}$, then the truth value of ϕ in $(\mathfrak{s},\mathfrak{h})$ does not depend on $\mathcal{R}$. We thus may write $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}} \phi$ instead of $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$. If, moreover, ϕ is pure, then $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}} \phi$ holds only if $\mathfrak{h}$ is empty. We will write $\mathfrak{s} \models^{\mathfrak{P}} \phi$ to state that $(\mathfrak{s},\emptyset) \models^{\mathfrak{P}} \phi$. Finally, if ϕ contains only equalities between variables then its semantics does not depend on $\mathcal{R}$ and $\mathfrak{P}$ thus we write $\mathfrak{s} \models \phi$ to state that $(\mathfrak{s},\emptyset) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$. Note that the semantics of $\phi_1 \circ \phi_2$ and $\phi_1 * \phi_2$ coincide if ϕ_1 or ϕ_2 is pure, and also coincide with that of the usual standard conjunction if *both* ϕ_1 and ϕ_2 are pure.

Shorthands. If $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_m)$ are sequences of variables in $\mathcal{V}_1$ then $\boldsymbol{x} \simeq \boldsymbol{y}$ denotes the formula $\bot$ if $n \neq m$ and $(x_1 \simeq y_1) \circ \ldots \circ (x_n \simeq y_n)$ otherwise. For every permission term p, we denote by $def(p)$ the atom $p \simeq p$. By definition, $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} def(p)$ iff $\mathfrak{s}(p)$ is defined and $\mathfrak{h} = \emptyset$.

3 $\mathfrak{h}$-Regular Systems

We focus on SIDs of some particular form, defined below.

Definition 7. *A rule is* $\mathfrak{h}$-regular *if it is of the following form:*

$$P(x, \boldsymbol{y}) \Leftarrow \exists u_1, \ldots, u_n \, (x \overset{p}{\mapsto} (v_1, \ldots, v_k) \circ Q_1(u_1, \boldsymbol{y}_1) \ldots \circ Q_n(u_n, \boldsymbol{y}_n) \circ \phi)$$

where $\{u_1, \ldots, u_n\} \subseteq \{v_1, \ldots, v_k\}$, $\boldsymbol{y}_i$ *is a vector of variables*[3], $Q_i \in \mathcal{P}$ *and* ϕ *is pure. We assume by* α*-renaming that* $x, \boldsymbol{y}$ *do not occur in* $\{u_1, \ldots, u_n\}$*. A SID* $\mathcal{R}$ *is* $\mathfrak{h}$-regular *if all the rules in* $\mathcal{R}$ *are* $\mathfrak{h}$*-regular.*

Note that the right-hand side formula contains only the disjoint separation connective $\circ$ and not the usual separating conjunction $*$. As we will see (Theorem 33) this is crucial for the decidability of the satisfiability problem. However, as already observed in [5], this is also justified from a practical point of view. Assume for instance that we want to define the predicate `lseg` introduced in [10], denoting a list segment from x to y with some permission z. The following rules can be used[4]: $\texttt{lseg}(x, y, z) \Leftarrow x \overset{z}{\mapsto} (y) \quad \texttt{lseg}(x, y, z) \Leftarrow \exists u \, (x \overset{z}{\mapsto} (u) \circ \texttt{lseg}(u, y, z))$. A structure $(\mathfrak{s}, \mathfrak{h})$ satisfies $\texttt{lseg}(x, y, z)$ if $\mathfrak{h} = \{(\ell_i, \ell_{i+1}, \mathfrak{s}(z)) \mid i = 1, \ldots, n\}$ with $n > 0$, $\mathfrak{s}(x) = \ell_1$, $\mathfrak{s}(y) = \ell_{n+1}$ and $\ell_i \neq \ell_j$ if $i \neq j$ and $i, j \in \{1, \ldots, n\}$. This fits in with the definition in [10] (except that $n > 0$). In contrast, if one uses instead the connective $*$: $\texttt{lseg}(x, y, z) \Leftarrow \exists u \, (x \overset{z}{\mapsto} (u) * \texttt{lseg}(u, y, z))$, then one could obtain models where the list "loops" on itself an arbitrary number of times, such as, for instance $(\mathfrak{s}, \{(\mathfrak{s}(x), \mathfrak{s}(x), p))\})$, with $\mathfrak{s}(y) = \mathfrak{s}(x)$ and $p = \mathfrak{s}(z)^n$, for any $n > 0$ such that $\mathfrak{s}(z)^n$ is defined. In the former definition, $\mathfrak{s}(y)$ possibly occurs in $\{\ell_1, \ldots, \ell_n\}$, but each location can only be allocated once.

Intuitively, $\mathfrak{h}$-regular sets of inductive rules generate heaps with a regular structure (in the sense that it may be represented by a tree automaton [9]), enriched with some additional edges (referring to the nodes corresponding to the variables passed as parameters to the spatial predicates at some recursive calls). These additional edges may refer to locations corresponding to free variables (e.g. the root of the structure) but also to existential variables (for instance they may refer to the parent node in the tree). $\mathfrak{h}$-Regular SID are related to the Pce systems introduced in [13] (for **p**rogressing, **c**onnected and **e**stablished), extended to formulas with permissions, but our conditions are slightly stronger, because we require that every existential variable be allocated at the next recursive call. Note that structures with mixed permissions are allowed, for instance

[3] i.e., compound permission terms are not allowed in predicate atoms.

[4] As $\mathfrak{h}$-regular rules allocate exactly one location, we assume that the segment is non empty, the case of an empty segment must be considered apart.

the rules $P(x, z_1, z_2) \Leftarrow x \overset{z_1}{\mapsto} ()$ and $P(x, z_1, z_2) \Leftarrow \exists u\,(x \overset{z_1}{\mapsto} (u) \circ P(u, z_2, z_1))$ defines a list with permissions alternating between z_1 and z_2. Rules with compound permission terms in points-to or permission atoms are allowed (such as $P(x, y_1, y_2) \Leftarrow x \overset{y_1 \oplus y_2}{\mapsto} () \circ \mathit{def}(y_1 \oplus y_1)$), but not those with compound permission terms in spatial predicate atoms[5] (e.g., $P(x, y_1, y_2) \Leftarrow x \overset{y_1}{\mapsto} () \circ Q(x, y_1 \oplus y_2)$ is *not* $\mathfrak{h}$-regular).

For every quantifier-free formula ϕ, we denote by $\mathit{roots}(\phi)$ the set of variables x (called the *roots of* ϕ) inductively defined as follows: $\mathit{roots}(x \overset{p}{\mapsto} (y_1, \ldots, y_k)) \overset{\text{def}}{=} \{x\}$, $\mathit{roots}(P(x, y_1, \ldots, y_k)) \overset{\text{def}}{=} \{x\}$, $\mathit{roots}(\exists y\, \phi) = \mathit{roots}(\phi) \setminus \{y\}$, $\mathit{roots}(\phi) = \emptyset$ if ϕ is pure and $\mathit{roots}(\phi_1 * \phi_2) = \mathit{roots}(\phi_1 \circ \phi_2) = \mathit{roots}(\phi_1) \cup \mathit{roots}(\phi_2)$. By Definition 7, roots are always allocated:

Proposition 8. *Let $\mathcal{R}$ be a $\mathfrak{h}$-regular SID. If $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$ and $x \in \mathit{roots}(\phi)$ then $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h})$. Consequently, every formula of the form $\phi_1 \circ \phi_2$ with $\mathit{roots}(\phi_1) \cap \mathit{roots}(\phi_2) \neq \emptyset$ is $(\mathcal{R}, \mathfrak{P})$-unsatisfiable.*

The conditions in Definition 7 are actually not sufficient to ensure that the satisfiability problem is decidable:

Theorem 9. *If there exist (not necessary distinct) permissions $\pi_1, \pi_2 \in \mathcal{P}_{\mathfrak{P}}$ such that $\pi_1 \oplus_{\mathfrak{P}} \pi_2$ is defined, then the $(\mathcal{R}, \mathfrak{P})$-satisfiability problem is undecidable for $\mathfrak{h}$-regular SID $\mathcal{R}$.*

To ensure decidability, we need to further restrict the way existential variables are passed as parameters during recursive calls. This is the goal of the next definition.

Definition 10. *Assume that $\mathcal{R}$ is $\mathfrak{h}$-regular. Given two spatial predicates P and Q, of arities n and m respectively, we write $P \bowtie_{\mathcal{R}} Q$ if $P(x, x_1, \ldots, x_{n-1}) * Q(x, y_1, \ldots, y_{m-1})$ is $(\mathcal{R}, \mathfrak{P})$-unsatisfiable[6] (where $x_1, \ldots, x_{n-1}, y_1, \ldots, y_{m-1}$ denote pairwise distinct variables of the appropriate sorts). We denote by $\gamma_{\mathcal{R}}$ the function associating every predicate symbol P of spatial arity n to a subset of $\{2, \ldots, n\}$ inductively defined as follows: for every rule $P(x_1, \ldots, x_n, \boldsymbol{u}) \Leftarrow \exists y_1, \ldots, y_m\, \phi$ in $\mathcal{R}$, for every predicate atom $Q(z_1, \ldots, z_k, \boldsymbol{u}_k)$ in ϕ with $\#_1(Q) = k$ and for all $i \in \{2, \ldots, k\}$:*

1. $z_i \in \{y_1, \ldots, y_m\} \Rightarrow i \in \gamma_{\mathcal{R}}(Q)$.
2. $z_i \in \{x_j \mid j \in \gamma_{\mathcal{R}}(P)\} \implies i \in \gamma_{\mathcal{R}}(Q)$.

[5] Otherwise the unfolding of spatial predicates could yield terms of arbitrary depth.

[6] In practice, as this condition is hard to test, some stronger syntactic condition can be tested instead, for instance one can check that all the formulas ϕ and ϕ' such that $P(x, x_1, \ldots, x_{n-1}) \Leftarrow_{\mathcal{R}} \phi$ and $Q(x, y_1, \ldots, y_{m-1}) \Leftarrow_{\mathcal{R}} \phi'$ are of the form $\phi = (x \mapsto (\boldsymbol{u}) \circ \psi)$ and $\phi' = (x \mapsto (\boldsymbol{u}') \circ \psi')$ with $|\boldsymbol{u}| \neq |\boldsymbol{u}'|$ (this condition is used in Theorem 33 and for the EXPTIME-hardness proof in Theorem 32.). More generally, it is sufficient to test that the "shape" of the structures generated by P and Q, up to a certain fixed unfolding depth, are incompatible.

Let $\mathcal{P}^\star$ be a subset of $\mathcal{P}$, such that: (3) $P \in \mathcal{P}^\star \implies \gamma_{\mathcal{R}}(P) = \emptyset$; and (4) $P \in \mathcal{P}^\star \wedge Q \in \mathcal{P} \setminus \mathcal{P}^\star \implies P \bowtie_{\mathcal{R}} Q$. A $\mathfrak{h}$-regular rule is $\exists$-restricted *(w.r.t. $\mathcal{R}$ and $\mathcal{P}^\star$) if it satisfies the following condition (using the notations of Definition 7):*

5. $\forall i \in \{1, \ldots, n\}\, \forall j \in \{1, \ldots, n\}\, (u_i \in \boldsymbol{y}_j \implies Q_i \in \mathcal{P}^\star)$.

A SID $\mathcal{R}$ is $\exists$-restricted *if all the rules in $\mathcal{R}$ are $\exists$-restricted.*

Conditions 1 and 2 in Definition 10 are meant to ensure that $\gamma_{\mathcal{R}}(P)$ denotes the indices of the parameters of P that may (but do not have to) be instantiated by some existential variable introduced during the unfolding of the inductive rules in $\mathcal{R}$ (the other parameters may only be instantiated by variables occurring in the initial formula). Condition 1 corresponds to a base case, where an existential variable is passed as a parameter to a predicate symbol, and Condition 2 handles the inductive case, when the variable is carried through recursive calls[7]. Then, Condition 5 ensures that an existential variable may only be passed as a parameter to a predicate symbol if it is the root of a structure defined by an atom $Q_i(\boldsymbol{y}_i)$ containing no variables introduced by unfolding (by Condition 3).

Example 11. The rules of the predicate $\mathtt{lseg}$ are $\exists$-restricted (with $\mathcal{P}^\star = \emptyset$). Indeed, they contain only one existential variable u, which occurs only as the first argument of a predicate. Hence Condition 5 in Definition 10 trivially holds. If $\mathcal{R}$ contains no other rule then $\gamma_{\mathcal{R}}(\mathtt{lseg}) = \emptyset$. Note that $\gamma_{\mathcal{R}}(\mathtt{lseg})$ depends on the entire set $\mathcal{R}$. For instance, if $\mathcal{R}$ contains a rule $P(x, y) \Leftarrow \exists u\, (x \overset{y}{\mapsto} (u) \circ \mathtt{lseg}(u, u, y))$ then the second argument of $\mathtt{lseg}$ may be instantiated by an existential variable hence $\gamma_{\mathcal{R}}(\mathtt{lseg}) = \{2\}$, and the latter rule is not $\exists$-restricted. On the other hand, if $\mathcal{P}^\star = \{Q\}$, then the rules $Q(x, y) \Leftarrow x \overset{y}{\mapsto} (), R(x, y) \Leftarrow \exists u, v\, (x \overset{y}{\mapsto} (u, v) \circ \mathtt{lseg}(u, v, y) \circ Q(v, y))$ are $\exists$-restricted, with $\mathcal{P}^\star = \{Q\}$. Indeed, the variable u occurs only at the root of a predicate, and the variable v is the root of $Q(v, y)$. Note that $\mathtt{lseg}(x, y, z) * Q(x, u)$ and $R(x, y) * Q(x, u)$ are $(\mathcal{R}, \mathfrak{P})$-unsatisfiable, thus $\mathtt{lseg} \bowtie_{\mathcal{R}} Q$ and $R \bowtie_{\mathcal{R}} Q$.

Intuitively, the structures generated by $\exists$-restricted rules are regular tree-shaped structures, enriched with two kinds of additional edges: (i) a *bounded* number of *arbitrary* edges (corresponding to free variables, which may be freely passed as arguments to any predicate, thus may be referred to in an arbitrary way); (ii) an *unbounded* number of other edges (corresponding to existential variables) which are only allowed to point to structures that contain no edge of type (ii). Condition 4 ensures that the structures containing only edges of type (i) do not overlap with those containing both kinds of edges. Note that the conditions of Definition 10 always hold if the existential variables occur only as roots (with $\mathcal{P}^\star = \mathcal{P}$ or $\mathcal{P}^\star = \emptyset$). In this case there is no edge of type (ii), i.e., the obtained structures are regular sets of trees with a bounded number of additional edges (for instance

[7] For generality, one could assume that all the equalities occurring in the rules are propagated before $\gamma_{\mathcal{R}}$ is computed (so that existential variables are eliminated if they are equal to a free variable), but this is not essential for our purposes hence the corresponding formal definitions are omitted.

trees with pointers to the root, or cyclic lists). Note that doubly linked lists cannot be captured (as they contain an unbounded number of additional edges from every node to the previous one). In the following we devise an algorithm to test the $(\mathcal{R}, \mathfrak{P})$-satisfiability of symbolic heaps when $\mathcal{R}$ is $\exists$-restricted.

4 A Decision Procedure For Testing Satisfiability

Before entering into technical details we start with a general overview of the procedure for testing satisfiability (assuming the considered SID is $\exists$-restricted).

1. Starting with a formula of the form $\delta_1 * \cdots * \delta_n$ where the δ_i's are atoms, we first reduce every spatial atom δ_i into an equivalent disjunction of $\circ$-conjunctions $\delta^i_1 \circ \ldots \circ \delta^i_{m_i}$ such that the *only* free variables allocated by an atom δ^i_j are its roots $\mathit{roots}(\delta^i_j)$ (as δ^i_j is an atom, $\mathit{card}(\mathit{roots}(\delta^i_j)) \leq 1$). Due to the particular properties of the $\mathfrak{h}$-regular rules (more precisely, due to the fact that the rules satisfy the "establishment" property of [13], i.e., every existential variable is allocated), this entails that, for all structures $(\mathfrak{s}, \mathfrak{h}_{i,j})$ satisfying δ^i_j, the domains of $\mathfrak{h}_{i,j}$ and $\mathfrak{h}_{i',j'}$ are either equal (if δ^i_j and $\delta^{i'}_{j'}$ have the same roots) or disjoint (otherwise). Indeed, the establishment property ensures that the considered heaps have no "pending edges" (i.e., no location that is referred to but not allocated), other than those denoted by free variables. This step can be considered as the key part of the procedure. It requires to (automatically) enrich the language with additional predicates and rules, and the termination of the transformation crucially depends on the conditions on $\exists$-restricted rules. For instance, an atom $\mathtt{lseg}(x, x)$ occurring in a formula with free variables x, y could be written $(x \simeq y \circ \mathtt{lseg}(x, x)) \vee \mathtt{lseg}'(x, x, y) \vee (\mathtt{lseg}'(x, y, y) \circ \mathtt{lseg}'(y, x, x))$ where $\mathtt{lseg}'(u, v, w)$ denotes a list segment from u to v not allocating w. The previous decomposition depends on whether y is equal to x and whether y occurs in the list segment from x to x.
2. By distributivity, we get at this point $*$-conjunctions of $\circ$-conjunctions of atoms. Taking advantage of the previous property, we then reduce these formulas into $\circ$-conjunctions of $*$-conjunctions of atoms, by regrouping the atoms with the same roots, e.g., $(P(x, y) \circ Q(y, x)) * (P'(x, y) \circ Q'(y, x))$ may be written $(P(x, y) * P'(x, y)) \circ (Q(y, x) * Q'(y, x))$.
3. Next, we show that a $*$-conjunction of atoms sharing the same root (such as $P(x, y) * P'(x, y)$ or $Q(y, x) * Q'(y, x)$) can be denoted by a single atom, the rules of which are obtained by "merging" the rules of the initial atoms.
4. At this point we get a $\circ$-conjunction of atoms. To ensure that the formula is satisfiable it suffices to test that all these atoms have a model and that all these models are compatible, w.r.t. the equality constraints, allocated locations and permission constraints. To this aim, we construct finite abstractions of the models of the considered atoms using a bottom-up fixpoint algorithm.

In the next subsections, each of these steps is explained in details.

4.1 Normalization

We first show that every formula can be transformed into an equivalent formula (that we call *normalized*) in which every allocated variable occurs as a root:

Definition 12. *A formula ϕ is* normalized *if it is of the form $\exists \boldsymbol{x}\, \psi$ where ψ is quantifier-free and for all spatial atoms δ in ψ, for all $(\mathcal{R}, \mathfrak{P})$-models $(\mathfrak{s}, \mathfrak{h})$ of δ and for all variables $y \in fv(\psi)$: $\mathfrak{s}(y) \in dom(\mathfrak{h}) \iff y \in roots(\psi)$.*

For instance, $\mathtt{lseg}(x, y)$ is not normalized, because y may be allocated (e.g., if $\mathfrak{s}(x) = \mathfrak{s}(y)$) and does not occur in $roots(\mathtt{lseg}(x, y)) = \{x\}$. To enforce this condition, we introduce new predicate symbols (called *derived predicates*), the rules of which can be automatically computed from those of the predicates already occurring in this formula. We first define predicate symbols that ensure that some given variable is not allocated.

Definition 13. *For all predicate atoms $P(\boldsymbol{x}, \boldsymbol{p})$ (where $\boldsymbol{x}$ and $\boldsymbol{p}$ are vectors of location variables and permission terms, respectively) and for all location variables v, we denote by $P(\boldsymbol{x}, \boldsymbol{p})[v]^-$ any atom of the form $Q(\boldsymbol{x}, v, \boldsymbol{p})$, where Q is a fresh predicate symbol, associated with the rules:*

$$Q(\boldsymbol{y}, w, \boldsymbol{z}) \Leftarrow \exists \boldsymbol{u}\, (Q_1(\boldsymbol{y}_1, \boldsymbol{p}_1)[w]^- \circ \ldots \circ Q_m(\boldsymbol{y}_m, \boldsymbol{p}_m)[w]^- \circ \phi \circ \boldsymbol{y}|_1 \not\simeq w)$$

for all rules $P(\boldsymbol{y}, \boldsymbol{z}) \Leftarrow \exists \boldsymbol{u}\, (Q_1(\boldsymbol{y}_1, \boldsymbol{p}_1) \circ \ldots \circ Q_m(\boldsymbol{y}_m, \boldsymbol{p}_m) \circ \phi)$ in $\mathcal{R}$ (up to AC), where $\boldsymbol{y}, \boldsymbol{y}_i$ are vectors of location variables, $\boldsymbol{z}$, $\boldsymbol{p}_i$ are vectors of permission variables, and ϕ contains no predicate atom.

For instance $\mathtt{lseg}(x, y, z)[u]^-$ is a predicate atom $Q(x, y, u, z)$ defined by the following rules: $\{Q(x, y, u, z) \Leftarrow \exists x'(x \overset{z}{\mapsto} (x') \circ Q(x', y, u, z) \circ x \not\simeq u), Q(x, y, u, z) \Leftarrow x \overset{z}{\mapsto} (y) \circ x \not\simeq u\}$. It denotes a list segment from x to y not allocating u. The following result is straightforward to prove:

Proposition 14. *For every $\exists$-restricted SID $\mathcal{R}$, the set $\mathcal{R}$ enriched with the rules associated with the predicate Q corresponding to $P(\boldsymbol{x}, p)[v]^-$ in Definition 13 is $\exists$-restricted, with $\gamma_{\mathcal{R}}(Q) = \gamma_{\mathcal{R}}(P)$ and $Q \in \mathcal{P}^\star \iff P \in \mathcal{P}^\star$.*

Intuitively the structures that satisfy $P(\boldsymbol{x}, \boldsymbol{p})[v]^-$ are exactly those that satisfy $P(\boldsymbol{x}, \boldsymbol{p})$ and do not allocate v:

Lemma 15. *For all $\mathfrak{h}$-regular SID $\mathcal{R}$, $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(\boldsymbol{x}, p)[v]^-$ iff $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(\boldsymbol{x}, p)$ and $\mathfrak{s}(v) \notin dom(\mathfrak{h})$.*

The operator $\delta \mapsto \delta[x]^-$ can be applied recursively, e.g., one can consider atoms of the form $\delta[x]^-[y]^-$, etc. For all predicate atoms δ, we denote by $unalloc(\delta)$ the set of variables inductively defined as follows: $unalloc(\delta[x]^-) \overset{\text{def}}{=} \{x\} \cup unalloc(\delta)$, and $unalloc(\delta) \overset{\text{def}}{=} \emptyset$ if δ is not of the form $\delta'[x]^-$. The following proposition is an immediate consequence of Lemma 15:

Proposition 16. *If $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \delta$ then $\mathfrak{s}(x) \notin dom(\mathfrak{h})$, for all $x \in unalloc(\delta)$.*

Next, we define predicate symbols allowing one to remove some part of a structure. Intuitively, the expression $(\phi \mathrel{-\!\!\bullet} \psi)$ will hold exactly in the structures that satisfy ψ when a disjoint structure satisfying ϕ is added. For instance given the rules $\mathtt{tree}(x,y) \Leftarrow \exists x_1, x_2\, x \overset{y}{\mapsto} (x_1, x_2) \circ \mathtt{tree}(x_1, y) \circ \mathtt{tree}(x_2, y)$ and $\mathtt{tree}(x,y) \Leftarrow x \overset{y}{\mapsto} ()$, $\mathtt{tree}(z,y)$ and $\mathtt{tree}(x,y)$ denote binary trees with roots z and x, respectively, and $\mathtt{tree}(z,y) \mathrel{-\!\!\bullet} \mathtt{tree}(x,y)$ denotes a tree of root x with a "hole" at z (the structures satisfying $\mathtt{tree}(z,y) \mathrel{-\!\!\bullet} \mathtt{tree}(x,y)$ are obtained from models of $\mathtt{tree}(x,y)$ by removing the part of the heap that corresponds to $\mathtt{tree}(z,y)$). The formula $\phi \mathrel{-\!\!\bullet} \psi$ is similar to the *strong magic wand* introduced in [17] and to the *context predicates* in [12] and also close in spirit to the separating implication of SL although the semantics are slightly different.

Definition 17. *For all finite sequences of predicate atoms $P_i(\boldsymbol{x}_i, \boldsymbol{p}_i)$ (with $i = 0, \ldots, n$), where $\boldsymbol{x}_i$ and $\boldsymbol{p}_i$ are vectors of location variables and permission terms, respectively, we denote by $(P_1(\boldsymbol{x}_1, \boldsymbol{p}_1) \circ \ldots \circ \hat{P}_n(\boldsymbol{x}_n, \boldsymbol{p}_n)) \mathrel{-\!\!\bullet} P_0(\boldsymbol{x}_0, \boldsymbol{p}_0)$ any atom $P(\boldsymbol{x}, \boldsymbol{p})$ with $\boldsymbol{x} = \boldsymbol{x}_0.\ldots.\boldsymbol{x}_n$, $\boldsymbol{p} = \boldsymbol{p}_0.\ldots.\boldsymbol{p}_n$, and such that $P = P_0$ if $n = 0$ and otherwise P is a fresh symbol associated with rules of the form*

$$P(\boldsymbol{y}, \boldsymbol{z}) \Leftarrow \exists \boldsymbol{w}\, (\psi_1 \circ \ldots \circ \psi_m \circ \phi)$$

for all rules

$$P_0(\boldsymbol{y}_0, \boldsymbol{z}_0) \Leftarrow \exists \boldsymbol{w}\, (Q_1(\boldsymbol{u}_1, \boldsymbol{q}_1) \circ \ldots \circ Q_m(\boldsymbol{u}_m, \boldsymbol{q}_m) \circ \phi)$$

in $\mathcal{R}$ and for all decompositions $\alpha_1 \circ \ldots \circ \alpha_m = P_1(\boldsymbol{y}_1, \boldsymbol{z}_1) \circ \ldots \circ P_n(\boldsymbol{y}_n, \boldsymbol{z}_n)$ (up to AC, where the α_i's may be empty), where:

- *$\boldsymbol{y}_i$ and $\boldsymbol{z}_i$ are sequences of pairwise distinct location and permission variables, respectively, with $|\boldsymbol{y}_i| = |\boldsymbol{x}_i|$ and $|\boldsymbol{z}_i| = |\boldsymbol{p}_i|$;*
- *$\boldsymbol{y} = \boldsymbol{y}_0.\ldots.\boldsymbol{y}_n$, $\boldsymbol{z} = \boldsymbol{z}_1.\ldots.\boldsymbol{z}_n$;*
- *ψ_i is of one of the following forms:*
  - *either $\alpha_i \mathrel{-\!\!\bullet} Q_i(\boldsymbol{u}_i, \boldsymbol{q}_i)$;*
  - *or $\boldsymbol{y}_j \simeq \boldsymbol{u}_i \circ \boldsymbol{z}_j \simeq \boldsymbol{q}_i$, if $\alpha_i = P_j(\boldsymbol{y}_j, \boldsymbol{z}_j)$ and $P_j = Q_i$.*

For instance $\mathtt{tree}(z,y) \mathrel{-\!\!\bullet} \mathtt{tree}(x,y)$ denotes an atom $P(x,z,y,y)$ with the rules:

$$P(x,z,y_1,y_2) \Leftarrow \exists x_1, x_2\, (x \overset{y_1}{\mapsto} (x_1, x_2) \circ P(x_1, z, y_1, y_2) \circ \mathtt{tree}(x_2, z, y_1))$$
$$P(x,z,y_1,y_2) \Leftarrow \exists x_1, x_2\, (x \overset{y_1}{\mapsto} (x_1, x_2) \circ \mathtt{tree}(x_1, z, y_1) \circ P(x_2, z, y_1, y_2))$$
$$P(x,z,y_1,y_2) \Leftarrow \exists x_1, x_2\, (x \overset{y_1}{\mapsto} (x_1, x_2) \circ x_1 \simeq z \circ y_1 \simeq y_2 \circ \mathtt{tree}(x_2, z, y_1))$$
$$P(x,z,y_1,y_2) \Leftarrow \exists x_1, x_2\, (x \overset{y_1}{\mapsto} (x_1, x_2) \circ \mathtt{tree}(x_1, z, y_1) \circ x_2 \simeq z \circ y_1 \simeq y_2)$$

For readability, all the expressions of the form $\mathtt{emp} \mathrel{-\!\!\bullet} \mathtt{tree}(x_2, z, y_1)$ have been replaced by $\mathtt{tree}(x_2, z, y_1)$. Note that the rules are not $\mathfrak{h}$-regular, as x_1 and x_2 do not occur as roots in every rule, but they can easily be transformed into $\mathfrak{h}$-regular rules by replacing x_1 and x_2 by z in the third and fourth rule, respectively (using the equations $x_1 \simeq z$ and $x_2 \simeq z$). The definition can be applied recursively (i.e., $P_0, \ldots, P_n$ may be derived predicates). The next proposition is an immediate consequence of Definition 17:

Proposition 18. *Let $\mathcal{R}$ be a $\mathfrak{h}$-regular SID. The rules associated with any predicate P corresponding to an expression $\alpha \mathrel{-\!\!\bullet} \delta$ (Definition 17) are $\mathfrak{h}$-regular, up to the following equivalence: $\exists x\,(x \simeq y \circ \phi) \equiv^{\mathfrak{P}}_{\mathcal{R}} \phi\{x \leftarrow y\}$. Moreover, the rules are also $\exists$-restricted, with $\gamma_{\mathcal{R}}(P) = \gamma_{\mathcal{R}}(P_0)$ and $P \in \mathcal{P}^\star \iff P_0 \in \mathcal{P}^\star$. Finally if $\alpha = \mathtt{emp}$ then $(\alpha \mathrel{-\!\!\bullet} \delta) = \delta$.*

Note that, however, the implication $P \in \mathcal{P}^\star \wedge Q \in \mathcal{P} \setminus \mathcal{P}^\star \implies P \bowtie_{\mathcal{R}} Q$ (Condition 4 in Definition 10) does *not* necessarily hold for derived predicates P, Q. The following lemma states a form of modus ponens, relating the connective $\circ$ with $\mathrel{-\!\!\bullet}$:

Lemma 19. *If $\mathcal{R}$ is $\mathfrak{h}$-regular then $P(\boldsymbol{x},\boldsymbol{p}) \circ ((P(\boldsymbol{x},\boldsymbol{p}) \circ \alpha) \mathrel{-\!\!\bullet} Q(\boldsymbol{y},\boldsymbol{q})) \models^{\mathfrak{P}}_{\mathcal{R}} \alpha \mathrel{-\!\!\bullet} Q(\boldsymbol{y},\boldsymbol{q})$.*

The next lemma states that every predicate atom allocating x can be written as a $\circ$-formula in which x occurs as a root.

Lemma 20. *Assume that $\mathcal{R}$ is $\exists$-restricted. Let $\boldsymbol{y},\boldsymbol{p}$ be vectors of location variables and permission terms, respectively. If $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} Q(\boldsymbol{y},\boldsymbol{p})$, $\mathfrak{s}(x) \neq \mathfrak{s}(\boldsymbol{y}|_1)$ and $\mathfrak{s}(x) \in dom(\mathfrak{h})$, then there exist atoms of the form $P(x,\boldsymbol{z},\boldsymbol{q})$, $P_i(x_i,\boldsymbol{y}_i,\boldsymbol{q}_i)$ (with $i \in \{1,\ldots,n\}$), where $\boldsymbol{z} \subseteq \boldsymbol{y} \cup \{x_1,\ldots,x_n\}$, $\boldsymbol{y}_i \subseteq \{\boldsymbol{y}|_j \mid j \notin \gamma_{\mathcal{R}}(Q)\}$, $\boldsymbol{q} \subseteq \boldsymbol{p}$ and $\boldsymbol{q}_i \subseteq \boldsymbol{p}$, such that: $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \exists x_1,\ldots,x_n\ (\beta \circ (\beta \mathrel{-\!\!\bullet} Q(\boldsymbol{y},\boldsymbol{p})))$, with $\beta = P(x,\boldsymbol{z},\boldsymbol{q}) \circ \bigcirc_{i=1}^{m} P_i(x_i,\boldsymbol{y}_i,\boldsymbol{q}_i)$. Moreover, $P_i \in \mathcal{P}^\star$, $\{x_1,\ldots,x_n\} \subseteq (x,\boldsymbol{z})|_{\gamma_{\mathcal{R}}(P)}$ and $y \in \boldsymbol{y} \cap \boldsymbol{z} \wedge y \notin \{\boldsymbol{y}|_j \mid j \notin \gamma_{\mathcal{R}}(Q)\} \implies y \in (x,\boldsymbol{z})|_{\gamma_{\mathcal{R}}(P)}$.*

Intuitively, since x is allocated and the rules are $\mathfrak{h}$-regular, then necessarily some predicate atom of the form $P(x,\boldsymbol{z},\boldsymbol{q})$ must be called at some point during the unfolding of the rules. Using $\mathrel{-\!\!\bullet}$, this predicate can be removed from the call tree of $Q(\boldsymbol{y},\boldsymbol{p})$ and lifted at the root level in the formula. The atom $P(x,\boldsymbol{z},\boldsymbol{q})$ may contain variables not occurring in $Q(\boldsymbol{y},\boldsymbol{p})$ corresponding to existential variables introduced by unfolding. As the rules are $\exists$-restricted, all such variables x_i must themselves appear as the root of some predicate atom $P_i(x_i,\boldsymbol{y}_i,\boldsymbol{q}_i)$ which contains (beside x_i) only variables occurring in $Q(\boldsymbol{y},\boldsymbol{p})$ (since $\gamma_{\mathcal{R}}(P_i) = \emptyset$, due to Condition 5 in Definition 10). Again, these atoms can be moved at the root level. See Appendix E for details.

Definition 21. *For all atoms $Q(\boldsymbol{y},\boldsymbol{p})$ we denote by $\delta[x]^+$ the set of formulas of the form $\exists x_1,\ldots,x_n\ (\beta \circ (\beta \mathrel{-\!\!\bullet} Q(\boldsymbol{y},\boldsymbol{p})))$ as defined in Lemma 20. We also denote by $\delta[x]^=$ the formula: $\delta \circ (x \simeq \boldsymbol{y}|_1)$.*

For every model of δ, $\delta[x]^-$ holds if x is not allocated in δ, $\delta[x]^=$ holds if x is equal to the root of δ and $\delta[x]^+$ holds if x is allocated but is not the root of δ. The following result follows immediately from Lemmata 19 and 20:

Lemma 22. *Assume that $\mathcal{R}$ is $\exists$-restricted. Let $x \in \mathcal{V}_1$. For every predicate atom δ such that $x \notin roots(\delta)$, and for all structures $(\mathfrak{s},\mathfrak{h})$: $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \delta$ iff there exists $\psi \in \{\delta[x]^-, \delta[x]^=\} \cup \delta[x]^+$ such that $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$.*

For instance the atom $\mathtt{lseg}(x, y, z)$ holds iff one of the formulas $\mathtt{lseg}(x, y, z) \circ x \simeq y$, $\mathtt{lseg}(x, y, z)[y]^-$ or $\mathtt{lseg}(y, y, z) \circ (\mathtt{lseg}(y, y, z) \mathrel{-\!\!\bullet} \mathtt{lseg}(x, y, z))$ holds. The second formula corresponds to the case where y is not allocated, and the first and third ones correspond to the case where there is a loop on y. By applying repeatedly Lemma 22 on every variable x and atom δ we eventually obtain a disjunction of normalized formulas:

Lemma 23. *Let $\mathcal{R}$ be a $\exists$-restricted SID. There exists an algorithm transforming any symbolic heap ϕ containing no points-to atom into a set of normalized formulas Ψ such that for all structures $(\mathfrak{s}, \mathfrak{h})$: $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$ iff there exists $\psi \in \Psi$ such that $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$. Furthermore, every formula in Ψ is a (quantified) separating conjunction of $\circ$-formulas.*

4.2 Commuting Separating and Disjoint Connections

The next step consists in showing that – under some particular conditions enforced by the previous transformation – the operator $*$ can be pushed innermost in the formula (below the operator $\circ$). To this aim, we exploit an essential property of $\mathfrak{h}$-regular SIDs, namely that all the locations that occur in the heap of some model of a formula ϕ but are not allocated correspond to a variable in $fv(\phi)$. We shall denote by $cut(L, L', \mathfrak{h})$ the set of locations reachable from L in $\mathfrak{h}$, from a path not crossing L':

Definition 24. *Let $\mathfrak{h}$ be a heap, let $L, L' \subseteq \mathcal{L}$. We denote by $cut(L, L', \mathfrak{h})$ the set of locations inductively defined as follows: $L \subseteq cut(L, L', \mathfrak{h})$, and if $\ell' \in cut(L, L', \mathfrak{h})$, $\mathfrak{h}(\ell') = (\ell_1, \ldots, \ell_k, \pi)$, $i \in \{1, \ldots, k\}$ and $\ell_i \notin L'$ then $\ell_i \in cut(L, L', \mathfrak{h})$.*

The following lemma characterizes the domain of the part of the heap satisfying some formula ϕ:

Lemma 25. *Let $\mathcal{R}$ be a $\mathfrak{h}$-regular SID and let ϕ be a $\circ$-formula containing no quantifier. Let $\mathfrak{s}$ be a store and let $\mathfrak{h}, \mathfrak{h}'$ be heaps, with $\mathfrak{h}' \leq \mathfrak{h}$. Let V be a set of variables, with $fv(\phi) \subseteq V \cup roots(\phi)$ and $\mathfrak{s}(V) \cap dom(\mathfrak{h}') = \emptyset$. If $(\mathfrak{s}, \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} \phi$ then $dom(\mathfrak{h}') = cut(\mathfrak{s}(roots(\phi)), \mathfrak{s}(V), \mathfrak{h})$.*

The commutation property, pushing $*$ below $\circ$, is given by Lemma 26:

Lemma 26. *Let $\mathcal{R}$ be a $\mathfrak{h}$-regular SID. Let $V \subseteq \mathcal{V}_1$ and let ϕ be a normalized formula, of the form $\phi = \phi' \circ (\mathbin{\ast}_{i=1}^{n} (\phi_i \circ \psi_i) * \psi')$, where, for all $i \in \{1, \ldots, n\}$, $roots(\phi_i) = V$ and $(roots(\psi_i) \cup roots(\psi')) \cap V = \emptyset$. Then ϕ is $(\mathcal{R}, \mathfrak{P})$-satisfiable iff $(\phi' \circ \mathbin{\ast}_{i=1}^{n} \phi_i) \circ ((\mathbin{\ast}_{i=1}^{n} \psi_i) * \psi')$ is $(\mathcal{R}, \mathfrak{P})$-satisfiable.*

Roughly speaking, as $roots(\phi_i) = V$ and ϕ_i is normalized, it is possible to prove, using the characterization given in Lemma 25, that the parts of the heap that correspond to the formulas ϕ_i have all the same domain. This entails that the heaps corresponding to the formulas ψ_i and $\phi_{i'}$ are disjoint, which permits to prove that $\mathbin{\ast}_{i=1}^{n} (\phi_i \circ \psi_i)$ can be written $(\mathbin{\ast}_{i=1}^{n} \phi_i) \circ (\mathbin{\ast}_{i=1}^{n} \psi_i)$, yielding the result. The detailed proof is given in Appendix H.

4.3 Merging of Spatial Predicates

We show that, under some particular conditions, it is possible to replace the separating conjunction of two spatial atoms having the same root by a single spatial atom. The rules defining this atom are obtained by combining the rules of the two initial atoms. More precisely, consider any $\mathfrak{h}$-regular SID $\mathcal{R}$ and two spatial atoms $P(x,\boldsymbol{y},\boldsymbol{p})$ and $P'(x,\boldsymbol{y'},\boldsymbol{p'})$ sharing the same root x, where $\boldsymbol{y},\boldsymbol{y'}$ are vectors of location variables and $\boldsymbol{p}$ and $\boldsymbol{p'}$ are vectors of permission terms. We denote by $P(x,\boldsymbol{y},\boldsymbol{p})\triangledown P'(x,\boldsymbol{y'},\boldsymbol{p'})$ any atom $Q(x,\boldsymbol{y},\boldsymbol{y'},\boldsymbol{p},\boldsymbol{p'})$ where Q is associated with rules of the form:

$$Q(v,\boldsymbol{w},\boldsymbol{w'},\boldsymbol{z},\boldsymbol{z'}) \Leftarrow \exists u_1,\ldots,u_n \quad v \overset{q}{\mapsto} (v_1,\ldots,v_k) \circ \bigcirc_{i=1}^{n}(Q_i(u_i,\boldsymbol{y}_i,\boldsymbol{q}_i)\triangledown Q'_i(u_i,\boldsymbol{y'_i},\boldsymbol{q_i'})) \circ \phi \circ \phi' \circ \psi$$

with $q \overset{\text{def}}{=} p \oplus p'$, for all pairs of rules of the following forms in $\mathcal{R}$ (with the same numbers k and n, and up to α-renaming, so that the rules share the same existential variables):

$$\begin{array}{ll} P(v,\boldsymbol{w},\boldsymbol{z}) & \Leftarrow \exists u_1,\ldots,u_n \quad v \overset{p}{\mapsto} (v_1,\ldots,v_k) \circ \bigcirc_{i=1}^{n} Q_i(u_i,\boldsymbol{y}_i,\boldsymbol{q}_i) \circ \phi \\ P'(v,\boldsymbol{w'},\boldsymbol{z'}) & \Leftarrow \exists u_1,\ldots,u_n \quad v \overset{p'}{\mapsto} (v'_1,\ldots,v'_k) \circ \bigcirc_{i=1}^{n} Q'_i(u_i,\boldsymbol{y'_i},\boldsymbol{q'_i}) \circ \phi' \end{array}$$

where $\psi = \bigcirc_{i=1}^{k}(v_i \simeq v'_i)$. Note that all the produced rules are $\mathfrak{h}$-regular[8].

Lemma 27. *Let $\mathcal{R}$ be a $\mathfrak{h}$-regular SID. Let $x \in \mathcal{V}_1$ and let $(\mathfrak{s},\mathfrak{h})$ be a structure such that $\mathfrak{s}(y) \notin dom(\mathfrak{h})$ holds for all variables y such that $\mathfrak{s}(x) \neq \mathfrak{s}(y)$. Then $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(x,\boldsymbol{y},\boldsymbol{p})\triangledown P'(x,\boldsymbol{y'},\boldsymbol{p'}) \iff (\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(x,\boldsymbol{y},\boldsymbol{p}) * P'(x,\boldsymbol{y'},\boldsymbol{p'})$.*

The result crucially depends on the fact that the parts of the heap that correspond to $P(x,\boldsymbol{y},\boldsymbol{p})$ and $P'(x,\boldsymbol{y'},\boldsymbol{p'})$ respectively must share the same domain, since otherwise, as $\mathcal{R}$ is $\mathfrak{h}$-regular, a free variable would be allocated, contradicting the hypothesis. This ensures that the heap can be generated by the above rules(see Appendix I for details).

4.4 Heap Abstractions and Main Result

As we shall see later, the previous transformations can be used to transform any symbolic heap into a $\circ$-formula (while preserving satisfiability). The final step is to devise an algorithm to test the satisfiability of $\circ$-formulas. As it is done in [6] for standard heap models, the algorithm works by constructing relevant abstractions of the models of the predicate atoms. It suffices to keep track of the truth value of the equational atoms, of the allocated variables and of the permission atoms satisfied by the structure.

[8] However $\exists$-restrictedness is not necessarily preserved.

Definition 28. *A* heap abstraction *is a tuple* $\mathfrak{a} = (V_\mathfrak{a}, \sim_\mathfrak{a}, A_\mathfrak{a}, \rho_\mathfrak{a})$ *where* $V_\mathfrak{a}$ *is a finite set of variables,* $\sim_\mathfrak{a}$ *is an equivalence relation on the variables of sort* $\mathtt{l}$ *occurring in* $V_\mathfrak{a}$*,* $A_\mathfrak{a}$ *is a subset of* $V_\mathfrak{a} \cap \mathcal{V}_\mathtt{l}$*, closed under* $\sim_\mathfrak{a}$ *(i.e., for all* $x, y \in \mathcal{V}_\mathtt{l}$*:* $x \in A_\mathfrak{a} \wedge x \sim_\mathfrak{a} y \implies y \in A_\mathfrak{a}$*), and* $\rho_\mathfrak{a}$ *is a permission formula (with variables in* $V_\mathfrak{a}$*).*

Definition 29. *Let* $(\mathfrak{s}, \mathfrak{h})$ *be a structure and let* $\mathfrak{a} = (V_\mathfrak{a}, \sim_\mathfrak{a}, A_\mathfrak{a}, \rho_\mathfrak{a})$ *be a heap abstraction. We write* $(\mathfrak{s}, \mathfrak{h}) \models^\mathfrak{P} \mathfrak{a}$ *if all the following conditions are satisfied: (i) For all variables* $x, y \in V_\mathfrak{a} \cap \mathcal{V}_\mathtt{l}$*:* $x \sim_\mathfrak{a} y \iff \mathfrak{s}(x) = \mathfrak{s}(y)$*; (ii) for all* $x \in V_\mathfrak{a} \cap \mathcal{V}_\mathtt{l}$*,* $x \in A_\mathfrak{a} \iff \mathfrak{s}(x) \in dom(\mathfrak{h})$*; and (iii)* $\mathfrak{s} \models^\mathfrak{P} \rho_\mathfrak{a}$*. A heap abstraction is* $\mathfrak{P}$-satisfiable *if there exists a structure* $(\mathfrak{s}, \mathfrak{h})$ *such that* $(\mathfrak{s}, \mathfrak{h}) \models^\mathfrak{P} \mathfrak{a}$*.*

Proposition 30. *A heap abstraction* $\mathfrak{a}$ *is* $\mathfrak{P}$*-satisfiable iff* $\rho_\mathfrak{a}$ *is* $\mathfrak{P}$*-satisfiable.*

For all $\circ$-formulas ϕ, we define a set of heap abstractions $\mathfrak{A}(\phi)$ by mutual induction as follows. The sets $\mathfrak{A}(\phi)$ are the least sets of heap abstractions satisfying the following properties, for all finite sets of variables[9] $V \supseteq fv(\phi)$ and for all equivalence relations $\sim$ on $V \cap \mathcal{V}_\mathtt{l}$: (i) if $\phi = x \overset{p}{\mapsto} (y_1, \ldots, y_n)$ then $(V, \sim, \{y \mid y \mid y \sim x\}, def(p)) \in \mathfrak{A}(\phi)$. (ii) if $\phi = x \simeq y$ (resp. $x \not\simeq y$) with $x, y \in \mathcal{V}_\mathtt{l}$ and $x \sim y$ (resp. $x \not\sim y$) then $(V, \sim, \emptyset, \mathtt{emp}) \in \mathfrak{A}(\phi)$; (iii) if ϕ is a permission formula then $(V, \sim, \emptyset, \phi) \in \mathfrak{A}(\phi)$; (iv) if $\phi = \exists x\, \psi$, $(V, \sim, A, \rho) \in \mathfrak{A}(\psi)$ then $(V \setminus \{x\}, \sim', A \setminus \{x\}, \rho) \in \mathfrak{A}(\phi)$, where $\sim'$ denotes the restriction of $\sim$ to the variables distinct from x, i.e., $\sim' \overset{\text{def}}{=} \{(u, v) \mid u \sim v \wedge u, v \neq x\}$ (note that x cannot occur in ρ, since quantification over permission variables is not allowed); (v) if $\phi = \phi_1 \circ \phi_2$, $(V, \sim, A_i, \rho_i) \in \mathfrak{A}(\phi_i)$ (for all $i = 1, 2$) with $A_1 \cap A_2 = \emptyset$, then $(V, \sim, A_1 \cup A_2, \rho_1 \circ \rho_2) \in \mathfrak{A}(\phi)$; (vi) if $\phi = P(\boldsymbol{x}, \boldsymbol{p})$ and $\phi \Leftarrow_\mathcal{R} \xi$ then $\mathfrak{A}(\xi) \subseteq \mathfrak{A}(\phi)$.

Lemma 31. *A* $\circ$*-formula* ϕ *is* $(\mathcal{R}, \mathfrak{P})$*-satisfiable iff at least one of the abstractions in* $\mathfrak{A}(\phi)$ *is* $\mathfrak{P}$*-satisfiable.*

Putting things together we get the following result:

Theorem 32. *If* $\mathfrak{P}$*-satisfiability is decidable for permission formulas, then there exists an algorithm that, for every* $\exists$*-restricted SID, decides whether a given formula* ϕ *is* $(\mathcal{R}, \mathfrak{P})$*-satisfiable. If, moreover,* $\mathfrak{P}$*-satisfiability is in* EXPTIME*, then* $(\mathcal{R}, \mathfrak{P})$*-satisfiability is also in* EXPTIME *(for* $\exists$*-restricted SID). Finally, for every permission model* $\mathfrak{P}$*,* $(\mathcal{R}, \mathfrak{P})$*-satisfiability is* EXPTIME*-hard (for* $\exists$*-restricted SID).*

5 Using Separating Conjunctions Inside Rules

To end the paper, we wish to point out that the satisfiability problem is undecidable from $\exists$-restricted SID if the disjoint separation $\circ$ is replaced by the standard

[9] For technical convenience we do not impose any bound on the cardinality of V, hence the set $\mathfrak{A}(\phi)$ is infinite. This simplifies the theoretical definition of the abstraction for disjoint conjunctions. In practice only variables occurring in the initial formula or in the rules need to be considered.

separating connective $*$ in the inductive definitions (see Definition 7). We think that the result is of some theoretical interest, although, as explained above, rules using $\circ$ are actually more convenient for describing data structures. The notions of $*$-$\mathfrak{h}$-*regular* and $*$-$\exists$-*restricted* SID are defined exactly as $\mathfrak{h}$-regular SID and $\exists$-restricted SID (Definitions 7 and 10) except that the symbol $\circ$ is replaced by $*$ everywhere (for conciseness the formal definitions are omitted).

Theorem 33. *Let $\mathfrak{P}$ be any permission model and assume that for every $n \in \mathbb{N}$, there exists $\pi \in \mathcal{P}_{\mathfrak{P}}$ such that π^n is defined. The $(\mathcal{R}, \mathfrak{P})$-satisfiability problem is undecidable for $*$-$\exists$-restricted SID.*

6 Conclusion and Future Work

An algorithm was devised to test the satisfiability of symbolic heaps in Separation Logic with inductively defined predicates and permissions, under some (syntactic) conditions on the inductive rules giving the semantics of the spatial predicates. The algorithm runs in exponential time, provided the satisfiability of permission formulas is in EXPTIME. In addition, we showed that some natural relaxings of these conditions make the problem undecidable (under some minimal assumptions on the permission model). The next step is to investigate the entailment problem for the considered fragment. The techniques devised in the present paper for transforming symbolic heaps into disjoint conjunctions of atoms should serve as a basis for this purpose, but the extension is not straightforward. Another (much easier) extension that could be of practical relevance is to consider formulas with labels (in the sense of [5]) which allow one to express additional equality conditions on some parts of the structures. In our context, labels would simply yield additional conditions on the decomposition generated during the normalization step: two formulas sharing the same label should be decomposed into formulas with the same set of roots. It could also be interesting to relax some of the conditions on the rules, for instance to allow for existential variables not occurring as roots in the rules. This is required to encode data structures with forward pointers, such as skip lists. It is also unclear whether Condition 4 in Definition 10 is required for decidability. Finally, the decision algorithm could probably be extended to handle arbitrary combinations of disjoint and separating conjunctions.

Acknowledgments. This work has been partially funded by the the French National Research Agency (`ANR-21-CE48-0011`)

References

1. J. Berdine, C. Calcagno, and P. W. O'Hearn. Smallfoot: Modular automatic assertion checking with separation logic. In F. S. de Boer, M. M. Bonsangue, S. Graf, and W. P. de Roever, editors, *Formal Methods for Components and Objects, 4th International Symposium, FMCO 2005, Amsterdam, The Netherlands, November 1-4, 2005, Revised Lectures*, volume 4111 of *Lecture Notes in Computer Science*, pages 115–137. Springer, 2005.

2. J. Berdine, B. Cook, and S. Ishtiaq. Slayer: Memory safety for systems-level code. In G. G. andShaz Qadeer, editor, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *LNCS*, pages 178–183. Springer, 2011.
3. R. Bornat, C. Calcagno, P. W. O'Hearn, and M. J. Parkinson. Permission accounting in separation logic. In J. Palsberg and M. Abadi, editors, *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2005, Long Beach, California, USA, January 12-14, 2005*, pages 259–270. ACM, 2005.
4. J. Boyland. Fractional permissions. In D. Clarke, J. Noble, and T. Wrigstad, editors, *Aliasing in Object-Oriented Programming. Types, Analysis and Verification*, volume 7850 of *Lecture Notes in Computer Science*, pages 270–288. Springer, 2013.
5. J. Brotherston, D. Costa, A. Hobor, and J. Wickerson. Reasoning over permissions regions in concurrent separation logic. In S. K. Lahiri and C. Wang, editors, *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II*, volume 12225 of *Lecture Notes in Computer Science*, pages 203–224. Springer.
6. J. Brotherston, C. Fuhs, J. A. N. Pérez, and N. Gorogiannis. A decision procedure for satisfiability in separation logic with inductive predicates. In T. A. Henzinger and D. 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.
7. C. Calcagno and D. Distefano. Infer: An automatic program verifier for memory safety of C programs. In M. G. Bobaru, K. Havelund, G. J. Holzmann, and R. Joshi, editors, *NASA Formal Methods - Third International Symposium, NFM 2011, Pasadena, CA, USA, April 18-20, 2011. Proceedings*, volume 6617 of *Lecture Notes in Computer Science*, pages 459–465. Springer, 2011.
8. C. Calcagno, P. W. O'Hearn, and H. Yang. Local action and abstract separation logic. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*, pages 366–378. IEEE Computer Society, 2007.
9. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 1997.
10. S. Demri, É. Lozes, and D. Lugiez. On symbolic heaps modulo permission theories. In S. V. Lokam and R. Ramanujam, editors, *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*, volume 93 of *LIPIcs*, pages 25:1–25:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
11. M. Echenim, R. Iosif, and N. Peltier. Entailment checking in separation logic with inductive definitions is 2-exptime hard. In E. Albert and L. 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.
12. M. Echenim, R. Iosif, and N. Peltier. Decidable entailments in separation logic with inductive definitions: Beyond establishment. In *CSL 2021: 29th International Conference on Computer Science Logic*, EPiC Series in Computing. EasyChair, 2021.
13. R. Iosif, A. Rogalewicz, and J. Simacek. The tree width of separation logic with recursive definitions. In *Proc. of CADE-24*, volume 7898 of *LNCS*, 2013.

14. S. S. Ishtiaq and P. W. O'Hearn. Bi as an assertion language for mutable data structures. In *ACM SIGPLAN Notices*, volume 36, pages 14–26, 2001.
15. J. Katelaan and F. Zuleger. Beyond symbolic heaps: Deciding separation logic with inductive definitions. In E. Albert and L. 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 390–408. EasyChair, 2020.
16. Q. L. Le. Compositional satisfiability solving in separation logic. In F. Henglein, S. Shoham, and Y. Vizel, editors, *Verification, Model Checking, and Abstract Interpretation - 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17-19, 2021, Proceedings*, volume 12597 of *Lecture Notes in Computer Science*, pages 578–602. Springer, 2021.
17. K. Nakazawa, M. Tatsuta, D. Kimura, and M. Yamamura. Cyclic Theorem Prover for Separation Logic by Magic Wand. In *ADSL 18 (First Workshop on Automated Deduction for Separation Logics)*, July 2018. Oxford, United Kingdom.
18. P. W. O'Hearn and D. J. Pym. The logic of bunched implications. *Bull. Symb. Log.*, 5(2):215–244, 1999.
19. J. A. N. Pérez and A. Rybalchenko. Separation logic modulo theories. In C. Shan, editor, *Programming Languages and Systems - 11th Asian Symposium, APLAS 2013, Melbourne, VIC, Australia, December 9-11, 2013. Proceedings*, volume 8301 of *LNCS*, pages 90–106. Springer, 2013.
20. R. Piskac, T. Wies, and D. Zufferey. Automating separation logic using SMT. In N. Sharygina and H. Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *LNCS*, pages 773–789. Springer, 2013.
21. X. Qiu, P. Garg, A. Stefanescu, and P. Madhusudan. Natural proofs for structure, data, and separation. In H. Boehm and C. Flanagan, editors, *ACM SIGPLAN PLDI '13*, pages 231–242. ACM, 2013.
22. J. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *Proc. of LICS'02*, 2002.
23. Z. Xu, T. Chen, and Z. Wu. Satisfiability of compositional separation logic with tree predicates and data constraints. In L. de Moura, editor, *CADE 26*, volume 10395 of *LNCS*, pages 509–527. Springer, 2017.

A Proof of Proposition 8

The proof is by induction on the satisfiability relation. ϕ cannot be purely-spatial as $\mathit{roots}(\phi)$ would be empty, contradicting the hypothesis of the proposition. If ϕ is a points-to atom then by definition of $\mathit{roots}(\phi)$ if must be of the form $\phi = x \overset{p}{\mapsto} (y_1, \ldots, y_k)$, so that $\mathit{dom}(\mathfrak{h}) = \{\mathfrak{s}(x)\}$ by definition of the semantics of points-to atoms. If ϕ is a spatial predicate atom then by definition if must be of the form $P(x, y_1, \ldots, y_k)$, and by definition of the semantics of predicate atoms we have $\phi \Leftarrow_{\mathcal{R}} \psi$ with $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$. Since $\mathcal{R}$ is $\mathfrak{h}$-regular, ψ is of the form $\exists \boldsymbol{u}(x \overset{p}{\mapsto} (y'_1, \ldots, y'_k) \circ \psi')$, with $x \notin \boldsymbol{u}$, so that $x \in \mathit{roots}(\psi)$, and we get $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h})$ by the induction hypothesis. If $\phi = \exists y\, \psi$ then by definition $x \in \mathit{roots}(\psi)$ and $x \neq y$, moreover, there exists a location ℓ such that $\mathfrak{s}\{y \leftarrow \ell\} \models^{\mathfrak{P}}_{\mathcal{R}} \psi$. By the induction hypothesis we get $\mathfrak{s}\{y \leftarrow \ell\}(x) \in \mathit{dom}(\mathfrak{h})$, thus $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h})$ as $x \neq y$. If $\phi = \phi_1 \circ \phi_2$ or $\phi = \phi_1 * \phi_2$ then there exists $i \in \{1, 2\}$ such that $x \in \mathit{roots}(\phi_i)$. Moreover, there exist heaps $\mathfrak{h}_i$ (for $i = 1, 2$) with $\mathfrak{h} = \mathfrak{h}_1 \sqcup \mathfrak{h}_2$ and $(\mathfrak{s}, \mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$. By the induction hypothesis we get $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h}_i)$, thus $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h})$, as $\mathit{dom}(\mathfrak{h}) = \mathit{dom}(\mathfrak{h}_1) \cup \mathit{dom}(\mathfrak{h}_2)$.

B Proof of Theorem 9

The proof is by reduction from the Post Correspondence Problem (PCP), that is well-known to be undecidable. Consider a natural number n and two sequences of words $\mu_1, \ldots, \mu_n$ and $\nu_1, \ldots, \nu_n$. A *potential witness* is a word ω such that for all $\lambda \in \{\mu, \nu\}$, there exists a sequence $I^\lambda(1), \ldots, I^\lambda(\kappa^\lambda)$ with $\omega = \lambda_{I^\lambda(1)} \ldots \lambda_{I^\lambda(\kappa^\lambda)}$ and $\kappa^\lambda > 0$. It is a *witness* if, moreover, $\kappa^\mu = \kappa^\nu = \kappa$ and $I^\mu(i) = I^\nu(i)$ for all $i \in \{1, \ldots, \kappa\}$. The PCP consists in determining whether a witness exists for two given sequences of words. We assume, to ease the encoding, that the solution sequence $I^\lambda(1), \ldots, I^\lambda(\kappa^\lambda)$ ends with a dummy index 0, with $\mu_0 = \nu_0 = \#$ (this entails that we must have $\kappa^\lambda > 1$). For every potential witness w, for every $\lambda \in \{\mu, \nu\}$ and for every $i \in \{1, \ldots, \kappa^\lambda\}$, we denote by $\zeta^\lambda(i)$ the position of the word $\lambda_{I^\lambda(i)}$ inside w, formally defined as follows: $\zeta^\lambda(i) \overset{\text{def}}{=} |\lambda_{I^\lambda(1)} \ldots . \lambda_{I^\lambda(i-1)}| + 1$. For every $j \in \{1, \ldots, |w|\}$, we also denote by $\eta^\lambda(j)$ the element $i \in \{1, \ldots, \kappa^\lambda\}$ such that $\omega|_j$ occurs in $\lambda_{I^\lambda(i)}$, formally defined as follows: $\eta^\lambda(j) = i$ if $\zeta^\lambda(i) \leq j < \zeta^\lambda(i+1)$.

For simplicity, we assume that all the natural numbers in $\{1, \ldots, n\}$ are taken as variables of sort $\mathtt{1}$. Under this assumption, a potential witness ω may be represented by a structure $(\mathfrak{s}, \mathfrak{h})$ defined as follows: $\mathfrak{h} = \mathfrak{h}^\mu \sqcup \mathfrak{h}^\nu \sqcup \mathfrak{h}'$, with $\mathfrak{h}' \overset{\text{def}}{=} \{(\ell'_j, {\ell^\mu}_{\eta^\mu(j)}, {\ell^\nu}_{\eta^\nu(j)}, \ell'_{j+1}, \pi_1) \mid 1 \leq j \leq |\omega|\}$ and for all $\lambda \in \{\mu, \nu\}$: $\mathfrak{h}^\lambda \overset{\text{def}}{=} \{(\ell^\lambda{}_{i+1}, \mathfrak{s}(I^\lambda(i)), \ell^\lambda{}_i, \pi_1) \mid 1 \leq i \leq \kappa\}$. Intuitively, every heap $\mathfrak{h}^\lambda$ encodes the sequence $I^\lambda(\kappa^\lambda), \ldots, I^\lambda(1)$ as a linked list in which the i-th element refers to (the image of) $I^\lambda(i)$ (note that the list is reversed, the reason will become clear later). Then $\mathfrak{h}'$ encodes the word ω as a list where the j-nth element of the list refers to the cells ${\ell^\mu}_{\eta^\mu(j)}$ and ${\ell^\nu}_{\eta^\nu(j)}$ in the lists $\mathfrak{h}^\mu$ and $\mathfrak{h}^\nu$ that correspond to the words in which $\omega|_j$ occurs.

Let $\boldsymbol{v}$ be a sequence of variables containing all numbers of $\{1,\dots,n\}$, as well as a special variable $\mathtt{nil}$ (marking the end of the lists), and variables u_μ and u_ν denoting the first cell in the list $\mathfrak{h}^\lambda$ and $\mathfrak{h}^\nu$. It is straightforward to check that the following rules generate structures of the above form (for all $i_\mu, i_\nu, i'_\mu, i'_\nu \in \{1,\dots,n\}$ and for all $j_\mu, j_\nu \in \{1,\dots,M+1\}$, where M denotes the maximal length of the word in the sequences μ or ν).

Intuitively, P allocates the first element of the potential witness, by guessing the first element i in the solution sequence[10], then call $P_{2,2,i,i}$. The predicate $P_{j_\mu,j_\nu,i_\mu,i_\nu}$ allocates the next elements, starting from a character occurring simultaneously at position j_μ in the word i_μ in the sequence μ and at position j_ν in the word i_ν in the sequence ν (thus initially $i_\mu = i_\nu = i$ and $j_\mu = j_\nu = 2$, as the first character of the first word has already been allocated by the predicate P). Conditions are added on the rules to ensure that the two characters are identical in both sequences. Each time one reaches the start of a new word $I^\lambda(i)$ in the witness, $Q(y, y', i, z)$ is called to allocate the elements of the list $\mathfrak{h}^\lambda$, where y' is a pointer to the *previous* element in the list $I^\lambda(1),\dots,I^\lambda(\kappa^\lambda)$ (thus initially $y' = \mathtt{nil}$). If the two words end on the same character then this means that we have reached the end of the potential witness[11], and we only have to allocate the locations corresponding to the dummy index 0 and the dummy character #. Note that the locations corresponding to 0 are associated with the variables u_μ and u_ν. The list must be constructed in reverse order to ensure that the obtained rules are $\mathfrak{h}$-regular (if the lists were constructed in the usual order, then the pointer to the next cell could not be allocated before one gets to the next word, hence the corresponding rule would not be $\mathfrak{h}$-regular, as one would need to introduce an existential variable without immediately allocating it).

$$P(x,\boldsymbol{v},z) \Leftarrow \exists x', y_\mu, y_\nu \quad x \overset{z}{\mapsto} (y_\mu, y_\nu, x') \circ Q(y_\mu, \mathtt{nil}, i, z) \circ Q(y_\nu, \mathtt{nil}, i, z) \circ P_{2,2,i,i}(x', y_\mu, y_\nu, \boldsymbol{v}, z) \quad \text{if } \mu_i|_1 = \nu_i|_1 \tag{1}$$

$$Q(y, y', i, z) \Leftarrow y \overset{z}{\mapsto} (i, y') \tag{2}$$

$$P_{j_\mu,j_\nu,i_\mu,i_\nu}(x, y_\mu, y_\nu, \boldsymbol{v}, z) \Leftarrow \exists x' \quad x \to (y_\mu, y_\nu, x', z) \circ P_{j_\mu+1,j_\nu+1,i_\mu,i_\nu}(x', y_\mu, y_\nu, \boldsymbol{v}, z) \quad \text{if } j_\mu \le |\mu_{i_\mu}|,\ j_\nu \le |\nu_{i_\nu}| \text{ and } \mu_{i_\mu}|_{j_\mu} = \nu_{i_\nu}|_{j_\nu} \tag{3}$$

$$P_{j_\mu,j_\nu,i_\mu,i_\nu}(x, y_\mu, y_\nu, \boldsymbol{v}, z) \Leftarrow \exists x, y'_\mu \quad x \to (y'_\mu, y_\nu, x', z) \circ Q(y'_\mu, i'_\mu, y_\mu, z) \circ P_{2,j_\nu+1,i'_\mu,i_\nu}(x', y'_\mu, y_\nu, \boldsymbol{v}, z) \quad \text{if } j_\mu = |\mu_{i_\mu}| + 1,\ j_\nu \le |\nu_{i_\nu}| \text{ and } \mu_{i'_\mu}|_1 = \nu_{i_\nu}|_{j_\nu} \tag{4}$$

[10] For simplicity we only encode potential witnesses where the two sequences of indices start with the same index, i.e., $I^\mu(1) = I^\nu(1)$. It is clear that this is not restrictive as we eventually want to encode witnesses.

[11] We assume, w.l.o.g., that only witnesses of minimal length are considered.

$$P_{j_\mu,j_\nu,i_\mu,i_\nu}(x,y_\mu,y_\nu,\boldsymbol{v},z) \Leftarrow \exists x,y'_\nu \quad x \to (y_\mu,y'_\nu,x',z) \circ Q(y'_\nu,i'_\nu,y_\nu,z) \circ P_{j_\mu+1,2,i_\mu,i'_\nu}(x',y_\mu,y'_\nu,\boldsymbol{v},z) \text{ if } j_\mu \leq |\mu_{i_\mu}|,\ j_\nu = |\nu_{i_\nu}|+1 \text{ and and } \mu_{i_\mu}|_{j_\mu} = \nu_{i'_\nu}|_1 \quad (5)$$

$$P_{j_\mu,j_\nu,i_\mu,i_\nu}(x,y_\mu,y_\nu,\boldsymbol{v},z) \Leftarrow \exists u',u'' \, x \to (u',u'',\mathtt{nil},z) \circ Q(u',y_\mu,0,z) \circ Q(u'',y_\nu,0,z) \circ u' \simeq u_\mu \circ u'' \simeq u_\nu \text{ if } j_\mu = |\mu_{i_\mu}|+1 \text{ and } j_\nu = |\nu_{i_\nu}|+1 \quad (6)$$

To ensure that the considered potential witness is indeed a witness, it only remains to check that the sequences Q^μ and Q^ν are identical. To this aim, we introduce a predicate that generates structures of the form $\mathfrak{h}'' \sqcup \mathfrak{h}^\mu \sqcup \mathfrak{h}^\nu$ with $\mathfrak{h}'' \stackrel{\text{def}}{=} \{(\ell''_i, {\ell^\mu}_i, {\ell^\nu}_i, \ell''_{i+1}, \pi_2) \mid i \in \{1,\ldots,\kappa\}\}$, ${\mathfrak{h}^\lambda}' \stackrel{\text{def}}{=} \{({\ell^\lambda}_{i+1}, \mathfrak{s}(I^\lambda(i)), {\ell^\lambda}_i, \pi_2) \mid 1 \leq i \leq \kappa\}$ (${\mathfrak{h}^\lambda}'$ is identical to $\mathfrak{h}^\lambda$, except for the permissions) and $\mathfrak{s}(I^\lambda) = \mathfrak{s}(I^\nu)$, for all $i \in \{1,\ldots,\kappa\}$.

$$R(x,y_\mu,y_\nu,\boldsymbol{v},z) \Leftarrow \exists x',y'_\mu,y'_\nu \quad x \overset{z}{\mapsto} (y'_\mu,y'_\nu,x') \circ R(x',y'_\mu,y'_\nu,z) \circ Q(y'_\mu,y_\mu,i,z) \circ Q(y'_\nu,y_\nu,i,z) \quad (7)$$

$$R(x,y_\mu,y_\nu,\boldsymbol{v},z) \Leftarrow \exists u',u'' x \overset{z}{\mapsto} (u',u'',\mathtt{nil}) \circ Q(u',y_\mu,i,z) \circ Q(u'',y_\nu,i,z) \circ u' \simeq u_\mu \circ u'' \simeq u_\nu \quad (8)$$

It is clear that the formula $P(x_1,\boldsymbol{v},z_1) \circ R(x_2,\mathtt{nil},\mathtt{nil},\boldsymbol{v},z_2)$ is $(\mathcal{R},\mathfrak{P})$-satisfiable iff the considered instance of the PCP admits a solution.

C Proof of Lemma 15

The proof is by induction on $|\mathfrak{h}|$.

– Assume that $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(\boldsymbol{x},\boldsymbol{p})[v]^-$. By definition of the semantics of predicate atoms, there exists a formula ψ such that $P(\boldsymbol{x},p)[v]^- \Leftarrow_{\mathcal{R}} \psi$ and $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$. By definition of the rules defining $P(\boldsymbol{x},p)[v]^-$ and of the relation $\Leftarrow_{\mathcal{R}}$, this entails that $\mathcal{R}$ contains a rule $P(\boldsymbol{y},\boldsymbol{z}) \Leftarrow \exists \boldsymbol{u}\,(\bigcirc_{i=1}^m Q_i(\boldsymbol{y}_i,\boldsymbol{p}_i) \circ \phi)$ and we have:

$$\psi = \exists \boldsymbol{u}\,(\bigcirc_{i=1}^m Q_i(\boldsymbol{y}_i,\boldsymbol{p}_i)[w]^- \circ \phi \circ \boldsymbol{y}|_1 \not\simeq w)\sigma$$

with $\sigma = \{\boldsymbol{y} \leftarrow \boldsymbol{x}, w \leftarrow v, \boldsymbol{z} \leftarrow \boldsymbol{p}\}$. Again by definition of the semantics, there exist a sequence of locations $\boldsymbol{\ell}$ and disjoint heaps $\mathfrak{h}_0,\ldots,\mathfrak{h}_m$ such that $\mathfrak{h} = \mathfrak{h}_0 \sqcup \ldots \sqcup \mathfrak{h}_m$, $(\mathfrak{s}',\mathfrak{h}_0) \models^{\mathfrak{P}}_{\mathcal{R}} \phi\sigma$, $\mathfrak{s}'(\boldsymbol{y}|_1\sigma) \neq \mathfrak{s}'(w\sigma)$ and $(\mathfrak{s}',\mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q_i(\boldsymbol{y}_i,\boldsymbol{p})[w]^-\sigma = Q_i(\boldsymbol{y}_i\sigma,\boldsymbol{p}\sigma)[v]^-$ (for all $i \in \{1,\ldots,m\}$), with $\mathfrak{s}' = \mathfrak{s}\{\boldsymbol{u} \leftarrow \boldsymbol{\ell}\}$. Since $\mathcal{R}$ is $\mathfrak{h}$-regular, ϕ contains exactly one points-to atom with left-hand side $\boldsymbol{y}|_1$, hence $dom(\mathfrak{h}_0) = \{\mathfrak{s}'(\boldsymbol{y}|_1\sigma)\} = \{\mathfrak{s}(\boldsymbol{x}|_1)\}$. Thus $\mathfrak{h}_0 \neq \emptyset$ and $|\mathfrak{h}_i| < |\mathfrak{h}|$ (for all $i \in \{1,\ldots,m\}$). By the induction hypothesis this entails that $(\mathfrak{s}',\mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q_i(\boldsymbol{y}_i\sigma,\boldsymbol{p}\sigma)$ and $\mathfrak{s}(v) = \mathfrak{s}'(v) \notin dom(\mathfrak{h}_i)$. As $\mathfrak{s}'(\boldsymbol{y}|_1\sigma) \neq \mathfrak{s}'(w\sigma)$, we have $\mathfrak{s}(\boldsymbol{x}|_1) \neq \mathfrak{s}(v)$ i.e. $\mathfrak{s}(v) \notin dom(\mathfrak{h}_0)$, hence $\mathfrak{s}(v) \notin dom(\mathfrak{h}) = \bigcup_{i=0}^m dom(\mathfrak{h}_i)$. Furthermore, $(\mathfrak{s}',\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \bigcirc_{i=1}^n Q_i(\boldsymbol{y}_i\sigma,\boldsymbol{p}\sigma) \circ \phi\sigma = (\bigcirc_{i=1}^n Q_i(\boldsymbol{y}_i,p) \circ \phi)\sigma$ and $P(\boldsymbol{x},\boldsymbol{p}) \Leftarrow_{\mathcal{R}} \exists \boldsymbol{u}\,(\bigcirc_{i=1}^m Q_i(\boldsymbol{y}_i,\boldsymbol{p}) \circ \phi)\sigma$, consequently $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(\boldsymbol{x},\boldsymbol{p})$.

– Conversely, assume that $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(\boldsymbol{x},\boldsymbol{p})$ and that $\mathfrak{s}(v) \notin dom(\mathfrak{h})$. By definition of the semantics, this entails that $\mathcal{R}$ contains a rule $P(\boldsymbol{y},\boldsymbol{z}) \Leftarrow \psi$ with $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi\sigma$ and $\sigma = \{\boldsymbol{y} \leftarrow \boldsymbol{x}, \boldsymbol{z} \leftarrow \boldsymbol{p}\}$. As $\mathcal{R}$ is $\mathfrak{h}$-regular, ψ is necessarily of the form $\exists \boldsymbol{u}\,(\bigcirc_{i=1}^{m} Q_i(\boldsymbol{y}_i,\boldsymbol{p}_i) \circ \phi)$, where ϕ contains no predicate atom and exactly one points-to atom with left-hand side $y|_1$. Moreover, there exist a vector of locations $\boldsymbol{\ell}$ and disjoint heaps $\mathfrak{h}_0,\ldots,\mathfrak{h}_m$ such that $(\mathfrak{s}',\mathfrak{h}_0) \models^{\mathfrak{P}}_{\mathcal{R}} \phi\sigma$, $(\mathfrak{s}',\mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q_i(\boldsymbol{y}_i,\boldsymbol{p}_i)\sigma$ and $\mathfrak{s}' = \mathfrak{s}\{\boldsymbol{u} \leftarrow \boldsymbol{\ell}\}$. Since $\mathfrak{s}'(v) = \mathfrak{s}(v) \notin dom(\mathfrak{h})$, we get $\mathfrak{s}'(v) \notin dom(\mathfrak{h})$ thus $\mathfrak{s}'(v) \notin dom(\mathfrak{h}_i)$. Moreover, $dom(\mathfrak{h}_0) = \{\mathfrak{s}'(\boldsymbol{y}|_1\sigma)\}$, hence $\mathfrak{s}'(v) \neq \mathfrak{s}'(\boldsymbol{y}|_1\sigma\}$ and $|\mathfrak{h}_i| < |\mathfrak{h}|$ (for all $i \in \{1,\ldots,m\}$). By the induction hypothesis, this entails that $(\mathfrak{s}',\mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q_i(\boldsymbol{y}_i\sigma,\boldsymbol{p}\sigma)[v]^-$. By definition of the rules defining $P(\boldsymbol{x},\boldsymbol{p})[v]^-$, $P(\boldsymbol{x},\boldsymbol{p})[v]^- \Leftarrow_{\mathcal{R}} \exists \boldsymbol{u}\,(\bigcirc_{i=1}^{m} Q_i(\boldsymbol{y}_i,\boldsymbol{p})[w]^- \circ \phi \circ y|_1 \not\simeq w)\sigma'$, with $\sigma' = \sigma\{w \leftarrow v\}$. We get $(\mathfrak{s}',\mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q_i(\boldsymbol{y}_i,\boldsymbol{p})[w]^-\sigma'$ (for all $i \in \{1,\ldots,m\}$), $(\mathfrak{s}',\mathfrak{h}_0) \models^{\mathfrak{P}}_{\mathcal{R}} \phi\sigma = \phi\sigma'$ and $\mathfrak{s}'(y|_1\sigma) \neq \mathfrak{s}'(w\sigma') = \mathfrak{s}'(v)$. Thus $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(\boldsymbol{x},\boldsymbol{p})[v]^-$.

D Proof of Lemma 19

Assume that $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(\boldsymbol{x},\boldsymbol{p}) \circ ((P(\boldsymbol{x},\boldsymbol{p}) \circ \alpha) \mathrel{-\!\!\bullet} Q(\boldsymbol{y},\boldsymbol{q}))$. By definition, there exist disjoint heaps $\mathfrak{h}_1,\mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \sqcup \mathfrak{h}_2$, $(\mathfrak{s},\mathfrak{h}_1) \models^{\mathfrak{P}}_{\mathcal{R}} P(\boldsymbol{x},\boldsymbol{p})$ and $(\mathfrak{s},\mathfrak{h}_2) \models^{\mathfrak{P}}_{\mathcal{R}} (P(\boldsymbol{x},\boldsymbol{p}) \circ \alpha) \mathrel{-\!\!\bullet} Q(\boldsymbol{y},\boldsymbol{q})$. We show, by induction on $|\mathfrak{h}_2|$, that $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \alpha \mathrel{-\!\!\bullet} Q(\boldsymbol{y},\boldsymbol{q})$. By definition of the rules defining $(P(\boldsymbol{x},\boldsymbol{p}) \circ \alpha) \mathrel{-\!\!\bullet} Q(\boldsymbol{y},\boldsymbol{q})$, necessarily $(\mathfrak{s},\mathfrak{h}_2) \models^{\mathfrak{P}}_{\mathcal{R}} \exists \boldsymbol{w}\,(\psi_1 \circ \ldots \circ \psi_m \circ \phi)$, with $P(\boldsymbol{x},\boldsymbol{p}) \Leftarrow_{\mathcal{R}} \bigcirc_{i=1}^{m} Q_i(\boldsymbol{u}_i,\boldsymbol{p}_i) \circ \phi$, $\alpha_1 \circ \ldots \circ \alpha_m$ is a decomposition of $P(\boldsymbol{x},\boldsymbol{p}) \circ \alpha$ (up to AC and neutrality of $\mathtt{emp}$), and for every $i \in \{1,\ldots,m\}$, either $\psi_i = \alpha_i \mathrel{-\!\!\bullet} Q_i(\boldsymbol{u}_i,\boldsymbol{q}_i)$ or $\psi_i = (\boldsymbol{y}_j \simeq \boldsymbol{u}_i \circ \boldsymbol{p}_j \simeq \boldsymbol{q}_i)$ with $\alpha_i = P_j(\boldsymbol{y}_j,\boldsymbol{p}_j)$ and $P_j = Q_i$. Thus there exist locations $\boldsymbol{\ell}$ and disjoint heaps $\mathfrak{h}_2^0,\ldots,\mathfrak{h}_2^m$ such that $(\mathfrak{s}',\mathfrak{h}_2^0) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$, $(\mathfrak{s}',\mathfrak{h}_2^i) \models^{\mathfrak{P}}_{\mathcal{R}} \psi_i$ (for all $i \in \{1,\ldots,m\}$) and $\mathfrak{s}' = \mathfrak{s}\{\boldsymbol{w} \leftarrow \boldsymbol{\ell}\}$. By definition, there exists $i \in \{1,\ldots,m\}$ such that $P(\boldsymbol{x},\boldsymbol{p})$ occurs in α_i, we assume by symmetry that $i = 1$, with $\alpha_1 = P(\boldsymbol{x},\boldsymbol{p}) \circ \alpha_1'$. As $\mathcal{R}$ is $\mathfrak{h}$-regular, ϕ contains exactly one points-to atom, thus $\mathfrak{h}_0 \neq \emptyset$ and $|\mathfrak{h}_i| < |\mathfrak{h}|$ (for all $i \in \{1,\ldots,m\}$). We distinguish two cases.

– Assume that ψ_1 is of the first form above. We have $(\mathfrak{s}',\mathfrak{h}_1 \sqcup \mathfrak{h}_2^1) \models^{\mathfrak{P}}_{\mathcal{R}} P(\boldsymbol{x},\boldsymbol{p}) \circ ((P(\boldsymbol{x},\boldsymbol{p}) \circ \alpha_1') \mathrel{-\!\!\bullet} Q_1(\boldsymbol{y}_1,\boldsymbol{q}_1))$. By the induction hypothesis, we get $(\mathfrak{s}',\mathfrak{h}_1 \sqcup \mathfrak{h}_2^1) \models^{\mathfrak{P}}_{\mathcal{R}} \alpha_1' \mathrel{-\!\!\bullet} Q_1(\boldsymbol{y}_1,\boldsymbol{q}_1)$. This entails that $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \exists \boldsymbol{w}\,(\alpha_1' \mathrel{-\!\!\bullet} Q_1(\boldsymbol{y}_1,\boldsymbol{q}_1) \circ \bigcirc_{i=2}^{m} \psi_m \circ \phi)$, hence $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \alpha \mathrel{-\!\!\bullet} Q(\boldsymbol{y},\boldsymbol{q}_1)$, by definition of the rules defining $\alpha \mathrel{-\!\!\bullet} Q(\boldsymbol{y},\boldsymbol{q}_1)$ (since $\alpha = \alpha_1' \circ \bigcirc_{i=2}^{m} \alpha_i$).
– Assume that ψ_1 is of the second form above. Then $\mathfrak{s}'(\boldsymbol{x}) = \mathfrak{s}'(\boldsymbol{y}_1)$, $\mathfrak{s}'(\boldsymbol{p}) = \mathfrak{s}'(\boldsymbol{q}_1)$, and $Q_1 = P$ (with $\alpha' = \mathtt{emp}$), so that $(\mathfrak{s}',\mathfrak{h}_1) \models^{\mathfrak{P}}_{\mathcal{R}} Q_1(\boldsymbol{y}_1,\boldsymbol{q}) = \mathtt{emp} \mathrel{-\!\!\bullet} Q_1(\boldsymbol{y}_1,p)$. Therefore $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \exists \boldsymbol{w}\,(\mathtt{emp} \mathrel{-\!\!\bullet} Q_1(\boldsymbol{y}_1,\boldsymbol{q}_1) \circ \bigcirc_{i=2}^{m} \psi_m \circ \phi)$, with $\alpha = \mathtt{emp} \circ \bigcirc_{i=2}^{m} \alpha_i$. This entails the result, by definition of the rules defining $\alpha \mathrel{-\!\!\bullet} Q(\boldsymbol{y},\boldsymbol{q}_1)$.

E Proof of Lemma 20

The proof is by induction on $|\mathfrak{h}|$. Assume that $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} Q(\boldsymbol{y},\boldsymbol{p})$, $\mathfrak{s}(x) \neq \boldsymbol{y}|_1$ and $\mathfrak{s}(x) \in dom(\mathfrak{h})$. By definition of the semantics, there exists a formula ψ such that $Q(\boldsymbol{y},\boldsymbol{p}) \Leftarrow_{\mathcal{R}} \psi$ and $(\mathfrak{s},\mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$, and, since $\mathcal{R}$ is $\mathfrak{h}$-regular, ψ is of the form $\exists \boldsymbol{w}\,(\bigcirc_{i=1}^{n} Q_i(\boldsymbol{u}_i,\boldsymbol{p}_i) \circ \phi)$, where ϕ contains exactly one points-to atom (which left-hand side is necessarily $\boldsymbol{y}|_1$) and $\boldsymbol{p}_i \subseteq \boldsymbol{p}$ (for all $i = 1,\ldots,n$). Therefore there exist a sequence of locations $\boldsymbol{\ell}$ and disjoint heaps $\mathfrak{h}_0,\ldots,\mathfrak{h}_n$ such that $\mathfrak{h} = \mathfrak{h}_0 \sqcup \ldots \sqcup \mathfrak{h}_n$, $(\mathfrak{s}',\mathfrak{h}_0) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$ and $(\mathfrak{s}',\mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q_i(\boldsymbol{u}_i,\boldsymbol{p}_i)$, for all $i \in \{1,\ldots,n\}$, with $\mathfrak{s}' = \mathfrak{s}\{\boldsymbol{w} \leftarrow \boldsymbol{\ell}\}$. Furthermore, we have $dom(\mathfrak{h}_0) = \{\mathfrak{s}'(\boldsymbol{y}|_1)\} \neq \emptyset$, and thus $|\mathfrak{h}_i| < |\mathfrak{h}|$, for all $i \in \{1,\ldots,n\}$. Since $\mathfrak{s}(x) \neq \mathfrak{s}(\boldsymbol{y}|_1)$ we have $\mathfrak{s}(x) \notin dom(\mathfrak{h}_0)$, and, as $\mathfrak{s}(x) \in dom(\mathfrak{h})$, this entails that $\mathfrak{s}(x) \in dom(\mathfrak{h}_i)$, for some $i \in \{1,\ldots,n\}$. Assume by symmetry that $\mathfrak{s}(x) \in dom(\mathfrak{h}_1)$. We distinguish two cases:

- If $\mathfrak{s}'(\boldsymbol{u}_1|_1) \neq \mathfrak{s}'(x)$, then, by the induction hypothesis, we deduce that there exist atoms $P(x,\boldsymbol{z},\boldsymbol{q})$ and $P_i(x,\boldsymbol{y}_i,\boldsymbol{q}_i)$ (for $i \in \{1,\ldots,m\}$) such that

$$(\mathfrak{s}',\mathfrak{h}_1) \models^{\mathfrak{P}}_{\mathcal{R}} \exists \boldsymbol{x}\ (\beta \circ (\beta \mathrel{-\!\!\bullet} Q_1(\boldsymbol{u}_1,\boldsymbol{p}_1)))$$

  with $\boldsymbol{x} = (x_1,\ldots,x_n)$, $\beta = P(x,\boldsymbol{z},\boldsymbol{q}) \circ \bigcirc_{i=1}^{m} P_i(x_i,\boldsymbol{y}_i,\boldsymbol{q}_i)$, $\boldsymbol{z} \subseteq \boldsymbol{u_1} \cup \boldsymbol{x}$, $\boldsymbol{q} \cup \boldsymbol{q_i} \subseteq \boldsymbol{p_1} \subseteq \boldsymbol{p}$, $\boldsymbol{y}_i \subseteq \{\boldsymbol{u_1}|_j \mid j \notin \gamma_{\mathcal{R}}(Q_1)\}$ (for all $i \in \{1,\ldots,m\}$), $P_i \in \mathcal{P}^\star$, $\boldsymbol{x} \subseteq (x,\boldsymbol{z})|_{\gamma_{\mathcal{R}}(P)}$ and $y \in \boldsymbol{u}_1 \cap \boldsymbol{z} \wedge y \notin \{\boldsymbol{u}_1|_j \mid j \notin \gamma_{\mathcal{R}}(Q_1)\} \implies y \in (x,\boldsymbol{z})|_{\gamma_{\mathcal{R}}(P)}$. By α-renaming, we assume that $\boldsymbol{x} \cap \boldsymbol{y} = \emptyset$. Thus there exist a store $\mathfrak{s}''$ coinciding with $\mathfrak{s}'$ on all variables not occurring in $\boldsymbol{x}$, and disjoint heaps $\mathfrak{h}_1', \mathfrak{h}_1''$ such that $\mathfrak{h}_1 = \mathfrak{h}_1' \cup \mathfrak{h}_1''$, $(\mathfrak{s}'',\mathfrak{h}_1') \models^{\mathfrak{P}}_{\mathcal{R}} \beta$ and $(\mathfrak{s}'',\mathfrak{h}_1'') \models^{\mathfrak{P}}_{\mathcal{R}} (\beta \mathrel{-\!\!\bullet} Q_1(\boldsymbol{u}_1,\boldsymbol{p}_1))$.
  We show that $y \in \boldsymbol{y} \cap \boldsymbol{z} \wedge y \notin \{\boldsymbol{y}|_j \mid j \notin \gamma_{\mathcal{R}}(Q)\} \implies y \in (x,\boldsymbol{z})|_{\gamma_{\mathcal{R}}(P)}$. Assume that $z \in \boldsymbol{y} \cap \boldsymbol{z}$, with $z \notin \{\boldsymbol{y}|_j \mid j \notin \gamma_{\mathcal{R}}(Q)\}$. As $\boldsymbol{z} \subseteq \boldsymbol{x} \cup \boldsymbol{u}_1$, necessarily $z = \boldsymbol{u}_1|_{j'}$ for some $j' \in \{1,\ldots,\#_1(Q_1)\}$. Assume that $j' \notin \gamma_{\mathcal{R}}(Q_1)$. By Condition 2 in Definition 10 we deduce that there exists $j \notin \gamma_{\mathcal{R}}(Q)$ such that $z = \boldsymbol{y}|_j$, which contradicts the above assumption. Otherwise (i.e., if $z = \boldsymbol{u}_1|_{j'} \Rightarrow j' \in \gamma_{\mathcal{R}}(Q_1)$, for all j') then $y \notin \{\boldsymbol{u}_1|_j \mid j \notin \gamma_{\mathcal{R}}(Q_1)\}$ and the proof follows from the induction hypothesis above.
  We show that $\boldsymbol{y}_i \subseteq \{\boldsymbol{y}|_{j'} \mid j' \notin \gamma_{\mathcal{R}}(Q)\}$, for all $i \in \{1,\ldots,m\}$. Let $v \in \boldsymbol{y}_i$. Since $\boldsymbol{y}_i \subseteq \{\boldsymbol{u_1}|_j \mid j \notin \gamma_{\mathcal{R}}(Q_1)\}$, we get $v = \boldsymbol{u_1}|_j$, for some $j \notin \gamma_{\mathcal{R}}(Q_1)$. By definition of $\gamma_{\mathcal{R}}(Q_1)$ (see Conditions 1 and 2 in Definition 10), this entails that $v = \boldsymbol{y}|_{j'}$, for some $j' \notin \gamma_{\mathcal{R}}(Q)$.
  We show that $\boldsymbol{z} \subseteq \boldsymbol{y} \cup \boldsymbol{x} \cup \{\boldsymbol{u}_i|_1 \mid i = 2,\ldots,n\}$. Let V be the set of variables occurring in $\boldsymbol{z}$, but not in $\boldsymbol{y} \cup \boldsymbol{x}$. Consider any variable $v \in V$. Necessarily, $v \in \boldsymbol{u}_1$ (as $\boldsymbol{z} \subseteq \boldsymbol{u_1} \cup \boldsymbol{x}$), and $v \in \boldsymbol{w}$ (as $\boldsymbol{u}_1 \subseteq \boldsymbol{y} \cup \boldsymbol{w}$, by definition of the unfolding). Note that, by Condition 1 in Definition 10, this entails that $v \in \boldsymbol{u}_1$ with $v = \boldsymbol{u}_1|_j \implies j \in \gamma_{\mathcal{R}}(Q_1)$, so that $v \in (x,\boldsymbol{z})|_{\gamma_{\mathcal{R}}(P)}$. As $\mathcal{R}$ is $\mathfrak{h}$-regular and $v \in \boldsymbol{w}$, necessarily there exists $k \in \{2,\ldots,m\}$ such that $v = \boldsymbol{u}_k|_1$. We assume, by symmetry, that the set of indices k such that $\boldsymbol{u}_k|_1 \in V$ is of the form $\{2,\ldots,n'\}$ (with possibly $n' = 1$, as V may be empty), so that $\boldsymbol{z} \subseteq \boldsymbol{y} \cup \boldsymbol{x} \cup \{\boldsymbol{u}_i|_1 \mid i = 2,\ldots,n'\}$.
  Since $\mathcal{R}$ is $\exists$-restricted (see Condition 5 in Definition 10) we get $\gamma_{\mathcal{R}}(Q_k) = \emptyset$, for all $k \in \{2,\ldots,n'\}$, so that $\{\boldsymbol{u}_k|_j \mid j \neq 1\} \subseteq \{\boldsymbol{y}|_j \mid j \notin \gamma_{\mathcal{R}}(Q)\}$.

Let $\mathfrak{h}' \stackrel{\text{def}}{=} \mathfrak{h}_0 \sqcup \mathfrak{h}_1 \sqcup \mathfrak{h}_{n'+1} \sqcup \ldots \sqcup \mathfrak{h}_n$ and $\mathfrak{h}'' \stackrel{\text{def}}{=} \mathfrak{h}_2 \sqcup \ldots \sqcup \mathfrak{h}_{n'}$. Observe that $\mathfrak{h}'$ and $\mathfrak{h}''$ are disjoint and that $\mathfrak{h} = \mathfrak{h}' \sqcup \mathfrak{h}''$, We get: $(\mathfrak{s}'', \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} (\beta \mathrel{-\!\!\bullet} Q(\boldsymbol{y}, \boldsymbol{p})) \circ \bigcirc_{i=n'+1}^{n} Q_i(\boldsymbol{y}_i, \boldsymbol{q}_i) \circ \phi$. By definition of the rules defining $\beta \mathrel{-\!\!\bullet} Q(\boldsymbol{y}, p)$, this entails that $(\mathfrak{s}'', \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} \beta' \mathrel{-\!\!\bullet} Q(\boldsymbol{y}, \boldsymbol{p})$ with $\beta' \stackrel{\text{def}}{=} \beta \circ \bigcirc_{i=2}^{n'} Q_i(\boldsymbol{u}_i, \boldsymbol{p}_i)$. Moreover, $(\mathfrak{s}'', \mathfrak{h}'') \models^{\mathfrak{P}}_{\mathcal{R}} P(x, \boldsymbol{z}, \boldsymbol{q}) \circ \bigcirc_{i=1}^{m} P_i(x_i, \boldsymbol{y}_i, \boldsymbol{q}_i) \circ \bigcirc_{i=2}^{n'} Q_i(\boldsymbol{u}_i, \boldsymbol{q}_i)$. Thus:

$$(\mathfrak{s}'', \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(x, \boldsymbol{z}, \boldsymbol{q}) \circ \bigcirc_{i=1}^{m} P_i(x_i, \boldsymbol{y}_i, \boldsymbol{q}_i) \circ \bigcirc_{i=2}^{n'} Q_i(\boldsymbol{u}_i, \boldsymbol{q}_i) \circ (\beta' \mathrel{-\!\!\bullet} Q(\boldsymbol{y}, p))$$

with $\beta' = P(x, \boldsymbol{z}, \boldsymbol{q}) \circ \bigcirc_{i=1}^{m} P_i(x_i, \boldsymbol{y}_i, \boldsymbol{q}_i) \circ \bigcirc_{i=2}^{n'} Q_i(\boldsymbol{u}_i, \boldsymbol{p}_i)$. Consequently,

$$(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \exists \boldsymbol{v} (P(x, \boldsymbol{z}, \boldsymbol{q}) \circ \bigcirc_{i=1}^{m} P_i(x_i, \boldsymbol{y}_i, \boldsymbol{q}_i) \circ \bigcirc_{i=2}^{n'} Q_i(\boldsymbol{u}_i, \boldsymbol{q}_i) \circ (\beta' \mathrel{-\!\!\bullet} Q(\boldsymbol{y}, p)))$$

with $\boldsymbol{v} = \boldsymbol{x} \cup V = \boldsymbol{x} \cup \{\boldsymbol{u}_i|_1 \mid i = 2, \ldots, n'\}$, which completes the proof, as we have proven that $\boldsymbol{z} \subseteq \boldsymbol{y} \cup \boldsymbol{x} \cup \{\boldsymbol{u}_i|_1 \mid i = 2, \ldots, n'\}$, $\boldsymbol{y}_i \cup \{\boldsymbol{u}_k|_j \mid j \neq 1\} \subseteq \{\boldsymbol{y}|_{j'} \mid j' \notin \gamma_{\mathcal{R}}(Q)\}$, $\boldsymbol{q} \cup \boldsymbol{q_i} \cup \boldsymbol{p_i} \subseteq \boldsymbol{p}$, $P_i \in \mathcal{P}^\star$ and $\boldsymbol{x} \cup V \cup (\boldsymbol{y}|_{\gamma_{\mathcal{R}}(Q)} \cap \boldsymbol{z}) \subseteq (x, \boldsymbol{z})|_{\gamma_{\mathcal{R}}(P)}$.

– If $\mathfrak{s}'(\boldsymbol{u}_1|_1) = \mathfrak{s}'(x)$, i.e., $\mathfrak{s}'(\boldsymbol{u}_1|_1) = \mathfrak{s}(x)$, then we let $P = Q_1$, $\boldsymbol{p} = \boldsymbol{p}_1$, and $\boldsymbol{u}_1|_1.\boldsymbol{z} = \boldsymbol{u}_1$. As in the previous case, the variables $\boldsymbol{v}$ occurring in $\boldsymbol{z}$ but not in $\boldsymbol{y}$ must occur in $\boldsymbol{w}$, thus must be the root of some atom $Q_i(\boldsymbol{u}_i, \boldsymbol{p}_i)$, for some $i \in \{2, \ldots, n'\}$ with $n' \in \{1, \ldots, n\}$, $Q_i \in \mathcal{P}^\star$ and $\{\boldsymbol{u}_i|_j \mid j \neq 1\} \subseteq \{\boldsymbol{y}|_j \mid j \notin \gamma_{\mathcal{R}}(Q)\}$. By definition of $\mathrel{-\!\!\bullet}$, we have $(\mathfrak{s}'', \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(x, \boldsymbol{z}, \boldsymbol{q}) \circ \bigcirc_{i=2}^{n'} Q_i(\boldsymbol{u}_i, \boldsymbol{q}_i) \circ (\beta \mathrel{-\!\!\bullet} Q(\boldsymbol{y}, p))$ with $\beta = P(x, \boldsymbol{z}, \boldsymbol{q}) \circ \bigcirc_{i=2}^{n'} Q_i(\boldsymbol{u}_i, \boldsymbol{p}_i)$, $\boldsymbol{z} \subseteq \boldsymbol{y} \cup \{\boldsymbol{u}_i|_1 \mid i = 2, \ldots, n'\}$, $\boldsymbol{u}_i \subseteq \{\boldsymbol{y}|_{j'} \mid j' \notin \gamma_{\mathcal{R}}(Q)\}$ and $\boldsymbol{q} \cup \boldsymbol{p_i} \subseteq \boldsymbol{p}$, so that $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \exists \boldsymbol{v} (P(x, \boldsymbol{z}, \boldsymbol{q}) \circ \bigcirc_{i=2}^{n'} Q_i(\boldsymbol{u}_i, \boldsymbol{q}_i) \circ (\beta \mathrel{-\!\!\bullet} Q(\boldsymbol{y}, p)))$. By definition of $\gamma_{\mathcal{R}}$, we have $\boldsymbol{v} \subseteq \boldsymbol{u}_1|_{\gamma_{\mathcal{R}}(P)}$ and $y \in \boldsymbol{y} \cap \boldsymbol{z} \wedge y \notin \{\boldsymbol{y}|_j \mid j \notin \gamma_{\mathcal{R}}(Q)\} \implies y \in (x, \boldsymbol{z})|_{\gamma_{\mathcal{R}}(P)}$. Again, the proof is completed.

F Proof of Lemma 23

We need the following:

Proposition 34. *Let δ be a predicate atom and let ϕ be a formula. If $x \in roots(\phi)$ then $\delta \circ \phi \equiv^{\mathfrak{P}}_{\mathcal{R}} \delta[x]^- \circ \phi$.*

Proof. The result is an immediate consequence of Lemma 15 and Proposition 8 (using the fact that x cannot be allocated in both parts of the heaps that correspond to $roots(\phi)$ and ϕ, respectively).

We apply the following (non deterministic) transformation, for all variables $x \in fv(\phi)$. We replace every atom δ occurring in ϕ such that $x \notin roots(\delta) \cup unalloc(\delta)$ by some (indeterministically chosen) formula in $\{\delta[x]^=, \delta[x]^-\} \cup \delta[x]^+$, yielding a set of formulas $\Phi(x)$. In the case where δ is replaced by $\delta[x]^=$, since by definition $\delta[x]^= = \delta \circ (x \simeq y)$ with $\{y\} = roots(\delta)$, the variable y may be replaced by x in the entire formula, so that we get an atom with root x. By Lemma 22, for all structures $(\mathfrak{s}, \mathfrak{h})$, $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$ iff $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$, for some formula $\psi \in \Phi(x)$, thus satisfiability is preserved. Observe that, for all formulas $\psi \in \Phi$ and for all atoms δ' in ψ, either x occurs at the root of some predicate atom

occurring in the same $\circ$-formula as δ' in ψ, or $x \in \mathit{unalloc}(\delta')$. The former case occurs when δ' occurs in $\delta[x]^=$ (after the replacement of y by x) or in some formula in $\delta[x]^+$, as, by definition, x occurs as a root in all formulas in $\delta[x]^+$, and all such formulas are $\circ$-formulas. The latter case occurs when δ' is of the form $\delta[x]^-$, as $x \in \mathit{unalloc}(\delta[x]^-)$, by definition of $\delta[x]^-$. Since the same variable x cannot be allocated in distinct parts of the heaps, this entails, by Lemma 15, that any atom δ' in ϕ with $x \notin \mathit{roots}(\delta) \cup \mathit{unalloc}(\delta)$ can be replaced by $\delta[x]^-$ without affecting the semantics of the formula. We then get a formula ϕ' such that $x \in \mathit{roots}(\delta') \cup \mathit{unalloc}(\delta')$ holds for all atom δ' in ϕ', so that the implication $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \delta' \wedge \mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h}) \implies x \in \mathit{roots}(\delta')$ holds, by Propositions 8 and 34.

This process is applied on every variable x, which eventually yield a set of normalized formulas Ψ. Note that new variables may be introduced into the formula, as $\delta[x]^+$ may contain variables (namely the variables $x_1, \ldots, x_n$ in Definition 21) not occurring in δ. The above transformation must be applied also on such variables, which, in principle, could lead to non termination. We prove that the algorithm terminates, by showing that no new variables are added when x does not occur in the initial formula ϕ (more precisely, we prove that the obtained formula is always $(\mathcal{R}, \mathfrak{P})$-unsatisfiable in this case, hence can be removed from the set of formulas at hand). By Lemma 20, as x is necessarily added by some replacement $\delta \to \delta[y]^+$, we observe the variable x necessarily occurs as the root of some atom $Q_i(x, \boldsymbol{y})$, with $Q_i \in \mathcal{P}^\star$. We denote by ξ_1 the $\circ$-formula containing $Q_i(x, \boldsymbol{y})$. Assume that the transformation above is in turn applied on the variable x, yielding a formula ψ. By definition of $\delta[x]^+$, this entails that x occurs as the root of some atom $P(x, \boldsymbol{z})$. Moreover, if new variables are added, then, by Lemma 20, we must have $\gamma_{\mathcal{R}}(P) \neq \emptyset$, thus $P \in \mathcal{P} \setminus \mathcal{P}^\star$ (by Condition 3 in Definition 10). We denote by ξ_2 the $\circ$-formula of ψ containing $P(x, \boldsymbol{z})$. The case where $\xi_1 = \xi_2$ can be dismissed, as x would occur twice as a root in the same $\circ$-formula, which would then be unsatisfiable. Therefore, for every model $(\mathfrak{s}, \mathfrak{h})$ of ψ, there exist heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $(\mathfrak{s}, \mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \xi_i$ and $\mathfrak{h}_1 \sqcup \mathfrak{h}_2 \leq \mathfrak{h}$. Observe that each time an atom of the form $\alpha \mathrel{-\!\!\bullet} P'(x, \boldsymbol{u})$ is introduced in the formula, the $\circ$-formula α is simultaneously added in the same $\circ$-formula, and we have (by Lemma 19) $\alpha \circ (\alpha \mathrel{-\!\!\bullet} P'(x, \boldsymbol{u})) \models^{\mathfrak{P}}_{\mathcal{R}} P'(x, \boldsymbol{u})$. Moreover, for every atom $P'(x, \boldsymbol{u})$ and variable y, we have (by Lemma 15) $P'(x, \boldsymbol{u})[y]^- \models^{\mathfrak{P}}_{\mathcal{R}} P'(x, \boldsymbol{u})$. By an easy induction on the structure of derived predicates, we deduce that for every atom $Q(x, \boldsymbol{u}')$ occurring in some $\circ$-formula ξ in ψ, and every model $(\mathfrak{s}, \mathfrak{h})$ of ξ, there exist an atom $Q'(x, \boldsymbol{u})$ and a heap $\mathfrak{h}'$ such that $(\mathfrak{s}, \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} Q'(x, \boldsymbol{u})$ and $\mathfrak{h}' \leq \mathfrak{h}$, Q is either equal to Q' or derived from Q', and Q' is not a derived predicate, i.e., occurs in the *initial* SID. Moreover, by Propositions 14 and 18, we have $Q' \in \mathcal{P}^\star \iff Q \in \mathcal{P}^\star$. Consequently, for all heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $(\mathfrak{s}, \mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \xi_i$ there exist heaps $\mathfrak{h}'_1, \mathfrak{h}'_2$ such that $(\mathfrak{s}, \mathfrak{h}'_1) \models^{\mathfrak{P}}_{\mathcal{R}} Q'_i(x_i, \boldsymbol{y}')$, $(\mathfrak{s}, \mathfrak{h}'_2) \models^{\mathfrak{P}}_{\mathcal{R}} P'(x_i, \boldsymbol{y}'')$, $\mathfrak{h}'_i \leq_1 \mathfrak{h}_i$ with $Q'_i \in \mathcal{P}^\star$, $P' \in \mathcal{P} \setminus \mathcal{P}^\star$ and Q'_i, P' are not derived predicates. However, by Condition 4 in Definition 10, $Q'_i(x_i, \boldsymbol{y}') * P'(x_i, \boldsymbol{y}'')$ is $(\mathcal{R}, \mathfrak{P})$-unsatisfiable. This entails that $\xi_1 * \xi_2$ (hence ψ) is also $(\mathcal{R}, \mathfrak{P})$-unsatisfiable.

G Proof of Lemma 25

We use the following result, that is an easy consequence of Definition 24:

Proposition 35. *$cut(L_1 \cup L_2, L', \mathfrak{h}) = cut(L_1, L' \cup L'_1, \mathfrak{h}) \cup cut(L_2, L' \cup L'_2, \mathfrak{h})$, if $L_1 \cap L_2 = \emptyset$ and $L'_i \subseteq L_{3-i}$.*

The proof of Lemma 25 is by induction on the satisfiability relation:

- If ϕ is pure then $roots(\phi) = dom(\mathfrak{h}') = \emptyset$ and the proof is immediate.
- If ϕ is a points-to atom $x \overset{p}{\mapsto} (y_1, \ldots, y_k)$ then by definition of the semantics $\mathfrak{h}' = \{(\mathfrak{s}(x), \mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_n), \mathfrak{s}(p))\}$, hence, by definition of $\leq$, $\mathfrak{h}$ is necessarily of the form $\{(\mathfrak{s}(x), \mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_n), \pi)\}$, where either $\pi = \mathfrak{s}(p)$ or $\pi = (\mathfrak{s}(p) \oplus_{\mathfrak{P}} \pi')$, for some $\pi' \in \mathcal{P}_{\mathfrak{P}}$. In both cases, we get $dom(\mathfrak{h}') = \{\mathfrak{s}(x)\}$, moreover, by Definition 24, $\{\mathfrak{s}(x)\} = cut(\mathfrak{s}(roots(\phi)), \mathfrak{s}(V), \mathfrak{h})$ since $\{x\} = roots(\phi)$ and for all $i \in \{1, \ldots, n\}$ such that $\mathfrak{s}(y_i) \neq \mathfrak{s}(x)$, we have $y_i \in fv(\phi) \setminus \{x\}$, hence $y_i \in V$.
- If ϕ is of the form $\phi_1 \circ \phi_2$, then there exist heaps $\mathfrak{h}'_1, \mathfrak{h}'_2$ such that $dom(\mathfrak{h}'_1) \cap dom(\mathfrak{h}'_2) = \emptyset$, $\mathfrak{h}'_1 \sqcup \mathfrak{h}'_2 = \mathfrak{h}'$ and $(\mathfrak{s}, \mathfrak{h}'_i) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$, for all $i = 1, 2$. Let $V_i = V \cup roots(\phi_{3-i})$. Note that $\mathfrak{s}(V_i) \cap dom(\mathfrak{h}'_i) = \emptyset$ (indeed $dom(\mathfrak{h}'_i) \subseteq dom(\mathfrak{h}')$, hence $\mathfrak{s}(V) \cap dom(\mathfrak{h}'_i) = \emptyset$, and by Proposition 8 $\mathfrak{s}(roots(\phi_{3-i})) \subseteq dom(\mathfrak{h}'_{3-i})$, thus $\mathfrak{s}(roots(\phi_{3-i})) \cap dom(\mathfrak{h}'_i) = \emptyset$, as $dom(\mathfrak{h}'_1) \cap dom(\mathfrak{h}'_2) = \emptyset$). Furthermore, $fv(\phi_i) \subseteq fv(\phi) \subseteq V \cup roots(\phi) = V \cup roots(\phi_i) \cup roots(\phi_{3-i}) = V_i \cup roots(\phi_i)$. By definition of $\leq$, we have $\mathfrak{h}'_i \leq \mathfrak{h}$ for all $i = 1, 2$, thus, by the induction hypothesis, $dom(\mathfrak{h}'_i) = cut(\mathfrak{s}(roots(\phi_i)), \mathfrak{s}(V_i), \mathfrak{h})$. Consequently $dom(\mathfrak{h}') = dom(\mathfrak{h}'_1) \cup dom(\mathfrak{h}'_2) = cut(\mathfrak{s}(roots(\phi_1)), \mathfrak{s}(V_1), \mathfrak{h}) \cup cut(\mathfrak{s}(roots(\phi_2)), \mathfrak{s}(V_2), \mathfrak{h})$, and by Proposition 35 (applied with $L_i = L'_{3-i} = roots(\mathfrak{h}'_i)$ and $L' = V$), $dom(\mathfrak{h}') = cut(\mathfrak{s}(roots(\phi_1)) \cup \mathfrak{s}(roots(\phi_2)), \mathfrak{s}(V), \mathfrak{h})$. Since we have $\mathfrak{s}(roots(\phi_1)) \cup \mathfrak{s}(roots(\phi_2)) = \mathfrak{s}(roots(\phi_1) \cup roots(\phi_2)) = \mathfrak{s}(roots(\phi))$, we get the result.
- If ϕ is a predicate atom $P(x, \boldsymbol{y}, z)$, then $roots(\phi) = \{x\}$ and $\phi \Leftarrow_{\mathcal{R}} \phi'$, with $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi'$. Since $\mathcal{R}$ is $\mathfrak{h}$-regular, ϕ' is of the form $\exists u_1, \ldots, u_n\, (x \overset{p}{\mapsto} (v_1, \ldots, v_k) \circ \psi \circ \psi')$, where ψ' is pure, $roots(\psi) = \{u_1, \ldots, u_n\}$ and $\{u_1, \ldots, u_n\} \subseteq \{v_1, \ldots, v_k\}$. Consequently, there exists a store $\mathfrak{s}'$ coinciding with $\mathfrak{s}$ on all variables not occurrence in $\{u_1, \ldots, u_n\}$ such that $(\mathfrak{s}', \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} x \overset{p}{\mapsto} (v_1, \ldots, v_k) \circ \psi \circ \psi'$. Since ψ' is pure, we also have $(\mathfrak{s}', \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} x \overset{p}{\mapsto} (v_1, \ldots, v_k) \circ \psi$. As $fv(x \overset{p}{\mapsto} (v_1, \ldots, v_k) \circ \psi) \subseteq fv(\phi) \cup \{u_1, \ldots, u_n\} \subseteq V \cup roots(x \overset{p}{\mapsto} (v_1, \ldots, v_k) \circ \psi))$, we get by the induction hypothesis:

$$\begin{array}{c} dom(\mathfrak{h}') = cut(\mathfrak{s}'(roots(x \overset{p}{\mapsto} (v_1, \ldots, v_k) \circ \psi)), \mathfrak{s}'(V), \mathfrak{h}) \\ \text{i.e. } dom(\mathfrak{h}') = cut(\{\mathfrak{s}(x)\} \cup \mathfrak{s}'(\{u_1, \ldots, u_n\}), \mathfrak{s}'(V), \mathfrak{h}) \end{array}$$

  We assume by α-renaming that $\{u_1, \ldots, u_n\} \cap V = \emptyset$ so that $\mathfrak{s}'(V) = \mathfrak{s}(V)$. By definition of the semantics we must have $\mathfrak{h}'(\mathfrak{s}(x)) = (\mathfrak{s}(v_1), \ldots, \mathfrak{s}(v_k), \mathfrak{s}(p))$. Since $\mathfrak{h}' \leq \mathfrak{h}$, we get $\mathfrak{h}(\mathfrak{s}(x)) = (\mathfrak{s}'(v_1), \ldots, \mathfrak{s}'(v_k), \pi)$ for some $\pi \in \mathcal{P}_{\mathfrak{P}}$. By Proposition 8, necessarily $\{\mathfrak{s}'(u_1), \ldots, \mathfrak{s}'(u_n)\} \subseteq dom(\mathfrak{h}')$, since $roots(\psi) = \{u_1, \ldots, u_n\}$. As $\mathfrak{s}(V) \cap dom(\mathfrak{h}') = \emptyset$, we get $\{\mathfrak{s}'(u_1), \ldots, \mathfrak{s}'(u_n)\} \cap \mathfrak{s}(V) = \emptyset$. Since $\mathfrak{h}(\mathfrak{s}(x)) = (\mathfrak{s}'(v_1), \ldots, \mathfrak{s}'(v_k), \pi)$ and $\{u_1, \ldots, u_n\} \subseteq \{v_1, \ldots, v_k\}$, this

entails, by Definition 24, that $\{\mathfrak{s}'(u_1), \ldots, \mathfrak{s}'(u_n)\} \subseteq cut(\mathfrak{s}(\{x\}), \mathfrak{s}(V), \mathfrak{h})$. Thus $cut(\{\mathfrak{s}(x)\} \cup \mathfrak{s}'(\{x, u_1, \ldots, u_n\}), \mathfrak{s}(V), \mathfrak{h}) = cut(\mathfrak{s}(\{x\}), \mathfrak{s}(V), \mathfrak{h})$, hence $dom(\mathfrak{h}') = cut(\mathfrak{s}(roots(\phi)), \mathfrak{s}(V), \mathfrak{h})$.

H Proof of Lemma 26

Let λ be a bijective mapping from $\mathcal{L}$ to $\mathcal{L}$. For any tuple $(\ell_1, \ldots, \ell_n, \pi)$ with $\ell_i \in \mathcal{L}$ and $\pi \in \mathcal{P}_{\mathfrak{P}}$, $\lambda((\ell_1, \ldots, \ell_n, \pi))$ denotes the tuple $(\lambda(\ell_1), \ldots, \lambda(\ell_n), \pi)$. For every heap $\mathfrak{h}$ we denote by $\lambda(\mathfrak{h})$ the heap defined as follows: $dom(\lambda(\mathfrak{h})) \stackrel{\text{def}}{=} \lambda(dom(\mathfrak{h}))$ and for all $\ell \in dom(\mathfrak{h})$, $\lambda(\mathfrak{h})(\ell) \stackrel{\text{def}}{=} \lambda(\mathfrak{h}(\ell))$. The following proposition, showing that the truth value of a formula in a structure does not depend on the name of locations, can be established by an immediate induction on formulas:

Proposition 36. *Let λ be a bijective mapping from $\mathcal{L}$ to $\mathcal{L}$. If $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$ then $(\lambda \circ \mathfrak{s}, \lambda(\mathfrak{h})) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$.*

Now, assume that $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$. Then there exist heaps $\mathfrak{h}_i, \hat{\mathfrak{h}}_i, \mathfrak{h}'$ and $\hat{\mathfrak{h}}'$ (for $i \in \{1, \ldots, n\}$) such that $\mathfrak{h} = \mathfrak{h}' \sqcup \bigsqcup_{i=1}^{n} (\mathfrak{h}_i \sqcup \hat{\mathfrak{h}}_i) \sqcup \hat{\mathfrak{h}}'$, $(\mathfrak{s}, \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} \phi'$, $(\mathfrak{s}, \mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$, $(\mathfrak{s}, \hat{\mathfrak{h}}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \psi_i$, $(\mathfrak{s}, \hat{\mathfrak{h}}') \models^{\mathfrak{P}}_{\mathcal{R}} \psi'$, $dom(\mathfrak{h}_i) \cap dom(\hat{\mathfrak{h}}_i) = \emptyset$ (for all $i \in \{1, \ldots, n\}$) and $dom(\mathfrak{h}') \cap dom(\bigsqcup_{i=1}^{n} (\mathfrak{h}_i \sqcup \hat{\mathfrak{h}}_i) \sqcup \hat{\mathfrak{h}}') = \emptyset$. Let $V' = fv(\phi) \setminus V$. By the hypothesis of the lemma $roots(\phi_i) = V$ (for all $i \in \{1, \ldots, n\}$), thus $fv(\phi_i) \subseteq V' \cup roots(\phi_i)$. Since ϕ is normalized and $(\mathfrak{s}, \mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$, we have $\forall x \in fv(\phi_i) : \mathfrak{s}(x) \in dom(\mathfrak{h}_i) \implies x \in roots(\phi_i)$, hence $\mathfrak{s}(V') \cap dom(\mathfrak{h}_i) = \emptyset$. Moreover, $\mathfrak{h}_i \leq \mathfrak{h}$, and by Lemma 25 (applied on ϕ_i, $\mathfrak{s}(V')$ and $\mathfrak{h}_i$), we deduce that $dom(\mathfrak{h}_i) = cut(\mathfrak{s}(roots(\phi_i)), \mathfrak{s}(V'), \mathfrak{h}) = cut(\mathfrak{s}(V), \mathfrak{s}(V'), \mathfrak{h})$. Therefore, $dom(\mathfrak{h}_i) = dom(\mathfrak{h}_j)$, for all $i, j \in \{1, \ldots, n\}$, which entails that $dom(\mathfrak{h}_i) \cap dom(\hat{\mathfrak{h}}_j) = \emptyset$ (as $dom(\mathfrak{h}_j) \cap dom(\hat{\mathfrak{h}}_j) = \emptyset$). Let L be any infinite subset of $\mathcal{L}$ containing no locations occurring in $\mathfrak{s}(fv(\phi))$ or $loc(\mathfrak{h})$ (such a set always exists as $fv(\phi)$ and $\mathfrak{h}$ are both finite and $\mathcal{L}$ is infinite). Let λ be any injective mapping from $\mathcal{L}$ to $L \cup \mathfrak{s}(fv(\phi))$ such that $\lambda(\mathfrak{s}(x)) = \mathfrak{s}(x)$ for all $x \in fv(\phi)$. By Proposition 36, we have $(\lambda \circ \mathfrak{s}, \lambda(\mathfrak{h}_i)) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$, so that $(\mathfrak{s}, \bigsqcup_{i=1}^{n} \lambda(\mathfrak{h}_i)) \models^{\mathfrak{P}}_{\mathcal{R}} \mathbin{\ast}_{i=1}^{n} \phi_i$ (as $\lambda \circ \mathfrak{s}$ coincides with $\mathfrak{s}$ on all variables in $fv(\phi)$ and $\bigsqcup_{i=1}^{n} \lambda(\mathfrak{h}_i)$ is defined, as $\bigsqcup_{i=1}^{n} \mathfrak{h}_i$ is defined)).

We show that $\bigsqcup_{i=1}^{n} \lambda(\mathfrak{h}_i)$ and $\mathfrak{h}'$ are disjoint. Assume, for the sake of contradiction, that $\ell \in dom(\lambda(\mathfrak{h}_i)) \cap dom(\mathfrak{h}')$. This entails that $\ell \in (L \cup \mathfrak{s}(fv(\phi))) \cap dom(\mathfrak{h}') \subseteq (L \cup \mathfrak{s}(fv(\phi))) \cap loc(\mathfrak{h})$. Since by definition of L, $L \cap loc(\mathfrak{h}) = \emptyset$ we deduce that $\ell = \mathfrak{s}(x)$, for some $x \in fv(\phi)$. By definition of λ, this entails that $\lambda(\ell) = \ell$, i.e., $\lambda^{-1}(\ell) = \ell$, so that $\ell \in dom(\mathfrak{h}_i)$. This entails that $\ell \in dom(\mathfrak{h}_i) \cap dom(\mathfrak{h}')$, which contradicts the fact that $\mathfrak{h}_i$ and $\mathfrak{h}'$ are disjoint.

Therefore, $\mathfrak{h}' \sqcup \bigsqcup_{i=1}^{n} \lambda(\mathfrak{h}_i)$ is defined and we get $(\mathfrak{s}, \mathfrak{h}' \sqcup \bigsqcup_{i=1}^{n} \lambda(\mathfrak{h}_i)) \models^{\mathfrak{P}}_{\mathcal{R}} \phi' \circ \mathbin{\ast}_{i=1}^{n} \phi_i$. As $(\mathfrak{s}, \hat{\mathfrak{h}}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \psi_i$ (for all $i \in \{1, \ldots, n\}$) and $(\mathfrak{s}, \hat{\mathfrak{h}}') \models^{\mathfrak{P}}_{\mathcal{R}} \psi'$, we also have $(\mathfrak{s}, (\bigsqcup_{i=1}^{n} \hat{\mathfrak{h}}_i) \sqcup \hat{\mathfrak{h}}') \models^{\mathfrak{P}}_{\mathcal{R}} \mathbin{\ast}_{i=1}^{n} \psi_i \ast \psi'$, thus it only remains to prove that $\mathfrak{h}' \sqcup \bigsqcup_{i=1}^{n} \lambda(\mathfrak{h}_i)$ and $(\bigsqcup_{i=1}^{n} \hat{\mathfrak{h}}_i) \sqcup \hat{\mathfrak{h}}'$ are disjoint to prove that $(\phi' \circ \mathbin{\ast}_{i=1}^{n} \phi_i) \circ ((\mathbin{\ast}_{i=1}^{n} \psi_i) \ast \psi')$ is $(\mathcal{R}, \mathfrak{P})$-satisfiable. We know that $dom(\mathfrak{h}') \cap (\bigcup_{i=1}^{n} dom(\hat{\mathfrak{h}}_i) \cup dom(\hat{\mathfrak{h}}')) = \emptyset$,

thus it suffices to show that $dom(\lambda(\mathfrak{h}_i)) \cap dom(\hat{\mathfrak{h}}_j) = \emptyset$ (for all $i, j \in \{1, \ldots, n\}$) and that $dom(\lambda(\mathfrak{h}_i)) \cap dom(\hat{\mathfrak{h}}') = \emptyset$ (for all $i \in \{1, \ldots, n\}$).

- Let $\ell \in dom(\lambda(\mathfrak{h}_i)) \cap dom(\hat{\mathfrak{h}}_j)$. By definition $\ell \in L \cup \mathfrak{s}(fv(\phi))$ and $\ell \in loc(\mathfrak{h})$, thus $\ell \in \mathfrak{s}(fv(\phi))$. By definition of λ, this entails that $\lambda^{-1}(\ell) = \ell$, so that $\ell \in dom(\mathfrak{h}_i)$, which contradicts the fact that $\mathfrak{h}_i$ and $\hat{\mathfrak{h}}_j$ are disjoint, as previously shown.
- Let $\ell \in dom(\lambda(\mathfrak{h}_i)) \cap dom(\hat{\mathfrak{h}}')$. By definition $\ell \in L \cup \mathfrak{s}(fv(\phi))$ and $\ell \in loc(\mathfrak{h})$, thus there exists $x \in fv(\phi)$ such that $\ell = \mathfrak{s}(x)$. As ϕ is normalized, $(\mathfrak{s}, \hat{\mathfrak{h}}') \models^{\mathfrak{P}}_{\mathcal{R}} \psi'$ and $\ell \in dom(\hat{\mathfrak{h}}')$ we get $x \in roots(\psi')$. Similarly, since $(\mathfrak{s}, \lambda(\mathfrak{h}_i)) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$ and $\ell \in dom(\lambda(\mathfrak{h}_i))$ we get $x \in roots(\phi_i)$, thus $roots(\phi_i) \cap roots(\psi') \neq \emptyset$, which contradicts the hypothesis of the lemma.

Conversely, assume that $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} (\phi' \circ \mathbin{\ast}^n_{i=1} \phi_i) \circ ((\mathbin{\ast}^n_{i=1} \psi_i) \ast \psi')$. Then there exist heaps $\mathfrak{h}_i, \hat{\mathfrak{h}}_i, \mathfrak{h}'$ and $\hat{\mathfrak{h}}'$ (for $i \in \{1, \ldots, n\}$) such that $\mathfrak{h} = \mathfrak{h}' \sqcup \bigsqcup^n_{i=1} \mathfrak{h}_i \sqcup \bigsqcup^n_{i=1} \hat{\mathfrak{h}}_i \sqcup \hat{\mathfrak{h}}'$, $(\mathfrak{s}, \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} \phi'$, $(\mathfrak{s}, \mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$, $(\mathfrak{s}, \hat{\mathfrak{h}}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \psi_i$, $(\mathfrak{s}, \hat{\mathfrak{h}}') \models^{\mathfrak{P}}_{\mathcal{R}} \psi'$, $dom(\mathfrak{h}') \cap dom(\bigsqcup^n_{i=1} \mathfrak{h}_i) = \emptyset$ and $dom(\mathfrak{h}' \sqcup \bigsqcup^n_{i=1} \mathfrak{h}_i) \cap dom(\bigsqcup^n_{i=1} \hat{\mathfrak{h}}_i \sqcup \hat{\mathfrak{h}}') = \emptyset$. This entails that $dom(\mathfrak{h}_i) \cap dom(\hat{\mathfrak{h}}_i) = \emptyset$ (for all $i \in \{1, \ldots, n\}$), so that $(\mathfrak{s}, \mathfrak{h}_i \sqcup \hat{\mathfrak{h}}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i \circ \psi_i$. Thus $(\mathfrak{s}, \bigsqcup^n_{i=1} (\mathfrak{h}_i \sqcup \hat{\mathfrak{h}}_i)) \models^{\mathfrak{P}}_{\mathcal{R}} \mathbin{\ast}^n_{i=1} (\phi_i \circ \psi_i)$ and $(\mathfrak{s}, \bigsqcup^n_{i=1} (\mathfrak{h}_i \sqcup \hat{\mathfrak{h}}_i) \sqcup \hat{\mathfrak{h}}') \models^{\mathfrak{P}}_{\mathcal{R}} \mathbin{\ast}^n_{i=1} (\phi_i \circ \psi_i) \ast \psi'$. Moreover, we also have $dom(\mathfrak{h}') \cap (\bigcup^n_{i=1} (dom(\mathfrak{h}_i) \cup dom(\hat{\mathfrak{h}}_i)) \cup dom(\hat{\mathfrak{h}}')) = \emptyset$, hence $(\mathfrak{s}, \hat{\mathfrak{h}} \sqcup \bigsqcup^n_{i=1} (\mathfrak{h}_i \sqcup \hat{\mathfrak{h}}_i) \sqcup \hat{\mathfrak{h}}') \models^{\mathfrak{P}}_{\mathcal{R}} \phi' \circ \mathbin{\ast}^n_{i=1} (\phi_i \circ \psi_i) \ast \psi'$, i.e., $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$.

I Proof of Lemma 27

We prove the two implications separately by induction on $|\mathfrak{h}|$. For the first implication, we show for technical convenience that the entailment is valid also in the case where the hypothesis $\mathfrak{s}(x) \neq \mathfrak{s}(y) \implies \mathfrak{s}(y) \notin dom(\mathfrak{h})$ is not satisfied.

$\Rightarrow$ Assume that $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(x, \boldsymbol{y}, \boldsymbol{p}) \triangledown P'(\boldsymbol{y}', \boldsymbol{p}')$. Then there exists a formula ξ such that $P(x, \boldsymbol{y}, \boldsymbol{p}) \triangledown P'(\boldsymbol{y}', \boldsymbol{p}) \Leftarrow_{\mathcal{R}} \xi$ and $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \xi$. By definition of the rules defining $\triangledown$, ξ is of the form:

$$\exists \boldsymbol{u}\, x \overset{q}{\mapsto} (\boldsymbol{v}) \circ \bigcirc^n_{i=1} (Q_i(u_i, \boldsymbol{y}_i, \boldsymbol{q}_i) \triangledown Q'_i(u_i, \boldsymbol{y}'_i, \boldsymbol{q}'_i)) \circ \phi \circ \phi' \circ \psi$$

with $q = p \oplus q$, $\boldsymbol{u} = (u_1, \ldots, u_n)$, $\boldsymbol{v} = (v_1, \ldots, v_k)$, ϕ and ϕ' are pure, $\psi = \boldsymbol{v} \simeq \boldsymbol{v}'$ and:

$$\begin{array}{ll} P(x, \boldsymbol{y}, \boldsymbol{p}) & \Leftarrow_{\mathcal{R}} \exists \boldsymbol{u} \quad x \overset{p}{\mapsto} (\boldsymbol{v}) \circ \bigcirc^n_{i=1} Q_i(u_i, \boldsymbol{y}_i, \boldsymbol{q}_i) \circ \phi \\ P(x, \boldsymbol{y}', \boldsymbol{p}') & \Leftarrow_{\mathcal{R}} \exists \boldsymbol{u} \quad x \overset{p'}{\mapsto} (\boldsymbol{v}) \circ \bigcirc^n_{i=1} Q'_i(u_i, \boldsymbol{y}'_i, \boldsymbol{q}'_i) \circ \phi' \end{array}$$

By definition of the semantics, there exist locations $\boldsymbol{\ell}$ and disjoints heaps $\mathfrak{h}_0, \ldots, \mathfrak{h}_n$ such that $(\mathfrak{s}', \mathfrak{h}_0) \models^{\mathfrak{P}}_{\mathcal{R}} x \overset{q}{\mapsto} (\boldsymbol{v})$ and $(\mathfrak{s}', \mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q_i(u_i, \boldsymbol{y}_i, \boldsymbol{q}_i) \triangledown Q'_i(u_i, \boldsymbol{y}'_i, \boldsymbol{q}'_i)$, for all $i \in \{1, \ldots, n\}$. This entails that $|\mathfrak{h}_0| = 1$, so that $|\mathfrak{h}_i| < |\mathfrak{h}|$ (for all $i \in \{1, \ldots, n\}$) and by the induction hypothesis, we deduce that there exist heaps $\mathfrak{h}'_i$ and $\mathfrak{h}''_i$ such that: $dom(\mathfrak{h}'_i) \subseteq dom(\mathfrak{h}_i)$, $dom(\mathfrak{h}''_i) \subseteq dom(\mathfrak{h}_i)$,

$\mathfrak{h}'_i \sqcup \mathfrak{h}''_i = \mathfrak{h}_i$, $(\mathfrak{s}', \mathfrak{h}'_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q_i(u_i, \boldsymbol{y}_i, \boldsymbol{q}_i)$ and $(\mathfrak{s}', \mathfrak{h}''_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q_i(u_i, \boldsymbol{y}'_i, \boldsymbol{q}'_i)$, with $\mathfrak{s}' \stackrel{\text{def}}{=} \mathfrak{s}\{\boldsymbol{u} \leftarrow \boldsymbol{\ell}\}$. As $\mathfrak{s}'(q)$ is defined, necessarily $\mathfrak{s}'(p)$ and $\mathfrak{s}'(p')$ are both defined and $\mathfrak{s}'(q) = \mathfrak{s}'(p) \oplus_{\mathfrak{P}} \mathfrak{s}'(p')$. Consider the heaps $\mathfrak{h}'_0 \stackrel{\text{def}}{=} \{(\mathfrak{s}(x), \mathfrak{s}'(\boldsymbol{v}), \mathfrak{s}(p))\}$ and $\mathfrak{h}''_0 \stackrel{\text{def}}{=} \{(\mathfrak{s}(x), \mathfrak{s}'(\boldsymbol{v}), \mathfrak{s}(p'))\}$. By definition, we have $dom(\mathfrak{h}'_0) = dom(\mathfrak{h}''_0) = dom(\mathfrak{h}_0)$ and $\mathfrak{h}'_0 \sqcup \mathfrak{h}''_0 = \mathfrak{h}_0$ (since $\mathfrak{s}'(\boldsymbol{v}) = \mathfrak{s}'(\boldsymbol{v}')$ as $\mathfrak{s}' \models^{\mathfrak{P}}_{\mathcal{R}} \psi$). It is clear that the heaps $\mathfrak{h}'_0, \ldots, \mathfrak{h}'_n$ are pairwise disjoint (as $\mathfrak{h}_0, \ldots, \mathfrak{h}_n$ are disjoint) thus $\mathfrak{h}' \stackrel{\text{def}}{=} \mathfrak{h}'_0 \sqcup \ldots \sqcup \mathfrak{h}'_n$ is defined. Moreover, as $\mathfrak{s}' \models^{\mathfrak{P}} \phi$, we get $(\mathfrak{s}', \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} x \stackrel{p}{\mapsto} (\boldsymbol{v}) \circ \bigcirc_{i=1}^{n} Q_i(u_i, \boldsymbol{y}_i, \boldsymbol{q}_i) \circ \phi$, hence $(\mathfrak{s}, \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} P(x, \boldsymbol{y}, \boldsymbol{p})$. By symmetry, we also have $(\mathfrak{s}', \mathfrak{h}'') \models^{\mathfrak{P}}_{\mathcal{R}} P'(x, \boldsymbol{y}', \boldsymbol{p}')$ with $\mathfrak{h}'' \stackrel{\text{def}}{=} \mathfrak{h}''_0 \sqcup \ldots \sqcup \mathfrak{h}''_n$. Furthermore, $\mathfrak{h}' \sqcup \mathfrak{h}'' = (\mathfrak{h}'_0 \sqcup \ldots \sqcup \mathfrak{h}'_n) \sqcup (\mathfrak{h}''_0 \sqcup \ldots \sqcup \mathfrak{h}''_n) = (\mathfrak{h}'_0 \sqcup \mathfrak{h}''_0) \sqcup \ldots \sqcup (\mathfrak{h}'_n \sqcup \mathfrak{h}''_n) = (\mathfrak{h}_0 \sqcup \ldots \sqcup \mathfrak{h}_n) = \mathfrak{h}$. Thus $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(x, \boldsymbol{y}, \boldsymbol{p}) * P'(x, \boldsymbol{y}', \boldsymbol{p}')$.

$\Leftarrow$ Let $\mathfrak{h}', \mathfrak{h}''$ be heaps such that $\mathfrak{h}' \sqcup \mathfrak{h}'' = \mathfrak{h}$, $(\mathfrak{s}, \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} P(x, \boldsymbol{y}, \boldsymbol{p})$ and $(\mathfrak{s}, \mathfrak{h}'') \models^{\mathfrak{P}}_{\mathcal{R}} P'(x, \boldsymbol{y}', \boldsymbol{p}')$. By definition, we have $P(x, \boldsymbol{y}, \boldsymbol{p}) \Leftarrow_{\mathcal{R}} \xi$ and $P(x, \boldsymbol{y}', \boldsymbol{p}') \Leftarrow_{\mathcal{R}} \xi'$, with $(\mathfrak{s}, \mathfrak{h}') \models^{\mathfrak{P}}_{\mathcal{R}} \xi$ and $(\mathfrak{s}, \mathfrak{h}'') \models^{\mathfrak{P}}_{\mathcal{R}} \xi'$. Since $\mathcal{R}$ is $\mathfrak{h}$-regular, ξ and ξ' are of the form $\exists \boldsymbol{u} \, (x \stackrel{p}{\mapsto} (\boldsymbol{v}) \circ \bigcirc_{i=1}^{n} Q_i(u_i, \boldsymbol{y}_i, \boldsymbol{q}_i) \circ \phi)$ and $\exists \boldsymbol{u}' \, (x \stackrel{p'}{\mapsto} (\boldsymbol{v}') \circ \bigcirc_{i=1}^{n'} Q_i(u'_i, \boldsymbol{y}_i, \boldsymbol{q}'_i) \circ \phi'$, respectively, with $\boldsymbol{u} \subseteq \boldsymbol{v}$, $\boldsymbol{u}' \subseteq \boldsymbol{v}'$, $\boldsymbol{u} = (u_1, \ldots, u_n)$, $\boldsymbol{u}' = (u'_1, \ldots, u'_{n'})$, $\boldsymbol{v} = (v_1, \ldots, v_k)$, $\boldsymbol{v}' = (v'_1, \ldots, v'_{k'})$, and ϕ, ϕ' are pure. Then there exist stores $\mathfrak{s}', \mathfrak{s}''$, coinciding with $\mathfrak{s}$ on all variables not occurring in $\boldsymbol{u}, \boldsymbol{u}'$, respectively, as well as sequences of pariwise disjoint heaps $\mathfrak{h}'_0, \ldots, \mathfrak{h}'_n$ and $\mathfrak{h}''_0, \ldots, \mathfrak{h}''_{n'}$ such that $(\mathfrak{s}', \mathfrak{h}'_0) \models^{\mathfrak{P}}_{\mathcal{R}} x \stackrel{p}{\mapsto} (\boldsymbol{v})$, $(\mathfrak{s}'', \mathfrak{h}''_0) \models^{\mathfrak{P}}_{\mathcal{R}} x \stackrel{p'}{\mapsto} (\boldsymbol{v}')$, $(\mathfrak{s}', \mathfrak{h}'_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q_i(u_i, \boldsymbol{y}_i, \boldsymbol{q}_i)$ (for all $i \in \{1, \ldots, n\}$), $(\mathfrak{s}'', \mathfrak{h}''_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q'_i(u'_i, \boldsymbol{y}'_i, \boldsymbol{q}_i)$ (for all $i \in \{1, \ldots, n'\}$), $\mathfrak{s}' \models^{\mathfrak{P}} \phi$ and $\mathfrak{s}'' \models^{\mathfrak{P}} \phi'$. Thus $(\mathfrak{s}(x), \mathfrak{s}'(\boldsymbol{v}), \mathfrak{s}(p)) \in \mathfrak{h}'$ and $(\mathfrak{s}(x), \mathfrak{s}''(\boldsymbol{v}'), \mathfrak{s}(p')) \in \mathfrak{h}''$. As $\mathfrak{h}' \sqcup \mathfrak{h}'' = \mathfrak{h}$, we deduce that $k = k'$, $\mathfrak{s}'(\boldsymbol{v}) = \mathfrak{s}''(\boldsymbol{v}')$ and $\mathfrak{s}(p) \oplus \mathfrak{s}(p')$ is defined, so that $\mathfrak{h}_0 \stackrel{\text{def}}{=} \{(\mathfrak{s}(x), \mathfrak{s}'(\boldsymbol{v}), \mathfrak{s}(p) \oplus \mathfrak{s}(p'))\} = \mathfrak{h}'_0 \sqcup \mathfrak{h}''_0 \leq \mathfrak{h}$.

Now assume (for the sake of contradiction) that there exists $i \in \{1, \ldots, n\}$ such that $v_i \notin \boldsymbol{u}$ and $v'_i \in \boldsymbol{u}'$. Then $v'_i \in roots(\xi')$, we get $\mathfrak{s}''(v'_i) \in dom(\mathfrak{h}'')$. As $\mathfrak{s}'(\boldsymbol{v}) = \mathfrak{s}''(\boldsymbol{v}')$ and $dom(\mathfrak{h}'') \subseteq dom(\mathfrak{h})$ we deduce that $\mathfrak{s}'(v_i) \in dom(\mathfrak{h})$ hence $\mathfrak{s}(v_i) \in dom(\mathfrak{h})$ (since $v_i \notin \boldsymbol{u}$), thus $\mathfrak{s}(v_i) = \mathfrak{s}(x)$, by the hypothesis of the lemma. But then $\mathfrak{s}(x) \in dom(\mathfrak{h}''_0) \cap dom(\mathfrak{h}''_i)$, which contradicts the fact that $\mathfrak{h}''_0$ and $\mathfrak{h}''_i$ are disjoint. Similarly, there is no $i \in \{1, \ldots, n'\}$ such that $v'_i \notin \boldsymbol{u}'$ and $v_i \in \boldsymbol{u}$. This entails that $\{i \in \{1, \ldots, k\} \mid v_i \in \boldsymbol{u}\} = \{i \in \{1, \ldots, k\} \mid v'_i \in \boldsymbol{u}'\}$ and, as $\boldsymbol{u} \subseteq \boldsymbol{v}$ and $\boldsymbol{u}' \subseteq \boldsymbol{v}'$, we deduce that $n = n'$ (since $\mathfrak{s}'(\boldsymbol{v}) = \mathfrak{s}''(\boldsymbol{v}')$ and $\mathfrak{s}'$ and $\mathfrak{s}''$ are injective on $u_1, \ldots, u_n$ and $u'_1, \ldots, u'_{n'}$, respectively). By α-renaming, we may then assume that $\boldsymbol{u} = \boldsymbol{u}'$ and $\mathfrak{s}' = \mathfrak{s}''$, thus $\mathfrak{s}' \models^{\mathfrak{P}}_{\mathcal{R}} \boldsymbol{v} \simeq \boldsymbol{v}'$. We have $\mathfrak{h}' \sqcup \mathfrak{h}'' = \mathfrak{h}$, i.e., $(\mathfrak{h}'_0 \sqcup \ldots \sqcup \mathfrak{h}'_n) \sqcup (\mathfrak{h}''_0 \sqcup \ldots \sqcup \mathfrak{h}''_n) = \mathfrak{h}$, and $\mathfrak{h} = (\mathfrak{h}'_0 \sqcup \mathfrak{h}''_0) \sqcup \ldots \sqcup (\mathfrak{h}'_n \sqcup \mathfrak{h}''_n)$. Let $\mathfrak{h}_i \stackrel{\text{def}}{=} \mathfrak{h}'_i \sqcup \mathfrak{h}''_i$ for all $i \in \{0, \ldots, n\}$, so that $\mathfrak{h} = \mathfrak{h}_0 \sqcup \ldots \sqcup \mathfrak{h}_n$. Note that $\mathfrak{s}'(u_i) \in dom(\mathfrak{h}'_i)$ (by Proposition 8), and $\mathfrak{s}(x) \in dom(\mathfrak{h}'_0)$ which entails, as $\mathfrak{h}'_0, \ldots, \mathfrak{h}'_n$ are disjoint and $dom(\mathfrak{h}'_i) \subseteq dom(\mathfrak{h})$, that the entailment $\mathfrak{s}'(y) \in dom(\mathfrak{h}'_i) \implies \mathfrak{s}'(y) = \mathfrak{s}'(u_i)$ holds for all $i \in \{1, \ldots, n\}$. By symmetry, $\mathfrak{s}'(y) \in dom(\mathfrak{h}''_i) \implies \mathfrak{s}'(y) = \mathfrak{s}'(u_i)$ also holds for all $i \in \{1, \ldots, n\}$, hence $\mathfrak{s}'(y) \in dom(\mathfrak{h}_i) \implies \mathfrak{s}'(y) = \mathfrak{s}'(u_i)$. Since $|\mathfrak{h}_0| = 1$, necessarily $|\mathfrak{h}_i| < |\mathfrak{h}|$, for all $i \in \{1, \ldots, n\}$. By the induction hypothesis, we deduce that $(\mathfrak{s}', \mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} Q_i(u_i, \boldsymbol{y}_i, \boldsymbol{p}) \triangledown Q'_i(u_i, \boldsymbol{y}'_i, \boldsymbol{p}'')$, so that

$(\mathfrak{s}', \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} x \overset{q}{\mapsto} (\boldsymbol{v}) \circ \bigcirc_{i=1}^{n} Q_i(u_i, \boldsymbol{y}_i, \boldsymbol{p}) \triangledown Q'_i(u_i, \boldsymbol{y}'_i, \boldsymbol{p}'') \circ \phi \circ \phi' \circ (\boldsymbol{v} \simeq \boldsymbol{v}')$. By definition of the rules of $P(x, \boldsymbol{y}, \boldsymbol{p}) \triangledown P'(x, \boldsymbol{y}', \boldsymbol{p}')$, this entails that $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} P(x, \boldsymbol{y}, \boldsymbol{p}) \triangledown P'(x, \boldsymbol{y}', \boldsymbol{p}')$.

J Proof of Proposition 30

The direct implication is immediate to prove. We now establish the converse. If $\rho_{\mathfrak{a}}$ is $\mathfrak{P}$-satisfiable then there exists a store $\mathfrak{s}$ such that $\mathfrak{s} \models^{\mathfrak{P}} \rho_{\mathfrak{a}}$ (i.e., $(\mathfrak{s}, \emptyset) \models^{\mathfrak{P}} \rho_{\mathfrak{a}}$). Since $\sim_{\mathfrak{a}}$ is an equivalence relation on $V_{\mathfrak{a}} \cap \mathcal{V}_1$, it is clear that there exists a store $\hat{\mathfrak{s}}$ such that $\hat{\mathfrak{s}}(x) = \mathfrak{s}(x)$ holds for all $x \in \mathcal{V}_{\mathtt{p}}$, and for all $y, z \in V_{\mathfrak{a}} \cap \mathcal{V}_1$: $\hat{\mathfrak{s}}(y) = \hat{\mathfrak{s}}(z) \iff y \sim_{\mathfrak{a}} z$ (it suffices to associates all equivalence classes of $\sim_{\mathfrak{a}}$ with pairwise distinct arbitrarily chosen locations). As $V_{\mathfrak{a}}$ is finite, there exists a heap $\hat{\mathfrak{h}}$ such that $dom(\mathfrak{h}) = \hat{\mathfrak{s}}(V_{\mathfrak{a}})$ (e.g., $\hat{\mathfrak{h}} = \{(\ell, \pi) \mid \ell \in dom(\mathfrak{h})\}$ where π is some arbitrary permission in $\mathcal{P}_{\mathfrak{P}}$). As the truth value of $\mathfrak{a}$ only depends on the interpretation of variables of sort $\mathtt{p}$, we have $\hat{\mathfrak{s}} \models^{\mathfrak{P}} \mathfrak{a}$, so that $(\hat{\mathfrak{s}}, \hat{\mathfrak{h}}) \models^{\mathfrak{P}} \mathfrak{a}$.

K Proof of Lemma 31

We need to establish a slightly more general property: Let ϕ be a $\circ$-formula, let $(\mathfrak{s}, \mathfrak{h})$ be a structure and let $V \supseteq fv(\phi)$. We show that the two following assertions hold: (i) If $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$ then there exists a heap abstraction $(V, \sim, A, \rho) \in \mathfrak{A}(\phi)$ such that $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} (V, \sim, A, \rho)$. (ii) If $(V, \sim, A, \rho) \in \mathfrak{A}(\phi)$, $\sim = \{(u, v) \in (V \cap \mathcal{V}_1)^2 \mid \mathfrak{s}(u) = \mathfrak{s}(v)\}$ and $\mathfrak{s} \models^{\mathfrak{P}} \rho$, then for every store $\hat{\mathfrak{s}}$ such that $\hat{\mathfrak{s}}(x) = \mathfrak{s}(x)$ for all $x \in \mathcal{V}_{\mathtt{p}}$ and $\mathfrak{s}(\hat{x}) = \hat{\mathfrak{s}}(y) \iff \mathfrak{s}(x) = \mathfrak{s}(y)$ for all $x, y \in V \cap \mathcal{V}_1$, and for every infinite subset L of $\mathcal{L}$, there exists a heap $\hat{\mathfrak{h}}$ such that $(\hat{\mathfrak{s}}, \hat{\mathfrak{h}}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$, with $dom(\hat{\mathfrak{h}}) \subseteq L \cup \hat{\mathfrak{s}}(A)$.

1. The proof is by induction on the satisfiability relation. Let $\sim = \{(x, y) \in (V \cap \mathcal{V}_1)^2 \mid \mathfrak{s}(x) = \mathfrak{s}(y)\}$.
   - If $\phi = x \overset{p}{\mapsto} (y_1, \ldots, y_k)$ then necessarily $dom(\mathfrak{h}) = \{\mathfrak{s}(x)\}$ and $\mathfrak{s}(p)$ is defined, so that $\mathfrak{s} \models^{\mathfrak{P}} def(p)$. Moreover $(V, \sim, \{y \in V \mid y \sim x\}, def(p)) \in \mathfrak{A}(\phi)$, by definition of $\mathfrak{A}(\phi)$. By definition of $\sim$, we have $\forall x, y \in V_{\mathfrak{a}} \cap \mathcal{V}_1$: $x \sim y \iff \mathfrak{s}(x) = \mathfrak{s}(y)$ and $\{y \in V \mid y \sim x\} = \{y \in V \mid \mathfrak{s}(y) = \mathfrak{s}(x)\} = \{y \in V \mid \mathfrak{s}(y) \in dom(\mathfrak{h})\}$, thus $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} (V, \sim, \{y \in V \mid y \sim x\}, def(p))$.
   - If $\phi = x \simeq y$ (resp. $x \not\simeq y$) with $x, y \in \mathcal{V}_1$, then $\mathfrak{h} = \emptyset$. Moreover $\mathfrak{s}(x) = \mathfrak{s}(y)$ (resp. $\mathfrak{s}(x) \neq \mathfrak{s}(y)$) so that $x \sim y$ (resp. $x \not\sim y$) and $(V, \sim, \emptyset, \mathtt{emp}) \in \mathfrak{A}(\phi)$, by definition of $\mathfrak{A}(\phi)$. As $\sim = \{(x, y) \in (V \cap \mathcal{V}_1)^2 \mid \mathfrak{s}(x) = \mathfrak{s}(y)\}$, $dom(\mathfrak{h}) = \emptyset$, and $\mathfrak{s} \models^{\mathfrak{P}} \mathtt{emp}$ we have $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} (V, \sim, \emptyset, \mathtt{emp})$.
   - If ϕ is a permission formula then $\mathfrak{h} = \emptyset$, and $(V, \sim, \emptyset, \phi) \in \mathfrak{A}(\phi)$ (by definition of $\mathfrak{A}(\phi)$). As $\sim = \{(x, y) \in (V \cap \mathcal{V}_1)^2 \mid \mathfrak{s}(x) = \mathfrak{s}(y)\}$, $dom(\mathfrak{h}) = \emptyset$ and $(\mathfrak{s}, \emptyset) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$, we get $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} (V, \sim, \emptyset, \phi)$.
   - Assume that $\phi = \exists x\, \psi$. By α-renaming, we assume that $x \notin V$. There exists a store $\mathfrak{s}'$ coinciding with $\mathfrak{s}$ on all the variables distinct from x such that $(\mathfrak{s}', \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$. Let $V' = V \cup \{x\}$, so that $fv(\psi) \subseteq V'$. By

the induction hypothesis, $\mathfrak{A}(\psi)$ contains a tuple $(V', \sim', A', \rho)$ such that $(\mathfrak{s}', \mathfrak{h}) \models^{\mathfrak{P}} (V', \sim', A', \rho)$. Then $\sim' = \{(y, z) \in (V' \cap \mathcal{V}_1) \mid \mathfrak{s}'(y) = \mathfrak{s}'(z)\}$, $A' = \{y \in V' \mid \mathfrak{s}'(y) \in dom(\mathfrak{h})\}$ and $\mathfrak{s}' \models^{\mathfrak{P}} \rho$. By definition of $\mathfrak{A}(\phi)$, we deduce that the heap abstraction $\mathfrak{a} \stackrel{\text{def}}{=} (V' \setminus \{x\}, \{(y, z) \in V' \mid y \sim' z \wedge y, z \neq x\}, A' \setminus \{x\}, \rho)$ is in $\mathfrak{A}(\psi)$. By definition of V', we have $V' \setminus \{x\} = V$, $\{(y, z) \in (V' \cap \mathcal{V}_1)^2 \mid y \sim' z \wedge y \neq x \wedge z \neq x\} = \{(y, z) \in (V \cap \mathcal{V}_1)^2 \mid \mathfrak{s}'(y) = \mathfrak{s}'(z)\}$ and $A' \setminus \{x\} = \{y \in V \mid \mathfrak{s}'(y) \in dom(\mathfrak{h})\}$, so that $\mathfrak{a} = (V, \{(y, z) \in (V \cap \mathcal{V}_1)^2 \mid \mathfrak{s}'(y) = \mathfrak{s}'(z)\}, \{y \in V \mid \mathfrak{s}'(y) \in dom(\mathfrak{h}), \rho\}$. As $\mathfrak{s}'$ and $\mathfrak{s}$ coincides on all variables $y \neq x$ (hence on all variables in V), and since ρ does not contain x (since existential variables are of sort $\mathtt{l}$) we get $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} \mathfrak{a}$.

- Assume that $\phi = \phi_1 \circ \phi_2$. Then there exist disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ with $(\mathfrak{s}, \mathfrak{h}_i) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$ and $\mathfrak{h} = \mathfrak{h}_1 \sqcup \mathfrak{h}_2$. Let $A_i = \{x \in V \mid \mathfrak{s}(x) \in dom(\mathfrak{h}_i)\}$. We have $fv(\phi_i) \subseteq fv(\phi) \subseteq V$, thus, by the induction hypothesis, there exist $(V, \sim_i, A_i, \rho_i) \in \mathfrak{A}(\phi_i)$ such that $(\mathfrak{s}, \mathfrak{h}_i) \models^{\mathfrak{P}} (V, \sim_i, A_i, \rho_i)$, so that $\sim_1 = \sim_2 = \{(x, y) \in (V \cap \mathcal{V}_1)^2 \mid \mathfrak{s}(x) = \mathfrak{s}(y)\}$, $A_i = \{x \in V \mid \mathfrak{s}(x) \in dom(\mathfrak{h}_i)\}$ and $\mathfrak{s} \models^{\mathfrak{P}} \rho_i$. As $dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2) = \emptyset$ necessarily $A_1 \cap A_2 = \emptyset$ thus $(V, \sim_1, A_1 \cup A_2, \rho_1 \circ \rho_2) \in \mathfrak{A}(\phi)$. Then $\mathfrak{s} \models^{\mathfrak{P}} \rho_1 \circ \rho_2$, and since $dom(\mathfrak{h}) = dom(\mathfrak{h}_1) \cup dom(\mathfrak{h}_2)$, we get $\{x \mid \mathfrak{s}(x) \in dom(\mathfrak{h})\} = \{x \mid \mathfrak{s}(x) \in dom(\mathfrak{h}_1)\} \cup \{x \mid \mathfrak{s}(x) \in dom(\mathfrak{h}_2)\} = A_1 \cup A_2$, thus $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} (V, \sim_1, A_1 \cup A_2, \rho_1 \circ \rho_2)$.
- Assume that ϕ is a predicate atom. Then there exists ψ such that $\phi \Leftarrow_{\mathcal{R}} \psi$ and $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$. We have $fv(\psi) \subseteq fv(\phi) \subseteq V$. By the induction hypothesis, there exists a heap abstraction $(V, \sim, A, \rho) \in \mathfrak{A}(\phi)$ such that $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} (V, \sim, A, \rho)$. By definition of $\mathfrak{A}(\phi)$, $(V, \sim, A, \rho)$ is in $\mathfrak{A}(\phi)$.

2. The proof is by induction on $\mathfrak{A}(\phi)$.
   - Assume that $\phi = x \overset{p}{\mapsto} (y_1, \ldots, y_k)$. Since $(V, \sim, A, \rho) \in \mathfrak{A}(\phi)$, we must have $A = \{y \in (V \cap \mathcal{V}_1) \mid y \sim x\}$ (thus $x \in A$) and $\rho = def(p)$, by definition of $\mathfrak{A}(\phi)$. Since $\mathfrak{s} \models^{\mathfrak{P}} \rho$ necessarily $\hat{\mathfrak{s}} \models^{\mathfrak{P}} \rho$, as $\hat{\mathfrak{s}}$ and $\mathfrak{s}$ coincide on all variables of sort $\mathtt{p}$. Consequently, $\hat{\mathfrak{s}}(p)$ must be defined. Consider the heap $\hat{\mathfrak{h}} \stackrel{\text{def}}{=} \{(\hat{\mathfrak{s}}(x), \hat{\mathfrak{s}}(y_1), \ldots, \hat{\mathfrak{s}}(y_k), \hat{\mathfrak{s}}(p))\}$. By definition of the semantics, $(\hat{\mathfrak{s}}, \hat{\mathfrak{h}}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$, moreover $dom(\mathfrak{h}) = \{\hat{\mathfrak{s}}(x)\} \subseteq \hat{\mathfrak{s}}(A)$, since $x \in A$.
   - Assume that $\phi = x \simeq y$. As $(V, \sim, A, \rho) \in \mathfrak{A}(\phi)$, we must have $x \sim y$, by definition of $\mathfrak{A}(\phi)$. Since $\sim = \{(u, v) \in (V \cap \mathcal{V}_1)^2 \mid \mathfrak{s}(u) = \mathfrak{s}(v)\}$, this entails that $\mathfrak{s}(x) = \mathfrak{s}(y)$, so that $\hat{\mathfrak{s}}(x) = \hat{\mathfrak{s}}(y)$, and by letting $\hat{\mathfrak{h}} = \emptyset$, we get $(\hat{\mathfrak{s}}, \hat{\mathfrak{h}}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$.
   - The proof is similar if $\phi = x \not\simeq y$.
   - Assume that ϕ is a permission formula. In this case, we must have $\rho = \phi$, hence $\mathfrak{s} \models^{\mathfrak{P}} \phi$. Since $\hat{\mathfrak{s}}$ coincides with $\mathfrak{s}$ on all variables of sort $\mathtt{p}$, this entails that $\hat{\mathfrak{s}} \models^{\mathfrak{P}} \phi$, so that $(\hat{\mathfrak{s}}, \emptyset) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$.
   - Assume that $\phi = \exists x \, \psi$. As $(V, \sim, A, \rho) \in \mathfrak{A}(\phi)$, we deduce, by definition of $\mathfrak{A}(\phi)$, that $(V', \sim', A', \rho') \in \mathfrak{A}(\psi)$ with $V = V' \setminus \{x\}$, $\sim = \{(u, v) \mid u \sim' v \wedge u \neq x \wedge v \neq x\}$ $A = A' \setminus \{x\}$ and $\rho = \rho'$. Moreover, since $(\mathfrak{s}, \mathfrak{h}) \models^{\mathfrak{P}} (V, \sim, A, \rho)$, we deduce that $y \sim z \implies \mathfrak{s}(y) = \mathfrak{s}(z)$, for

all $y, z \in (V \cap \mathcal{V}_1)$. By the hypothesis of the lemma, this entails that the implication $\forall y, z \in (V \cap \mathcal{V}_1)\ (y \sim z \implies \hat{\mathfrak{s}}(y) = \hat{\mathfrak{s}}(z))$ also holds. Consider any stores $\mathfrak{s}'$ and $\hat{\mathfrak{s}'}$ coinciding with $\mathfrak{s}$ and $\hat{\mathfrak{s}}$ (respectively) on all variables distinct from x and such that:

- If $x \sim' y$ for some variable $y \in V$ then $\mathfrak{s}'(x) \stackrel{\text{def}}{=} \mathfrak{s}(y)$ and $\hat{\mathfrak{s}}'(x) \stackrel{\text{def}}{=} \hat{\mathfrak{s}}(y)$ (note that $\mathfrak{s}(y)$ and $\hat{\mathfrak{s}}(y)$ does not depend on the choice of $y \in V$ in the same equivalence class for $\sim$).
- Otherwise, we let $\mathfrak{s}'(x) = \hat{\mathfrak{s}}'(x) = \ell$, where ℓ is an arbitrarily chosen location in $L \backslash \hat{\mathfrak{s}}(V)$. Note that such a location exists since L is infinite.

By construction, it is clear that the equivalence $\mathfrak{s}'(y) = \mathfrak{s}'(z) \iff y \sim' z \iff \hat{\mathfrak{s}}'(y) = \hat{\mathfrak{s}}'(z)$ holds for all $y, z \in V' \cap \mathcal{V}_1$. Moreover, $\mathfrak{s}'$ coincides with $\mathfrak{s}$ on all permission variables (as x is of sort $\mathtt{1}$, since quantification over permission variables is not allowed), thus $\mathfrak{s}' \models^{\mathfrak{P}} \rho' = \rho$. Let $L' = L \setminus \{\hat{\mathfrak{s}}'(x)\}$. By the induction hypothesis, applied on $(V', \sim', A', \rho')$, ψ, $\mathfrak{s}'$, $\hat{\mathfrak{s}}'$ and L', there exists a heap $\hat{\mathfrak{h}}$ such that $(\hat{\mathfrak{s}}', \hat{\mathfrak{h}}) \models^{\mathfrak{P}}_{\mathcal{R}} \psi$ and $dom(\hat{\mathfrak{h}}) \subseteq L' \cup \hat{\mathfrak{s}}'(A')$. This entails that $(\hat{\mathfrak{s}}, \hat{\mathfrak{h}}) \models^{\mathfrak{P}}_{\mathcal{R}} \exists x\, \psi$. It only remains to show that $dom(\hat{\mathfrak{h}}) \subseteq L \cup \hat{\mathfrak{s}}(A)$. Assume, for the sake of contradiction, that $dom(\hat{\mathfrak{h}})$ contains a location ℓ not occurring in $L \cup \hat{\mathfrak{s}}(A)$. As $\ell \in L' \cup \hat{\mathfrak{s}}'(A')$, with $L' = L \setminus \{\hat{\mathfrak{s}}'(x)\}$ and $A = A' \setminus \{x\}$, this entails that $\ell = \hat{\mathfrak{s}}'(x)$ with $x \in A'$ and $\hat{\mathfrak{s}'}(x) \notin L$. By definition of $\hat{\mathfrak{s}}'$, the latter assertion entails that there exists a variable $y \in \mathcal{V}_1 \cap V$ such that $x \sim y$ and $\hat{\mathfrak{s}}'(x) = \hat{\mathfrak{s}}(y)$. We must have $y \in A'$ (since $x \in A'$ and A' is closed under $\sim$) thus $y \in A$ (as $x \neq y$, since $x \notin V$), hence $\hat{\mathfrak{s}}(y) = \ell \in \hat{\mathfrak{s}}(A)$, which contradicts our assumption.

– Assume that $\phi = \phi_1 \circ \phi_2$. As $(V, \sim, A, \rho) \in \mathfrak{A}(\phi)$, we deduce, by definition of $\mathfrak{A}(\phi)$, that for all $i = 1, 2$, $\mathfrak{A}(\phi_i)$ contains a heap abstraction $(V, \sim, A_i, \rho_i)$ with $A = A_1 \cup A_2$, $\rho = \rho_1 \circ \rho_2$ and $A_1 \cap A_2 = \emptyset$. As $\mathfrak{s} \models^{\mathfrak{P}} \rho$, we get $\mathfrak{s} \models^{\mathfrak{P}} \rho_i$. Moreover, $\mathfrak{s}(A_1) \cap \mathfrak{s}(A_2) = \emptyset$. Indeed, if $\mathfrak{s}(x_i) \in \mathfrak{s}(A_i)$ with $x_i \in A_i$, then $x_1 \sim x_2$ (as $\sim = \{(u, v) \in (V \cap \mathcal{V}_1)^2 \mid \mathfrak{s}(u) = \mathfrak{s}(v)\}$) which entails (by Definition 28) that $x_2 \in A_1$ and $x_1 \in A_2$, contradicting the fact that $A_1 \cap A_2 = \emptyset$. Let L_1, L_2 be disjoint infinite subsets of L also disjoint from $\hat{\mathfrak{s}}(V)$ (such sets exist since L is infinite). By the induction hypothesis, there exist heaps $\hat{\mathfrak{h}}_i$ such that $(\hat{\mathfrak{s}}, \hat{\mathfrak{h}_i}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_i$ and $dom(\hat{\mathfrak{h}}_i) \subseteq L_i \cup \hat{\mathfrak{s}}(A_i)$. As $L_1 \cap L_2 = \emptyset$, $\mathfrak{s}(A_1) \cap \mathfrak{s}(A_2) = \emptyset$ and $(L_1 \cup L_2) \cap \hat{\mathfrak{s}}(V) = \emptyset$, we have $dom(\hat{\mathfrak{h}}_1) \cap dom(\hat{\mathfrak{h}}_2) = \emptyset$, so that $\hat{\mathfrak{h}}_1$ and $\hat{\mathfrak{h}}_2$ are disjoint and $(\hat{\mathfrak{s}}, \hat{\mathfrak{h}}_1 \sqcup \hat{\mathfrak{h}}_2) \models^{\mathfrak{P}}_{\mathcal{R}} \phi_1 \circ \phi_2 = \phi$. Moreover, $dom(\hat{\mathfrak{h}}_1 \sqcup \hat{\mathfrak{h}}_2) \subseteq L \cup \hat{\mathfrak{s}}(A)$.

– Assume that ϕ is a predicate atom. Since $(V, \sim, A, \rho) \in \mathfrak{A}(\phi)$, necessarily $(V, \sim, A, \rho) \in \mathfrak{A}(\xi)$ with $\phi \Leftarrow_{\mathcal{R}} \xi$. By the induction hypothesis there exists a heap $\hat{\mathfrak{s}}$ such that $(\hat{\mathfrak{s}}, \hat{\mathfrak{h}}) \models^{\mathfrak{P}}_{\mathcal{R}} \xi$ and $dom(\mathfrak{h}) \subseteq L \cup \hat{\mathfrak{s}}(A)$. By definition of the semantics, we also have $(\hat{\mathfrak{s}}, \hat{\mathfrak{h}}) \models^{\mathfrak{P}}_{\mathcal{R}} \phi$.

L Proof of Theorem 32

We assume for simplicity that ϕ contains no quantifier and no points-to atom[12].

[12] It is clear that this is not restrictive: any existential variable may be replaced by a fresh free variable, and any points-to atom $x \overset{z}{\mapsto} (y_1, \ldots, y_k)$ may be replaced

Step 1 *(Normalization).* Using Lemma 23, one first compute a set of normalized formulas Φ_2 such that ϕ is $(\mathcal{R}, \mathfrak{P})$-satisfiable iff there exists $\phi_2 \in \Phi_2$ such that ϕ_2 is $(\mathcal{R}, \mathfrak{P})$-satisfiable, and all formulas in Φ_2 are separating conjunctions of $\circ$-formulas (all the existential variables occurring in formulas in Φ_2 are replaced by fresh free variables).

Step 2 *(Elimination of* $*$*).* By definition, every formula $\phi_2 \in \Phi_2$ may be written on the form $\phi' \circ (\mathbin{\circledast}_{i=1}^{n} \chi_i)$ (†), where $\phi', \chi_1, \ldots, \chi_n$ are $\circ$-formulas (with initially $\phi' = \mathtt{emp}$). We show that, if $n > 1$, then the above formula can be reduced into a strictly smaller sat-equivalent formula that is still of the form (†). Let x be any variable occurring in $\mathit{roots}(\chi_i)$, for some $i \in \{1, \ldots, n\}$ (if no such x exists then $\mathbin{\circledast}_{i=1}^{n} \chi_i$ is pure and the symbol $*$ may be replaced by $\circ$, yielding a formula of the form (†) with $n = 1$). Let I be the set of indices in $\{1, \ldots, n\}$ such that $x \in \mathit{roots}(\chi_i)$. By symmetry, we assume that $I = \{1, \ldots, m\}$ for some $m \in \{1, \ldots, n\}$. Every formula χ_i must be of the form $\delta_i * \chi_i'$, where δ_i is a predicate atom, $\mathit{roots}(\delta_i) = \{x\}$ and $x \notin \mathit{roots}(\chi_i')$ (if χ_i contains two distinct atoms with the same root x then by Proposition 8, χ_i is $(\mathcal{R}, \mathfrak{P})$-unsatisfiable and the entire formula can be dismissed). By Lemma 26, applied with $\phi_i = \delta_i$, $\psi_i = \chi_i'$ and $\psi' = \mathbin{\circledast}_{i=m+1}^{n} \chi_i$, we deduce that $\phi' \circ (\mathbin{\circledast}_{i=1}^{n} \chi_i)$ is sat-equivalent to $(\phi' \circ \mathbin{\circledast}_{i=1}^{m} \delta_i) \circ (\mathbin{\circledast}_{i=1}^{m} \chi_i' * \mathbin{\circledast}_{i=m+1}^{n} \chi_i)$. Let $\phi_2' = \phi' \circ (\delta_1 \triangledown \ldots \triangledown \delta_m) \circ (\mathbin{\circledast}_{i=1}^{m} \chi_i' * \mathbin{\circledast}_{i=m+1}^{n} \chi_i)$. By Lemma 27, $\mathbin{\circledast}_{i=1}^{m} \delta_i \equiv_{\mathcal{R}}^{\mathfrak{P}} \delta_1 \triangledown \ldots \triangledown \delta_m$ (the hypothesis of the lemma is satisfied since all formulas are normalized), thus ϕ_2' and $\phi' \circ (\mathbin{\circledast}_{i=1}^{n} \chi_i)$ are sat-equivalent. As the weight of the symbol $*$ is strictly greater than that of $\circ$, it is clear that $|\phi_2'| < |\phi' \circ (\mathbin{\circledast}_{i=1}^{n} \chi_i)|$. Moreover, ϕ_2' is also of form (†). By repeating the above transformation, we eventually obtain a formula ϕ_3 of the form (†) above, with $n = 1$, i.e., a $\circ$-formula. We thus get a set of $\circ$-formulas Φ_3 such that ϕ is $(\mathcal{R}, \mathfrak{P})$-satisfiable iff there exists $\phi_3 \in \Phi_3$ such that ϕ_3 is $(\mathcal{R}, \mathfrak{P})$-satisfiable.

Step 3 *(Abstractions).* It only remains to check that one of $\circ$-formulas $\phi_3 \in \Phi_3$ is $(\mathcal{R}, \mathfrak{P})$-satisfiable. By Lemma 31, it is sufficient to compute the set $\mathfrak{A}(\phi_3)$ and test whether it contains a heap abstraction $(V, \sim, A, \rho)$ that is $\mathfrak{P}$-satisfiable. By Proposition 30, we only have to check that ρ is $\mathfrak{P}$-satisfiable, which is decidable by the hypothesis of the lemma.

We now analyze the complexity of the algorithm and we show that it runs in exponential time.

1. At Step 1, one replacement is performed for each atom δ and for each variable. Moreover, each replacement may in turn introduce new atoms and new variables in the formula (however, as shown above, the application of the replacement operation on these new variables do not add further variables). Note that the replacement does not increase the number of $\circ$-formulas occurring as operands of $*$: only the number of atoms inside $\circ$-formulas may increase. Thus the number of $\circ$-formulas is at most $|\phi|$, as

by a predicate atom $P(x, y_1, \ldots, y_k, z)$, with the rule $P(u, v_1, \ldots, v_k, w) \Leftarrow u \overset{w}{\mapsto} (v_1, \ldots, v_k)$.

initially ϕ contains at most $|\phi|$ spatial predicate atoms and no occurrence of $\circ$. Moreover, the replacement cannot be applied twice with the same variable and the same $\circ$-formula. We denote by N be the maximal number of variables introduced by one single replacement. We get a total of at most $fv(\phi) + (N \times fv(\phi) \times |\phi|) \leq (N+1) \times |\phi|^2$ variables. As all the spatial atoms occurring in the same $\circ$-formula must have distinct roots (otherwise the formula is $(\mathcal{R}, \mathfrak{P})$-unsatisfiable by Proposition 8 and can be dismissed), this yields a total of at most $(N+1) \times |\phi|^2$ atoms in each $\circ$-formula, hence of at most $(N+1) \times |\phi|^3$ atoms in every formula $\phi_2 \in \Phi_2$. Each replacement may produce a new derived predicate, which may be of the form $\delta[x]^-$ or $(P(x, \boldsymbol{z}, p) \circ \bigcirc_{i=1}^{n} P_i(x_i, \boldsymbol{y}_i, p)) \mathrel{-\!\!\bullet} Q(\boldsymbol{y}, p)$ (with $\delta = Q(\boldsymbol{y}, p)$). In the former case, no new variable is introduced and the maximal arity increases by at most 1, whereas in the latter case, new variables $x_1, \ldots, x_n$ (with $n \leq N$) are added into the formula and the maximal arity of the predicates occurring in the formula is increased by at most n (see Definition 17). By Lemma 20, we have $\{x_1, \ldots, x_n\} \subseteq (x, \boldsymbol{z})|_{\gamma_{\mathcal{R}}(P)}$, thus $n \leq \mathit{card}(\gamma_{\mathcal{R}}(P))$. By Propositions 14 and 18, the computation of derived predicates cannot increase the maximal value of $\mathit{card}(\gamma_{\mathcal{R}}(P))$, so that $N \leq |\mathcal{R}|$ and $\mathit{card}(\gamma_{\mathcal{R}}(P)) \leq |\mathcal{R}|$, for every derived predicate P. Thus, in every formula $\phi_2 \in \Phi_2$, the number of variables added in the formula, the number of atoms and the number of derived predicates and their maximal arity are all polynomial w.r.t. $|\phi| + |\mathcal{R}|$, so that $|\phi_2|$ is polynomial w.r.t. $|\phi|$. Moreover, the total number of derived predicates is at most exponential w.r.t. $|\phi| + |\mathcal{R}|$. Indeed, it is easy to see that these derived predicates may be uniquely determined by choosing the initial atom δ in ϕ from which the derived predicate is computed and the sets of atoms that are removed from the call tree of δ (inside the initial set of predicates). At most $N+1$ such atoms are introduced at each replacement, and there are at most $(N+1) \times fv(\phi)$ replacements, yielding at most $2^{|\mathcal{R}| \times (N+1)^2 \times fv(\phi)}$ possible choices. Therefore, the total number of formulas in Φ_2 is at most exponential w.r.t. $|\phi| + |\mathcal{R}|$. It is clear that each formula ϕ_2 can be computed in polynomial time.

2. It is straightforward to check that for each formula $\phi_2 \in \Phi_2$, the corresponding $\circ$-formula $\phi_3 \in \Phi_3$ can be computed in polynomial time w.r.t. $|\phi_2|$ (each step of the transformation strictly decreases the size of the formula). Hence Step 2 can be performed in exponential time w.r.t. $|\phi|$.

3. At Step 3, for each formula $\phi_3 \in \Phi_3$, the test $\mathfrak{A}(\phi_3)$ may be computed by using a standard fixpoint computation algorithm, following the inductive rules in Definition 28. Observe that we only need to compute $\mathfrak{A}(\psi)$ for formulas ψ that occur in either ϕ_3 or in $\mathcal{R}$ (up to a renaming of variables), as other formulas does not interfere with the computation of $\mathfrak{A}(\phi_3)$. Moreover, we only need to consider heap abstractions $(V, \sim, A, \rho)$ such that V contains only variables in $fv(\psi)$ and existential variables in $\mathcal{R}$, and ρ contains only permission terms and permission predicates in ψ. It is clear that the number of such heap abstractions is simply exponential w.r.t. $|\phi_3|$, hence the computation can be performed in exponential time.

The EXPTIME-hardness proof goes by an easy reduction from the halting problem for alternating Turing machines (ATM) running in polynomial space. As APSPACE (the class of languages decidable in polynomial space by alternating Turing machines) is identical to EXPTIME, we get the result. Let $M = (Q, \Gamma, \delta, q_0, g)$ be an ATM, where Q is a finite set of states, Γ is a finite tape alphabet, $\delta \subseteq (Q \times \Gamma \times Q \times \Gamma \times \{\leftarrow, \rightarrow\})$ is a transition relation, $q_0 \in Q$ is the initial state and $g : Q \to \{\vee, \wedge\}$ specifies the type of each state. We assume, w.l.o.g., that $\Gamma \subseteq \mathcal{V}_1$, and we denote by $\boldsymbol{\gamma}$ any sequence containing all the elements of Γ. For every move $\mu \in \{\leftarrow, \rightarrow\}$, $\mu(i)$ is defined as $i-1$ if $\mu = \leftarrow$ and $i+1$ otherwise. Assume that the considered ATM runs in time at most $n \in \mathbb{N}$ on some word w (where n is polynomial w.r.t. $|w|$.) Let w_i $(1 \leq i \leq n)$ be the i-th character in w, where w_i is a blank symbol if $i > |w|$. For every natural number i with $1 \leq i \leq n$ and every $q \in Q$, we consider the predicate q^i of arity $2 + n + |\boldsymbol{\gamma}|$ associated with the rules:

$$q^i(x, y_1, \ldots, y_n, \boldsymbol{\gamma}, z) \Leftarrow \exists x' \quad x \overset{z}{\mapsto} (x') \\ \circ\, q^{\mu(i)}(x', y_1, \ldots, y_{i-1}, b, y_{i+1}, \ldots, y_n, \gamma, z) \circ y_i \simeq a \quad (9)$$

if $g(q) = \vee$, for all transitions $(q, a, q', b, \mu) \in \delta$ such that $1 \leq \mu(i) \leq n$.

$$q^i(x, y_1, \ldots, y_n, \boldsymbol{\gamma}, z) \Leftarrow \exists x'_1, \ldots, x'_k \quad x \overset{z}{\mapsto} (x'_1, \ldots, x'_k) \\ \circ \bigcirc_{j=1}^{k} q^{\mu_j(i)}(x'_j, y_1, \ldots, y_{i-1}, b_j, y_{i+1}, \ldots, y_n, \gamma, z) \circ y_i \simeq a \quad (10)$$

if $g(q) = \wedge$, for all $a \in \Gamma$, where (q, a, q'_j, b_j, μ_j) (for $j \in \{1, \ldots, k\}$) is the set of transitions of the form $(q, a, q', b, \mu) \in \delta$ with $1 \leq \mu(i) \leq n$.

It is clear that $p^1(x, w_1, \ldots, w_n, \gamma, z)$ is $(\mathcal{R}, \mathfrak{P})$-satisfiable iff the considered ATM terminates on the word w and accepts w (every model of $p^1(x, w_1, \ldots, w_n, \gamma, z)$ encodes an accepting derivation from the initial state p and the tape $w_1, \ldots, w_n$, where the head is at position 1). Moreover, the rules are $\exists$-restricted, with $\gamma_{\mathcal{R}}(q^i) = \emptyset$ and $\mathcal{P}^\star = \emptyset$.

M Proof of Theorem 33

As for Theorem 9, the proof is by reduction from the PCP (the same notations are used). Potential witnesses ω are encoded in a similar way than in the proof of Theorem 9, except that the use of the heaps $\mathfrak{h}^\mu$ and $\mathfrak{h}^\nu$ is avoided, instead the indices $I^\mu(\eta^\mu(j))$ and $I^\nu(\eta^\nu(j))$ are stored directly in the list $\mathfrak{h}'$. Moreover, an additional location ℓ'_j, called a *mark*, is added in each tuple, that is allocated and always refers to (). Finally, each cell has a double link to the next one (this will be useful so that the tail can be generated twice in the body of the rule). We get a heap of the following form:

$$\mathfrak{h}' = \{(\ell_j, \mathfrak{s}(I^\mu(\eta^\mu(j))), \mathfrak{s}(I^\nu(\eta^\nu(j))), \ell'_j, \ell_{j+1}, \ell_{j+1}, \pi'), (\ell'_j, \pi') \mid 1 \leq j \leq |\omega|\}$$

Moreover, the list is cyclic, i.e., we assume that $\ell_{|\omega|+1} = \ell_1$. The rules defining the predicate generating the lists $\mathfrak{h}'$ corresponding to potential witnesses are very similar to those given in the proof of Theorem 9, except that Q is not used (a predicate D is used instead to allocate marks). We assume that $\boldsymbol{v}$ contains all natural numbers in $\{1, \ldots, n\}$ and a variable u, denoting the first cell in the list. The predicate $P'(x'', y, \boldsymbol{v}, z)$ is called (taking advantage of the double link to the next element and from the fact that $*$ is used instead of $\circ$) to ensure that all the marks are pairwise distinct (it allocates a list in which all marks are distinct from y). This ensures that the mapping $\ell_j \mapsto \ell'_j$ is bijective, so that the cells can be unambiguously denoted by their marks.

$$P(x, \boldsymbol{v}, z) \Leftarrow \exists x', x'', y \quad x \overset{z}{\mapsto} (i, i, y, x', x'') * P'(x'', y, \boldsymbol{v}, z) * x'' \simeq x' * P_{2,2,i,i}(x', \boldsymbol{v}, z) * D(y, z) \quad \text{if } \mu_i|_1 = \nu_i|_1 \tag{11}$$

$$D(x, z) \quad \Leftarrow \quad x \quad \overset{z}{\mapsto} \quad () \tag{12}$$

$$P_{j_\mu, j_\nu, i_\mu, i_\nu}(x, \boldsymbol{v}, z) \Leftarrow \exists x', x'', y \quad x \overset{z}{\mapsto} (i_\mu, i_\nu, y, x', x'') * P'(x'', y) * x'' \simeq x' * * D(y, z) * P_{j_\mu+1, j_\nu+1, i_\mu, i_\nu}(x', \boldsymbol{v}, z) \quad \text{if } j_\mu \leq |\mu_{i_\mu}|,\ j_\nu \leq |\nu_{i_\nu}| \text{ and } \mu_{i_\mu}|_{j_\mu} = \nu_{i_\nu}|_{j_\nu} \tag{13}$$

$$P_{j_\mu, j_\nu, i_\mu, i_\nu}(x, \boldsymbol{v}, z) \Leftarrow \exists x', x'', y \quad x \overset{z}{\mapsto} (i'_\mu, i_\nu, y, x', x'') * P'(x'', y) * x'' \simeq x' * * D(y, z) * P_{2, j_\nu+1, i'_\mu, i_\nu}(x', \boldsymbol{v}, z) \quad \text{if } j_\mu = |\mu_{i_\mu}| + 1,\ j_\nu \leq |\nu_{i_\nu}| \text{ and } \mu_{i'_\mu}|_1 = \nu_{i_\nu}|_{j_\nu} \tag{14}$$

$$P_{j_\mu, j_\nu, i_\mu, i_\nu}(x, \boldsymbol{v}, z) \Leftarrow \exists x', x'', y \quad x \overset{z}{\mapsto} (i_\mu, i'_\nu, y, x', x'') * P'(x'', y) * x'' \simeq x' * * D(y, z) * P_{j_\mu+1, 2, i_\mu, i'_\nu}(x', \boldsymbol{v}, z) \quad \text{if } j_\mu \leq |\mu_{i_\mu}|,\ j_\nu = |\nu_{i_\nu}| + 1 \text{ and and } \mu_{i_\mu}|_{j_\mu} = \nu_{i'_\nu}|_1 \tag{15}$$

$$P_{j_\mu, j_\nu, i_\mu, i_\nu}(x, \boldsymbol{v}, z) \Leftarrow \exists y \quad x \overset{z}{\mapsto} (0, 0, y, u, u) * D(y, z) \quad \text{if } j_\mu = |\mu_{i_\mu}| \text{ and } j_\nu = |\nu_{i_\nu}| \tag{16}$$

$$P'(x, y', \boldsymbol{v}, z) \Leftarrow \exists x', y \quad x \overset{z}{\mapsto} (i_\mu, i_\nu, y, x', x') * y \not\simeq y' * P'(x', y', \boldsymbol{v}, z) * D(y, z) \tag{17}$$

$$P'(x, y', \boldsymbol{v}, z) \quad \Leftarrow \quad \exists y \quad x \quad \overset{z}{\mapsto} \quad (i_\mu, i_\nu, y, u, u) \ * \ y \quad \not\simeq \quad y' \ * \ D(y, z) \tag{18}$$

To check that the sequences I^μ and I^ν are identical, we proceed as follows, exploiting the fact that $\mathfrak{h}$ is cyclic and can be generated an unbounded number of times (since by hypothesis, for all $n \in \mathbb{N}$, there is a permission π such that π^n is defined). By construction, we already know that $I^\mu(1) = I^\nu(1)$, we need to check that $I^\mu(i) = I^\nu(i)$ holds for all $i \in \{2, \ldots, \kappa\}$. We first guess an index ι (which is intended to denote the value of I^μ and I^ν) and we keep track of the mark of the first cell by storing it into parameters y_μ and y_ν. These parameters are intended to denote the mark of the start of the current word in the witness (note that it would be much simpler to keep track of the cells themselves, instead of their marks, but this would make the rules non $*$-$\exists$-restricted). At this point we have $y_\mu = y_\nu$ as the words $\mu_{I^\mu(1)}$ and $\nu_{I^\nu(1)}$ always start at the same cell u, later y_μ and y_ν will take different values. Then we check that $I^\mu(2) = I^\nu(2) = \iota$. This is done by skipping the word $\mu_{I^\mu(1)}$ to find the start of the word $\mu_{I^\mu(2)}$. To this aim, we define a predicate $R^\lambda_{\iota,i,j}$ skipping the characters $j, \ldots, |\lambda_i|$ in the word λ_i (rule 23). Once this is done, we check (rule 24) that $I^\mu(2) = \iota$. Simultaneously, we keep track of the mark of the cell that corresponds to the start of $\mu_{I^\mu(2)}$ by storing it into y_μ. Then, we call a predicate R^ν_ι which performs the same operation for the sequence ν. To this aim, we must go through the entire list (rule 20) to go back to the cell marked with y_ν (rule 21). Note that this is feasible as the list is cyclic. Afterwards, we may call the predicate $R^\nu_{\iota,i,2}$ to check that $I^\nu(2) = \iota$ (rule 25) and find the start of the word $\nu_{I^\nu(2)}$ as done for μ. We also keep track of the mark of the cell that correspond to the start of $\mu_{I^\nu(2)}$ by storing it into y_ν. We then guess a new index ι'' and call the predicate R^μ_ι that performs exactly the same operations, but starting at cells marked y_μ and y_ν, respectively. This will check that the indices of the next words after the ones marked with y_μ and y_ν are identical, i.e., that $I^\mu(3) = I^\nu(3)$. We repeat this operation until y_μ and y_ν are equal (rule 22). In the rules below, λ ranges over the set $\{\mu, \nu\}$ and i, i', ι range over $\{1, \ldots, n\}$:

$$R(x, \boldsymbol{v}, z) \Leftarrow \exists x', y \quad x \overset{z}{\mapsto} (i, i', y, x', x') * D(y, z) * R^\mu_{\iota,i,2}(x', y, y, \boldsymbol{v}, z) \quad (19)$$

$$R^\lambda_\iota(x, y_\mu, y_\nu, \boldsymbol{v}, z) \Leftarrow \exists x', y \quad x \overset{z}{\mapsto} (i, i', y, x', x') * y \not\simeq y_\lambda * D(y, z) * R^\lambda_\iota(x', y_\mu, y_\nu, \boldsymbol{v}, z) \quad (20)$$

$$R^\lambda_\iota(x, y_\mu, y_\nu, \boldsymbol{v}, z) \Leftarrow \exists x', y \quad x \overset{z}{\mapsto} (i, i', y, x', x') * y \simeq y_\lambda * D(y, z) * R^\lambda_{\iota,i,2}(x', y_\mu, y_\nu, \boldsymbol{v}, z) \quad (21)$$

$$R^\mu_\iota(x, y_\mu, y_\nu, \boldsymbol{v}, z) \Leftarrow \exists x', y \; x \overset{z}{\mapsto} (0, 0, y, u, u) * y_\mu \simeq y_\nu * D(y, z) \quad (22)$$

$$R^\lambda_{\iota,i,j}(x, y_\mu, y_\nu, \boldsymbol{v}, z) \Leftarrow \exists x', y \quad x \overset{z}{\mapsto} (i, i', y, x', x') * D(y, z) * R^\lambda_{\iota,i,j+1}(x', y_\mu, y_\nu, \boldsymbol{v}, z) \quad \text{if } j \leq |\lambda_i| \quad (23)$$

$$R^{\mu}_{\iota,i,j}(x, y_\mu, y_\nu, \boldsymbol{v}, z) \Leftarrow \exists x', y \quad x \overset{z}{\mapsto} (\iota, i', y, x', x') * D(y, z) * R^{\nu}_{\iota}(x', y, y_\nu, \boldsymbol{v}, z) \quad \text{if } j = |\mu_i| + 1 \quad (24)$$

$$R^{\nu}_{\iota,i,j}(x, y_\mu, y_\nu, \boldsymbol{v}, z) \Leftarrow \exists x', y \quad x \overset{z}{\mapsto} (i', \iota, y, x', x') * D(y, z) * R^{\mu}_{\iota''}(x', y_\mu, y, \boldsymbol{v}, z) \quad \text{if } j = |\nu_i| + 1 \quad (25)$$

It is easy to see that the formula $P(x, \boldsymbol{v}, z) * R(x, \boldsymbol{v}, z)$ is $(\mathcal{R}, \mathfrak{P})$-satisfiable iff the corresponding instance of the PCP admits a solution. Moreover, the rules are $*$-$\exists$-restricted, with $\mathcal{P}^\star = \{D\}$, $\gamma_{\mathcal{R}}(R) = \gamma_{\mathcal{R}}(P) = \gamma_{\mathcal{R}}(P_{j_\mu, j_\nu, i_\mu, i_\nu}) = \emptyset$, $\gamma_{\mathcal{R}}(P') = \{2\}$ and $\gamma_{\mathcal{R}}(R^{\lambda}_{\iota}) = \gamma_{\mathcal{R}}(R^{\lambda}_{\iota,i,j}) = \{2, 3\}$.