Fundamenta Informaticae, Volume 194, Issue 1, Article 4, 2025
DOI: 10.46298/fi.11318
https://fi.episciences.org/

Tractable and Intractable Entailment Problems in Separation Logic with Inductively Defined Predicates

Mnacho Echenim[*]

Nicolas Peltier[*]

Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France

Abstract. We establish various complexity results for the entailment problem between formulas in Separation Logic (SL) with user-defined predicates denoting recursive data structures. The considered fragments are characterized by syntactic conditions on the inductive rules that define the semantics of the predicates. We focus on so-called `P`-*rules*, which are similar to (but simpler than) the so-called *bounded treewidth fragment* of SL studied by Iosif et al. in 2013. In particular, for a specific fragment where predicates are defined by so-called `loc`-*deterministic* inductive rules, we devise a sound and complete cyclic proof procedure running in polynomial time. Several complexity lower bounds are provided, showing that any relaxing of the provided conditions makes the problem intractable.

Keywords: Separation logic, Inductive reasoning, Decision procedures, Cyclic proofs

ACM Computing Classification System: –Theory of computation, Logic, Automated reasoning; –Theory of computation, Logic, Separation logic

1. Introduction

Separation Logic [15, 20] (SL) is widely used in verification to reason on programs manipulating pointer-based data structures. It forms the basis of several automated static program analyzers such as Smallfoot [1], Infer [3] (Facebook) or SLAyer [2] (Microsoft Research) and several correctness proofs were carried out by embedding SL in interactive theorem provers such as Coq [21], see for instance

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

[16]. SL uses a special connective $*$, called *separating conjunction*, modeling heap compositions and allowing for concise and natural specifications. More precisely, atoms in SL are expressions of the forms $x \mapsto (y_1, \dots, y_n)$, where $x, y_1, \dots, y_n$ are variables denoting locations (i.e., memory addresses), asserting that location x is allocated and refers to the tuple (record) $(y_1, \dots, y_n)$. The special connective $\phi * \psi$ asserts that formulas ϕ and ψ hold on disjoint parts of the memory. Recursive data structures may then be described by considering predicates associated with inductive rules, such as:

$$\begin{array}{lcl@{\quad}lcl} \mathtt{ls}(x,y) & \Leftarrow & x \mapsto y & \mathtt{tree}(x) & \Leftarrow & x \mapsto () \\ \mathtt{ls}(x,y) & \Leftarrow & x \mapsto z * \mathtt{ls}(z,y) & \mathtt{tree}(x) & \Leftarrow & x \mapsto (y,z) * \mathtt{tree}(y) * \mathtt{tree}(z) \end{array}$$

where $\mathtt{ls}(x,y)$ denotes a nonempty list segment and $\mathtt{tree}(x)$ denotes a tree. For the sake of genericity, such rules are not built-in but may be provided by the user. Due to the expressive power of such inductive definitions, the input language is usually restricted in this context to so-called *symbolic heaps*. These are existentially quantified conjunctions and separating conjunctions of atoms, including inductively defined predicates and equational atoms but dismissing for instance universal quantifications, negations and separating implications. Many problems in verification require to solve entailment problems between such SL formulas, for instance when these formulas denote pre- or post-conditions of programs. Unfortunately, the entailment problem between symbolic heaps is undecidable in general [19], but it is decidable if the considered inductive rules satisfy the so-called *PCE conditions* (standing for **P**rogress, **C**onnectivity and **E**stablishment) [13]. However even for the PCE fragment the complexity of the entailment problem is still very high; more precisely, this problem is 2-EXPTIME-complete [8, 9, 17]. Less expressive fragments have thus been considered, for which more efficient algorithms were developed. In [14] a strict subclass of PCE entailments is identified with an EXPTIME complexity based on a reduction to the language inclusion problem for tree automata [6]. In [11], an algorithm is developed to handle various kinds of (possibly nested) singly-linked lists based on a reduction to the membership problem for tree automata. The complexity of the procedure is dominated by the boolean satisfiability and unsatisfiability tests, that are NP and co-NP complete, respectively. A polynomial proof procedure has been devised for the specific case of singly-linked lists [7]. In [5], the tractability result is extended to more expressive fragments, with formulas defined on some unique nonlinear compositional inductive predicate with distinguished source, destination, and static parameters. The compositional properties satisfied by the considered predicate (as originally introduced in [12]) ensure that the entailment problem can be solved efficiently. Recently [18] introduced a polynomial-time cyclic proof system to solve entailment problem efficiently, under some conditions on the inductive rules.

In the present paper, we study the complexity of the entailment problem for a specific fragment that is similar to the PCE fragment, but simpler. The fragment inherits most of the conditions given in [13] and admits an additional restriction that is meant to ensure that entailment problems can be solved in a more efficient way[1]: every predicate is bound to allocate *exactly* one of its parameters (forbidding for instance predicates denoting doubly-linked list segments from x to y, as both x and y would be allocated). This means that the rules do not allow for multiple pointers *into* a data structure (whereas multiple pointers *out* of the structure are allowed). We first show that this additional restriction is actu-

[1] At the cost, of course, of a loss of expressivity.

ally not sufficient to ensure tractability. More precisely, we establish several lower-bound complexity results for the entailment problem under various additional hypotheses. Second, we define a new class of inductive definitions for which the entailment problem can be solved in polynomial time, based mainly on the two following additional restrictions: (i) the arity of the predicates is bounded; and (ii) the rules defining the same predicate do not overlap, in a sense that will be formally defined below. Both conditions are rather natural restrictions in the context of programming. Indeed, the number of parameters is usually small in this context. Also, data structures are typically defined using a finite set of free constructors, which yields inductive definitions that are trivially non-overlapping.

If Condition (i) is not satisfied, then the complexity is simply exponential. In contrast with other polynomial-time algorithms, the formulas we consider may contain several inductive predicates, and these predicates are possibly non-compositional (in the sense of [11]). The algorithm for testing entailment is defined as a sequent-like cyclic proof procedure, with standard unfolding and decomposition rules, together with a specific strategy ensuring efficiency and additional syntactic criteria to detect and dismiss non-provable sequents. Our approach is close to that of [18], in the sense that the two procedures use cyclic proof procedures with non-disjunctive consequents. However, the conditions on the rules are completely different: our definition allows for multiple inductive rules with mutually recursive definitions, yielding richer recursive data structures. On the other hand, the SHLIDe rules in [18] support ordering and equality relations on non-addressable values, whereas the predicate we consider are purely spatial. Moreover, the base cases of the rules in [18] correspond to empty heaps, which are forbidden in our approach.

To provide some intuition on what can and cannot be expressed in the fragment we consider, we provide some examples (formal definitions will be given later); consider the predicate $\mathtt{P}$ defined by the following rules, which encode a combination of lists and trees, possibly looping on an initial element y, and ending with an empty tuple:

$$\begin{array}{lcl}
\mathtt{P}(x,y) & \Leftarrow & x \mapsto (\mathtt{list}, u) * \mathtt{P}(u,y) \\
\mathtt{P}(x,y) & \Leftarrow & x \mapsto (\mathtt{tree}, u_1, u_2) * \mathtt{P}(u_1,y) * \mathtt{P}(u_2,y) \\
\mathtt{P}(x,y) & \Leftarrow & x \mapsto (\mathtt{loop}, y) \\
\mathtt{P}(x,y) & \Leftarrow & x \mapsto ()
\end{array}$$

All variables are implicitly universally quantified at the root level in every rule[2]. The constants $\mathtt{list}, \mathtt{tree}$ and $\mathtt{loop}$ may be viewed as constructors for the data structure. This predicate does not fall in the scope of the fragment considered in [18] since it involves a definition with several inductive rules, but it falls in the scope of the fragment considered in the present paper. Our restrictions require that the definition must be deterministic, in the sense that there can be no overlap between distinct rules. This is the case here, as the tuples $(\mathtt{list}, u)$, $(\mathtt{tree}, u_1, u_2)$ and $(\mathtt{loop}, y)$ are pairwise distinct (not unifiable), but replacing for instance the constant $\mathtt{loop}$ by $\mathtt{list}$ in the third rule would not be possible, as the resulting rule would overlap with the first one (both rules could allocate the same heap cell). As explained above, a key limitation of our fragment (compared to that of [13]) is that it does not allow predicates allocating several parameters, such as the following predicate $\mathtt{dllseg}(x,y,z,u)$

[2] Alternatively, the variables u, u_1, u_2, which do not occur in the left-hand side of the rules, may be viewed as being quantified existentially on the right-hand side of $\Leftarrow$.

defining a doubly-linked list segment from x to z (each cell points to a pair containing the previous and next element and y and u denote the previous and next element in the list, respectively):

$$\begin{array}{lcl}\mathtt{dllseg}(x,y,z,u) & \Leftarrow & (x \mapsto (y,x') * \mathtt{dllseg}(x',x,z,u)) \wedge x \not\approx z \\ \mathtt{dllseg}(x,y,z,u) & \Leftarrow & x \mapsto (y,u) \wedge x \approx z\end{array}$$

Other definitions of `dllseg` are possible, but none would fit in with our restrictions: in every case, both x (the beginning of the list) and z (its end) must be eventually allocated, which is not permitted in the fragment we consider. On the other hand, the following predicate, defining a doubly-linked list, ending with $()$, can be defined (y denotes the previous element in the list):

$$\begin{array}{lcl}\mathtt{dll}(x,y) & \Leftarrow & x \mapsto (y,z) * \mathtt{dll}(z,x) \\ \mathtt{dll}(x,y) & \Leftarrow & x \mapsto ()\end{array}$$

The rest of the paper is organised as follows. In Section 2, the syntax and semantics of the logic are defined. The definitions are mostly standard, although we consider a multisorted framework, with a special sort `loc` denoting memory locations and additional sorts for data or constructors. We then introduce a class of inductive definitions called `P`-*rules*. In Section 3, various lower bounds on the complexity of the entailment problem for SL formulas with `P`-rules are established which allow one to motivate additional restrictions on the inductive rules. These lower bounds show that all the restrictions are necessary to ensure that the entailment problem is tractable. This leads to the definition of the notion of a `loc`-*deterministic* set of rules, that is a subset of `P`-rules for which entailment can be decided in polynomial time. The proof procedure is defined in Section 4. For the sake of readability and generality we first define generic inference rules and establish their correctness, before introducing a specific strategy to further restrict the application of the rules that is both complete and efficient. Section 5 contains all soundness, completeness and complexity results and Section 6 concludes the paper.

2. Definitions

2.1. Syntax

We use a multisorted framework, which is essentially useful to distinguish locations from data. Let $\mathfrak{S}$ be a set of *sorts*, containing a special sort `loc`, denoting memory locations. Let $\mathcal{V}_{\mathtt{s}\in\mathfrak{S}}$ be a family of countably infinite disjoint sets of *variables of sort* $\mathtt{s}$, with $\mathcal{V} \stackrel{def}{=} \bigcup_{\mathtt{s}\in\mathfrak{S}} \mathcal{V}_{\mathtt{s}}$. Let $\mathcal{C}_{\mathtt{s}\in\mathfrak{S}}$ be a family of disjoint sets of *constant symbols of sort* $\mathtt{s}$, also disjoint from $\mathcal{V}$, with $\mathcal{C} \stackrel{def}{=} \bigcup_{\mathtt{s}\in\mathfrak{S}} \mathcal{C}_{\mathtt{s}}$. The set of *terms of sort* $\mathtt{s}$ is $\mathcal{T}_{\mathtt{s}} \stackrel{def}{=} \mathcal{V}_{\mathtt{s}} \cup \mathcal{C}_{\mathtt{s}}$, and we let $\mathcal{T} \stackrel{def}{=} \bigcup_{\mathtt{s}\in\mathfrak{S}} \mathcal{T}_{\mathtt{s}}$. Constants are especially useful in our framework to denote constructors in data structures. To simplify technicalities, we assume that there is no constant of sort `loc`, i.e., $\mathcal{C}_{\mathtt{loc}} = \emptyset$.

An equation (resp. a disequation) is an expression of the form $t \approx s$ (resp. $t \not\approx s$) where $t, s \in \mathcal{T}_{\mathtt{s}}$ for some $\mathtt{s} \in \mathfrak{S}$. The set of *pure formulas* $\mathcal{F}_P$ is the set of formulas of the form $e_1 \wedge \cdots \wedge e_n$, where every expression e_i is either an equation or a disequation. Such formulas are considered modulo contraction, e.g., a pure formula $\xi \wedge \xi$ is considered identical to ξ, and also modulo associativity and commutativity of conjunction. We denote by $\bot$ (false) any formula of the form $t \not\approx t$. If $n = 0$, then $\bigwedge_{i=1}^{n} e_i$ may be denoted by $\top$ (true). If $(t_1, \dots, t_n)$ and $(s_1, \dots, s_m)$ are vectors of terms, then $(t_1, \dots, t_n) \approx (s_1, \dots, s_m)$ denotes the formula $\bot$ if either $n \neq m$ or $n = m$ and there exists $i \in \{1, \dots, n\}$ such that s_i and t_i are of different sorts; and denotes $\bigwedge_{i=1}^{n} t_i \approx s_i$ otherwise.

Let $\mathcal{P}$ be a set of *predicate symbols*. Each symbol in $\mathcal{P}$ is associated with a unique *profile* of the form $(\mathtt{s}_1, \dots, \mathtt{s}_n)$ with $n \geq 1$, $\mathtt{s}_1 = \mathtt{loc}$ and $\mathtt{s}_i \in \mathfrak{S}$, for all $i \in \{2, \dots, n\}$. We write $p : \mathtt{s}_1, \dots, \mathtt{s}_n$ to denote a symbol with profile $(\mathtt{s}_1, \dots, \mathtt{s}_n)$ and we write $p : \mathtt{s}_1, \dots, \mathtt{s}_n \in \mathcal{P}$ to state that p is a predicate symbol of profile $\mathtt{s}_1, \dots, \mathtt{s}_n$ in $\mathcal{P}$. A *spatial atom* α is either a *points-to atom* $x \mapsto (t_1, \dots, t_n)$ with $x \in \mathcal{V}_{\mathtt{loc}}$ and $t_1, \dots, t_n \in \mathcal{T}$, or a *predicate atom* of the form $p(x, t_1, \dots, t_n)$, where p is a predicate of profile $\mathtt{loc}, \mathtt{s}_1, \dots, \mathtt{s}_n$ in $\mathcal{P}$, the term x is a variable in $\mathcal{V}_{\mathtt{loc}}$ and $t_i \in \mathcal{T}_{\mathtt{s}_i}$ for all $i \in \{1, \dots, n\}$. In both cases, the variable x is called the *root* of α and is denoted by $root(\alpha)$.

The set of *spatial formulas* $\mathcal{F}_S$ is the set of formulas of the form $\beta_1 * \dots * \beta_n$, where every expression β_i is a spatial atom. If $n = 0$ then $\beta_1 * \dots * \beta_n$ is denoted by *emp*. The number n of occurrences of spatial atoms in a spatial formula $\phi = \beta_1 * \dots * \beta_n$ is denoted by $len(\phi)$. We write $\phi \sqsubseteq \psi$ if ψ is of the form $\phi * \phi'$, modulo associativity and commutativity of $*$. The set of (non quantified) *symbolic heaps* $\mathcal{F}_H$ is the set of expressions of the form $\phi \curlywedge \xi$, where $\phi \in \mathcal{F}_S$ and $\xi \in \mathcal{F}_P$. Note that for clarity we use $\curlywedge$ to denote conjunctions between spatial and pure formulas and $\wedge$ to denote conjunctions occurring within pure formulas. If $\xi = \top$, then $\phi \curlywedge \xi$ may be written ϕ (i.e., any spatial formula may be viewed as a symbolic heap). For any formula λ, $|\lambda|$ denotes the size of λ (which is defined inductively as usual). Note that $\top$ is not a symbolic heap (but $emp \curlywedge \top$ is a symbolic heap).

We denote by $\mathcal{V}(\beta)$ (resp. $\mathcal{V}_{\mathtt{s}}(\beta)$) the set of variables (resp. of variables of sort $\mathtt{s}$) occurring in a variable or formula β. A *substitution* is a sort-preserving total mapping from $\mathcal{V}$ to $\mathcal{T}$. We denote by $dom(\sigma)$ the set of variables such that $\sigma(x) \neq x$, and by $codom(x)$ the set $\{\sigma(x) \mid x \in dom(\sigma)\}$. The substitution σ such that $\sigma(x_i) = y_i$ for all $i = 1, \dots, n$ and $dom(\sigma) \subseteq \{x_1, \dots, x_n\}$ is denoted by $\{x_i \leftarrow y_i \mid i = 1, \dots, n\}$. For any expression β and substitution σ, we denote by $\beta\sigma$ the expression obtained from β by replacing every variable x by $\sigma(x)$. A *unifier* of two expressions or tuples of expressions β and β' is a substitution σ such that $\beta\sigma = \beta'\sigma$.

An *inductive rule* is an expression of the form $p(x_1, \dots, x_n) \Leftarrow \lambda$, where $p : \mathtt{s}_1, \dots, \mathtt{s}_n \in \mathcal{P}$, $x_1, \dots, x_n$ are pairwise distinct variables of sorts $\mathtt{s}_1, \dots, \mathtt{s}_n$ respectively and $\lambda \in \mathcal{F}_H$. The set of variables in $\mathcal{V}(\lambda) \setminus \{x_1, \dots, x_n\}$ are the *existential variables* of the rule. Let $\mathfrak{R}$ be a set of inductive rules. We write $p(t_1, \dots, t_n) \Leftarrow_{\mathfrak{R}} \lambda$ iff $\mathfrak{R}$ contains (up to a renaming, and modulo AC) a rule of the form $p(y_1, \dots, y_n) \Leftarrow \gamma$ and $\lambda = \gamma\{y_i \leftarrow t_i \mid i = 1, \dots, n\}$. We assume by renaming that γ contains no variable in $\{t_1, \dots, t_n\}$. We write $p(t_1, \dots, t_n) \leadsto_{\mathfrak{R}} E$ if E is the set of symbolic heaps λ such that $p(t_1, \dots, t_n) \Leftarrow_{\mathfrak{R}} \lambda$. Note that if $\mathfrak{R}$ is finite then E is finite up to a renaming of variables not occurring in $\{t_1, \dots, t_n\}$. Note also that the considered logic does not allow for negations (hence entailment is not reducible to satisfiability) or separating implications, as this would make satisfiability undecidable (see for instance [19]).

The symbol $\subseteq_m$ denotes the inclusion relation between multisets. With a slight abuse of notations, we will sometimes identify sequences with sets when the order and number of repetitions is not important, for instance we may write $\boldsymbol{x} \subseteq \boldsymbol{y}$ to state that every element of $\boldsymbol{x}$ occurs in $\boldsymbol{y}$.

In the present paper, we shall consider entailment problems between symbolic heaps.

2.2. Semantics

We assume for technical convenience that formulas are interpreted over a fixed universe and that constants are interpreted as pairwise distinct elements. Let $\mathfrak{U}_{\mathtt{s}\in\mathfrak{S}}$ be pairwise disjoint countably infinite sets and let $\mathfrak{U} \overset{\mathit{def}}{=} \bigcup_{\mathtt{s}\in\mathfrak{S}} \mathfrak{U}_{\mathtt{s}}$. We assume that an injective function is given, mapping every constant $c \in \mathcal{C}_{\mathtt{s}}$ to an element of $\mathfrak{U}_{\mathtt{s}}$, denoted by $\dot{c}$.

A *heap* is a partial finite function from $\mathfrak{U}_{\mathtt{loc}}$ to $\mathfrak{U}^*$, where $\mathfrak{U}^*$ denotes as usual the set of finite sequences of elements of $\mathfrak{U}$. An element $\ell \in \mathfrak{U}_{\mathtt{loc}}$ is *allocated* in a heap $\mathfrak{h}$ if $\ell \in \mathit{dom}(\mathfrak{h})$. Two heaps $\mathfrak{h}$ and $\mathfrak{h}'$ are *disjoint* if $\mathit{dom}(\mathfrak{h}) \cap \mathit{dom}(\mathfrak{h}') = \emptyset$, in which case $\mathfrak{h} \uplus \mathfrak{h}'$ denotes their disjoint union. We write $\mathfrak{h} \subseteq \mathfrak{h}'$ if there is a heap $\mathfrak{h}''$ such that $\mathfrak{h}' = \mathfrak{h} \uplus \mathfrak{h}''$. For every heap $\mathfrak{h}$, we denote by $\mathit{ref}(\mathfrak{h})$ the set of elements $\ell \in \mathfrak{U}_{\mathtt{loc}}$ such that there exists $\ell_0, \dots, \ell_n$ with $\ell_0 \in \mathit{dom}(\mathfrak{h})$, $\mathfrak{h}(\ell_0) = (\ell_1, \dots, \ell_n)$ and $\ell = \ell_i$ for some $i = 0, \dots, n$. We write $\ell \rightarrow_{\mathfrak{h}} \ell'$ if $(\ell, \ell') \in \mathfrak{U}_{\mathtt{loc}}^2$, $\ell \in \mathit{dom}(\mathfrak{h})$, $\mathfrak{h}(\ell) = (\ell_1, \dots, \ell_n)$ and $\ell' = \ell_i$, for some $i = 1, \dots, n$.

Proposition 2.1. Let $\mathfrak{h}, \mathfrak{h}'$ be two heaps such that $\mathfrak{h} \subseteq \mathfrak{h}'$. For all $\ell, \ell' \in \mathfrak{U}_{\mathtt{loc}}$, if $\ell \rightarrow_{\mathfrak{h}}^* \ell'$ then $\ell \rightarrow_{\mathfrak{h}'}^* \ell'$.

Proof:
By definition of the relation $\rightarrow_{\mathfrak{h}}$ we have $\rightarrow_{\mathfrak{h}} \subseteq \rightarrow_{\mathfrak{h}'}$, thus $\rightarrow_{\mathfrak{h}}^* \subseteq \rightarrow_{\mathfrak{h}'}^*$ □

A *store* $\mathfrak{s}$ is a total function mapping every term in $\mathcal{T}_{\mathtt{s}}$ to an element in $\mathfrak{U}_{\mathtt{s}}$ such that $\mathfrak{s}(c) = \dot{c}$, for all $c \in \mathcal{C}$ (note that this entails that $\mathfrak{s}$ is injective on $\mathcal{C}$). A store $\mathfrak{s}$ is *injective on a multiset of variables* V if $\{x, y\} \subseteq_m V \implies \mathfrak{s}(x) \neq \mathfrak{s}(y)$. When a store is injective on a multiset of variables, this entails that the latter is a set, i.e., that contains at most one occurrence of each variable. For any $V \subseteq \mathcal{V}$, and for any store $\mathfrak{s}$, a store $\mathfrak{s}'$ is an *associate of* $\mathfrak{s}$ *w.r.t.* V if $\mathfrak{s}(x) = \mathfrak{s}'(x)$ holds for all $x \notin V$.

Definition 2.2. An *SL-structure* is a pair $(\mathfrak{s}, \mathfrak{h})$, where $\mathfrak{s}$ is a store and $\mathfrak{h}$ is a heap.

The satisfiability relation on SL-formulas is defined inductively as follows:

Definition 2.3. An SL-structure $(\mathfrak{s}, \mathfrak{h})$ *validates a formula (pure formula, spatial formula, or symbolic heap)* λ *modulo a set of inductive rules* $\mathfrak{R}$, written $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \lambda$, if one of the following conditions holds:

- $\lambda = \mathit{emp}$ and $\mathfrak{h} = \emptyset$;
- $\lambda = (t \approx s)$ and $\mathfrak{s}(t) = \mathfrak{s}(s)$;
- $\lambda = (t \not\approx s)$ and $\mathfrak{s}(t) \neq \mathfrak{s}(s)$;
- $\lambda = x \mapsto (t_1, \dots, t_k)$ and $\mathfrak{h} = \{(\mathfrak{s}(x), \mathfrak{s}(t_1), \dots, \mathfrak{s}(t_k))\}$;
- either $\lambda = \lambda_1 \wedge \lambda_2$ or $\lambda = \lambda_1 \curlywedge \lambda_2$, and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \lambda_i$ for all $i = 1, 2$;

- $\lambda = \lambda_1 * \lambda_2$ and there exist disjoint heaps $\mathfrak{h}_1$ and $\mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$ and $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \lambda_i$ for all $i = 1, 2$;

- $\lambda = p(t_1, \dots, t_n)$ and $p(t_1, \dots, t_n) \Leftarrow_{\mathfrak{R}} \gamma$, where there exists an associate $\mathfrak{s}'$ of $\mathfrak{s}$ w.r.t. the set $\mathcal{V}(\gamma) \setminus \mathcal{V}(\lambda)$ such that $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \gamma$.

An $\mathfrak{R}$-*model* of a formula λ is a structure $(\mathfrak{s}, \mathfrak{h})$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \lambda$. A formula is *satisfiable* (w.r.t. $\mathfrak{R}$) if it admits at least one $\mathfrak{R}$-model.

Remark 2.4. Note that a formula $x \mapsto (t_1, \dots, t_k)$ asserts not only that x refers to $(t_1, \dots, t_k)$ but also that x is the only allocated location. This fits with usual definitions (see, e.g., [15]). The assertions are meant to describe elementary heaps, which can be combined afterwards using the connective $*$. Simply asserting that x refers to $(t_1, \dots, t_k)$ could be done in full SL using the following formula: $x \mapsto (t_1, \dots, t_k) * \top$, but such a formula is not a symbolic heap and is thus outside of the fragment we consider in the present paper.

We write $(\mathfrak{s}, \mathfrak{h}) \models \lambda$ instead of $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \lambda$ if λ contains no predicate symbol, since the relation is independent of $\mathfrak{R}$ in this case. Similarly, if λ is a pure formula then the relation does not depend on the heap, thus we may simply write $\mathfrak{s} \models \lambda$.

Remark 2.5. Note that there is no symbolic heap that is true in every structure. For instance *emp* $\curlywedge \top$ is true only in structures with an empty heap.

Proposition 2.6. Let $(\mathfrak{s}, \mathfrak{h})$ be a structure and let σ be a substitution. If $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \lambda\sigma$ then we have $(\mathfrak{s} \circ \sigma, \mathfrak{h}) \models_{\mathfrak{R}} \lambda$.

Proof:
By an immediate induction on the satisfiability relation. □

In the present paper, we shall consider inductive rules of a particular form, defined below.

Definition 2.7. An inductive rule is a P-*rule* if it is of the form

$$p(x_1, \dots, x_n) \Leftarrow x_1 \mapsto (y_1, \dots, y_k) * q_1(z_1, \boldsymbol{u}_1) * \dots q_m(z_m, \boldsymbol{u}_m) \curlywedge \xi$$

possibly with $m = 0$, where:

1. ξ is a conjunction of disequations of the form $u \not\approx v$, where $u \in \{x_1, \dots, x_n, y_1, \dots, y_k\}$ and $v \in \{y_1, \dots, y_k\} \setminus \{x_1, \dots, x_n\}$;

2. $\{z_1, \dots, z_m\} = (\{y_1, \dots, y_k\} \setminus \{x_1, \dots, x_n\}) \cap \mathcal{V}_{\mathtt{loc}}$, and $z_1, \dots, z_m$ are pairwise distinct;

3. All the elements of $\boldsymbol{u}_i$ occur in $\{x_1, \dots, x_n\} \cup \{y_1, \dots, y_k\} \cup \mathcal{C}$.

The predicate symbol p is called the *head* of the rule.

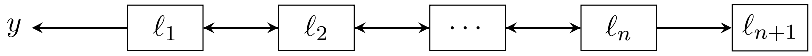

Figure 1. Doubly-linked list ending with $()$ (ℓ_{n+1} is allocated but has no successor)

Example 2.8. The rules associated with $\mathtt{ls}$ and $\mathtt{tree}$ in the introduction are $\mathtt{P}$-rules, as well as the following rules. Intuitively $\mathtt{als}(x,y) \curlywedge x \not\approx y$ denotes an acyclic list ($\mathtt{als}(x,y)$ is thus "quasi-acyclic", in the sense that it may loop only on the first element). Note that the constraint $x \not\approx y$ cannot be added to the right-hand side of the rules because the obtained rule would not be a $\mathtt{P}$-rule, hence it must be added in the formula. The atom $\mathtt{dll}(x,y)$ denotes a doubly-linked list starting at x, with the convention that each element of the list points to a pair containing the previous and next elements. The parameter y denotes the element before x (if any) and the last element points to the empty tuple $()$. The structures validating $\mathtt{dll}(x,y)$ are of the form (see Figure 1) $(\mathfrak{s}_n, \mathfrak{h}_n)$ with $n \geq 0$, $dom(\mathfrak{h}) = \{\ell_1, \dots, \ell_{n+1}\}$, $\mathfrak{h}(\ell_i) = (\ell_{i-1}, \ell_{i+1})$ for all $i \in \{1, \dots, n\}$, $\mathfrak{h}(\ell_{n+1}) = ()$, $\mathfrak{s}(x) = \ell_1$ and $\mathfrak{s}(y) = \ell_0$. Locations $\ell_1, \dots, \ell_{n+1}$ must be pairwise distinct and ℓ_0 is arbitrary. The atom $\mathtt{tptr}(x,y,z)$ denotes a binary tree in which every node refers to its two successors and to its parent and sibling nodes. The parameters y and z denote the sibling and parent nodes, respectively. Leaves point to $()$. See Figure 2 for an example with 5 allocated locations.

$$
\begin{array}{lcll}
\mathtt{als}(x,y) & \Leftarrow & (x \mapsto (z) * \mathtt{als}(z,y)) \curlywedge y \not\approx z & \texttt{\% (quasi-)acyclic list} \\
\mathtt{als}(x,y) & \Leftarrow & x \mapsto (y) & \\
\mathtt{tll}(x,y) & \Leftarrow & x \mapsto (y,z) * \mathtt{tree}(z) & \texttt{\% binary trees with} \\
\mathtt{tll}(x,y) & \Leftarrow & (x \mapsto (z,u) * \mathtt{tll}(z,y) * \mathtt{tree}(u)) & \texttt{\% leftmost leaf } y \\
 & & \qquad \curlywedge (y \not\approx z) & \\
\mathtt{dll}(x,y) & \Leftarrow & x \mapsto (y,z) * \mathtt{dll}(z,x) & \texttt{\% doubly-linked lists} \\
\mathtt{dll}(x,y) & \Leftarrow & x \mapsto () & \\
\mathtt{tptr}(x,y,z) & \Leftarrow & x \mapsto (u,v,y,z) * \mathtt{tptr}(u,v,x) & \texttt{\% binary trees with} \\
 & & \qquad * \mathtt{tptr}(v,u,x) & \texttt{\% pointers to brother} \\
\mathtt{tptr}(x,y,z) & \Leftarrow & x \mapsto () & \texttt{\% and parent nodes}
\end{array}
$$

The following rules are not $\mathtt{P}$-rules (if all variables are of sort $\mathtt{loc}$):

$$
\begin{array}{lcll}
p(x) & \Leftarrow & x \mapsto (z) & \text{Condition 2 violated} \\
p(x) & \Leftarrow & \mathtt{ls}(x,z) * p(z) & \text{No points-to atom} \\
q(x,y) & \Leftarrow & x \mapsto (z) \curlywedge y \approx z & \text{Condition 2 violated} \\
q(x,y) & \Leftarrow & \mathtt{ls}(x,y) & \text{No points-to atom} \\
\mathtt{als}(x,y) & \Leftarrow & (x \mapsto (z) * \mathtt{als}(z,y)) \curlywedge x \not\approx y & \text{Condition 1 violated} \\
\mathtt{als}(x,y) & \Leftarrow & x \mapsto (y) \curlywedge x \not\approx y & \text{Condition 1 violated}
\end{array}
$$

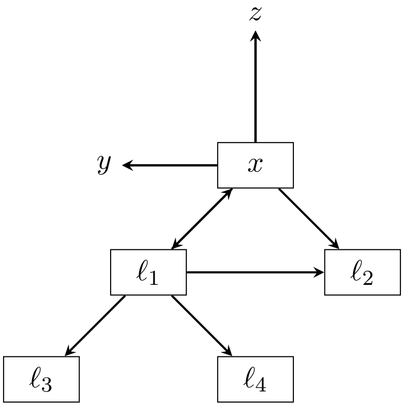

Figure 2. Binary tree with pointers to parent and sibling (ℓ_2, ℓ_3, ℓ_4 are allocated but have no successor)

Remark 2.9. As evidenced by the rules in Example 2.8, the tuple $()$ is frequently used as a base case, to end a data structure. This departs from standard conventions in SL in which a non-allocated constant `nil` is frequently used instead. We avoid considering constants of sort `loc` in our framework because this would complicate definitions: one would have to keep track of allocated and non-allocated constants and/or to add syntactic conditions on the formulas and rules to ensure that such constants are never allocated. Note that this convention introduces an asymmetry in the structure. For instance, in doubly-linked lists, the "next" field of the last element points to $()$ while the "previous" field of the first element contains an arbitrary location y. To achieve symmetry in the structure, one could introduce an atom $y \mapsto ()$. For example, the formula $y \mapsto () * \mathtt{dll}(x, y)$ represents a doubly-linked list ending with $()$ in both directions.

Note that P-rules are progressing and connected (in the sense of [13]): every rule allocates exactly one location –the first parameter of the predicate– and the first parameter of every predicate in the body of the rule occurs in the right-hand side of the (necessarily unique) points-to atom of the rule. They are not necessarily established (again in the sense of[13]) as non-allocated existential variables are allowed provided they are not of sort `loc`.

Example 2.10. The following (non established, in the sense of [13]) rules, denoting list segments with unallocated elements are P-rules iff $u \notin \mathcal{V}_{\mathtt{loc}}$:

$$\mathtt{ls}(x, y) \Leftarrow x \mapsto (u, y) \qquad \mathtt{ls}(x, y) \Leftarrow x \mapsto (u, z) * \mathtt{ls}(z, y)$$

The heap of any model of $\mathtt{ls}(x, y)$ is of the form $\{\ell_i \mapsto (u_i, \ell_{i+1}) \mid i \in \{1, \ldots, n\}\}$, where $u_1, \ldots, u_n$ denote arbitrary elements (of a sort distinct from `loc`).

P-Rules containing no variable of a sort distinct from `loc` are established. P-Rules also differ from PCE rules in that every predicate allocates *exactly* one of its parameters, namely the first one (the other allocated locations are associated with existential variables). In other words, there may be only one "entry point" to the structure allocated by a predicate, namely its root. For instance the rule $p(x,y) \Leftarrow x \mapsto (y) * q(y)$ (along with another rule for symbol q, e.g., $q(y) \Leftarrow y \mapsto ()$) is PCE but it is not a P-rule, whereas $p(x) \Leftarrow x \mapsto (y) * q(y)$ is a P-rule. Such a restriction makes the entailment problem easier to solve because it rules out data structures that can be constructed in different orders (for instance doubly-linked lists with a reference to the end of the list).

We introduce some useful notations and measures on sets of P-rules. For every set of P-rules $\mathfrak{R}$, we denote by $\mathcal{P}(\mathfrak{R})$ the set of predicate symbols occurring in a rule in $\mathfrak{R}$. We define:

$$ar_{max}(\mathfrak{R}) = \max\{n \mid p : \mathtt{s}_1, \dots, \mathtt{s}_n \in \mathcal{P}(\mathfrak{R})\}$$

and

$$record_{max}(\mathfrak{R}) = \max\{k \mid p(x_1, \dots, x_n) \Leftarrow (x \mapsto (t_1, \dots, t_k) * \phi) \curlywedge \xi \in \mathfrak{R}\}$$

The numbers $ar_{max}(\mathfrak{R})$ and $record_{max}(\mathfrak{R})$ respectively denote the maximum arity of the predicate symbols in $\mathfrak{R}$ and the maximum number of record fields in a points-to atom occurring in $\mathfrak{R}$. The *width* of $\mathfrak{R}$ is defined as follows: $width(\mathfrak{R}) \overset{def}{=} \max(ar_{max}(\mathfrak{R}), record_{max}(\mathfrak{R}))$.

We make two additional assumptions about the considered set of rules: we assume that every predicate is productive (Assumption 2.12) and that no parameter is useless (Assumption 2.15). More precisely, the set of *productive* predicate symbols is inductively defined as follows: $p \in \mathcal{P}$ is productive w.r.t. a set of inductive rules $\mathfrak{R}$ if $\mathfrak{R}$ contains a rule $p(\boldsymbol{x}) \Leftarrow \lambda$ such that all the predicate symbols occurring in λ are productive. In particular, a rule with no predicate in its right-hand side is always productive (base case). Productive rules can easily be computed using a straightforward least fixpoint algorithm.

Example 2.11. Let $\mathfrak{R} = \{p(x) \Leftarrow q(x), q(x) \Leftarrow p(x), r(x) \Leftarrow x \mapsto (y) * p(y)\}$. The predicates p, q, r are not productive.

It is easy to check that every formula containing at least one non-productive predicate symbol is unsatisfiable. Indeed, if a predicate symbol p is non-productive, then an atom $p(\boldsymbol{x})$ cannot be unfolded into a formula containing no predicate symbol. This justifies the following:

Assumption 2.12. For all sets of P-rules $\mathfrak{R}$, we assume that all the predicate symbols are productive w.r.t. $\mathfrak{R}$.

For all predicates $p : \mathtt{s}_1, \dots, \mathtt{s}_n$, the set $out_{\mathfrak{R}}(p)$ denotes the least set of indices i in $\{1, \dots, n\}$ such that $\mathtt{s}_i = \mathtt{loc}$ and there exists a rule $p(x_1, \dots, x_n) \Leftarrow \lambda$ in $\mathfrak{R}$ such that λ contains either a points-to atom $x_1 \mapsto (t_1, \dots, t_k)$ where $x_i \in \{t_1, \dots, t_k\}$ or a predicate atom $q(t_1, \dots, t_m)$ with $t_j = x_i$, for some $j \in out_{\mathfrak{R}}(q)$. Intuitively, $out_{\mathfrak{R}}(p)$ denote the set of "out-going" nodes of the structures corresponding to p, i.e., the set of parameters corresponding to locations that can be referred to but not necessarily allocated.

Example 2.13. Consider the following rules:

$$p(x,y,z) \Leftarrow x \mapsto (x,y) \qquad p(x,y,z) \Leftarrow x \mapsto (x,u) * q(u,z,z) \qquad q(x,y,z) \Leftarrow x \mapsto (y)$$

Then $out_{\mathfrak{R}}(p) = \{1,2\}$ and $out_{\mathfrak{R}}(q) = \{2\}$.

Proposition 2.14. Let $\mathfrak{R}$ be a set of P-rules. If $(\mathfrak{s},\mathfrak{h}) \models_{\mathfrak{R}} p(t_1,\dots,t_n)$ and the index $i \neq 1$ is such that $i \notin \mathit{out}_{\mathfrak{R}}(p)$ (i.e., i is not an outgoing parameter of p) and $\mathtt{s}_i = \mathtt{loc}$, then the entailment $(\mathfrak{s},\mathfrak{h}) \models_{\mathfrak{R}} p(t_1,\dots,t_{i-1},s,t_{i+1},t_n)$ holds for all terms s.

Proof:
By an induction on the satisfiability relation. By hypothesis there exists a formula γ such that $p(t_1,\dots,t_n) \Leftarrow_{\mathfrak{R}} \gamma$, where and γ is of the form $t_1 \mapsto (t'_1,\dots,t'_k) * \gamma'$ and there exists an associate $\mathfrak{s}'$ of $\mathfrak{s}$ w.r.t. $\mathcal{V}(\gamma) \setminus \mathcal{V}(\lambda)$ such that $(\mathfrak{s}',\mathfrak{h}) \models_{\mathfrak{R}} \gamma$. By hypothesis $i \neq 1$, and since $i \notin \mathit{out}_{\mathfrak{R}}(p)$, t_i cannot occur in $\{t'_1,\dots,t'_k\}$. This entails that $p(t_1,\dots,t_{i-1},s,t_{i+1},t_n) \Leftarrow_{\mathfrak{R}} t_1 \mapsto (t'_1,\dots,t'_k)*\gamma''$. If γ' contains a predicate $q(s_1,\dots,s_m)$ and there exists an index j such that $s_j = t_i$, then we cannot have $j = 1$ because $t_i \notin \{t'_1,\dots,t'_k\}$ and the rule under consideration is a P-rule. Since $i \notin \mathit{out}_{\mathfrak{R}}(p)$ by hypothesis, j cannot belong to $\mathit{out}_{\mathfrak{R}}(q)$ and by induction, we deduce that $(\mathfrak{s}',\mathfrak{h}) \models_{\mathfrak{R}} t_1 \mapsto (t'_1,\dots,t'_k) * \gamma''$, hence the result. □

Proposition 2.14 states that if $i \notin \mathit{out}_{\mathfrak{R}}(p) \cup \{1\}$ and $\mathtt{s}_i = \mathtt{loc}$, then the semantics of $p(t_1,\dots,t_n)$ does not depend on t_i, thus the i-th argument of p is redundant and can be removed. This justifies the following:

Assumption 2.15. For all sets of P-rules $\mathfrak{R}$ and for all predicate symbols $p : \mathtt{loc}, \mathtt{s}_1,\dots,\mathtt{s}_n \in \mathcal{P}$, we assume that $\mathit{out}_{\mathfrak{R}}(p) \supseteq \{2 \leq i \leq n \mid \mathtt{s}_i = \mathtt{loc}\}$.

Definition 2.16. For any formula λ, we write $x \rightarrow_{\lambda} y$ if $x, y \in \mathcal{V}_{\mathtt{loc}}$ and λ contains an atom $p(t_1,\dots,t_n)$ (resp. $t_1 \mapsto (t_2,\dots,t_n)$) such that $t_1 = x$ and $t_i = y$, for some $i \in \mathit{out}_{\mathfrak{R}}(p)$ (resp. for some $i \in \{2,\dots,n\}$).

A structure $(\mathfrak{s},\mathfrak{h})$ is called a *$\rightarrow$-compatible model* of a formula λ iff $(\mathfrak{s},\mathfrak{h}) \models_{\mathfrak{R}} \lambda$ and for every $x, y \in \mathcal{V}_{\mathtt{loc}}$, $\mathfrak{s}(x) \rightarrow_{\mathfrak{h}}^{*} \mathfrak{s}(y) \implies x \rightarrow_{\lambda}^{*} y$.

Intuitively, $x \rightarrow_{\lambda} y$ states that the formula λ allocates an edge from x to y.

Definition 2.17. A *sequent* is an expression of the form $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$, where λ, γ are symbolic heaps, V is a multiset of variables of sort $\mathtt{loc}$ and $\mathfrak{R}$ is a finite set of inductive rules. If $V = \emptyset$ then the sequent is written $\lambda \vdash_{\mathfrak{R}} \gamma$. A sequent is *equality-free* if λ and γ contain no atoms of the form $u \approx v$. A *counter-model* of a sequent $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$ is a structure $(\mathfrak{s},\mathfrak{h})$ such that:

- $(\mathfrak{s},\mathfrak{h}) \models_{\mathfrak{R}} \lambda$ and $(\mathfrak{s},\mathfrak{h}) \not\models_{\mathfrak{R}} \gamma$,
- $\forall x \in V$, $\mathfrak{s}(x) \notin \mathit{dom}(\mathfrak{h})$,
- $\mathfrak{s}$ is injective on the multiset V.

A sequent is *valid* iff it has no counter-model.

3. Lower Bounds

We establish various lower bounds for the validity problems for sequents $\lambda \vdash_{\mathfrak{R}} \gamma$, where $\mathfrak{R}$ satisfies some additional conditions. These lower bounds will motivate the additional restrictions that are imposed to devise a polynomial-time proof procedure.

Checking the validity of sequents $\lambda \vdash_{\mathfrak{R}} \gamma$ where $\mathfrak{R}$ is a set of P-rules is actually undecidable in general. This result can be established by an argument similar to the one used in [10] to prove the undecidability of PCE entailments modulo theories; it is not given here for the sake of conciseness, and because the goal of this paper is to investigate tractable cases. The undecidability proof relies on the existence of variables of a sort distinct from `loc`. If such variables are forbidden, then the rules are PCE hence entailment is decidable [13], but we still get an ExpTime lower bound:

Proposition 3.1. Checking the validity of sequents $\lambda \vdash_{\mathfrak{R}} \gamma$ is ExpTime-hard, even if $\mathfrak{R}$ is a set of P-rules and all the variables occurring in $\lambda \vdash_{\mathfrak{R}} \gamma$ are of sort `loc`.

Proof:
The proof is by a straightforward reduction from the inclusion problem for languages accepted by tree automata (see [6]). Indeed, a tree automaton $(Q, V, \{q_0\}, R)$ can be straightforwardly encoded as a set of P-rules, where each rule $q \to f(q_1, \dots, q_n)$ in R is encoded by an inductive rule of the form $q(x) \Leftarrow x \mapsto (f, x_1, \dots, x_n) * *_{i=1}^{n} q_i(x_i)$. Each function symbol f is considered as a constant of a sort $\mathtt{s} \neq \mathtt{loc}$, and a term $f(t_1, \dots, t_n)$ is represented as a heap $\mathfrak{h}_1 \uplus \dots \uplus \mathfrak{h}_n \uplus \{(\ell_0, \ell_1, \dots, \ell_n)\}$, where $\ell_0, \dots, \ell_n$ are pairwise distinct locations and $\mathfrak{h}_1, \dots, \mathfrak{h}_n$ are disjoint representations of t_i with $\ell_0 \notin dom(\mathfrak{h}_i)$, for $i = 1, \dots, n$. It is straightforward to verify that the language accepted by $(Q, V, \{q_0\}, R)$ is included in that of $(Q', V, \{q'_0\}, R')$ iff the sequent $q_0(x) \vdash_{\mathfrak{R}} q'_0(x)$ is valid. □

Since the inclusion problem is polynomial for top-down deterministic tree automata [6], it is natural to further restrict the considered rules to make them deterministic, in the following sense:

Definition 3.2. A set of P-rules $\mathfrak{R}$ is *deterministic* if for all pairs of distinct rules of the form $p(\boldsymbol{x}_i) \Leftarrow (y_i \mapsto \boldsymbol{t}_i * \phi_i) \curlywedge \xi_i$ (where $i = 1, 2$) occurring in $\mathfrak{R}$, the formula $\boldsymbol{x}_1 \approx \boldsymbol{x}_2 \wedge \boldsymbol{t}_1 \approx \boldsymbol{t}_2 \wedge \xi_1 \wedge \xi_2$ is unsatisfiable (we assume by renaming that the rules share no variable).

For instance the rules associated with the predicate `ls` in the introduction are not deterministic, whereas the rules associated with `tree` are deterministic, as well as all those in Example 2.8. For the predicate `ls`, the formula $x \approx x' \wedge y \approx z$ is satisfiable, whereas for the predicate `tree`, the formula $x \approx x' \wedge () \approx (y, z)$ is unsatisfiable (in both cases the variable x is renamed by x' in the second rule).

The following proposition shows that the restriction to deterministic sets of P-rules is still not sufficient to obtain a tractable validity problem:

Proposition 3.3. Checking the validity of sequents $\lambda \vdash_{\mathfrak{R}} \gamma$ is PSpace-hard, even if $\mathfrak{R}$ is a deterministic set of P-rules and all variables in $\lambda \vdash_{\mathfrak{R}} \gamma$ are of sort `loc`.

Proof:
Let "$w \in E$" be any problem in PSPACE. By definition, there exists a Turing machine $M = (Q, \Sigma, B, \Gamma, \delta, q_0, F)$ accepting exactly the words in E and a polynomial R such that M runs in space $R(n)$ on all words $w \in \Sigma^n$. The set Q denotes the set of states of M, Σ is the input alphabet, B is the blank sumbol, Γ is the tape alphabet, δ is the transition function, q_0 is the initial state and F is the set of final states. We shall reduce the problem "$w \in E$" to the entailment problem, for a sequent fulfilling the conditions above. Consider a word w of length n, and let $N = R(n)$. Assume that $\mathcal{C}$ contains all the elements in Γ. We consider $card(Q) \cdot N$ predicates q^i of arity $N + 3$, for all $q \in Q$ and $i \in \{1, \ldots, N\}$, associated with the following rules:

$$\begin{array}{l} q^i(x, y_1, \ldots, y_N, u, v) \Leftarrow x \mapsto (x', u, v, a) * p^{i+\mu}(x', y_1, \ldots, y_{i-1}, b, y_{i+1}, \ldots, y_N, y_i, a) \\ \qquad\qquad\qquad\qquad \text{if } q \notin F \text{ and } \delta \text{ contains a rule } (q, a) \to (p, b, \mu) \\ \qquad\qquad\qquad\qquad\qquad \text{with } i + \mu \in \{1, \ldots, N\} \end{array}$$

$$q^i(x, y_1, \ldots, y_N, u, v) \Leftarrow x \mapsto (x, u, v, B), \text{ if } q \in F.$$

Intuitively, q is the state of the machine, the arguments $y_1, \ldots, y_N$ denote the tape (that is of length N by hypothesis) and i denotes the position of the head on the tape. The constants a, b denote the symbols read and written on the tape, respectively, p is the final state of the transition rule and the integer μ denotes the move, i.e., an element of $\{-1, 0, +1\}$, so that $i + \mu$ is the final position of the head on the tape. Note that at this point the inductive rule does not test whether the symbol a is indeed identical to the symbol at position i, namely y_i. Instead, it merely stores both y_i and a within the next tuple of the heap, by passing them as parameters to $p^{i+\mu}$. The arguments u and v are used to encode respectively the symbol read on the tape at the previous state and the symbol that was expected. By definition of the above rules, it is clear that $q^i(x, y_1, \ldots, y_N, B, B)$ holds if the heap is a list of tuples (x_j, u_j, v_j, a_j), for $j = 1, \ldots, k$, linked on the first argument (the last element loops on itself). The heap encodes a "candidate run" of length k of M, i.e., a run for which one does not check, when applying a transition $(p, a) \to (q, b, \mu)$, that the symbol read on the tape is identical to the expected symbol a. The symbols u_i, v_i, a_i stored at each node are precisely the symbols that are read (u_i) and expected (v_i) at the previous step, respectively, along with the symbol (a_i) that is expected at the current step (this last symbol is added to ensure that the rules are deterministic). Note that for $i = 1$ there is no previous step and for $i = k$ no symbol is read since the state is final; thus by convention, a_k is set to B (see the last rule of q^i). Furthermore, u_1, v_1 will also be set to B by invoking the initial state predicate with B as the last two arguments (see the definition of the sequent below). To check that the list corresponds to an actual run of M, it thus suffices to check that $u_i = v_i$ holds for all $i = 1, \ldots, k$.

The right-hand side of the sequent will allocate all structures not satisfying this condition. To this purpose, we associate with each state $r \in Q$ two predicate symbols r_0 and r_1 defined by the following rules (where $i, j \in \{0, 1\}$):

$$\begin{array}{l} r_i(x) \Leftarrow x \mapsto (x', a, b, c) * r_j(x') \\ \qquad\qquad \text{for all } r \notin F,\ a, b, c \in \Gamma, \text{ where } j = 1 \text{ iff either } i = 1 \text{ or } a \neq b; \\ r_i(x) \Leftarrow x \mapsto (x, a, b, B) \quad \text{if } a, b \in \Gamma,\ r \in F \text{ and either } i = 1 \text{ or } a \neq b. \end{array}$$

Intuitively, the index i in predicate r_i is equal to 1 iff a faulty location has been encountered (i.e., a tuple (x', a, b) with $a \neq b$).

Note that the number of rules is polynomial w.r.t. N, since the machine M is fixed. Also, the obtained set of rules is deterministic, because M is deterministic and the expected symbol is referred to by the location allocated by each predicate symbol q^i, thus the tuples (x', u, v, a) corresponding to distinct rules associated with the same symbol q^i cannot be unifiable.

Let $w_1 \dots, w_N = w.B^{N-n}$ be the initial tape (where w is completed by blank symbols B to obtain a word of length N). It is clear that the sequent $q_0^1(x, w_1, \dots, w_N, B, B) \vdash_{\mathfrak{R}} r_0(x)$ is valid iff all the "candidate runs" of M fulfill the conditions of the right-hand side, i.e., falsify at least one equality between read and expected symbols. Thus $q_0^1(x, w_1, \dots, w_N, B, B) \vdash_{\mathfrak{R}} r_0(x)$ is valid iff M does not accept w. □

In view of this result, it is natural to investigate the complexity of the entailment problem when the maximal arity of the predicates is bounded. However, this is still insufficient to get a tractable problem, as the following lemma shows.

Lemma 3.4. Checking the validity of sequents $\lambda \vdash_{\mathfrak{R}} \gamma$ is co-NP-hard, even if $\mathit{width}(\mathfrak{R}) \leq 4$ (i.e., if the symbols and tuples are of arity at most 4) and $\mathfrak{R}$ is a deterministic set of P-rules.

Proof:
The proof is by a reduction from the complement of the 3-coloring problem, that is well-known to be NP-complete. Let $G = (V, E)$ be a graph, where $V = \{v_1, \dots, v_n\}$ is a finite set of vertices and E is a set of undirected edges, i.e., a set of unordered pairs of vertices. Let $\texttt{Colors} = \{a, b, c\}$ be a set of colors, with $\mathit{card}(\texttt{Colors}) = 3$. We recall that a solution of the 3-coloring problem is a function $f : V \to \texttt{Colors}$ such that $(x, y) \in E \implies f(x) \neq f(y)$. We assume, w.l.o.g., that all vertices occur in at least one edge. We consider two distinct sorts $\texttt{loc}$ and $\texttt{s}$. We assume, w.l.o.g., that $V \cup \texttt{Colors} \subseteq \mathcal{V}_{\texttt{s}}$ (i.e., a, b, c, as well as the set of vertices in G, are variables) and $V \cap \texttt{Colors} = \emptyset$.

Let $E = \{(x_i, y_i) \mid i = 1, \dots, m\}$ (where the edges are ordered arbitrarily) and let $u_1, \dots, u_{m+1}$ be pairwise distinct variables of sort $\texttt{loc}$. Let ϕ be the formula: $*_{i=1}^{m} u_i \mapsto (x_i, y_i, u_{i+1}) * u_{m+1} \mapsto ()$. Let p and q be predicate symbols associated with the following rules:

$$\begin{array}{lcl}
p(u, a, b, c) & \Leftarrow & u \mapsto (v, v, u') * q(u') \\
p(u, a, b, c) & \Leftarrow & u \mapsto (v_1, v_2, u') * v_1 \not\approx v_2 * v_1 \not\approx a * v_1 \not\approx b * v_1 \not\approx c * q(u') \\
p(u, a, b, c) & \Leftarrow & u \mapsto (d, v_2, u') * v_2 \not\approx a * v_2 \not\approx b * v_2 \not\approx c * q(u') \\
 & & \quad \text{for all } d \in \{a, b, c\} \\
p(u, a, b, c) & \Leftarrow & u \mapsto (d_1, d_2, u') * p(u', a, b, c) \\
 & & \quad \text{for all } d_1, d_2 \in \{a, b, c\} \text{ where } d_1 \neq d_2 \\
q(u) & \Leftarrow & u \mapsto (v_1, v_2, u') * q(u') \\
q(u) & \Leftarrow & u \mapsto ()
\end{array}$$

Intuitively, any model $(\mathfrak{s}, \mathfrak{h})$ of ϕ encodes a candidate solution of the 3-coloring problem, where each variable $z \in \{x_1, \dots, x_m\} \cup \{y_1, \dots, y_m\}$ is mapped to an element $\mathfrak{s}(z)$ in $\mathfrak{U}_{\texttt{s}}$. The heap $\mathfrak{h}$ is a list of tuples linked on the last element and containing a tuple $(\mathfrak{s}(u_i), \mathfrak{s}(x_i), \mathfrak{s}(y_i), \mathfrak{s}(u_{i+1}))$ for all $(x_i, y_i) \in E$. To check that this candidate solution indeed fulfills the required properties, one has to verify that all the pairs $(\mathfrak{s}(x_i), \mathfrak{s}(y_i))$ are composed of distinct elements in $\{a, b, c\}$.

By definition of the rules for predicate p, $(\mathfrak{s}, \mathfrak{h})$ is a model of $p(u_1, a, b, c)$ iff the list contains a pair $(\mathfrak{s}(x), \mathfrak{s}(y))$ such that one of the following holds:

- $\mathfrak{s}(x) = \mathfrak{s}(y)$ (first rule of p),
- $\mathfrak{s}(x) \neq \mathfrak{s}(y)$ and $\mathfrak{s}(x) \notin \{\mathfrak{s}(a), \mathfrak{s}(b), \mathfrak{s}(c)\}$ (second rule of p),
- $\mathfrak{s}(x) \in \{\mathfrak{s}(a), \mathfrak{s}(b), \mathfrak{s}(c)\}$ and $\mathfrak{s}(y) \notin \{\mathfrak{s}(a), \mathfrak{s}(b), \mathfrak{s}(c)\}$ (third rule of p).

After the cell corresponding to this faulty pair is allocated, q is invoked to allocate the remaining part of the list. Thus $p(u_1, a, b, c)$ holds iff the model does *not* encode a solution of the 3-coloring problem, either because $\mathfrak{s}(x_i) = \mathfrak{s}(y_i)$ for some $(x_i, y_i) \in E$ or because one of the variables is mapped to an element distinct from a, b, c – note that by the above assumption, each of these variables occurs in the list. Consequently $\phi \vdash_{\mathfrak{R}} p(u_1, a, b, c)$ admits a counter-model iff there exists a model of ϕ that does not satisfy $p(u_1, a, b, c)$, i.e., iff the 3-coloring problem admits a solution (thus the entailment is valid iff the 3-coloring problem admits no solution). □

The results above motivate the following definition, that strengthens the notion of a deterministic set of rules.

Definition 3.5. A set of $\mathtt{P}$-rules $\mathfrak{R}$ is $\mathtt{loc}$-*deterministic* if it is deterministic and all the disequations occurring in the rules in $\mathfrak{R}$ are of the form $x \not\approx y$ with $x, y \in \mathcal{V}_{\mathtt{loc}}$.

The intuition behind $\mathtt{loc}$-deterministic rules is that, to get an efficient proof procedure, we have to restrict the amount of equational reasoning needed to establish the validity of the sequents. Disequations between locations are relatively easy to handle because (by definition of $\mathtt{P}$-rules) all existential variables of sort $\mathtt{loc}$ must be pairwise distinct (as they are allocated in distinct atoms). However, dealing with disequations between data is much more difficult, as evidenced by the proof of Lemma 3.4. Thus we restrict such disequations to those occurring in the initial sequent.

The rules associated with $\mathtt{als}$, $\mathtt{tree}$, $\mathtt{tll}$, $\mathtt{tptr}$ or $\mathtt{dll}$ in the introduction and in Example 2.8 are $\mathtt{loc}$-deterministic. In contrast, the following rules are deterministic, but not $\mathtt{loc}$-deterministic (where u, v denote variables of some sort distinct from $\mathtt{loc}$):

$$p(x,u) \quad \Leftarrow \quad x \mapsto (v) \curlywedge v \not\approx u \qquad\qquad p(x,u) \quad \Leftarrow \quad x \mapsto (u)$$

Rules that are $\mathtt{loc}$-deterministic are well-suited to model constructor-based data structures used in standard programming languages; for instance, lists could be represented as follows (where $\mathtt{cons}$ is a constant symbol denoting a constructor and y is a variable of some sort distinct from $\mathtt{loc}$, denoting data stored in the list):

$$\mathtt{ls}(x) \Leftarrow \quad x \mapsto (\mathtt{cons}, y, z) * \mathtt{ls}(z) \qquad\qquad \mathtt{ls}(x) \Leftarrow \quad x \mapsto ()$$

We end this section by establishing a key property of deterministic set of rules, namely the fact that every spatial formula ϕ is *precise*, in the sense of [4]: it is fulfilled on at most one subheap within a given structure.

Lemma 3.6. Let $\mathfrak{R}$ be a deterministic set of rules. For every spatial formula ϕ, for every store $\mathfrak{s}$ and for every heap $\mathfrak{h}$ there exists at most one heap $\mathfrak{h}'$ such that $\mathfrak{h}' \subseteq \mathfrak{h}$ and $(\mathfrak{s}, \mathfrak{h}') \models_{\mathfrak{R}} \phi$.

Proof:
The proof is by induction on the satisfiability relation $\models_{\mathfrak{R}}$. Note that by hypothesis ϕ is a spatial formula, hence contains no occurrences of $\approx$, $\not\approx$, $\curlywedge$ or $\wedge$. Assume that there exist two heaps $\mathfrak{h}'_1, \mathfrak{h}'_2$ such that $\mathfrak{h}'_i \subseteq \mathfrak{h}$ and $(\mathfrak{s}, \mathfrak{h}'_i) \models_{\mathfrak{R}} \phi$ (for $i = 1, 2$). We show that $\mathfrak{h}'_1 = \mathfrak{h}'_2$.

- If $\phi = \mathit{emp}$ then necessarily $\mathfrak{h}_i = \emptyset$ for $i = 1, 2$ thus $\mathfrak{h}'_1 = \mathfrak{h}'_2$.
- If $\phi = y_0 \mapsto (y_1, \ldots, y_n)$ then by Definition 2.3 we have $\mathfrak{h}'_i = \{(\mathfrak{s}(y_0), \ldots, \mathfrak{s}(y_n))\}$ for $i = 1, 2$ thus $\mathfrak{h}'_1 = \mathfrak{h}'_2$.
- If $\phi = \phi_1 * \phi_2$ then for all $i = 1, 2$ there exist two disjoint heaps $\mathfrak{h}^j_i$ (for $j = 1, 2$) such that $\mathfrak{h}'_i = \mathfrak{h}^1_i \uplus \mathfrak{h}^2_i$ for $i = 1, 2$ and $(\mathfrak{s}, \mathfrak{h}^j_i) \models_{\mathfrak{R}} \phi_j$, for $i, j \in \{1, 2\}$. Since $\mathfrak{h}^j_i \subseteq \mathfrak{h}'_i \subseteq \mathfrak{h}$ we get by the induction hypothesis $\mathfrak{h}^j_1 = \mathfrak{h}^j_2$ for $j = 1, 2$. Therefore $\mathfrak{h}^1_1 \uplus \mathfrak{h}^2_1 = \mathfrak{h}^1_2 \uplus \mathfrak{h}^2_2$, i.e., $\mathfrak{h}'_1 = \mathfrak{h}'_2$.
- If ϕ is a predicate atom of root x, then for $i = 1, 2$ we have $\phi \Leftarrow_{\mathfrak{R}} \lambda_i$, and there exists an associate $\mathfrak{s}_i$ of $\mathfrak{s}$ w.r.t. $\mathcal{V}(\psi_i) \setminus \mathcal{V}(\phi)$ such that $(\mathfrak{s}_i, \mathfrak{h}'_i) \models_{\mathfrak{R}} \lambda_i$. Since $\mathfrak{R}$ is a set of P-rules, λ_i is of the form $(x_i \mapsto \boldsymbol{y}_i * \phi_i) \curlywedge \xi_i$ and there exist disjoint heaps $\mathfrak{h}^j_i$ (for $j = 1, 2$) such that the following conditions are satisfied: (i) $\mathfrak{h}'_i = \mathfrak{h}^1_i \uplus \mathfrak{h}^2_i$; (ii) $(\mathfrak{s}_i, \mathfrak{h}^1_i) \models_{\mathfrak{R}} x_i \mapsto (\boldsymbol{y}_i)$; (iii) $(\mathfrak{s}_i, \mathfrak{h}^2_i) \models_{\mathfrak{R}} \phi_i$; (iv) and $\mathfrak{s}_i \models \xi_i$. Furthermore, x_i must be the root of ϕ, thus $x_1 = x_2 = x$. For $i = 1, 2$ we have $\mathfrak{h}^1_i = \{(\mathfrak{s}(x_i), \mathfrak{s}_i(\boldsymbol{y}_i))\}$, and since $\mathfrak{h}^1_i \subseteq \mathfrak{h}$, necessarily $\mathfrak{s}_1(\boldsymbol{y}_1) = \mathfrak{s}_2(\boldsymbol{y}_2)$ and $\mathfrak{h}^1_1 = \mathfrak{h}^1_2$. The heap $\mathfrak{h}^2_i$ is the restriction of $\mathfrak{h}'_i$ to the locations distinct from $\mathfrak{s}(x)$. We distinguish two cases.
  - Assume that the inductive rules applied on ϕ to respectively derive λ_1 and λ_2 are different. We may assume by α-renaming that $(\mathcal{V}(\lambda_1) \setminus \mathcal{V}(\phi)) \cap (\mathcal{V}(\lambda_2) \setminus \mathcal{V}(\phi)) = \emptyset$, which entails that there exists a store $\mathfrak{s}'$ that coincides with $\mathfrak{s}_i$ on $\mathcal{V}(\lambda_i)$ (since $\mathfrak{s}_1$ and $\mathfrak{s}_2$ coincide on the variables in $\mathcal{V}(\phi)$). Then we have $\mathfrak{s}' \models \boldsymbol{y}_1 \approx \boldsymbol{y}_2$ (since $\mathfrak{s}_1(\boldsymbol{y}_1) = \mathfrak{s}_2(\boldsymbol{y}_2)$) and $\mathfrak{s}' \models \xi_i$ (since $\mathfrak{s}_i \models \xi_i$), which entails that the formula $\boldsymbol{y}_1 \approx \boldsymbol{y}_2 \wedge \xi_1 \wedge \xi_2$ is satisfiable, contradicting the fact that $\mathfrak{R}$ is deterministic.
  - Assume that the same rule is used to derive both λ_1 and λ_2. We may assume in this case (again by α-renaming) that the vector of variables occurring in λ_1 and λ_2 are the same, so that $\boldsymbol{y}_1 = \boldsymbol{y}_2$ and $\phi_1 = \phi_2$. Since $\mathfrak{R}$ is a set of P-rules, all variables z in $\mathcal{V}(\phi_i) \setminus \mathcal{V}(\phi)$ occur in $\boldsymbol{y}_i$. As $\mathfrak{s}_1(\boldsymbol{y}_1) = \mathfrak{s}_2(\boldsymbol{y}_2)$, this entails that $\mathfrak{s}_1(z) = \mathfrak{s}_2(z)$ holds for all such variables, thus $\mathfrak{s}_1 = \mathfrak{s}_2$. Consequently, $(\mathfrak{s}_1, \mathfrak{h}^2_i) \models_{\mathfrak{R}} \phi_1$, for all $i = 1, 2$ with $\mathfrak{h}^2_i \subseteq \mathfrak{h}'_i \subseteq \mathfrak{h}$. By the induction hypothesis this entails that $\mathfrak{h}^2_1 = \mathfrak{h}^2_2$, thus $\mathfrak{h}'_1 = \mathfrak{h}'_2$.

□

Example 3.7. Lemma 3.6 does not hold if the rules are not deterministic. For instance, the formula $\mathtt{ls}(x, y)$ (with the rules given in the introduction) has two models $(\mathfrak{s}, \mathfrak{h})$ and $(\mathfrak{s}, \mathfrak{h}')$ where $\mathfrak{h}'$ is a strict subheap of $\mathfrak{h}$: $\mathfrak{s}(x) = \ell_1$, $\mathfrak{s}(y) = \ell_2$, $\mathfrak{h} = \{\ell_1 \mapsto (\ell_2), \ell_2 \mapsto (\ell_2)\}$ and $\mathfrak{h}' = \{\ell_1 \mapsto (\ell_2)\}$. Intuitively, the formula $\mathtt{ls}(y, y)$ (which is useful to derive $\mathtt{ls}(x, y)$) can be derived by any of the two rules of $\mathtt{ls}$, yielding two different models. In contrast $\mathtt{als}(x, y)$ (with the rules of Example 2.8) has only one model with the store $\mathfrak{s}$ and a heap included in $\mathfrak{h}$, namely $(\mathfrak{s}, \mathfrak{h}')$.

4. Proof Procedure

From now on, we consider a fixed loc-deterministic set of P-rules $\mathfrak{R}$, satisfying Assumptions 2.12 and 2.15. For technical convenience, we also assume that $\mathfrak{R}$ is nonempty and that every constant in $\mathcal{C}$ occurs in a rule in $\mathfrak{R}$.

4.1. Some Basic Properties of P-Rules

We begin by introducing some definitions and deriving straightforward consequences of the definition of P-rules. We shall denote by $alloc(\lambda)$ the multiset of variables allocated by a formula λ:

Definition 4.1. For every formula λ, we denote by $alloc(\lambda)$ the multiset of variables x such that λ contains a spatial atom with root x.

Lemma 4.2 states that the variables in $alloc(\lambda)$ are necessarily allocated in every model of λ, which entails (Corollary 4.3) that they must be associated with pairwise distinct locations. Moreover, a formula distinct from emp has at least one root, hence allocates at least one variable (Corollary 4.4).

Lemma 4.2. Let λ be a formula. If $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \lambda$ and $x \in alloc(\lambda)$ then $\mathfrak{s}(x) \in dom(\mathfrak{h})$.

Proof:
By hypothesis, λ is of the form $(\alpha * \phi) \curlywedge \xi$ where α is a spatial atom with root x. Thus $(\mathfrak{s}, \mathfrak{h}) \models \alpha * \phi$ and there exist disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{R}} \alpha$, $(\mathfrak{s}, \mathfrak{h}_2) \models_{\mathfrak{R}} \phi$, and $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$. Since the root of α is x, α is either of the form $x \mapsto \boldsymbol{y}$ or of the form $p(x, \boldsymbol{y})$ where $p \in \mathcal{P}$. In the former case, it is clear that $\mathfrak{s}(x) \in dom(\mathfrak{h}_1) \subseteq dom(\mathfrak{h})$ since $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{R}} \alpha$. In the latter case, we have $\alpha \Leftarrow_{\mathfrak{R}} \lambda'$ and $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \lambda'$, where $\mathfrak{s}'$ is an associate of $\mathfrak{s}$ w.r.t. $\mathcal{V}(\lambda') \setminus \mathcal{V}(\alpha)$. Since $\mathfrak{R}$ is a set of P-rules, necessarily λ' contains a points-to atom of the form $x \mapsto \boldsymbol{z}$, which entails that $\mathfrak{s}'(x) \in dom(\mathfrak{h}_1)$, hence $\mathfrak{s}(x) \in dom(\mathfrak{h})$. □

Corollary 4.3. Let λ be a formula and let $(\mathfrak{s}, \mathfrak{h})$ be an $\mathfrak{R}$-model of λ. If $\{x, y\} \subseteq_m alloc(\lambda)$ then $\mathfrak{s}(x) \neq \mathfrak{s}(y)$. In particular, if $alloc(\lambda)$ contains two occurrences of the same variable x then λ is unsatisfiable.

Proof:
By definition, λ is of the form $(\alpha_1 * \alpha_2 * \phi) \curlywedge \xi$, where α_1 and α_2 are spatial atoms of roots x and y, respectively, with $alloc(\alpha_1) = \{x\}$ and $alloc(\alpha_2) = \{y\}$. If λ admits a model $(\mathfrak{s}, \mathfrak{h})$, then there exists disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2, \mathfrak{h}'$ such that $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2 \uplus \mathfrak{h}'$, $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \alpha_i$ (for $i = 1, 2$) and $(\mathfrak{s}, \mathfrak{h}') \models_{\mathfrak{R}} \phi$. By Lemma 4.2 we have $\mathfrak{s}(x) \in dom(\mathfrak{h}_1)$ and $\mathfrak{s}(y) \in dom(\mathfrak{h}_2)$, thus $\mathfrak{s}(x) \neq \mathfrak{s}(y)$ since $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are disjoint. □

Corollary 4.4. Let ϕ be a spatial formula and let $(\mathfrak{s}, \mathfrak{h})$ be an $\mathfrak{R}$-model of ϕ. If $\phi \neq emp$ then $\mathfrak{h} \neq \emptyset$.

Proof:
Since $\phi \neq emp$, necessarily ϕ contains at least one atom α, thus $root(\alpha) \in alloc(\phi)$. Then the result follows immediately from Lemma 4.2. □

Corollary 4.3 motivates the following definition, which provides a simple syntactic criterion to identify some formulas that cannot be satisfiable, due to the fact that the same variable is allocated twice.

Definition 4.5. A formula λ is *heap-unsatisfiable* if $alloc(\lambda)$ contains two occurrences of the same variable. Otherwise, it is *heap-satisfiable*.

The next proposition states that every location that is referred to in the heap of some model of λ must be reachable from one of the roots of λ. This follows from the fact that, by definition of P-rules, the set of allocated locations has a tree-shaped structure: the root of each atom invoked in an inductive rule must be connected to the location allocated by the rule (see Condition 2 in Definition 2.7).

Proposition 4.6. Let λ be a symbolic heap and let $(\mathfrak{s}, \mathfrak{h})$ be an $\mathfrak{R}$-model of λ. For every $\ell \in \mathit{ref}(\mathfrak{h})$, there exists $x \in alloc(\lambda)$ such that $\mathfrak{s}(x) \to^*_{\mathfrak{h}} \ell$.

Proof:
The proof is by induction on the satisfiability relation. We establish the result also for spatial formulas and pure formulas.

- If $\lambda = emp$ or is λ is a pure formula then $\mathit{ref}(\lambda) = \emptyset$ hence the proof is immediate.

- If $\lambda = y_0 \mapsto (y_1, \dots, y_n)$, then $\mathfrak{h} = \{(\mathfrak{s}(y_0), \dots, \mathfrak{s}(y_n))\}$, by Definition 2.3, thus $\mathit{ref}(\mathfrak{h}) = \{\mathfrak{s}(y_i) \mid i = 0, \dots, n, \text{and } y_i \text{ is of sort } \mathtt{loc}\}$ and $\mathfrak{s}(y_0) \to^*_{\mathfrak{h}} \mathfrak{s}(y_i)$, for all $i = 1, \dots, n$ such that y_i is of sort loc. By Definition 4.1 $alloc(\lambda) = \{y_0\}$, thus the proof is completed.

- If $\lambda = \phi \curlywedge \xi$, where ϕ is a spatial formula and ξ is a pure formula distinct from $\top$, then we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi$ and $alloc(\lambda) = alloc(\phi)$, hence the result follows immediately from the induction hypothesis.

- If $\lambda = \phi_1 * \phi_2$, then there exist two disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$ and $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \phi_i$, for all $i = 1, 2$. If $\ell \in \mathit{ref}(\mathfrak{h})$ then necessarily $\ell \in \mathit{ref}(\mathfrak{h}_i)$ for some $i = 1, 2$. By the induction hypothesis, we deduce that there exists $x \in alloc(\phi_i)$ such that $\mathfrak{s}(x) \to^*_{\mathfrak{h}_i} \ell$. Since $alloc(\phi_i) \subseteq_m alloc(\phi)$ and $\to^*_{\mathfrak{h}_i} \subseteq \to^*_{\mathfrak{h}}$ by Proposition 2.1, we obtain the result.

- If λ is a predicate atom, then $\lambda \Leftarrow_{\mathfrak{R}} \gamma$ and $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \gamma$ for some formula γ and some associate $\mathfrak{s}'$ of $\mathfrak{s}$ w.r.t. $\mathcal{V}(\gamma) \setminus \mathcal{V}(\lambda)$. Let $\ell \in \mathit{ref}(\mathfrak{h})$. By the induction hypothesis, there exists $x \in alloc(\gamma)$ such that $\mathfrak{s}(x) \to^*_{\mathfrak{h}} \ell$. By definition, x is the root of some atom α in γ. If α is a points-to atom, then since $\lambda \Leftarrow_{\mathfrak{R}} \gamma$ is an instance of a rule in $\mathfrak{R}$ and all rules are P-rules, x must be the root of λ; in this case $x \in alloc(\lambda)$ and the proof is completed. Otherwise, x is the root of a spatial atom in γ, and, because all rules are P-rules, γ must contain an atom of the form $y_0 \mapsto (y_1, \dots, y_n)$, such that $y_0 = root(\lambda)$ and $y_i = x$, for some $i = 1, \dots, n$. Since $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \gamma$, we have $(\mathfrak{s}(y_0), \dots, \mathfrak{s}(y_n)) \in \mathfrak{h}$, hence $\mathfrak{s}(y_0) \to_{\mathfrak{h}} \mathfrak{s}(x)$. Using the fact that $\mathfrak{s}(x) \to^*_{\mathfrak{h}} \ell$, we deduce that $\mathfrak{s}(y_0) \to^*_{\mathfrak{h}} \ell$, hence the proof is completed since $alloc(\lambda) = \{y_0\}$.

□

The next lemma asserts a key property of the considered formulas: all the locations occurring in the heap of a model of some formula ϕ are either allocated or associated with a variable that is free in ϕ. This follows from the definition of P-rules: all variables of sort `loc` that are existentially quantified in an inductive rule must be allocated (at the next recursive call). Recall that spatial formulas contain no quantifiers.

Lemma 4.7. Let ϕ be a spatial formula and let $(\mathfrak{s}, \mathfrak{h})$ be an $\mathfrak{R}$-model of ϕ. Then the following inclusion holds: $\mathit{ref}(\mathfrak{h}) \subseteq \mathit{dom}(\mathfrak{h}) \cup \mathfrak{s}(\mathcal{V}(\phi))$.

Proof:
The proof is by induction on the relation $\models_{\mathfrak{R}}$. Note that as ϕ is spatial, it contains no occurrence of $\approx$, $\not\approx$, $\curlywedge$ and $\wedge$.

- If ϕ is of the form $y_0 \mapsto (y_1, \dots, y_n)$ then by definition $\mathfrak{h} = \{(\mathfrak{s}(y_0), \dots, \mathfrak{s}(y_n))\}$ and $\mathit{ref}(\mathfrak{h}) = \{\mathfrak{s}(y_0), \dots, \mathfrak{s}(y_n)\} = \mathfrak{s}(\mathcal{V}(\phi))$.

- If $\phi = \mathit{emp}$ then $\mathfrak{h} = \emptyset$ thus $\mathit{ref}(\mathfrak{h}) = \emptyset$ and the proof is immediate.

- If ϕ is a predicate atom then we have $\phi \Leftarrow_{\mathfrak{R}} \psi \curlywedge \xi$, and $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \psi$, for some associate $\mathfrak{s}'$ of $\mathfrak{s}$ w.r.t. $\mathcal{V}(\psi \curlywedge \xi) \setminus \mathcal{V}(\phi)$. Let $\ell \in \mathit{ref}(\mathfrak{h}) \setminus \mathit{dom}(\mathfrak{h})$. By the induction hypothesis, $\ell = \mathfrak{s}'(x)$ for some variable $x \in \mathcal{V}(\psi)$. If $x \in \mathcal{V}(\phi)$ then necessarily $\mathfrak{s}'(x) = \mathfrak{s}(x)$, thus $\ell \in \mathfrak{s}(\mathcal{V}(\phi))$ and the proof is completed. Otherwise, by Definition 2.7, since all the rules in $\mathfrak{R}$ are P-rules; x occurs as the root of some predicate atom in ψ, i.e., $x \in \mathit{alloc}(\psi)$. By Lemma 4.2 we deduce that $\mathfrak{s}'(x) \in \mathit{dom}(\mathfrak{h})$, i.e., $\ell \in \mathit{dom}(\mathfrak{h})$ which contradicts our assumption.

- If ϕ is of the form $\phi_1 * \phi_2$ then there exist disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \phi_i$ (for $i = 1, 2$) and $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$. Let $\ell \in \mathit{ref}(\mathfrak{h}) \setminus \mathit{dom}(\mathfrak{h})$. Necessarily we have $\ell \in \mathit{ref}(\mathfrak{h}_i)$, for some $i = 1, 2$ and $\ell \notin \mathit{dom}(\mathfrak{h}_i)$, hence by the induction hypothesis we deduce that $\ell = \mathfrak{s}(x)$, for some $x \in \mathcal{V}(\phi_i)$. Since $\mathcal{V}(\phi_i) \subseteq \mathcal{V}(\phi)$, the proof is completed.

□

4.2. A Restricted Entailment Relation

We introduce a simple syntactic criterion, used in the inference rules of Section 4.3, that is sufficient to ensure that a given pure formula ξ holds in every counter-model of a sequent with left-hand side λ and multiset of variables V. The idea is to test that ξ either occurs in λ, is trivial, or is a disequation entailed by the fact that the considered store must be injective on $\mathit{alloc}(\lambda) \cup V$ (using Definition 2.17 and Corollary 4.3). Lemma 4.10 states that the relation satisfies the expected property.

Definition 4.8. Let λ be a symbolic heap, ξ be a pure formula and let V be a multiset of variables. We write $\lambda \rhd_V \xi$ if for every atom ζ occurring in ξ, one of the following conditions holds:

1. ζ occurs in λ;

2. either $\zeta = (t \approx t)$ for some variable t, or $\zeta = (t_1 \not\approx t_2)$ and t_1, t_2 are distinct constants;

3. $\zeta = (x_1 \not\approx x_2)$ modulo commutativity, and one of the following holds: $\{x_1, x_2\} \subseteq_m alloc(\lambda)$, $(x_1 \in alloc(\lambda)$ and $x_2 \in V)$ or $\{x_1, x_2\} \subseteq_m V$. This is equivalent to stating that $\{x_1, x_2\} \subseteq_m alloc(\lambda) + V$ where $alloc(\lambda) + V$ denotes as usual the union of the multisets $alloc(\lambda)$ and V.

Example 4.9. Consider the symbolic heap $\lambda = (p(x, y) * q(z)) \curlywedge x \not\approx y$, and let $V = \{u\}$. We have $\lambda \rhd_V x \not\approx y \wedge x \not\approx z \wedge x \not\approx u$. Indeed, x and z are necessarily distinct since they are allocated by distinct atoms $p(x, y)$ and $q(z)$ (as, by definition of the P-rules, every predicate allocates it first parameter) x cannot be equal to u as $u \in V$ and V is intended to denote a set of non-allocated variables (see Definition 2.17) and x is allocated, and x cannot be equal to y as the disequation $x \not\approx y$ occurs in λ.

Lemma 4.10. Let λ be a symbolic heap and let ξ be a pure formula such that $\lambda \rhd_V \xi$. For every structure $(\mathfrak{s}, \mathfrak{h})$, if $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \lambda$, $\mathfrak{s}(V) \cap dom(\mathfrak{h}) = \emptyset$ and $\mathfrak{s}$ is injective on V then $\mathfrak{s} \models_{\mathfrak{R}} \xi$.

Proof:
We show that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \zeta$, for all atoms ζ in ξ. We consider each case in Definition 4.8 separately:

1. If ζ occurs in λ then since $(\mathfrak{s}, \mathfrak{h}) \models \gamma$, necessarily $\mathfrak{s} \models_{\mathfrak{R}} \zeta$.

2. If $\zeta = (t \approx t)$ then it is clear that $\mathfrak{s} \models \zeta$. If $\zeta = (t_1 \not\approx t_2)$ and t_1, t_2 are distinct constants then $\mathfrak{s}(t_1) \neq \mathfrak{s}(t_2)$ since all stores are injective on constants by definition.

3. If $\zeta = x_1 \not\approx x_2$ and $\{x_1, x_2\} \subseteq_m alloc(\lambda)$ then by Corollary 4.3, we get $\mathfrak{s}(x_1) \neq \mathfrak{s}(x_2)$ since $\mathfrak{s}(x_1)$ and $\mathfrak{s}(x_2)$ must be allocated in disjoint heaps. Thus $\mathfrak{s} \models_{\mathfrak{R}} \zeta$. If $x_1 \in alloc(\phi)$ and $x_2 \in V$ then we have $\mathfrak{s}(x_1) \in dom(\mathfrak{h})$ by Lemma 4.2, and since $\mathfrak{s}(V) \cap dom(\mathfrak{h}) = \emptyset$, we deduce that $\mathfrak{s} \models_{\mathfrak{R}} \zeta$. Finally, if $\{x_1, x_2\} \subseteq_m V$ then $\mathfrak{s}(x_1) \neq \mathfrak{s}(x_2)$, because $\mathfrak{s}$ is injective on V by hypothesis.

□

4.3. Inference Rules

We consider the rules represented in Figure 3. The rules apply modulo AC, they are intended to be applied bottom-up: a rule is *applicable* on a sequent $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$ if there exists an instance of the rule the conclusion of which is $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$. We recall that an inference rule is *sound* if the validity of the premises entails the validity of the conclusion, and *invertible* if the converse holds.

Remark 4.11. The application conditions given in Figure 3 are the most general ones ensuring that the rules are sound. Additional application conditions will be provided afterwards (see Definition 4.31) to obtain a proof procedure that runs in polynomial time. The latter conditions are rather complex, and in our opinion introducing them at this point could hinder the understanding of the rules.

We provide some examples and explanations concerning the inference rules.

Example 4.12. Rules R (replacement) and E (elimination) handle equational reasoning. For instance, given the sequent $(p(x, u) * p(y, u)) \curlywedge (u \approx v) \vdash_{\mathfrak{R}}^{\emptyset} (p(x, u) * p(y, v)) \curlywedge (x \not\approx y)$, one may first apply R, yielding: $p(x, u) * p(y, u) \vdash_{\mathfrak{R}}^{\emptyset} (p(x, u) * p(y, u)) \curlywedge (x \not\approx y)$. As $\{x, y\} \subseteq_m alloc(p(x, u) * p(y, x))$, E applies, which yields the trivially valid sequent $p(x, u) * p(y, u) \vdash_{\mathfrak{R}}^{\emptyset} p(x, u) * p(y, u)$.

$$\mathtt{R}: \frac{\phi\{x \leftarrow t\} \curlywedge \xi\{x \leftarrow t\} \vdash_{\mathfrak{R}}^{V\{x \leftarrow t\}} \gamma\{x \leftarrow t\}}{\phi \curlywedge (x \approx t \wedge \xi) \vdash_{\mathfrak{R}}^{V} \gamma} \quad \text{if } x \in \mathcal{V}$$

$$\mathtt{E}: \frac{\lambda \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta}{\lambda \vdash_{\mathfrak{R}}^{V} \psi \curlywedge (\zeta \wedge \zeta')} \quad \text{if } \lambda \triangleright_V \zeta'$$

$$\mathtt{S}: \frac{\phi_1 \curlywedge \xi_1 \vdash_{\mathfrak{R}}^{V \cup alloc(\phi_2)} \psi_1 \curlywedge \zeta_1 \qquad \phi_2 \curlywedge \xi_2 \vdash_{\mathfrak{R}}^{V \cup alloc(\phi_1)} \psi_2 \curlywedge \zeta_2}{(\phi_1 * \phi_2) \curlywedge (\xi_1 \wedge \xi_2) \vdash_{\mathfrak{R}}^{V} (\psi_1 * \psi_2) \curlywedge (\zeta_1 \wedge \zeta_2)} \quad \text{if } \phi_1 \neq \mathit{emp} \text{ and } \phi_2 \neq \mathit{emp}$$

$$\mathtt{U}: \frac{(\phi'_1 * \phi) \curlywedge (\xi'_1 \wedge \xi) \vdash_{\mathfrak{R}}^{V} \gamma \qquad \ldots \qquad (\phi'_n * \phi) \curlywedge (\xi'_n \wedge \xi) \vdash_{\mathfrak{R}}^{V} \gamma}{(p(\boldsymbol{t}) * \phi) \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \gamma} \text{ if } p(\boldsymbol{t}) \leadsto_{\mathfrak{R}} \{\phi'_i \curlywedge \xi'_i \mid i = 1, \ldots, n\}$$

$$\mathtt{I}: \frac{(x \mapsto (y_1, \ldots, y_k) * \phi) \curlywedge \xi \vdash_{\mathfrak{R}}^{V} (x \mapsto (y_1, \ldots, y_k) * \psi'\sigma * \psi) \curlywedge \zeta}{(x \mapsto (y_1, \ldots, y_k) * \phi) \curlywedge \xi \vdash_{\mathfrak{R}}^{V} (p(x, \boldsymbol{z}) * \psi) \curlywedge \zeta} \quad \text{if all the conditions below hold:}$$

(i) $p(x, \boldsymbol{z}) \Leftarrow_{\mathfrak{R}} (x \mapsto (u_1, \ldots, u_k) * \psi') \curlywedge \zeta'$; (ii) σ is a substitution whose domain is included in $\{u_1, \ldots, u_k\} \setminus (\{x\} \cup \boldsymbol{z})$; (iii) $\sigma(u_i) = y_i$, for all $i \in \{1, \ldots, k\}$; and (iv) $(x \mapsto (y_1, \ldots, y_k) * \phi) \curlywedge \xi \triangleright_V \zeta'\sigma$

$$\mathtt{W}: \frac{\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \gamma}{\phi \curlywedge (\xi \wedge \xi') \vdash_{\mathfrak{R}}^{V} \gamma} \quad \text{if one of the conditions below holds:}$$

(i) $\xi' = x \not\approx y$ and $\phi \triangleright_V x \not\approx y$; or (ii) $\xi' = \bigwedge_{i=1}^{n} x \not\approx y_i$ and $x \notin \mathcal{V}(\phi) \cup \mathcal{V}(\xi) \cup \mathcal{V}(\gamma) \cup \{y_1, \ldots, y_n\}$

$$\mathtt{C}: \frac{\phi\{x \leftarrow y\} \curlywedge \xi\{x \leftarrow y\} \vdash_{\mathfrak{R}}^{V\{x \leftarrow y\}} \gamma\{x \leftarrow y\} \qquad \phi \curlywedge (\xi \wedge x \not\approx y) \vdash_{\mathfrak{R}}^{V} \gamma}{\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \gamma} \quad \text{if } x, y \in \mathcal{V}_{\mathtt{loc}}$$

$$\mathtt{V}: \frac{\lambda \vdash_{\mathfrak{R}}^{V} \gamma}{\lambda \vdash_{\mathfrak{R}}^{V \cup \{x\}} \gamma} \quad \text{if } x \notin \mathcal{V}(\lambda) \cup \mathcal{V}(\gamma)$$

Figure 3. The Inference rules

Example 4.13. Rule S (separation) applies on the sequent $p(x) * p(y) \vdash_{\mathfrak{R}}^{\emptyset} q(x) * r(y)$, yielding $p(x) \vdash_{\mathfrak{R}}^{\{y\}} q(x)$ and $p(y) \vdash_{\mathfrak{R}}^{\{x\}} r(y)$ (as $\{x\} = \mathit{alloc}(p(x))$ and $\{y\} = \mathit{alloc}(p(y))$). The addition of y (resp. x) to the variables associated with the sequent allows us to keep track of the fact that these variables cannot be allocated in $p(x)$ (resp. $p(y)$) as they are already allocated in the other part of the heap. Note that the rule also yields $p(x) \vdash_{\mathfrak{R}}^{\{y\}} r(y)$ and $p(y) \vdash_{\mathfrak{R}}^{\{x\}} q(x)$, however as we shall see later (see Definition 4.24) the latter premises cannot be valid and this application can be dismissed.

Example 4.14. Rules U (unfolding) and I (imitation) both unfold inductively defined predicate symbols. U unfolds predicates occurring on the left-hand side of a sequent, yielding one premise for each inductive rule. In contrast, I applies on the right-hand side and selects one rule in a non-deterministic way (provided it fulfills the rule application condition), yielding exactly one premise. Let $\mathfrak{R}$ be the following set of rules, where $\mathtt{a}, \mathtt{b}$ are constant symbols and z, z_1, z_2 are variables of the same sort as $\mathtt{a}$ and $\mathtt{b}$:

$$\begin{array}{rcc}
p(x) & \Leftarrow & x \mapsto (\mathtt{a}, x) \\
p(x) & \Leftarrow & x \mapsto (\mathtt{b}, x) \\
q(x) & \Leftarrow & x \mapsto (z, y) \\
q(x) & \Leftarrow & x \mapsto (z_1, z_2, x)
\end{array}$$

Rule U applies on $p(x, y) \vdash_{\mathfrak{R}}^{\emptyset} q(x)$, yielding $x \mapsto (\mathtt{a}, x) \vdash_{\mathfrak{R}}^{\emptyset} q(x)$ and $x \mapsto (\mathtt{b}, x) \vdash_{\mathfrak{R}}^{\emptyset} q(x)$. Then I applies on both sequents, with the respective substitutions $\{y \leftarrow x, z \leftarrow \mathtt{a}\}$ and $\{y \leftarrow x, z \leftarrow \mathtt{b}\}$, yielding $x \mapsto (\mathtt{a}, x) \vdash_{\mathfrak{R}}^{\emptyset} x \mapsto (\mathtt{a}, x)$ and $x \mapsto (\mathtt{b}, x) \vdash_{\mathfrak{R}}^{\emptyset} x \mapsto (\mathtt{b}, x)$. Note that I cannot be applied with the inductive rule $q(x) \Leftarrow x \mapsto (z_1, z_2, x)$, as $(\mathtt{a}, x)$ and $(\mathtt{b}, x)$ are not instances of (z_1, z_2, x).

Example 4.15. Rules W (weakening) and V (variable weakening) allow to get rid of irrelevant information, which is essential for termination. For instance one may deduce $p(x) \vdash_{\mathfrak{R}}^{\emptyset} q(x)$ from $p(x) \curlywedge (u \not\approx v \wedge u \not\approx w) \vdash_{\mathfrak{R}}^{\emptyset} q(x)$. Indeed, if the former sequent admits a counter-model, then one gets a counter-model of the latter one by associating u, v, w with pairwise distinct elements.

Example 4.16. Rule C performs a case analysis on $x \approx y$. It is essential to allow further applications of Rule I in some cases. Consider the sequent $u \mapsto (x, x) * p(x) \vdash_{\mathfrak{R}}^{\emptyset} q(u, y)$, with the rules $q(u, y) \Leftarrow u \mapsto (y, z) * p(z)$, and $q(u, y) \Leftarrow (u \mapsto (z, z) * p(z)) \curlywedge z \not\approx y$. Rule I does not apply because there is no substitution σ with domain $\{z\}$ such that $(x, x) = (y, z)\sigma$, and $u \mapsto (x, x) * p(x) \not\models_V x \not\approx y$. The rule C yields $u \mapsto (y, y) * p(y) \vdash_{\mathfrak{R}}^{\emptyset} q(u, y)$ and $(u \mapsto (x, x) * p(x)) \curlywedge x \not\approx y \vdash_{\mathfrak{R}}^{\emptyset} q(u, y)$. Then the rule I applies on both sequents, yielding the premisses $u \mapsto (y, y) * p(y) \vdash_{\mathfrak{R}}^{\emptyset} u \mapsto (y, y) * p(y)$ and $(u \mapsto (x, x) * p(x)) \curlywedge x \not\approx y \vdash_{\mathfrak{R}}^{\emptyset} u \mapsto (x, x) * p(x)$.

We have the following facts, which can be verified by an inspection of the inference rules:

Proposition 4.17. Consider an equality-free sequent $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$ that is the conclusion of an inference rule, of which $\lambda' \vdash_{\mathfrak{R}}^{V'} \gamma'$ is a premise.

1. No rule can introduce any equality to $\lambda' \vdash_{\mathfrak{R}}^{V'} \gamma'$.
2. If $x \in V' \setminus V$, then the inference rule is either V or C.

3. If $alloc(\gamma') \subsetneq alloc(\gamma)$ then the inference rule is either S or C.

4. The only inference rules that can introduce new variables to γ' are I and C.

5. No rule introduces a disequation between terms of a sort distinct from `loc`.

6. The only rule that introduces a predicate atom to the right-hand side of a premise is I.

7. If $v \in (alloc(\gamma) \cup V) \setminus (alloc(\gamma') \cup V')$, then the inference rule must be C.

Proof:
The first six facts are straightforward to verify. Fact 7 holds because Rule R cannot apply if no equality occurs; if rule S is applied then the variables in $alloc(\gamma) \setminus alloc(\gamma')$ must occur in V'; Rule I deletes a predicate atom but introduces a points-to atom with the same root and rule V cannot be applied on variables occurring in $\mathcal{V}\gamma$. □

We establish additional basic properties of the inference rules. All the rules are sound and invertible, except for rule S that is only sound. The results follow easily from the semantics, except for the invertibility of I, which crucially depends on the fact that rules are deterministic. Indeed, I unfolds one atom on the right-hand side by selecting one specific inductive rule. In our case, at most one rule can be applied, which ensures that equivalence is preserved. This is a crucial point because otherwise one would need to consider disjunctions of formulas on the right-hand side of the sequent (one disjunct for each possible rule), which would make the procedure much less efficient.

Example 4.18. Consider the sequent $x \mapsto (y) * y \mapsto (z) \vdash_{\mathfrak{R}}^{\emptyset} p(x,z)$, with the rules

$$\begin{array}{lclclcl} p(x,z) & \Leftarrow & x \mapsto (y) * q(y,z) & \quad & p(x,z) & \Leftarrow & x \mapsto (y) * q'(y,y) \\ q(y,z) & \Leftarrow & y \mapsto (z) \curlywedge y \not\approx z & & q'(y,z) & \Leftarrow & y \mapsto (z) \end{array}$$

It is clear that the rules are not deterministic, as there is an overlap between the two rules associated with p. Applying rule I on the above sequent yields either $x \mapsto (y) * y \mapsto (z) \vdash_{\mathfrak{R}}^{\emptyset} x \mapsto (y) * q(y,z)$ or $x \mapsto (y) * y \mapsto (z) \vdash_{\mathfrak{R}}^{\emptyset} x \mapsto (y) * q'(y,y)$. None of these two possible premises is valid, although the initial sequent is valid. This shows that I is not invertible (although it is still sound) when $\mathfrak{R}$ is not deterministic. The intuition is that it is not possible to decide which rule must be applied on p before deciding whether z is equal to y or not.

Lemma 4.19. The rules R, E, U, W, V and C and I are sound and invertible. More specifically, for all heaps $\mathfrak{h}$, the conclusion of the rule admits a counter-model $(\mathfrak{s},\mathfrak{h})$ iff one of the premises admits a counter-model $(\mathfrak{s}',\mathfrak{h})$.

Proof:
We consider each rule separately (we refer to the definitions of the rules for notations) and establish the equivalence of the lemma. We recall (Definition 2.17) that a counter-model of a sequent is a structure $(\mathfrak{s},\mathfrak{h})$ that validates the antecedent, falsifies the conclusion, and is such that the store is injective on V and no variable in V is allocated.

R Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi \curlywedge (x \approx t \wedge \xi)$, that $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma$, that $\mathfrak{s}(V) \cap \mathit{dom}(\mathfrak{h}) = \emptyset$ and that $\mathfrak{s}$ is injective on V. Then we have $\mathfrak{s}(x) = \mathfrak{s}(t)$, thus $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi\{x \leftarrow t\} \curlywedge \xi\{x \leftarrow t\}$ and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma\{x \leftarrow t\}$. For all $y \in V\{x \leftarrow t\}$, we have either $y \in V$ and $\mathfrak{s}(y) \notin \mathit{dom}(\mathfrak{h})$, or $x \in V$ and $y = t$, thus $\mathfrak{s}(y) = \mathfrak{s}(t) = \mathfrak{s}(x) \notin \mathit{dom}(\mathfrak{h})$. Finally, assume (for the sake of contradiction) that $\{u, v\} \subseteq V\{x \leftarrow t\}$ with $\mathfrak{s}(u) = \mathfrak{s}(v)$. Then there exist variables u', v' such that $\{u', v'\} \subseteq V$, with $u'\{x \leftarrow t\} = u$ and $v'\{x \leftarrow t\} = v$. If u' and v' are both equal to x or both distinct from x then we have $\mathfrak{s}(u') = \mathfrak{s}(v')$, which contradicts the fact that $\mathfrak{s}$ is injective on V. If $u' = x$ and $v' \neq x$, then we have $\mathfrak{s}(v') = \mathfrak{s}(t) = \mathfrak{s}(x)$, again contradicting the fact that $\mathfrak{s}$ is injective on V. The proof is similar when $u' \neq x$ and $v' = x$. Consequently, $(\mathfrak{s}, \mathfrak{h})$ is also a counter-model of the premise.

Conversely, assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi\{x \leftarrow t\} \curlywedge \xi\{x \leftarrow t\}$; $\mathfrak{s}(V\{x \leftarrow t\}) \cap \mathit{dom}(\mathfrak{h}) = \emptyset$; the store $\mathfrak{s}$ is injective on $V\{x \leftarrow t\}$; and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma\{x \leftarrow t\}$. We consider the store $\mathfrak{s}'$ such that $\mathfrak{s}'(x) = \mathfrak{s}(t)$ and $\mathfrak{s}'(y) = \mathfrak{s}(y)$ if $y \neq x$. It is clear that $\mathfrak{s}' \models_{\mathfrak{R}} x \approx t$, $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \phi \curlywedge \xi$ and $(\mathfrak{s}', \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma$. For all $y \in V$, if $y \neq x$ then $y \in V\{x \leftarrow t\}$ and $\mathfrak{s}'(y) = \mathfrak{s}(y) \notin \mathit{dom}(\mathfrak{h})$; otherwise $y = x$ and $t \in V\{x \leftarrow t\}$, thus $\mathfrak{s}'(y) = \mathfrak{s}'(x) = \mathfrak{s}(t) \notin \mathit{dom}(\mathfrak{h})$. There only remains to check that $\mathfrak{s}'$ is injective on V, which is done by contradiction: if this is not the case then there exists $\{u, v\} \subseteq_m V$ such that $\mathfrak{s}'(u) = \mathfrak{s}'(v)$. Hence $\{u\{x \leftarrow t\}, v\{x \leftarrow t\}\} \subseteq_m V\{x \leftarrow t\}$, and we have $\mathfrak{s}'(u\{x \leftarrow t\}) = \mathfrak{s}(u\{x \leftarrow t\})$ and $\mathfrak{s}'(v\{x \leftarrow t\}) = \mathfrak{s}(v\{x \leftarrow t\})$. This contradicts the fact that $\mathfrak{s}$ is injective on $V\{x \leftarrow t\}$.

E Let $(\mathfrak{s}, \mathfrak{h})$ be a structure such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \lambda$; $\mathfrak{s}(V) \cap \mathit{dom}(\mathfrak{h}) = \emptyset$; the store $\mathfrak{s}$ is injective on V; and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \psi \curlywedge (\zeta \wedge \zeta')$. By the application condition of the rule we have $\lambda \triangleright_V \zeta'$, thus by Lemma 4.10, we deduce that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \zeta'$. Therefore $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \psi \curlywedge \zeta$. Conversely, it is clear that any counter-model of $\lambda \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ is a counter-model of $\lambda \vdash_{\mathfrak{R}}^{V} \psi \curlywedge (\zeta \wedge \zeta')$.

U Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} (p(\boldsymbol{t}) * \phi) \curlywedge \xi$, $\mathfrak{s}(V) \cap \mathit{dom}(\mathfrak{h}) = \emptyset$, $\mathfrak{s}$ is injective on V and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma$. By definition, $\mathfrak{s} \models \xi$, and there are disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{R}} p(\boldsymbol{t})$, $(\mathfrak{s}, \mathfrak{h}_2) \models_{\mathfrak{R}} \phi$ and $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$. We get that $p(\boldsymbol{t}) \Leftarrow_{\mathfrak{R}} \phi' \curlywedge \xi'$ and $(\mathfrak{s}', \mathfrak{h}_1) \models_{\mathfrak{R}} \phi' \curlywedge \xi'$, for some associate of $\mathfrak{s}$ w.r.t. $\mathcal{V}(\phi' \curlywedge \xi') \setminus \mathcal{V}(p(\boldsymbol{t}))$. We assume that $\mathcal{V}(\phi' \curlywedge \xi') \cap \mathcal{V}((p(\boldsymbol{t}) * \phi) \curlywedge \xi) \subseteq \boldsymbol{t}$ (by α-renaming). We get $\mathfrak{s}' \models \xi'$ and $(\mathfrak{s}', \mathfrak{h}_1 \uplus \mathfrak{h}_2) \models_{\mathfrak{R}} \phi' * \phi$, hence $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} (\phi' * \phi) \curlywedge (\xi' \wedge \xi)$. Furthermore, the formula $\phi' \curlywedge \xi'$ occurs in $\{\phi'_i \curlywedge \xi'_i \mid i = 1, \dots, n\}$, up to a renaming of the variables not occurring in $\boldsymbol{t}$, by definition of $\leadsto_{\mathfrak{R}}$. Thus $(\mathfrak{s}', \mathfrak{h})$ is a counter-model of a sequent $(\phi'_i * \phi) \curlywedge (\xi'_i \wedge \xi) \vdash_{\mathfrak{R}}^{V} \gamma$, for some $i = 1, \dots, n$.

Conversely, let $(\mathfrak{s}, \mathfrak{h})$ be a structure such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} (\phi'_i * \phi) \curlywedge (\xi'_i \wedge \xi)$, $\mathfrak{s}(V) \cap \mathit{dom}(\mathfrak{h}) = \emptyset$, $\mathfrak{s}$ is injective on V and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma$. We deduce that $\mathfrak{s} \models \xi'_i \wedge \xi$ and there exist disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{R}} \phi'_i$ and $(\mathfrak{s}, \mathfrak{h}_2) \models_{\mathfrak{R}} \phi$. By the application condition of the rule we have $p(\boldsymbol{t}) \Leftarrow_{\mathfrak{R}} \phi'_i \curlywedge \xi'_i$, thus $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{R}} p(\boldsymbol{t})$ by definition of the semantics of predicate atoms (since $\mathfrak{s}$ is an extension of itself). Consequently, $(\mathfrak{s}, \mathfrak{h}_1 \uplus \mathfrak{h}_2) \models_{\mathfrak{R}} p(\boldsymbol{t}) * \phi$, hence $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} (p(\boldsymbol{t}) * \phi) \curlywedge \xi$, and $(\mathfrak{s}, \mathfrak{h})$ is a counter-model of $(p(\boldsymbol{t}) * \phi) \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \gamma$.

W Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi \curlywedge \xi$, $\mathfrak{s}(V) \cap \mathit{dom}(\mathfrak{h}) = \emptyset$, $\mathfrak{s}$ is injective on V and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma$. If ξ' is $x \not\approx y$, with $\phi \triangleright_V x \not\approx y$, then, by Lemma 4.10, we have $\mathfrak{s}(x) \neq \mathfrak{s}(y)$, thus $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi \curlywedge (\xi \wedge \xi')$, hence $(\mathfrak{s}, \mathfrak{h})$ is a counter-model of the conclusion of the rule. Assume that $\xi' = \bigwedge_{i=1}^{n} x \not\approx y_i$

with $x \not\in \mathcal{V}(\phi) \cup \mathcal{V}(\xi) \cup \mathcal{V}(\gamma) \cup \{y_1, \dots, y_n\}$. Let $\mathfrak{s}'$ be a store such that $\mathfrak{s}'(y) = \mathfrak{s}(y)$ if $y \neq x$ and $\mathfrak{s}'(x)$ is an arbitrarily chosen location not occurring in $dom(\mathfrak{h}) \cup \{\mathfrak{s}(y_i) \mid i = 1, \dots, n\}$ (such a location exists since $\mathfrak{h}$ is finite and $\mathfrak{U}_{\mathtt{loc}}$ is infinite). By definition, we have $\mathfrak{s}'(V) \cap dom(\mathfrak{h}) = \emptyset$, $\mathfrak{s}' \models \bigwedge_{i=1}^{n} x \not\approx y_i$, $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \phi \curlywedge \xi$ (since $\mathfrak{s}'$ and $\mathfrak{s}$ coincide on $\mathcal{V}(\phi) \cup \mathcal{V}(\xi)$) and $(\mathfrak{s}', \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma$ (since $\mathfrak{s}'$ and $\mathfrak{s}$ coincide on $\mathcal{V}(\gamma)$). Thus $(\mathfrak{s}', \mathfrak{h})$ is a counter-model of $\phi \curlywedge (\xi \wedge \bigwedge_{i=1}^{n} x \not\approx y_i) \vdash_{\mathfrak{R}}^{V} \gamma$.

Conversely, it is clear that any counter-model of $\phi \curlywedge (\xi \wedge \xi') \vdash_{\mathfrak{R}}^{V} \gamma$ is a counter-model of $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \gamma$.

C Let $(\mathfrak{s}, \mathfrak{h})$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi \curlywedge \xi$, $\mathfrak{s}(V) \cap dom(\mathfrak{h}) = \emptyset$, $\mathfrak{s}$ is injective on V and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma$. We distinguish two cases.

- If $\mathfrak{s}(x) = \mathfrak{s}(y)$ then $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi\{x \leftarrow y\}$, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \xi\{x \leftarrow y\}$ and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma\{x \leftarrow y\}$. Moreover, $\mathfrak{s}(V\{x \leftarrow y\}) \cap dom(\mathfrak{h}) = \mathfrak{s}(V) \cap dom(\mathfrak{h}) = \emptyset$ and $\mathfrak{s}$ must be injective on $V\{x \leftarrow y\}$, hence $(\mathfrak{s}, \mathfrak{h})$ falsifies $\phi\{x \leftarrow y\} \curlywedge \xi\{x \leftarrow y\} \vdash_{\mathfrak{R}}^{V\{x \leftarrow y\}} \lambda\{x \leftarrow y\}$.
- Otherwise, we have $\mathfrak{s} \models x \not\approx y$, thus $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi \curlywedge (\xi \wedge x \not\approx y)$, and $(\mathfrak{s}, \mathfrak{h})$ is a counter-model of $\phi \curlywedge (\xi \wedge x \not\approx y) \vdash_{\mathfrak{R}}^{V} \lambda$.

Conversely, if $(\mathfrak{s}, \mathfrak{h})$ is a counter-model of $\phi \curlywedge (\xi \wedge x \not\approx y) \vdash_{\mathfrak{R}}^{V} \gamma$ then it is clear that it is also a counter-model of $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \gamma$. If $(\mathfrak{s}, \mathfrak{h})$ is a counter-model of $\phi\{x \leftarrow y\} \curlywedge \xi\{x \leftarrow y\} \vdash_{\mathfrak{R}}^{V\{x \leftarrow y\}} \gamma\{x \leftarrow y\}$, then consider the store $\mathfrak{s}'$ such that $\mathfrak{s}'(x) = \mathfrak{s}(y)$ and $\mathfrak{s}'(z) = \mathfrak{s}(z)$ if $z \neq x$. By definition, $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \phi \curlywedge \xi$ and $(\mathfrak{s}', \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma$. The set $\mathfrak{s}'(V) \cap dom(\mathfrak{h}) = \mathfrak{s}(V\{x \leftarrow y\}) \cap dom(\mathfrak{h})$ is empty, and $\mathfrak{s}'$ is injective on V, since $\mathfrak{s}$ is injective on $V\{x \leftarrow y\}$ and by definition $\mathfrak{s}'(u) = \mathfrak{s}(u\{x \leftarrow y\})$, for all variables u. Therefore $(\mathfrak{s}', \mathfrak{h})$ is a counter-model of $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \gamma$.

V Let $(\mathfrak{s}, \mathfrak{h})$ be a counter-model of $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$. Then $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \lambda$ and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma$. Let $\mathfrak{s}'$ be a store such that $\mathfrak{s}'(x) \not\in dom(\mathfrak{h}) \cup \mathfrak{s}(V)$ and $\mathfrak{s}'(y) = \mathfrak{s}(y)$, if $y \neq x$. Since $x \not\in \mathcal{V}(\lambda) \cup \mathcal{V}(\gamma)$ we have $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \lambda$, and $(\mathfrak{s}', \mathfrak{h}) \not\models_{\mathfrak{R}} \gamma$. Moreover $\mathfrak{s}'(x) \not\in dom(\mathfrak{h})$ and $\mathfrak{s}'$ is injective on $V \cup \{x\}$ (since $\mathfrak{s}$ is injective on V and $\mathfrak{s}'(x) \not\in V$), thus $(\mathfrak{s}', \mathfrak{h})$ is a counter-model of $\lambda \vdash_{\mathfrak{R}}^{V \cup \{x\}} \gamma$.

Conversely, it is clear that any counter-model of $\lambda \vdash_{\mathfrak{R}}^{V \cup \{x\}} \gamma$ is also a counter-model of $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$.

I Let $(\mathfrak{s}, \mathfrak{h})$ be a counter-model of $(x \mapsto (y_1, \dots, y_k) * \phi) \curlywedge \xi \vdash_{\mathfrak{R}}^{V} (p(x, \boldsymbol{z}) * \psi) \curlywedge \zeta$. By definition $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} (x \mapsto (y_1, \dots, y_k) * \phi) \curlywedge \xi$, $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} (p(x, \boldsymbol{z}) * \psi) \curlywedge \zeta$, the set $\mathfrak{s}(V) \cap dom(\mathfrak{h})$ is empty and $\mathfrak{s}$ is injective on V. By the application conditions of the rule and Lemma 4.10, we have $\mathfrak{s} \models \zeta'\sigma$. Assume for the sake of contradiction that $(\mathfrak{s}, \mathfrak{h})$ is not a counter-model of $(x \mapsto (y_1, \dots, y_k) * \phi) \curlywedge \xi \vdash_{\mathfrak{R}}^{V} (x \mapsto (y_1, \dots, y_k) * \psi'\sigma * \psi) \curlywedge \zeta$. This entails that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} (x \mapsto (y_1, \dots, y_k) * \psi'\sigma * \psi) \curlywedge \zeta$, hence $\mathfrak{s} \models \zeta$, and there exist disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ and $\mathfrak{h}_3$ such that $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2 \uplus \mathfrak{h}_3$, $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{R}} x \mapsto (y_1, \dots, y_k)$, $(\mathfrak{s}, \mathfrak{h}_2) \models_{\mathfrak{R}} \psi'\sigma$ and $(\mathfrak{s}, \mathfrak{h}_3) \models_{\mathfrak{R}} \psi$. Let $\lambda = x \mapsto (u_1, \dots, u_k) * \psi') \curlywedge \zeta'$, so that $p(x, \boldsymbol{z}) \Leftarrow_{\mathfrak{R}} \lambda$. Since all the rules in $\mathfrak{R}$ are $\mathtt{P}$-rules, necessarily $\mathcal{V}(\lambda) \setminus \mathcal{V}(p(x, \boldsymbol{z})) \subseteq \{u_1, \dots, u_k\}$. Let $\mathfrak{s}'$ be the store defined by: $\mathfrak{s}'(y) = \mathfrak{s}(\sigma(y))$, for all $y \in \mathcal{V}$. By the application condition of the rule, $dom(\sigma)$ is a subset of $\{u_1, \dots, u_k\} \setminus (\{x\} \cup \boldsymbol{z})$, hence $\mathfrak{s}'$ coincides with $\mathfrak{s}$ on all variables in $x, \boldsymbol{z}$. We deduce that $\mathfrak{s}'$ is an associate of $\mathfrak{s}$ w.r.t. $\{u_1, \dots, u_k\} \setminus \mathcal{V}(p(x, \boldsymbol{z}))$. We have $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{R}} x \mapsto (y_1, \dots, y_k)$, with $\sigma(x) = x$ and $\sigma(u_i) = y_i$, thus $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{R}} (x \mapsto (u_1, \dots, u_k))\sigma$. Moreover, we also have

$(\mathfrak{s}, \mathfrak{h}_2) \models_{\mathfrak{R}} \psi'\sigma$, hence $(\mathfrak{s}, \mathfrak{h}_1 \uplus \mathfrak{h}_2) \models_{\mathfrak{R}} \lambda\sigma$. By Proposition 2.6, we get $(\mathfrak{s}', \mathfrak{h}_1 \uplus \mathfrak{h}_2) \models_{\mathfrak{R}} \lambda$. Since $p(x, \boldsymbol{z}) \Leftarrow_{\mathfrak{R}} \lambda$, we deduce that $(\mathfrak{s}, \mathfrak{h}_1 \uplus \mathfrak{h}_2) \models_{\mathfrak{R}} p(x, \boldsymbol{z})$, hence $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} p(x, \boldsymbol{z}) * \psi$. Thus $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} (p(x, \boldsymbol{z}) * \psi) \curlywedge \zeta$, which contradicts our hypothesis.

Conversely, assume that $(x \mapsto (y_1, \dots, y_k) * \phi) \curlywedge \xi \vdash_{\mathfrak{R}}^{V} (p(x, \boldsymbol{z}) * \psi) \curlywedge \zeta$ is valid, and let $(\mathfrak{s}, \mathfrak{h})$ be an $\mathfrak{R}$-model of $(x \mapsto (y_1, \dots, y_k) * \phi) \curlywedge \xi$ such that $\mathfrak{s}(V) \cap dom(\mathfrak{h}) = \emptyset$ and $\mathfrak{s}$ is injective on V. We deduce that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} (p(x, \boldsymbol{z}) * \psi) \curlywedge \zeta$, thus $\mathfrak{s} \models \zeta$, and there exist disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{R}} (p(x, \boldsymbol{z})$ and $(\mathfrak{s}, \mathfrak{h}_2) \models_{\mathfrak{R}} \psi$. This entails that there exists a symbolic heap λ and a associate $\mathfrak{s}'$ of $\mathfrak{s}$ w.r.t. $\mathcal{V}(\lambda) \setminus \mathcal{V}(p(x, \boldsymbol{z}))$ such that $p(x, \boldsymbol{z}) \Leftarrow_{\mathfrak{R}} \lambda$, and $(\mathfrak{s}', \mathfrak{h}_1) \models_{\mathfrak{R}} \lambda$. Since the rules in $\mathfrak{R}$ are P-rules, λ is of the form $(x \mapsto (v_1, \dots, v_m) * \psi'') \curlywedge \zeta''$. Moreover, it is clear that $\mathfrak{h}(\mathfrak{s}(x)) = (\mathfrak{s}(y_1), \dots, \mathfrak{s}(y_k))$, so that $m = k$ and $\mathfrak{s}'(v_i) = \mathfrak{s}(y_i)$, for all $i = 1, \dots, k$. Let σ' be the substitution mapping every variable in $\{v_1, \dots, v_k\}$ not occurring in $x, \boldsymbol{z}$ to the first variable y_j such that $\mathfrak{s}'(v_i) = \mathfrak{s}(y_j)$. By definition, we have $\mathfrak{s}' = \mathfrak{s} \circ \sigma'$, thus by Proposition 2.6, we get $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{R}} \lambda\sigma'$.

Assume for the sake of contradiction that the inductive rule used to derive λ is distinct from the one used to derive the formula $\lambda' = (x \mapsto (u_1, \dots, u_k) * \psi') \curlywedge \zeta'$ in the application condition of the rule. We may assume by renaming that $\mathcal{V}(\lambda') \cap \mathcal{V}(\lambda) \subseteq \mathcal{V}(p(x, \boldsymbol{z}))$. Let $\mathfrak{s}''$ be a store coinciding with $\mathfrak{s}'$ on all constants and on all variables in $\mathcal{V}(\lambda)$, and such that, for all variables $y \in \mathcal{V}(\lambda') \setminus \mathcal{V}(p(x, \boldsymbol{z}))$, $\mathfrak{s}''(y) = \mathfrak{s}(\sigma(y))$. Since $\mathfrak{s}' \models \zeta''$ we have $\mathfrak{s}'' \models \zeta''$. By the application condition of the rule $\sigma(u_i) = y_i$, thus $\mathfrak{s}'' \models u_i \approx v_i$, for all $i = 1, \dots, k$. Since $\mathfrak{s} \models \xi$ and $\xi \models \zeta'\sigma$ (still by the application condition of the rule), we get $\mathfrak{s} \models \zeta'\sigma$, and by Proposition 2.6 we deduce that $\mathfrak{s} \models \zeta'$. Thus $(u_1, \dots, u_k) \approx (v_1, \dots, v_k) \wedge \zeta' \wedge \zeta''$ is satisfiable, which contradicts the fact that $\mathfrak{R}$ is deterministic.

This entails that the rules applied to derive λ and λ' are the same, and by renaming we may assume in this case that $(u_1, \dots, u_k) = (v_1, \dots, v_k)$ (which entails that $(x \mapsto (v_1, \dots, v_k))\sigma = x \mapsto (y_1, \dots, y_k)$), with $\psi' = \psi''$ and $\zeta' = \zeta''$. Using the fact that $\sigma(u_i) = y_i$ and $\mathfrak{s}'(v_i) = \mathfrak{s}(y_i)$, for all $i = 1, \dots, k$, it is easy to check that $\mathfrak{s}'(y) = \mathfrak{s}(\sigma(y))$, for all variables y. Since $(\mathfrak{s}', \mathfrak{h}_2) \models_{\mathfrak{R}} \lambda$ we get by Proposition 2.6, $(\mathfrak{s}, \mathfrak{h}_2) \models_{\mathfrak{R}} \lambda\sigma$, i.e., $(\mathfrak{s}, \mathfrak{h}_2) \models_{\mathfrak{R}} (x \mapsto (y_1, \dots, y_k) * \psi'\sigma)$. Therefore $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} (x \mapsto (y_1, \dots, y_k) * \psi'\sigma * \psi) \curlywedge \zeta$.

□

Rule S is sound but in contrast with the other rules, it is not invertible in general (intuitively, there is no guarantee that the decomposition of the left-hand side of the sequent corresponds to that of the right-hand side).

Example 4.20. Consider the (valid) sequent $x \mapsto (y) * y \mapsto (x) \vdash_{\mathfrak{R}}^{\emptyset} p(x, y) * p(y, x)$ with the rule $p(u, v) \Leftarrow u \mapsto (v)$. Rule S applies, yielding the valid premises $x \mapsto (y) \vdash_{\mathfrak{R}}^{\emptyset} p(x, y)$ and $y \mapsto (x) \vdash_{\mathfrak{R}}^{\emptyset} p(y, x)$. However, since the rules apply modulo commutativity of $*$ we may also get the premises: $x \mapsto (y) \vdash_{\mathfrak{R}}^{\emptyset} p(y, x)$ and $y \mapsto (x) \vdash_{\mathfrak{R}}^{\emptyset} p(x, y)$ which are not valid.

Lemma 4.21. Rule S is sound. More specifically, if $(\mathfrak{s}, \mathfrak{h})$ is a counter-model of the conclusion, then one of the premises admits a counter-model $(\mathfrak{s}, \mathfrak{h}')$, where $\mathfrak{h}'$ is a proper subheap of $\mathfrak{h}$.

Proof:
Let $(\mathfrak{s}, \mathfrak{h})$ be an $\mathfrak{R}$-model of $(\phi_1 * \phi_2) \curlywedge (\xi_1 \wedge \xi_2)$, where $\mathfrak{s}(V) \cap dom(\mathfrak{h}) = \emptyset$ and $\mathfrak{s}$ is injective on V. Assume that the premises admit no counter-model of the form $(\mathfrak{s}, \mathfrak{h}')$ with $\mathfrak{h}' \subset \mathfrak{h}$. By definition, there exist disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$, and $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \phi_i$, for $i = 1, 2$. Since $\phi_i \neq emp$, ϕ_i contains at least one predicate atom, with a root x_i. By Lemma 4.2, necessarily $\mathfrak{s}(x_i) \in dom(\mathfrak{h}_i)$, so $\mathfrak{h}_i$ is not empty and $\mathfrak{h}_i \subset \mathfrak{h}$ for $i = 1, 2$. Still by Lemma 4.2, $\mathfrak{s}(alloc(\phi_i)) \subseteq dom(\mathfrak{h}_i)$, and since $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are disjoint, we deduce that $\mathfrak{s}(alloc(\phi_{3-i})) \cap dom(\mathfrak{h}_i) = \emptyset$ for $i = 1, 2$. Thus $\mathfrak{s}(V \cup alloc(\phi_{3-i})) \cap dom(\mathfrak{h}_i) = \emptyset$. By Corollary 4.3 $\mathfrak{s}$ is injective on $alloc(\phi_1 * \phi_2)$, hence since $\mathfrak{s}$ is injective on V, we deduce that $\mathfrak{s}$ is injective on $V \cup alloc(\phi_{3-i})$. Since $(\mathfrak{s}, \mathfrak{h}_i)$ cannot be a counter-model of the premises because $\mathfrak{h}_i \subset \mathfrak{h}$, this entails that $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \psi_i \curlywedge \zeta_i$ for $i = 1, 2$, thus $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} (\psi_1 * \psi_2) \curlywedge (\zeta_1 \wedge \zeta_2)$. □

4.4. Axioms and Anti-Axioms

We define two sets of syntactic criteria on sequents that allow to quickly conclude that such sequents are respectively valid or non-valid. This will be useful to block the application of the inference rules in these cases. Axioms (i.e., necessarily valid sequents) are defined as follows.

Definition 4.22. An *axiom* is a sequent that is of one of the following forms modulo AC:

1. $\phi \curlywedge (\xi \wedge \xi') \vdash_{\mathfrak{R}}^{V} \phi \curlywedge \xi$;
2. $\phi \curlywedge (\xi \wedge x \not\approx x) \vdash_{\mathfrak{R}}^{V} \gamma$;
3. $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \gamma$ where ϕ is heap-unsatisfiable;
4. $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \gamma$ where either $alloc(\phi) \cap V \neq \emptyset$ or V contains two occurrences of the same variable.

Intuitively, a sequent is valid if the right-hand side is a trivial consequence of the left-hand side, if the left-hand side is (trivially) unsatisfiable, or if V contains a variable that is allocated by the left-hand side or two occurrences of the same variable (since by hypothesis counter-models must be injective on V).

Lemma 4.23. All axioms are valid.

Proof:
We consider each case separately (using the same notations as in Definition 4.22):

1. It is clear that every model of $\phi \curlywedge (\xi \wedge \xi')$ is a model of $\phi \curlywedge \xi$.
2. By definition, $\phi \curlywedge (\xi \wedge x \not\approx x)$ has no model, hence $\phi \curlywedge (\xi \wedge x \not\approx x) \vdash_{\mathfrak{R}}^{V} \gamma$ has no counter-model.
3. If ϕ is heap-unsatisfiable then $alloc(\phi)$ contains two occurrences of the same variable, which by Corollary 4.3, entails that ϕ has no model. Thus $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \gamma$ has no counter-model.

4. Let $(\mathfrak{s}, \mathfrak{h})$ be a counter-model of $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \gamma$. By definition $\mathfrak{s}$ is injective on V hence we cannot have $\{x, x\} \subseteq_m V$. Also, by definition, if $x \in V$ then we cannot have $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h})$, and if $x \in \mathit{alloc}(\phi)$ then if $x \in \mathit{alloc}(\phi) \cap V$ then $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h})$ by Lemma 4.2. We conclude that it is impossible to have $x \in \mathit{alloc}(\phi) \cap V$ either.

□

We also introduce the notion of an anti-axiom, which is a sequent satisfying some syntactic conditions that prevent it from being valid.

Definition 4.24. A sequent $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ is an *anti-axiom* if it is not an axiom, ξ contains no equality, $\zeta = \top$ and one of the following conditions holds:

1. $\mathit{alloc}(\psi) \not\subseteq \mathit{alloc}(\phi)$;
2. $\psi = \mathit{emp}$ and $\phi \neq \mathit{emp}$.
3. There exists a variable $x \in \mathit{alloc}(\phi) \setminus \mathit{alloc}(\psi)$, such that $y \not\rightarrow_{\phi}^{*} x$ holds, for all $y \in \mathit{alloc}(\psi)$;
4. $V \cap (\mathcal{V}(\phi) \setminus \mathcal{V}(\psi))$ is not empty.
5. $\mathcal{V}_{\mathtt{loc}}(\phi) \setminus (\mathcal{V}(\psi) \cup \mathit{alloc}(\phi))$ is not empty.

We provide examples illustrating every case in Definition 4.24:

Example 4.25. The following sequents (where p is some arbitrary predicate) are anti-axioms:

$$\begin{array}{llll@{\qquad}llll}
1: & p(x,y) & \vdash_{\mathfrak{R}}^{\emptyset} & p(y,x) & 2: & p(x,y) & \vdash_{\mathfrak{R}}^{\emptyset} & \mathit{emp} \\
3: & p(x,y) * p(z,y) & \vdash_{\mathfrak{R}}^{\emptyset} & q(x,y) & 4: & p(x,y) & \vdash_{\mathfrak{R}}^{\{y\}} & r(x) \\
5: & p(x,y) & \vdash_{\mathfrak{R}}^{\emptyset} & r(x) & & & &
\end{array}$$

Intuitively, 1 cannot be valid because there exist models of $p(x, y)$ in which y is not allocated whereas y is allocated in all models of $p(y, x)$ Note that by Assumption 2.12, all predicates are productive, hence $p(x, y)$ admits at least one model. Furthermore, a predicate cannot allocate any of its arguments other than the root, for instance rules of the form $p(x, y) \Leftarrow x \mapsto (y) * p(y, x)$, indirectly allocating y, are not allowed. 2 cannot be valid because the models of $p(x, y)$ allocate at least x. For 3, assuming that all variables are associated with distinct locations, one can construct a model of $p(x, y) * p(z, y)$ in which there is no path from x to z and by Proposition 4.6 all locations occurring in the heap of any model of $q(x, y)$ must be reachable from x. For 4 and 5, we can construct a counter-model by considering any structure in which y occurs in the heap but is not allocated, and by Lemma 4.7, all the locations occurring in the heap of any model of $r(x)$ must be allocated.

To show that all anti-axioms admit counter-models, we use the following lemma, which will also play a key rôle in the completeness proof. It states that all the formulas that are heap-satisfiable admit a model satisfying some particular properties:

Lemma 4.26. Let ϕ be a spatial formula, containing a variable x of sort $\mathtt{loc}$. Let $\mathfrak{s}$ be a store that is injective on $alloc(\phi)$. Let U be an infinite subset of $\mathfrak{U}_{\mathtt{loc}}$ such that $U \cap \mathfrak{s}(\mathcal{V}_{\mathtt{loc}}) = \emptyset$. If ϕ is heap-satisfiable, then it admits an $\mathfrak{R}$-model of the form $(\mathfrak{s}, \mathfrak{h})$, where $\mathfrak{s}(x) \in ref(\mathfrak{h})$, the set $dom(\mathfrak{h})$ is a subset of $U \cup \mathfrak{s}(alloc(\phi))$ and $ref(\mathfrak{h}) \subseteq U \cup \mathfrak{s}(\mathcal{V}(\phi))$. Moreover, if $\mathfrak{s}$ is injective then $(\mathfrak{s}, \mathfrak{h})$ is a $\to$-compatible model of ϕ.

Proof:

The proof is by induction on the formulas. Note that we cannot have $\phi = emp$ since $x \in \mathcal{V}(\phi)$.

- If $\phi = y_0 \mapsto (y_1, \dots, y_n)$ then we set: $\mathfrak{h} \overset{\mathit{def}}{=} \{(\mathfrak{s}(y_0), \dots, \mathfrak{s}(y_n))\}$. By definition, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi$. Moreover, $x \in \{y_0, \dots, y_n\}$, and $ref(\mathfrak{h}) = \{\mathfrak{s}(y_i) \mid i = 0, \dots, n \text{ and } y_i \in \mathcal{V}_{\mathtt{loc}}\}$, hence $\mathfrak{s}(x) \in ref(\mathfrak{h})$ and $ref(\mathfrak{h}) \subseteq \mathfrak{s}(\mathcal{V}(\phi))$. Furthermore, $dom(\mathfrak{h}) = \{\mathfrak{s}(y_0)\}$ and $alloc(\phi) = \{y_0\}$ thus $dom(\mathfrak{h}) \subseteq \mathfrak{s}(alloc(\phi))$. Finally, assume that $\mathfrak{s}$ is injective. We have by definition $\to_{\mathfrak{h}} = \{(\mathfrak{s}(y_0), \mathfrak{s}(y_i)) \mid i = 1, \dots, n\}$, thus if $\mathfrak{s}(u) \to_{\mathfrak{h}}^* \mathfrak{s}(v)$ for $u, v \in \mathcal{V}(\lambda)$ then we must have either $\mathfrak{s}(u) = \mathfrak{s}(v)$, so that $u = v$ because $\mathfrak{s}$ is injective, in which case it is clear that $u \to_{\phi}^* v$; or $\mathfrak{s}(u) = \mathfrak{s}(y_0)$ and $\mathfrak{s}(v) = \mathfrak{s}(y_i)$ for some $i = 1, \dots, n$. Since $\mathfrak{s}$ is injective this entails that $u = y_0$, $v = y_i$, thus $u \to_{\phi} v$ by definition of $\to_{\phi}$.

- Assume that $\phi = p(y_0, \dots, y_n)$ is a predicate atom. Then, since by Assumption 2.12 every predicate symbol is productive, there exists a symbolic heap γ such that $\phi \Leftarrow_{\mathfrak{R}} \gamma$. If $x = y_0$ then since the rules in $\mathfrak{R}$ are $\mathtt{P}$-rules, γ contains a points-to-atom with root x. Otherwise, by Assumption 2.15, $x = y_i$ for some $i \in out_{\mathfrak{R}}(p)$, hence there exists a rule application $\phi \Leftarrow_{\mathfrak{R}} \gamma$ such that x occurs in some predicate atom in γ. Thus in both cases we may assume that x occurs in a spatial atom in γ. Note that γ must be heap-satisfiable, since all considered rules are $\mathtt{P}$-rules and by Definition 2.7 the roots of the predicate symbols in γ are pairwise distinct existential variables, thus also distinct from the root y_0 of the points-to atom. Furthermore, γ is of the form $\psi \curlywedge \zeta$, where $x \in \mathcal{V}(\psi)$ and ζ is a conjunction of disequations $u \not\approx v$, with $u \neq v$.

  Let $\mathfrak{s}'$ be an associate of $\mathfrak{s}$ w.r.t. $\mathcal{V}(\gamma) \setminus \mathcal{V}(\phi)$ mapping the variables in $\mathcal{V}(\gamma) \setminus \mathcal{V}(\phi)$ to pairwise distinct locations in U. Since by hypothesis $U \cap \mathfrak{s}(\mathcal{V}) = \emptyset$, $\mathfrak{s}'$ is injective on any set of the form $E \cup (\mathcal{V}(\gamma) \setminus \mathcal{V}(\phi))$ when $\mathfrak{s}$ is injective on E. Let $U' \overset{\mathit{def}}{=} U \setminus \mathfrak{s}'(\mathcal{V}(\gamma) \setminus \mathcal{V}(\phi))$. By the induction hypothesis, there exists a heap $\mathfrak{h}$ such that $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \psi$, with $\mathfrak{s}'(x) \in ref(\mathfrak{h})$, $dom(\mathfrak{h}) \subseteq U' \cup \mathfrak{s}'(alloc(\psi))$ and $ref(\mathfrak{h}) \subseteq U' \cup \mathfrak{s}'(\mathcal{V}(\psi))$. Now if $\mathfrak{s}$ is injective then (as $\mathfrak{s}'$ is also injective in this case), $(\mathfrak{s}', \mathfrak{h})$ is a $\to$-compatible $\mathfrak{R}$-model of ψ. We show that $(\mathfrak{s}, \mathfrak{h})$ fulfills all the properties of the lemma.

  - Since $\mathfrak{s}'$ maps the variables in $\mathcal{V}(\gamma) \setminus \mathcal{V}(\phi)$ to pairwise distinct locations in U and $\mathfrak{s}(\mathcal{V}) \cap U$ is $\emptyset$, necessarily $\mathfrak{s}' \models_{\mathfrak{R}} \zeta$, thus $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \gamma$ which entails that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi$. We also have $\mathfrak{s}(x) = \mathfrak{s}'(x) \in ref(\mathfrak{h})$.

  - Let $\ell \in ref(\mathfrak{h})$. We show that $\ell \in U \cup \mathfrak{s}(\mathcal{V}(\phi))$. By the induction hypothesis, we have $\ell \in U' \cup \mathfrak{s}'(\mathcal{V}(\psi))$. If $\ell \in U' \subseteq U$ then the proof is completed, otherwise we have $\ell = \mathfrak{s}'(y)$ for some $y \in \mathcal{V}(\psi)$. If $y \in \mathcal{V}(\phi)$ then $\mathfrak{s}(y) = \mathfrak{s}'(y)$ thus $\ell \in \mathfrak{s}(\mathcal{V}(\phi))$. Otherwise, we must have $y \in \mathcal{V}(\gamma) \setminus \mathcal{V}(\phi)$, thus $\mathfrak{s}'(y) \in U$ by definition of $\mathfrak{s}'$ and the result holds.

- Let $\ell \in \mathit{dom}(\mathfrak{h})$, we show that $\ell \in U \cup \mathfrak{s}(\mathit{alloc}(\phi))$. By the induction hypothesis we have $\ell \in U' \cup \mathfrak{s}'(\mathit{alloc}(\psi))$. If $\ell \in U' \subseteq U$ then the proof is completed. Otherwise, $\ell = \mathfrak{s}'(y)$ with $y \in \mathit{alloc}(\psi)$. Since the rules in $\mathfrak{R}$ are P-rules, y is either the root y_0 of ϕ, in which case we have $y \in \mathit{alloc}(\phi)$ and $\mathfrak{s}(y) = \mathfrak{s}'(y)$, thus the result holds; or y occurs in $\mathcal{V}(\gamma) \setminus \mathcal{V}(\phi)$, in which that we have $\mathfrak{s}'(y) \in U$, by definition of $\mathfrak{s}'$.
- There remains to show that $(\mathfrak{s}, \mathfrak{h})$ is a $\rightarrow$-compatible $\mathfrak{R}$-model of ϕ, in the case where $\mathfrak{s}$ is injective. Assume that $\mathfrak{s}(u) \rightarrow_{\mathfrak{h}}^* \mathfrak{s}(v)$. If $\mathfrak{s}(u) = \mathfrak{s}(v)$ then $u = v$ by injectivity of $\mathfrak{s}$, hence $u \rightarrow_{\mathfrak{h}}^* v$. Otherwise, we must have $\{\mathfrak{s}(u), \mathfrak{s}(v)\} \subseteq \mathit{ref}(\mathfrak{h}) \subseteq U' \cup \mathfrak{s}'(\mathcal{V}(\psi))$, and since $U' \subseteq U$ and $U \cap \mathfrak{s}(\mathcal{V}) = \emptyset$, we have $\{\mathfrak{s}(u), \mathfrak{s}(v)\} \subseteq \mathfrak{s}'(\mathcal{V}(\psi))$. We also have $\mathfrak{s}'(\mathcal{V}(\gamma) \setminus \mathcal{V}(\phi)) \subseteq U$, which entails that $\{\mathfrak{s}(u), \mathfrak{s}(v)\} \subseteq \mathfrak{s}'(\mathcal{V}(\phi))$. Since $\mathfrak{s}'$ is an associate of $\mathfrak{s}$ w.r.t. $\mathcal{V}(\gamma) \setminus \mathcal{V}(\phi)$, $\mathfrak{s}$ and $\mathfrak{s}'$ coincide on all variables in $\mathcal{V}(\phi)$ and we deduce that $\{\mathfrak{s}(u), \mathfrak{s}(v)\} \subseteq \mathfrak{s}(\mathcal{V}(\phi))$. Because $\mathfrak{s}$ is injective, this entails that $u, v \in \mathcal{V}(\phi)$, so that $\mathfrak{s}(u) = \mathfrak{s}'(u)$ and $\mathfrak{s}(v) = \mathfrak{s}'(v)$. By hypothesis $(\mathfrak{s}', \mathfrak{h})$ is a $\rightarrow$-compatible $\mathfrak{R}$-model of ψ, and we deduce that $u \rightarrow_{\psi}^* v$. Since $u \neq v$, necessarily $u \in \mathit{alloc}(\psi)$ (by definition of $\rightarrow_{\psi}$), and since the rules in $\mathfrak{R}$ are P-rules, and $u \in \mathcal{V}(\phi)$, this entails that $u = \mathit{roots}(\phi)$. Since $v \in \mathcal{V}(\phi)$, by Assumption 2.15 we have $u \rightarrow_{\phi}^* v$.

- Assume that $\phi = \phi_1 * \phi_2$, with $\phi_i \neq \mathit{emp}$. Let U_1, U_2 be disjoint infinite subsets of U. Let $\{x_1, x_2\}$ be some arbitrary chosen variables such that $x_i \in \mathcal{V}(\phi_i)$ and $x \in \{x_1, x_2\}$ (it is easy to check that such a pair of variables always exists). By the induction hypothesis, there exist heaps $\mathfrak{h}_i$ such that $(\mathfrak{s}, \mathfrak{h}_i) \models \phi_i$ where $\mathfrak{s}(x_i) \in \mathit{ref}(\mathfrak{h}_i)$, $\mathit{dom}(\mathfrak{h}_i) \subseteq U_i \cup \mathfrak{s}(\mathit{alloc}(\phi_i))$ and $\mathit{ref}(\mathfrak{h}_i) \subseteq U_i \cup \mathfrak{s}(\mathcal{V}(\phi_i))$. Moreover, if $\mathfrak{s}$ is injective, then $(\mathfrak{s}, \mathfrak{h}_i)$ is an $\rightarrow$-compatible $\mathfrak{R}$-model of ϕ_i. We first show that $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are disjoint. Assume for the sake of contradiction that $\ell \in \mathit{dom}(\mathfrak{h}_1) \cap \mathit{dom}(\mathfrak{h}_2)$. Since $U_1 \cap U_2 = \emptyset$, necessarily $\ell = \mathfrak{s}(y_i)$ (for $i = 1, 2$), with $y_i \in \mathit{alloc}(\phi_i)$. Since $\mathfrak{s}$ is injective on $\mathit{alloc}(\phi)$, we deduce that $y_1 = y_2$. We have $\{y_1, y_2\} \subseteq_m \mathit{alloc}(\phi)$, hence ϕ is heap-unsatisfiable, which contradicts the hypotheses of the lemma. Thus $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are disjoint.

  Let $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$. We have $\mathfrak{s}(x) \in \{\mathfrak{s}(x_1), \mathfrak{s}(x_2)\} \subseteq \mathit{ref}(\mathfrak{h}_1) \cup \mathit{ref}(\mathfrak{h}_2) = \mathit{ref}(\mathfrak{h})$. Moreover, $\mathit{dom}(\mathfrak{h}) = \mathit{dom}(\mathfrak{h}_1) \cup \mathit{dom}(\mathfrak{h}_2) \subseteq U_1 \cup U_2 \cup \mathfrak{s}(\mathit{alloc}(\phi_1)) \cup \mathfrak{s}(\mathit{alloc}(\phi_2)) \subseteq U \cup \mathfrak{s}(\mathit{alloc}(\phi))$, and $\mathit{ref}(\mathfrak{h}) = \mathit{ref}(\mathfrak{h}_1) \cup \mathit{ref}(\mathfrak{h}_2) \subseteq U_1 \cup U_2 \cup \mathfrak{s}(\mathcal{V}(\phi_i)) \cup \mathfrak{s}(\mathcal{V}(\phi_2)) \subseteq U \cup \mathfrak{s}(\mathcal{V}(\phi))$. Furthermore, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi_1 * \phi_2 = \phi$.

  There only remains to prove that $(\mathfrak{s}, \mathfrak{h})$ is a $\rightarrow$-compatible $\mathfrak{R}$-model of ϕ when $\mathfrak{s}$ is injective. Assume that this is not the case, and let u, v be variables such that $\mathfrak{s}(u) \rightarrow_{\mathfrak{h}}^* \mathfrak{s}(v)$ and $u \not\rightarrow_{\phi}^* v$. This entails that $u \neq v$. By definition, there exist $\ell_0, \ldots, \ell_m$ such that $\ell_0 = \mathfrak{s}(u)$, $\ell_m = \mathfrak{s}(v)$, and $\forall i = 1, \ldots, m$, $\ell_{i-1} \rightarrow_{\mathfrak{h}} \ell_i$. We assume, w.l.o.g., that m is miminal, i.e., that there is no sequence $\ell'_0, \ldots, \ell'_k$ and no variables x_0, x_k such that $k < m$, $\ell'_0 = \mathfrak{s}(x_0)$, $\ell'_k = \mathfrak{s}(x_k)$ and $x_0 \not\rightarrow_{\phi}^* x_k$. We may also assume, by symmetry, that $\ell_0 \in \mathit{dom}(\mathfrak{h}_1)$. If all the locations $\ell_1, \ldots, \ell_{m-1}$ occur in $\mathit{dom}(\mathfrak{h}_1)$ then $\mathfrak{s}(u) \rightarrow_{\mathfrak{h}_1}^* \mathfrak{s}(v)$, thus $u \rightarrow_{\phi_i}^* v$ because $(\mathfrak{s}, \mathfrak{h}_i)$ is an $\rightarrow$-compatible $\mathfrak{R}$-model of ϕ_i, which entails that $u \rightarrow_{\phi}^* v$ since by definition $\rightarrow_{\phi_i} \subseteq \rightarrow_{\phi}$, contradicting our assumption. Otherwise, let j be the smallest index in $\ell_1, \ldots, \ell_{m-1}$ such that $\ell_j \notin \mathit{dom}(\mathfrak{h}_1)$. Since $\ell_j \in \mathit{dom}(\mathfrak{h})$ (as $\ell_j \rightarrow_{\lambda} \ell_{j+1}$) we deduce that $\ell_j \in \mathit{dom}(\mathfrak{h}_2) \subseteq \mathit{ref}(\mathfrak{h}_2)$, and $\ell_j \in \mathit{ref}(\mathfrak{h}_1)$. Since $U_1 \cap U_2 = \emptyset$, we get $\ell_j = \mathfrak{s}(\mathcal{V}(\phi_i))$, for some $i = 1, 2$. Thus

$\ell_j = \mathfrak{s}(u')$ with $u' \in \mathcal{V}(\phi)$. Since $u \not\rightarrow^*_\phi v$, we have by transitivity either $u \not\rightarrow^*_\phi u'$ or $u' \not\rightarrow^*_\phi v$, which contradicts the minimality of m (one of the sequences $\ell_1, \dots, \ell_j$ or $\ell_j, \dots, \ell_m$ satisfies the conditions above and has a length strictly less than $m + 1$).

□

Lemma 4.27. All anti-axioms admit counter-models.

Proof:
Consider an anti-axiom $\phi \curlywedge \xi \vdash^V_{\mathfrak{R}} \psi \curlywedge \zeta$, with the same notations as in Definition 4.24. Let $\mathfrak{s}$ be an injective store such that $\mathfrak{U}_{\mathtt{loc}} \setminus \mathfrak{s}(\mathcal{V})$ is infinite, such a store always exists since $\mathfrak{U}_{\mathtt{loc}}$ is infinite. First assume that $\phi \neq \mathit{emp}$. In this case ϕ necessarily contains at least one spatial atom α, hence at least one variable $x = \mathit{root}(\alpha)$ of sort $\mathtt{loc}$. Since $\phi \curlywedge \xi \vdash^V_{\mathfrak{R}} \psi \curlywedge \zeta$ is not an axiom, ϕ is heap-satisfiable, thus by Lemma 4.26 (applied with $U \overset{\mathit{def}}{=} \mathfrak{U}_{\mathtt{loc}} \setminus \mathfrak{s}(\mathcal{V})$), $\phi \curlywedge \xi$ admits a $\rightarrow$-compatible model $(\mathfrak{s}, \mathfrak{h})$, such that $\mathit{dom}(\mathfrak{h}) \subseteq U \cup \mathfrak{s}(\mathit{alloc}(\phi))$. Since $U \cap \mathfrak{s}(\mathcal{V}) = \emptyset$, we have, for all $u \in \mathcal{V}$, $\mathfrak{s}(u) \in \mathit{dom}(\mathfrak{h}) \iff \mathfrak{s}(u) \in \mathfrak{s}(\mathit{alloc}(\phi))$. Since $\mathfrak{s}$ is injective, we deduce that

$$\mathfrak{s}(u) \in \mathit{dom}(\mathfrak{h}) \iff u \in \mathit{alloc}(\phi) \quad (\dagger).$$

Note that if $\phi = \mathit{emp}$, then the structure $(\mathfrak{s}, \mathfrak{h})$ with $\mathfrak{h} = \emptyset$ also satisfies $(\dagger)$, since $\rightarrow_{\mathfrak{h}}$, $\rightarrow_\phi$, $\mathit{dom}(\mathfrak{h})$ and $\mathit{alloc}(\phi)$ are all empty in this case.

We show that $(\mathfrak{s}, \mathfrak{h})$ is a counter-model of $\phi \curlywedge \xi \vdash^V_{\mathfrak{R}} \psi \curlywedge \zeta$. Since $\mathfrak{s}$ is injective, in particular $\mathfrak{s}$ must be injective on V, because otherwise V would contain two occurrences of the same variable, hence $\phi \curlywedge \xi \vdash^V_{\mathfrak{R}} \psi \curlywedge \zeta$ would be an axiom, contradicting Definition 4.24. If V contains a variable y such that $\mathfrak{s}(y) \in \mathit{dom}(\mathfrak{h})$ then $(\dagger)$ entails that $y \in \mathit{alloc}(\phi)$, which is impossible since $\phi \curlywedge \xi \vdash^V_{\mathfrak{R}} \psi \curlywedge \zeta$ would then be an axiom. We prove that $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \psi \curlywedge \zeta$ by considering each case from Definition 4.24 separately.

1. If $\mathit{alloc}(\psi) \setminus \mathit{alloc}(\phi)$ contains a variable y, then by $(\dagger)$ we have $\mathfrak{s}(y) \notin \mathit{dom}(\mathfrak{h})$, which by Lemma 4.2 entails that $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \psi$.

2. If $\psi = \mathit{emp}$ and $\phi \neq \mathit{emp}$, then ϕ contains at least one atom, hence $\mathit{alloc}(\phi) \neq \emptyset$. By Lemma 4.2, $\mathfrak{s}(\mathit{alloc}(\phi)) \subseteq \mathit{dom}(\mathfrak{h})$, thus $\mathfrak{h} \neq \emptyset$ and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \psi \curlywedge \zeta$.

3. Assume that there exists a variable $x' \in \mathit{alloc}(\phi) \setminus \mathit{alloc}(\psi)$ such that $y \not\rightarrow_\phi x'$, for all $y \in \mathit{alloc}(\psi)$, and that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \psi$. By Lemma 4.2 we have $\mathfrak{s}(x') \in \mathit{dom}(\mathfrak{h}) \subseteq \mathit{ref}(\mathfrak{h})$, thus by Proposition 4.6 there exists $y \in \mathit{alloc}(\psi)$ such that $\mathfrak{s}(y) \rightarrow^*_{\mathfrak{h}} \mathfrak{s}(x)$. Since $(\mathfrak{s}, \mathfrak{h})$ is a $\rightarrow$-compatible model of ϕ necessarily $y \rightarrow^*_\phi x'$, which contradicts the above assumption.

For the remaining cases, we will apply Lemma 4.26 to obtain a heap, using a variable x that is in $V \cap (\mathcal{V}(\phi) \setminus \mathcal{V}(\psi))$ (for Condition 4) or in $\mathcal{V}_{\mathtt{loc}}(\phi) \setminus (\mathcal{V}(\psi) \cup \mathit{alloc}(\phi))$ (for Condition 5). Note that by Lemma 4.26 this entails in particular that $\mathfrak{s}(x) \in \mathit{ref}(\mathfrak{h})$.

4. Assume that $x \in V \cap \mathcal{V}(\phi)$ and $x \notin \mathcal{V}(\psi)$. Then $\mathfrak{s}(x) \in \mathit{ref}(\mathfrak{h})$ and since $x \in V$ we get $\mathfrak{s}(x) \notin \mathit{dom}(\mathfrak{h})$. By Lemma 4.7, if $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \psi$ then we have $\mathfrak{s}(x) \in \mathfrak{s}(\mathcal{V}(\psi))$, and $x \in \mathcal{V}(\psi)$ since $\mathfrak{s}$ is injective. Thus we get a contradiction.

5. Assume that $x \in \mathcal{V}_{\mathtt{loc}}(\phi)$, $x \notin \mathcal{V}(\psi)$ and $x \notin \mathit{alloc}(\phi)$. By (†), we deduce that $\mathfrak{s}(x) \notin \mathit{dom}(\mathfrak{h})$. Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \psi$. Since $x \notin \mathcal{V}(\psi)$ and $\mathfrak{s}$ is injective, we get $\mathfrak{s}(x) \notin \mathfrak{s}(\mathcal{V}(\psi))$. By Lemma 4.7, we deduce that $\mathfrak{s}(x) \notin \mathit{ref}(\mathfrak{h})$, a contradiction.

□

4.5. Proof Trees

A *proof tree* is a (possibly infinite) tree with nodes labeled by sequents, such that if a node labeled by $\mathcal{S}$ has successors labeled by $\mathcal{S}_1, \dots, \mathcal{S}_n$ then there exists a rule instance of the form $\frac{\mathcal{S}_1 \dots \mathcal{S}_n}{\mathcal{S}}$. We will usually identify the nodes in a proof tree with the sequents labeling them. A *path* from $\mathcal{S}$ to $\mathcal{S}'$ in a proof tree is a finite sequence $\mathcal{S}_0, \dots, \mathcal{S}_n$ such that $\mathcal{S} = \mathcal{S}_0$, $\mathcal{S}_n = \mathcal{S}'$ and for all $i = 1, \dots, n$, $\mathcal{S}_i$ is a successor of $\mathcal{S}_{i-1}$. A proof tree is *fully expanded* if all its leaves are axioms. It is *rational* if it contains a finite number of subtrees, up to a renaming of variables. Note that rational trees may be infinite, but they can be represented finitely. A sequent $\mathcal{S}'$ is a *descendant* of a sequent $\mathcal{S}$ if there exists a proof tree with a path from $\mathcal{S}$ to $\mathcal{S}'$.

Example 4.28. Let $\mathfrak{R}$ be the following set of rules, where $\mathtt{a}, \mathtt{b}$ denote constant symbols and u is a variable of the same sort as $\mathtt{b}$:

$$\begin{array}{ccc}
p(x,y) & \Leftarrow & x \mapsto (\mathtt{a}, y, z) * p(z,y) \\
p(x,y) & \Leftarrow & x \mapsto (\mathtt{b}) \\
r(x) & \Leftarrow & x \mapsto (\mathtt{a}, y, z) * r(z) \\
r(x) & \Leftarrow & x \mapsto (u)
\end{array}$$

The sequent $p(x,y) \vdash_{\mathfrak{R}}^{\emptyset} r(x)$ admits the following rational proof tree (the sequent $p(z,y) \vdash_{\mathfrak{R}}^{\emptyset} r(z)$ is identical to the conclusion up to a renaming):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x \mapsto (\mathtt{a}, y, z) \vdash_{\mathfrak{R}}^{\{z\}} x \mapsto (\mathtt{a}, y, z)}\,\text{axiom} \quad \dfrac{p(z,y) \vdash_{\mathfrak{R}}^{\emptyset} r(z)}{p(z,y) \vdash_{\mathfrak{R}}^{\{x\}} r(z)}\,\mathtt{V}}{x \mapsto (\mathtt{a}, y, z) * p(z,y) \vdash_{\mathfrak{R}}^{\emptyset} x \mapsto (\mathtt{a}, y, z) * r(z)}\,\mathtt{S}}{x \mapsto (\mathtt{a}, y, z) * p(z,y) \vdash_{\mathfrak{R}}^{\emptyset} r(x)}\,\mathtt{I} \quad \dfrac{\dfrac{}{x \mapsto (\mathtt{b}) \vdash_{\mathfrak{R}}^{\emptyset} x \mapsto (\mathtt{b})}\,\text{axiom}}{x \mapsto (\mathtt{b}) \vdash_{\mathfrak{R}}^{\emptyset} r(x)}\,\mathtt{I}}{p(x,y) \vdash_{\mathfrak{R}}^{\emptyset} r(x)}\,\mathtt{U}$$

Remark 4.29. Note that, since infinite proof trees are allowed, the fact that each rule is sound does not imply that the procedure itself is sound, i.e., that the root of every fully expanded proof tree is valid. The latter property holds only for the strategy introduced in the next section, see Theorem 5.13.

4.6. The Strategy

In this section, we introduce a strategy to restrict the application of the inference rules, which will ensure both the soundness and efficiency of the proof procedure. To this purpose, we define a set of variables $\mathcal{V}^{\dagger}(\mathcal{S})$ which denotes the variables occurring at non-root positions on the right-hand side of the considered sequent, but not in the set of non-allocated variables V. As we shall see, the strategy will handle the sequents in different ways depending on the number of such variables. Recall that $\mathit{width}(\mathfrak{R})$ denotes the maximal arity of the predicate symbols and tuples occurring in $\mathfrak{R}$.

Definition 4.30. For every sequent $\mathcal{S} = \lambda \vdash_{\mathfrak{R}}^{V} \gamma$, we denote by $\mathcal{V}^{\dagger}(\mathcal{S})$ the set $\mathcal{V}_{\mathtt{loc}}(\gamma) \setminus (V \cup \mathit{alloc}(\gamma))$. A sequent $\mathcal{S}$ is *narrow* if it is equality-free and $\mathit{card}(\mathcal{V}^{\dagger}(\mathcal{S})) \leq \mathit{width}(\mathfrak{R})$.

Note that in particular, if γ is a spatial atom, then $\mathcal{S}$ is necessarily narrow since in this case we must have $\mathit{card}(\mathcal{V}(\gamma)) \leq \mathit{ar}_{\mathit{max}}(\mathfrak{R}) \leq \mathit{width}(\mathfrak{R})$. The inference rules are applied with the following strategy:

Definition 4.31. We assume a *selection function* is given, mapping every nonempty finite set of expressions (i.e., formulas or rule applications) S to an element of S; this element is said to be *selected in* S. We also assume that this function can be computed in polynomial time. A proof tree is *admissible* if all the rule applications occurring in it fulfill the following conditions (see Figure 3 for the notations):

1. No rule is applied on an axiom or an an anti-axiom, and no rule is applied if one of the premises is an anti-axiom.

2. $\mathtt{I}$ is applied only if $\zeta = \top$ and $\psi = \mathit{emp}$.

3. $\mathtt{U}$ is applied only if γ is of one of the forms $q(x_1, \boldsymbol{y}) \curlywedge \top$ or $x_1 \mapsto \boldsymbol{y} \curlywedge \top$, where x_1 is the first component of the vector $\boldsymbol{t}$ (as defined in the rule).

4. $\mathtt{C}$ is applied only if γ is of the form $\psi \curlywedge \top$, $x \in \mathit{alloc}(\phi)$, $y \in \mathcal{V}(\gamma) \setminus (\mathit{alloc}(\phi) \cup V)$, ψ is a predicate atom and $x \not\approx y$ does not already occur in ξ.

5. $\mathtt{S}$ is applied only if $\xi_1 = \xi_2$, $\zeta_1 = \zeta_2 = \top$, and ψ_1 is selected in the set of atoms in $\psi_1 * \psi_2$. Furthermore, if the conclusion is not narrow, then the rule is applied only if the left-premise is valid (this will be tested by applying the decision procedure recursively).

6. The rules are applied with the following priority (meaning that no rule can be applied if a rule with a higher priority is also applicable):

$$\mathtt{W} > \mathtt{V} > \mathtt{R} > \mathtt{E} > \mathtt{S} > \mathtt{U} > \mathtt{C} > \mathtt{I}.$$

7. For all the rules other than $\mathtt{S}$, and for all applications of $\mathtt{S}$ on a sequent that is not narrow, if several applications of the rule are possible and all fulfill the conditions above, then only the one that is selected in the set of possible rule applications can be applied.

From now on, we assume that all the considered proof trees are admissible (the soundness[3] and completeness proofs below, as well as the complexity analysis hold with this requirement). No assumption is made on the selection function, for instance one may assume that ψ_1 is the leftmost atom

[3] The soundness of each individual rule, as stated by Lemmata 4.19 and 4.21, does not depend on Definition 4.31, but the global soundness of the procedure, as stated in Theorem 5.13 holds only for admissible proof trees.

in $\psi_1 * \psi_2$ (i.e., the atoms are ordered according to the order in which they occur in the formula) and that the first detected rule application is applied, except for S if the conclusion is narrow.

Many of the conditions in Definition 4.31 are quite natural, and simply aim at pruning the search space by avoiding irrelevant rule applications. For instance, applying a rule on an axiom or on an anti-axiom is clearly useless. The priority order is chosen in such a way that the rules with the minimal branching factor are applied first, postponing the most computationally costly rules. Similarly, the rules operating on spatial atoms (I or U) are postponed until all pure formulas have been handled. The unfolding of the atoms on the left-hand side of a sequent (Rule U) is postponed until the right-hand side has been fully decomposed (using rule S), yielding a unique spatial atom. This permits to guide the choice of the atom to be unfolded on the left-hand side: we only unfold the atom with the same root as the one on the right-hand side. Rule C may cause a costly explosion if applied blindly, thus the application conditions are carefully designed. They are meant to ensure that the rule application is really useful, in the sense that it permits further applications of rule I. Condition 7 in Definition 4.31 is meant to ensure that the considered rules are applied with a "don't care" strategy, meaning that if several rule applications are possible then one of them (the selected one) is chosen arbitrarily and the others are ignored. This is justified by the fact that these rules are invertible, hence exploring all possibilities is useless. Such a strategy is crucial for proving that the procedure runs in polynomial time. In contrast, if the conclusion of S is narrow, then the rule must applied with a "don't know" indeterminism, i.e., all possible applications must be considered. This is necessary for completeness, due to the fact that the rule is *not* invertible in this case. Of course, all these possible rule applications must be taken into account for the complexity analysis. In other words, one must cope with two different kinds of branching: an "and-branching" due to the fact that a given rule may have several premises, and an "or-branching" due to the fact that there may be several ways of applying a given rule on a given sequent. By Condition 7, the latter branching occurs only when the rule S is applied on a narrow sequent; in all other cases, only the selected rule application can be considered hence there is no or-branching.

The requirement that ψ_1 must be selected in Condition 5 ensures that only one decomposition $\psi_1 * \psi_2$ can be considered for the right-hand side of a given sequent. This avoids for instance the exponential blow-up that would occur if the same decomposition was performed in different orders. Note that, to check whether the left premise is valid in Condition 5, it is necessary to recursively invoke the proof procedure. As we shall see below, this is feasible because this premise and all its descendants are necessarily narrow:

Proposition 4.32. Let $\mathcal{S}_1$ be an equality-free sequent. For every successor $\mathcal{S}_2$ of $\mathcal{S}_1$, the relation $\mathcal{V}^\dagger(\mathcal{S}_2) \subseteq \mathcal{V}^\dagger(\mathcal{S}_1)$ holds. Consequently, if a sequent is narrow then all its descendants are narrow.

Proof:
Let $\mathcal{S}_i = \lambda_i \vdash_{\mathfrak{R}}^{V_i} \gamma_i$ for $i = 1, 2$. Assume for the sake of contradiction that $\mathcal{V}^\dagger(\mathcal{S}_2) \setminus \mathcal{V}^\dagger(\mathcal{S}_1)$ contains a variable u. By definition, $u \in \mathcal{V}(\gamma_2)$ and $u \not\in \mathit{alloc}(\gamma_2) \cup V_2$. First assume that $u \not\in \mathcal{V}(\gamma_1)$, i.e., that u was introduced by the application of an inference rule to $\mathcal{S}_1$. By Proposition 4.17 (4), the only rule that can introduce new variables to the right-hand side of a sequent is I. Indeed, since we assume that all inferences are admissible, by Condition 4 of Definition 4.31, C must replace a variable x by a variable y occurring in $\mathcal{V}(\gamma_1)$. Rule I replaces a predicate atom $p(x, \boldsymbol{z})$ that occurs in γ_1 by a formula

of the form $\gamma_1'\sigma$, where $p(x,\boldsymbol{z}) \Leftarrow_{\mathfrak{R}} \gamma_1' \curlywedge \zeta'$. Since the rules in $\mathfrak{R}$ are $\mathtt{P}$-rules, by Condition 2 of Definition 2.7, all the variables occurring in $\gamma_1'\sigma$ but not in $p(x,\boldsymbol{z})$ must occur as roots in $\gamma_1'\sigma$. This entails that $\mathcal{V}(\gamma_2) \setminus \mathcal{V}(\gamma_1) \subseteq \mathit{alloc}(\gamma_2)$, which contradicts the fact that $u \notin \mathit{alloc}(\gamma_2) \cup V$. We deduce that $u \in \mathcal{V}(\gamma_1)$.

Assume that $u \in V_1$. Since $u \notin V_2$, the inference rule applied to $\mathcal{S}_1$ must have deleted a variable from V_1. The only rules that can delete variables from V_1 are $\mathtt{V}$ and $\mathtt{C}$ by Proposition 4.17 (2). Rule $\mathtt{V}$ applies only on a variable $x \notin \mathcal{V}(\gamma_1)$, thus we cannot have $x = u$. If Rule $\mathtt{C}$ is applied and replaces a variable x by y, then x cannot occur in $\mathcal{S}_2$, hence $u \neq x$. Thus we must have $u \notin V_1$.

Finally assume that $u \in \mathit{alloc}(\gamma_1)$. Since $u \notin \mathit{alloc}(\gamma_2)$, the inference rule applied to $\mathcal{S}_1$ must have deleted a variable from $\mathit{alloc}(\gamma_1)$. The only rules that can delete variables from $\mathit{alloc}(\gamma_1)$ are $\mathtt{S}$ and $\mathtt{C}$ by Proposition 4.17 (3). Again, if rule $\mathtt{C}$ replaces a variable x by y, then x cannot occur in $\mathcal{S}_2$, hence $u \neq x$. Now, consider rule $\mathtt{S}$ and let $\mathcal{S}_2' \overset{\mathit{def}}{=} \lambda_2' \vdash_{\mathfrak{R}}^{V_2'} \gamma_2'$ be the other premise of the rule. Since $u \in \mathit{alloc}(\gamma_1)$ and $\mathcal{S}_1$ is not an anti-axiom, we must have $u \in \mathit{alloc}(\lambda_1)$, which entails by definition of the rule that $u \in \mathit{alloc}(\lambda_2) \cup V_2$ and $u \in \mathit{alloc}(\lambda_2') \cup V_2'$. Still by definition of rule $\mathtt{S}$, if a variable is the root of a spatial atom occurring on the right-hand side of the conclusion γ_1, then this spatial atom must occur on the right-hand side of one of the premises γ_2 or γ_2'. This entails that $u \in \mathit{alloc}(\gamma_2) \cup \mathit{alloc}(\gamma_2')$. Furthermore, we must have $\mathit{alloc}(\lambda_2) \cap \mathit{alloc}(\lambda_2') = \emptyset$ because otherwise $\mathcal{S}_1$ would contain two atoms with the same root, hence would be an axiom. Since $u \notin V_2$ we deduce that $u \in \mathit{alloc}(\lambda_2)$, and that $u \in V_2'$. If $u \notin \mathit{alloc}(\gamma_2)$ then necessarily $u \in \mathit{alloc}(\gamma_2')$. Since $\mathcal{S}_2'$ is not an anti-axiom we deduce that $u \in \mathit{alloc}(\lambda_2')$, which entails that $\mathcal{S}_2'$ is an axiom since $\mathit{alloc}(\lambda_2') \cap V_2' \neq \emptyset$, a contradiction.

The second part of the proposition follows by an immediate induction, using the fact that no rule can introduce any equality in its premises. □

We call *auxiliary successors* the sequents whose validity must be tested to check whether rule $\mathtt{S}$ is applicable or not, according to Definition 4.31:

Definition 4.33. A sequent $\mathcal{S}$ is an *auxiliary successor* of a sequent $\mathcal{S}'$ if:

- $\mathcal{S}$ is not an anti-axiom and is not narrow,
- $\mathcal{S}'$ is not an axiom,
- $\mathcal{S}'$ and $\mathcal{S}$ are of the form $\phi' \curlywedge \xi \vdash_{\mathfrak{R}}^{V'} \psi' \curlywedge \top$ and $(\phi' * \phi) \curlywedge \xi \vdash_{\mathfrak{R}}^{V} (\psi' * \psi) \curlywedge \top$ respectively, where ψ' is a spatial atom, and $V' = V \cup \mathit{alloc}(\phi)$.

If the sequent $\mathcal{S}$ in Definition 4.33 is valid, then it is the left premise of an application of $\mathtt{S}$ on $\mathcal{S}'$. The following proposition gives an upper-bound on the number of auxiliary successors.

Proposition 4.34. Assume that $\mathit{ar}_{\mathit{max}}(\mathfrak{R}) \leq \kappa$ (i.e., that the maximal arity of the predicates is bounded by κ). Then every sequent $\mathcal{S}$ has at most 2^{κ} auxiliary successors, and each of these auxiliary successors can be computed in polynomial time.

Proof:
If $\mathcal{S}$ admits an auxiliary successor, then $\mathcal{S}$ is necessarily of the form $\phi \wedge \xi \vdash_{\mathfrak{R}}^{V} \psi_1 * \psi_2$ where ψ_1 is the selected predicate atom in $\psi_1 * \psi_2$ according to Definition 4.31. To get an auxiliary successor of $\mathcal{S}$,

one has to decompose ϕ into $\phi_1 * \phi_2$, in such a way that the obtained premises $\mathcal{S}_i = \phi_i \curlywedge \xi \vdash_{\mathfrak{R}}^{V_i} \psi_i$ (with $V_i = V \cup \mathit{alloc}(\phi_{3-i})$) are not anti-axioms. For each atom α in ϕ such that $\mathit{root}(\alpha) \in \mathcal{V}(\psi_1)$, we first choose whether α occurs in ϕ_1 or ϕ_2. There is at most one such atom for each variable, because otherwise ϕ would be heap-unsatisfiable and $\mathcal{S}$ would be an axiom, and ψ_1 contains at most κ variables, thus there are at most 2^κ possible choices.

We show that, once this choice is performed, the decomposition $\phi = \phi_1 * \phi_2$ is also fixed. We denote by E the set of variables $\mathcal{V}(\psi_1) \setminus \mathit{alloc}(\phi_1)$. Let α be a predicate atom with root x in ϕ, and let y_0 be the root of ψ_1. If $x = y_0$ then necessarily α occurs in ϕ_1, since otherwise $x \in V_1$ and $\mathcal{S}_1$ would be an anti-axiom. If for all paths $y_0 \to_\phi y_1 \to_\phi \dots \to_\phi y_n \to_\phi x$, there exists $i = 1, \dots, n$ such that $y_i \in E$, then necessarily $y \not\to_{\phi_1}^* x$ and thus α cannot occur in ϕ_1, as otherwise $\mathcal{S}_1$ would be an anti-axiom. We finally show that for every atom α' in ϕ with root x', if there exists a path $y_0 \to_\phi y_1 \to_\phi \dots \to_\phi y_n \to_\phi x'$ such that $\{y_1, \dots, y_n, x'\} \cap E = \emptyset$, then α' occurs in ϕ_1 (thus, in particular, α occurs in ϕ_1 if the previous conditions are not satisfied, since $x \notin E$, as $x \notin \mathcal{V}(\psi_1)$). We assume, w.l.o.g., that the considered path is the minimal one not satisfying the property, so that the atoms with roots $y_0, \dots, y_n$ all occur in ϕ_1. This entails that $y_0 \to_{\phi_1}^* x'$. If $x' \notin \mathcal{V}(\psi_1)$ then α must occur in ϕ_1, otherwise $\mathcal{S}_1$ would be an anti-axiom, because we would have $x' \in V_1$, as $x' \in \mathit{alloc}(\phi_2)$. Otherwise, since $x' \notin E$, we have $x' \in \mathit{alloc}(\phi_2)$, by definition of E.

To sum up, assuming that $\mathcal{S} = \phi \wedge \xi \vdash_{\mathfrak{R}}^{V} \psi$ is neither an axiom nor an anti-axiom, the auxiliary successors of $\mathcal{S}$ are computed as follows. We first compute the selected atom ψ_1 in ψ (which can be done in polynomial time by the assumption in Definition 4.31). The atoms α in ϕ are added either to ϕ_1 or to ϕ_2 using the following algorithm. Initially, ϕ_1 and ϕ_2 are both empty. If $\mathit{root}(\alpha) = \mathit{root}(\psi_1)$ then α is moved from ϕ to ϕ_1 (there is exactly one atom with this property). For each atom α in ϕ with $\mathit{root}(\alpha) \in \mathcal{V}(\psi_1)$ and $\mathit{root}(\alpha) \neq \mathit{root}(\psi_1)$, we nondeterministically add α to either ϕ_1 or ϕ_2 (which yields at most 2^κ possible choices) and remove α from ϕ. Then, for each atom α in ϕ_1, all the remaining atoms α' in ϕ such that $\mathit{root}(\alpha') \in \mathcal{V}(\alpha)$ are also moved from ϕ to ϕ_1. This rule is applied recursively until no such atom exists. Afterwards, all the atoms still remaining in ϕ are added to ϕ_2. It is clear that all these operations can be performed in polynomial time w.r.t. the size of $\mathcal{S}$ (in particular, the number of applications of the previous rule is bounded by the number of atoms in ϕ). □

5. Properties of the Proof Procedure

5.1. Soundness

We prove that the proof procedure is sound, in the sense that the root of every fully expanded proof tree fulfilling the conditions of Definition 4.31 is valid. As infinite proof trees are allowed, this does not follow from the fact that all the rules are sound. We show that every infinite branch contains infinitely many applications of the rule S. As we shall see, this is sufficient to ensure that no counter-model exists, since otherwise the size of the smallest counter-model would be decreasing indefinitely along some infinite branch. We recall that the rules are meant to be applied bottom-up: a rule is applicable on some sequent $\mathcal{S}$ if it admits an instance with conclusion $\mathcal{S}$, yielding some premises $\mathcal{S}_1, \dots, \mathcal{S}_n$. We focus on sequents on which rule I is applied (which we call I-reducible), and we establish some useful properties.

Definition 5.1. A sequent is $\mathtt{I}$-*reducible* if rule $\mathtt{I}$ can be applied on it; this entails that no other rule is applicable, as all other rules have priority over $\mathtt{I}$. A sequent $\mathcal{S}$ is *quasi-*$\mathtt{I}$*-reducible* if it is of the form $(x \mapsto \boldsymbol{y} * \phi) \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \top$, where ξ is a conjunction of disequations, ψ is a predicate atom with root x, and $\mathcal{S}$ is not an axiom. If $\mathcal{S}$ is quasi-$\mathtt{I}$-reducible then we denote by $\mathcal{V}_{\mapsto}(\mathcal{S})$ the set of variables occurring in $\boldsymbol{y}$.

Note that the definition of $\mathcal{V}_{\mapsto}(\mathcal{S})$ is unambiguous because the left-hand side of $\mathcal{S}$ cannot contain more than one points-to atom with root $x = \mathit{root}(\psi)$, otherwise it would be heap-unsatisfiable and $\mathcal{S}$ would be an axiom. It is easy to check that every $\mathtt{I}$-reducible sequent is quasi-$\mathtt{I}$-reducible (see Definition 4.31, Condition 2). The converse does not hold in general, because rules $\mathtt{W}$, $\mathtt{V}$ or $\mathtt{C}$ may be applicable on the considered sequent, and they have priority over $\mathtt{I}$. We observe that the application of $\mathtt{I}$ is necessarily interleaved with that of $\mathtt{S}$:

Lemma 5.2. Every path in a proof tree between two distinct $\mathtt{I}$-reducible sequents contains an application of rule $\mathtt{S}$.

Proof:
Let $\mathcal{S}_1$ be an $\mathtt{I}$-reducible sequent and assume that $\mathcal{S}_2$ is a descendant of $\mathcal{S}_1$. By definition, the only rule that applies on $\mathcal{S}_1$ is $\mathtt{I}$, yielding a sequent $\mathcal{S}_1'$. Since the rules in $\mathfrak{R}$ are $\mathtt{P}$-rules, necessarily the right-hand side of $\mathcal{S}_1'$ contains exactly one points-to atom. Since $\mathcal{S}_2$ is $\mathtt{I}$-reducible, $\mathtt{I}$ applies on $\mathcal{S}_2$, which entails that no points-to atom can occur on the right-hand side of $\mathcal{S}_2$ (since $\mathtt{I}$ applies only when the right-hand side is a predicate atom). The only rule that can remove a points-to atom from the right-hand side of a sequent is $\mathtt{S}$, thus $\mathtt{S}$ necessarily applies along the path from $\mathcal{S}_1$ to $\mathcal{S}_2$. □

To analyze the termination behavior of the rules, we define a new set of variables $\mathcal{V}^{\star}(\mathcal{S})$, which is similar to $\mathcal{V}^{\dagger}(\mathcal{S})$: it contains variables that occur on the right-hand side of the considered sequent but are not allocated on the left-hand side, and that do not occur either in the set of non-allocated variables V.

Definition 5.3. For every sequent $\mathcal{S} = \lambda \vdash_{\mathfrak{R}}^{V} \gamma$, we denote by $\mathcal{V}^{\star}(\mathcal{S})$ the set $\mathcal{V}_{\mathtt{loc}}(\gamma) \setminus (V \cup \mathit{alloc}(\lambda))$.

Note that, since the considered inferences are admissible, if rule $\mathtt{C}$ applies on variables x, y (with the notations of the rule), then necessarily $y \in \mathcal{V}^{\star}(\mathcal{S})$ and $x \in \mathcal{V}(\lambda)$ (see Condition 4 in Definition 4.31). The next proposition states that the rules (except possibly $\mathtt{R}$) cannot add new variables in the set $\mathcal{V}^{\star}(\mathcal{S})$.

Proposition 5.4. Let $\mathcal{S}_1$ be a sequent containing no equality. For every successor $\mathcal{S}_2$ of $\mathcal{S}_1$, we have $\mathcal{V}^{\star}(\mathcal{S}_2) \subseteq \mathcal{V}^{\star}(\mathcal{S}_1)$.

Proof:
The proof is similar to that of Proposition 4.32. Let $\mathcal{S}_i = \lambda_i \vdash_{\mathfrak{R}}^{V_i} \gamma_i$ (for $i = 1, 2$). Assume for the sake of contradiction that $v \in \mathcal{V}^{\star}(\mathcal{S}_2) \setminus \mathcal{V}^{\star}(\mathcal{S}_1)$. By definition of $\mathcal{V}^{\star}(\mathcal{S}_i)$, v is of sort $\mathtt{loc}$, $v \in \mathcal{V}(\gamma_2)$, $v \notin \mathit{alloc}(\lambda_2) \cup V_2$, and either $v \notin \mathcal{V}(\gamma_1)$ or $v \in \mathit{alloc}(\gamma_1) \cup V_1$. First assume that $v \notin \mathcal{V}(\gamma_1)$. Since $\mathcal{S}_1$ contains no equality and $v \notin \mathcal{V}^{\star}(\mathcal{S}_1)$, by Proposition 4.17 (4), the only rule that can introduce new variables to the right-hand side of a sequent is $\mathtt{I}$. If Rule $\mathtt{I}$ is applied, then we must have $v = y_i\sigma$ for

some $i = 1, \dots, n$, with the notations used in the definition of the rule. Since all the rules in $\mathfrak{R}$ are P-rules, γ_2 necessarily contains an atom with root v and $v \in alloc(\gamma_2)$. Since $v \notin alloc(\lambda_2)$, this entails by Definition 4.24 (1) that $\mathcal{S}_2$ is an anti-axiom, which contradicts Condition 1 in Definition 4.31. Thus we necessarily have $v \in \mathcal{V}(\gamma_1)$. Now assume that $v \in alloc(\gamma_1) \cup V_1$. Then by Proposition 4.17 (7), the only rule that can remove a variable v from $alloc(\gamma_1) \cup V_1$ is C, setting $x = v$. However, C replaces x by another variable in the entire sequent, hence we have $x \notin \mathcal{V}(\gamma_2)$, thus x cannot be v. □

We show (Lemma 5.12) that the rules terminate in polynomial time if I is not applied. To this purpose we prove that both the depth of the proof tree and the number of branches are polynomial. We first introduce the following definition:

Definition 5.5. A path $\mathcal{S}_0, \dots, \mathcal{S}_n$ in a proof tree from $\mathcal{S}$ to $\mathcal{S}'$ is I-*free* if there is no application of the rule I on $\mathcal{S}_0, \dots, \mathcal{S}_{n-1}$. A descendant $\mathcal{S}'$ of $\mathcal{S}$ is called I-*free* if the path from $\mathcal{S}$ to $\mathcal{S}'$ is I-free.

The first goal is thus to prove that the length of all I-free paths is bounded. Then we will derive a bound on the total number of I-free descendants. To this purpose, we analyze the rules that can be applied on quasi-I-reducible sequents, and derive a number of easy but useful technical results.

Proposition 5.6. If a sequent $\mathcal{S}$ is quasi-I-reducible then the only rules that can be applied on $\mathcal{S}$ are I, W, C, or V. Moreover, for every I-free descendant $\mathcal{S}'$ of $\mathcal{S}$, the sequent $\mathcal{S}'$ is quasi-I-reducible and $\mathcal{V}_{\mapsto}(\mathcal{S}) \cap \mathcal{V}(\mathcal{S}') \subseteq \mathcal{V}_{\mapsto}(\mathcal{S}')$.

Proof:
The proof is by an inspection of the different rules. Note that if U applies on $(x \mapsto \boldsymbol{y} * \phi) \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \top$ (with the notations of Definition 5.1), then, by Definition 4.31 (Condition 3), ϕ must contain an atom with the same root as $root(\psi) = x$, hence $(x \mapsto \boldsymbol{y} * \phi)$ is heap-unsatisfiable and the sequent is an axiom. For the second part, it is clear that the only rule among W, C or V that can delete a variable x from $\mathcal{V}_{\mapsto}(\mathcal{S})$ is C, and this rule entirely removes x from the sequent. □

Proposition 5.7. The premises of U are quasi-I-reducible.

Proof:
We use the notations of the rule. By Definition 4.31 (Condition 3), γ is a predicate atom with the same root as $p(\boldsymbol{t})$ and ξ is a conjunction of disequations. Since the rules in $\mathfrak{R}$ are P-rules, every formula ϕ_i contains a points-to atom with root $root(p(\boldsymbol{t})) = root(\gamma)$. Thus the premises of U are necessarily quasi-I-reducible. □

Corollary 5.8. There is at most one application of U along a path containing no I-reducible sequent.

Proof:
The result follows immediately from Propositions 5.6 and 5.7. □

We eventually derive (Lemma 5.10) the result concerning the length of the I-free paths. To this aim we first introduce a new notation:

Definition 5.9. For every sequent $\mathcal{S} = \lambda \vdash_{\mathfrak{R}}^{V} \gamma$, we denote by $\texttt{NbDisEq}(\mathcal{S})$ the number of disequations $x \not\approx y$ not occurring in λ such that $x, y \in \mathcal{V}(\mathcal{S})$.

Lemma 5.10. The length of every $\texttt{I}$-free path from $\mathcal{S}$ is at most $O(|\mathcal{S}|^2)$.

Proof:
By Corollary 5.8, an $\texttt{I}$-free path contains at most one application of $\texttt{U}$. Rule $\texttt{R}$ applies at most $\mathit{card}(\mathcal{V}(\mathcal{S})^2)$ times, with highest priority, yielding an equality-free sequent, and afterward $\texttt{R}$ can no longer be applied, since no inference rules introduce any equality to a sequent. Thus it is sufficient to prove the result for $\texttt{I}$-free paths containing no application of rules $\texttt{R}$ or $\texttt{U}$. Let $\mathcal{S}_1, \dots, \mathcal{S}_n$ be a path with no application of $\texttt{R}$, $\texttt{U}$ or $\texttt{I}$, where $\mathcal{S}_1 = \mathcal{S}$. Let $\mathcal{S}_i = \phi_i \curlywedge \xi_i \vdash_{\mathfrak{R}}^{V_i} \psi_i \curlywedge \zeta_i$. An inspection of the rules shows that we have $5 \cdot \texttt{NbDisEq}(\mathcal{S}_i) + |\mathcal{S}_i| > 5 \cdot \texttt{NbDisEq}(\mathcal{S}_{i+1}) + |\mathcal{S}_{i+1}|$, for all $i = 1, \dots, n$. Indeed, none of the considered rules can add new variables to $\mathcal{S}_i$ by Proposition 4.17 (4) and Condition 4 of Definition 4.31, all the rules (except possibly $\texttt{C}$) decrease $|\mathcal{S}_i|$ and $\texttt{W}$ cannot remove disequations between variables occurring in predicate atoms. Rule $\texttt{C}$ may add a disequation $x \not\approx y$ in $\mathcal{S}_{i+1}$ (increasing the size by $4 = |* x \not\approx y|$), but simultaneously removes the disequation $x \not\approx y$ from $\texttt{NbDisEq}(\mathcal{S}_{i+1})$. Then the proof follows from the fact that $5 \cdot \texttt{NbDisEq}(\mathcal{S}_1) + |\mathcal{S}_1| = O(|\mathcal{S}_1|^2)$. □

To derive the result about the total number of $\texttt{I}$-free descendants, knowing the *length* of the paths is of course not sufficient: it is also necessary to estimate the *number* of such paths, which depends in particular on the number of applications of rule $\texttt{C}$. To this purpose, we prove the following result:

Lemma 5.11. Let $\mathcal{S}$ be a sequent, with $\mathit{card}(\mathcal{V}^\star(\mathcal{S})) = \kappa$. The exhaustive application of $\texttt{C}$ on $\mathcal{S}$ yields at most $\kappa \cdot \texttt{NbDisEq}(\mathcal{S})^\kappa$ different branches.

Proof:
The proof is by induction on the pair $(\kappa, \texttt{NbDisEq}(\mathcal{S}))$. By definition, every application of $\texttt{C}$ on $\mathcal{S}$ yields two sequents $\mathcal{S}_1$ and $\mathcal{S}_2$, where $\mathcal{S}_1$ is obtained from $\mathcal{S}$ by replacing a variable x by y and $\mathcal{S}_2$ is obtained by adding the disequation $x \not\approx y$. By the application condition of the rule (Definition 4.31, Condition 4) necessarily $y \in \mathcal{V}^\star(\mathcal{S})$. Since x is replaced by y and x is the root of an atom from the left-hand side of $\mathcal{S}$, we must have $y \not\in \mathcal{V}^\star(\mathcal{S}_1)$, thus $\mathit{card}(\mathcal{V}^\star(\mathcal{S}_1)) = \kappa - 1$. Now $\texttt{NbDisEq}(\mathcal{S}_1) \leq \texttt{NbDisEq}(\mathcal{S})$, $\texttt{NbDisEq}(\mathcal{S}_2) = \texttt{NbDisEq}(\mathcal{S}) - 1$ (since by Definition 4.31, Condition 4, $x, y \in \mathcal{V}(\mathcal{S})$ and $x \not\approx y$ does not occur in $\mathcal{S}$) and $\mathcal{V}^\star(\mathcal{S}_2) = \mathcal{V}^\star(\mathcal{S})$, hence $\mathit{card}(\mathcal{V}^\star(\mathcal{S}_2)) = \kappa$. By the induction hypothesis, the application of $\texttt{C}$ generates at most $(\kappa - 1) \cdot \texttt{NbDisEq}(\mathcal{S}_1)^{\kappa-1} \leq \kappa \cdot \texttt{NbDisEq}(\mathcal{S})^{\kappa-1}$ branches on $\mathcal{S}_1$ and $\kappa \cdot (\texttt{NbDisEq}(\mathcal{S}) - 1)^\kappa \leq \kappa \cdot \texttt{NbDisEq}(\mathcal{S})^{\kappa-1} \cdot (\texttt{NbDisEq}(\mathcal{S}) - 1) = \kappa \cdot \texttt{NbDisEq}(\mathcal{S})^\kappa - \kappa \cdot \texttt{NbDisEq}(\mathcal{S})^{\kappa-1}$ branches on $\mathcal{S}_2$. Thus the total number of branches is at most $\kappa \cdot \texttt{NbDisEq}(\mathcal{S})^\kappa$. □

Lemma 5.12. Assume that $\mathit{width}(\mathfrak{R}) \leq \kappa$, for some fixed $\kappa \in \mathbb{N}$ (i.e., that both the maximal arity of the symbols and the number of record fields are bounded by κ). The number of $\texttt{I}$-free descendants of any sequent $\mathcal{S}$ is polynomial w.r.t. $|\mathcal{S}| + |\mathfrak{R}|$ (but it is exponential w.r.t. κ).

Proof:
By Lemma 5.10, it is sufficient to prove that the total number of paths occurring in a proof tree with no application of $\texttt{I}$ is polynomial w.r.t. $|\mathcal{S}| + |\mathfrak{R}|$.

The rules R, W, V, E apply with the highest priority and yield only one branch, yielding a (unique) sequent $\phi \curlywedge \xi \vdash \psi \curlywedge \zeta$. If $\zeta \neq \top$, then by Definition 4.31, no other rule can be applied because these other rules apply only when the right-hand side is of the form $\psi \curlywedge \top$. Otherwise, the rule S may apply (possibly several times), and eventually transforms the sequent into sequents $\phi_i \curlywedge \xi_i \vdash_{\mathfrak{R}}^{V_i} \psi_i \curlywedge \top$, where each ψ_i is a predicate atom, $\phi = *_{i=1}^{n} \phi_i$ and $\psi = *_{i=1}^{n} \psi_i$ (we may have $\xi_i \neq \xi$ since the rule may interleaved with applications of rule W).

If the considered sequent is not narrow, then by definition of the strategy (Condition 7 in Definition 4.31), there is at most one application of rule S; indeed, if several rule applications are possible then only the one that is selected is considered. We thus obtain at most $|\mathcal{S}|$ branches, each ending with a narrow sequent (because ψ_i is a predicate atom, thus $\phi_i \curlywedge \xi_i \vdash_{\mathfrak{R}}^{V_i} \psi_i$ is necessarily narrow).

Now assume that $\mathcal{S}$ is narrow and let x_i be the root of ψ_i. We have $alloc(\psi_i) = \{x_i\}$, because ψ_i is a predicate atom. Since $\phi_i * \xi_i \vdash_{\mathfrak{R}}^{V_i} \psi_i$ is not an axiom, we have $x_i \in alloc(\phi_i)$. By definition of rule S, this entails that $x_i \in V_j$, for all $j \in \{1, \dots, i-1, i+1, \dots, n\}$. For S to be applied, it is only necessary to choose how to decompose the formula ϕ into $*_{i=1}^{n} \phi_i$. To this aim, we only have to associate each atom in ϕ with a formula ϕ_i, hence to associate each variable $y \in alloc(\phi)$ (that are by definition the roots of the spatial atoms in ϕ) to an index $i = 1, \dots, n$. By definition, the variable x_i must be associated with index i, as otherwise we would have $x_i \notin alloc(\phi_i)$ and the premise $\phi_i * \xi_i \vdash_{\mathfrak{R}}^{V_i} \psi_i$ would be an anti-axiom. We then arbitrarily choose the image of each variable occurring in $\mathcal{V}^{\dagger}(\mathcal{S})$. Afterward, the image of the other variables are fixed inductively as follows. Let $y \in alloc(\phi)$ and assume that $y \notin \mathcal{V}^{\dagger}(\mathcal{S}) \cup \{x_1, \dots, x_n\}$. If y occurs in some atom with root y' in ϕ and y' has already been associated with i, then we also associate y with i. For the sake of contradiction, assume that y is associated with an index $j \neq i$. By definition of rule S, this entails that $y \in V_i$, and since $\phi_i * \xi_i \vdash_{\mathfrak{R}}^{V_i} \psi_i$ is not an anti-axiom, we deduce that $y \in \mathcal{V}(\psi_i)$, hence $y \in \mathcal{V}(\psi) \setminus (alloc(\psi) \cup V) = \mathcal{V}^{\dagger}(\mathcal{S})$, which contradicts our assumption (we cannot have $y \in V$ since $\mathcal{S}$ would then be an axiom). Note that, for all variables $y \in alloc(\phi) \setminus \{x_1, \dots, x_n\}$, we have $x_i \rightarrow_{\phi}^{*} y$, for some $i = 1, \dots, n$, because otherwise $\mathcal{S}$ would be an anti-axiom. Hence all such variables y must be eventually associated with some indice $i = 1, \dots, n$. Since $\mathcal{S}$ is narrow, $card(\mathcal{V}^{\dagger}(\mathcal{S})) \leq \kappa$. Consequently, there exist at most $|\mathcal{S}|^{\kappa}$ possible applications of rule S, each yielding $n \leq |\mathcal{S}|$ branches. We thus get a total of at most $|\mathcal{S}|^{\kappa+1}$ branches.

Afterwards, rule U applies, yielding at most $|\mathfrak{R}|$ premises in each branch. Then C applies on each leaf sequent $\mathcal{S}'$, and by Lemma 5.11 we get at most $|\mathfrak{R}| \cdot \mathtt{NbDisEq}(\mathcal{S}')^{\kappa}$ branches. The variables in $\mathcal{S}'$ either occur in $\mathcal{S}$ or are introduced by the rule U. Since each application of U introduces at most $record_{max}(\mathfrak{R})$ variables, we get $\mathtt{NbDisEq}(\mathcal{S}') \leq (|\mathcal{S}| + record_{max}(\mathfrak{R}))^2$. □

We derive the main result of this section:

Theorem 5.13. The root sequent of every (possibly infinite) fully expanded proof tree is valid.

Proof:
Let $\mathcal{S}$ be a non-valid sequent, with a counter-model $(\mathfrak{s}, \mathfrak{h})$. By Lemma 4.23, $\mathcal{S}$ is not an axiom, hence by definition of a fully expanded proof tree, it admits successors. By Lemmata 4.19 and 4.21, one of these successors must be non-valid and must admit a counter-model $(\mathfrak{s}, \mathfrak{h}')$, with $card(\mathfrak{h}') \leq card(\mathfrak{h})$, and if the rule that applies on $\mathcal{S}$ is S, then $card(\mathfrak{h}') < card(\mathfrak{h})$. Starting from the root of the tree, we thus obtain (if this root is not valid) an infinite path $\mathcal{S}_0, \dots, \mathcal{S}_n, \dots$, such that all the $\mathcal{S}_i$ admit a

counter-model $(\mathfrak{s}_i, \mathfrak{h}_i)$, $card(\mathfrak{h}_{i+1}) \leq card(\mathfrak{h}_i)$ and if S is applies on $\mathcal{S}_i$ then $card(\mathfrak{h}_{i+1}) < card(\mathfrak{h}_i)$. Since $card(\mathfrak{h})$ is finite this entails that there exists $i \geq 0$ such that rule S does not apply on $\mathcal{S}_j$, for all $j \geq i$. By Lemma 5.2, this entails that there exists k such that I does not apply on $\mathcal{S}_j$, for all $j \geq k$. Thus $\mathcal{S}_k$ admits an infinite number of I-free descendants, which contradicts Lemma 5.12. □

5.2. Completeness

We now establish completeness, i.e., we prove that every valid sequent admits a fully expanded tree. To this aim, we prove that for every valid sequent, there exists a rule application yielding valid premises. Lemma 5.20 handles the case of rule S, and Lemma 5.21 handles all the other rules. We begin by establishing several preliminary results. First, we note that the truth value of a formula in a structure is not dependent on the *names* of the locations: these locations can be freely renamed, provided the relations between the variables are preserved, and provided allocated locations are not mapped to the same image. This result will be useful to construct copies of models when needed in forthcoming proofs. More formally, we introduce a notion of a $\mathfrak{U}$-*mapping* and state some conditions ensuring that the application of such $\mathfrak{U}$-mappings on a structure preserves the truth value of a formula.

Definition 5.14. A $\mathfrak{U}$-*mapping* is a function ν mapping every element of $\mathfrak{U}_{\mathtt{s}}$ to an element of $\mathfrak{U}_{\mathtt{s}}$. Let $\mathfrak{h}$ be a heap. If ν is injective on $dom(\mathfrak{h})$, then we denote by $\nu(\mathfrak{h})$ the heap with domain $\nu(dom(\mathfrak{h}))$, such that for all $\ell_0 \in dom(\mathfrak{h})$, with $\mathfrak{h}(\ell_0) = (\ell_1, \dots, \ell_n)$, we have $\nu(\mathfrak{h})(\nu(\ell_0)) = (\nu(\ell_1), \dots, \nu(\ell_n))$.

Lemma 5.15. Let λ be a symbolic heap, let $(\mathfrak{s}, \mathfrak{h})$ be an $\mathfrak{R}$-model of λ and let ν be a $\mathfrak{U}$-mapping that is injective on $\mathfrak{s}(\mathcal{V}(\lambda)) \cup dom(\mathfrak{h})$. Then $(\nu \circ \mathfrak{s}, \nu(\mathfrak{h})) \models_{\mathfrak{R}} \lambda$.

Proof:
The proof is by induction on the satisfiability relation. The result is established also for spatial formulas and pure formulas.

- If λ is an equation $x \approx y$, then $\mathfrak{s}(x) = \mathfrak{s}(y)$, hence $\nu(\mathfrak{s}(x)) = \nu(\mathfrak{s}(y))$ and $(\nu \circ \mathfrak{s}, \nu(\mathfrak{h})) \models_{\mathfrak{R}} \lambda$.
- If λ is a disequation $x \not\approx y$, then $\mathfrak{s}(x) \neq \mathfrak{s}(y)$, and since ν is injective on $\mathfrak{s}(\mathcal{V}(\lambda))$ we get $\nu(\mathfrak{s}(x)) \neq \nu(\mathfrak{s}(y))$ and $(\nu \circ \mathfrak{s}, \nu(\mathfrak{h})) \models_{\mathfrak{R}} \lambda$.
- If $\lambda = y_0 \mapsto (y_1, \dots, y_n)$ then we must have $\mathfrak{h} = \{(\mathfrak{s}(y_0), \dots, \mathfrak{s}(y_n))\}$, which entails that $\nu(\mathfrak{h})$ is $\{(\nu(\mathfrak{s}(y_0)), \dots, \nu(\mathfrak{s}(y_n)))\}$ and that $(\nu \circ \mathfrak{s}, \nu(\mathfrak{h})) \models_{\mathfrak{R}} \lambda$.
- If $\lambda = \phi \curlywedge \xi$ (or $\lambda = \phi \wedge \xi$) then we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi$ and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \xi$. By the induction hypothesis, we get $(\nu \circ \mathfrak{s}, \nu(\mathfrak{h})) \models_{\mathfrak{R}} \phi$ and $(\nu \circ \mathfrak{s}, \nu(\mathfrak{h})) \models_{\mathfrak{R}} \xi$, thus $(\nu \circ \mathfrak{s}, \nu(\mathfrak{h})) \models_{\mathfrak{R}} \lambda$.
- If $\lambda = \phi_1 * \phi_2$ then there exists disjoint heaps $\mathfrak{h}_i$ (for $i = 1, 2$) such that $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$ and $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \phi_i$. Since $\mathfrak{s}(\mathcal{V}(\phi_i)) \cup dom(\mathfrak{h}_i) \subseteq \mathfrak{s}(\mathcal{V}(\lambda)) \cup dom(\mathfrak{h})$, by the induction hypothesis we have $(\nu \circ \mathfrak{s}, \nu(\mathfrak{h}_i)) \models_{\mathfrak{R}} \phi_i$ (for $i = 1, 2$). But $\nu(\mathfrak{h}_1)$ and $\nu(\mathfrak{h}_2)$ must be disjoint since $dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2) = \emptyset$ and ν is injective, therefore $(\nu \circ \mathfrak{s}, \nu(\mathfrak{h})) \models_{\mathfrak{R}} \lambda$.

- If λ is a predicate atom, then we have $\lambda \Leftarrow_{\mathfrak{R}} \gamma$ and $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \gamma$, for some associate $\mathfrak{s}'$ of $\mathfrak{s}$ w.r.t. $\mathcal{V}(\gamma) \setminus \mathcal{V}(\lambda)$. Since the rules in $\mathfrak{R}$ are P-rules, for all variables $x \in \mathcal{V}(\gamma) \setminus \mathcal{V}(\lambda)$, we have $x \in \mathit{alloc}(\gamma)$ thus $\mathfrak{s}'(x) \in \mathit{dom}(\mathfrak{h})$ by Lemma 4.2. This entails that ν is injective on $\mathfrak{s}'(\mathcal{V}(\gamma)) \cup \mathit{dom}(\mathfrak{h})$, and by the induction hypothesis we get $(\nu \circ \mathfrak{s}', \nu(\mathfrak{h})) \models_{\mathfrak{R}} \xi$. Moreover, it is clear that $\nu(\mathfrak{s}')$ is an associate of $\nu(\mathfrak{s})$ w.r.t. $\mathcal{V}(\gamma) \setminus \mathcal{V}(\lambda)$, thus $(\nu \circ \mathfrak{s}, \nu(\mathfrak{h})) \models_{\mathfrak{R}} \lambda$.

□

This result entails that for every set U that is sufficiently large, counter-models may always be renamed, so that all locations occur in U:

Corollary 5.16. Let U be an infinite subset of $\mathfrak{U}_{\mathtt{s}}$. Any non-valid and equality-free sequent $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$ admits a counter-model $(\mathfrak{s}, \mathfrak{h})$ such that $\mathit{dom}(\mathfrak{h}) \cup \mathfrak{s}(\mathcal{V}_{\mathtt{loc}}) \subseteq U$.

Proof:
It suffices to consider any counter-model of γ and apply a bijective $\mathfrak{U}$-mapping (or any $\mathfrak{U}$-mapping that is injective on $\mathfrak{s}(\mathcal{V}(\lambda)) \cup \mathit{dom}(\mathfrak{h})$) that maps all locations $\ell \in \mathfrak{U}_{\mathtt{loc}}$ to an element in U. □

The first step toward establishing completeness is to show that rule S always applies (on a valid sequent) if the right-hand side of the sequent contains several spatial formulas (assuming that the rules of higher priority do not apply). This is essential because the strategy in Definition 4.31 forbids the application of the other rules in this case. The difficulty is that, of course, S cannot be applied arbitrarily: we are required to obtain valid premises. To this purpose we show that the heap decomposition of the left-hand side $(\phi_1 * \phi_2)$ matches that of the right-hand side $(\psi_1 * \psi_2)$. This will be proven thanks to the next lemma. Intuitively, this lemmas states that, under some additional conditions, if a structure $(\mathfrak{s}, \mathfrak{h}_1 \uplus \mathfrak{h}_2)$ validates a formula $\phi_1 * \phi_2$, then we may deduce that each structure $(\mathfrak{s}, \mathfrak{h}_i)$ validates ϕ_i, when all the variables allocated by ϕ_i are in the domain of $\mathfrak{h}_i$.

Lemma 5.17. Let ϕ_i, ψ_i (for $i = 1, 2$) be spatial formulas, with $\mathit{alloc}(\psi_1 * \psi_2) \subseteq \mathit{alloc}(\phi_1 * \phi_2)$. Let $\mathfrak{h}_1$ and $\mathfrak{h}_2$ be disjoint heaps such that $(\mathfrak{s}, \mathfrak{h}_1 \uplus \mathfrak{h}_2) \models_{\mathfrak{R}} \phi_1 * \phi_2$ and $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \psi_i$, for $i = 1, 2$. If $(\mathit{dom}(\mathfrak{h}_1) \cup \mathit{dom}(\mathfrak{h}_2)) \cap \mathfrak{s}(\mathcal{V}) \subseteq \mathfrak{s}(\mathit{alloc}(\phi_1 * \phi_2))$ and $\mathfrak{s}(\mathit{alloc}(\phi_i)) \subseteq \mathit{dom}(\mathfrak{h}_i)$ for $i = 1, 2$, then $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \phi_i$, for $i = 1, 2$.

Proof:
By definition, since $(\mathfrak{s}, \mathfrak{h}_1 \uplus \mathfrak{h}_2) \models_{\mathfrak{R}} \phi_1 * \phi_2$, there exist disjoint heaps $\widehat{\mathfrak{h}}_1, \widehat{\mathfrak{h}}_2$ such that $(\mathfrak{s}, \widehat{\mathfrak{h}}_i) \models_{\mathfrak{R}} \phi_i$, for all $i = 1, 2$, and $\mathfrak{h}_1 \uplus \mathfrak{h}_2 = \widehat{\mathfrak{h}}_1 \uplus \widehat{\mathfrak{h}}_2$. We prove that $\widehat{\mathfrak{h}}_i = \mathfrak{h}_i$ for $i = 1, 2$. Since $\widehat{\mathfrak{h}}_1 \uplus \widehat{\mathfrak{h}}_2 = \mathfrak{h}_1 \uplus \mathfrak{h}_2$ and $\widehat{\mathfrak{h}}_1$ and $\widehat{\mathfrak{h}}_2$ are disjoint, it is sufficient to prove that $\mathit{dom}(\mathfrak{h}_i) \subseteq \mathit{dom}(\widehat{\mathfrak{h}}_i)$, for $i = 1, 2$. By symmetry, we prove the result for $i = 1$. Assume, for the sake of contradiction, that $\ell \in \mathit{dom}(\mathfrak{h}_1)$ and $\ell \notin \mathit{dom}(\widehat{\mathfrak{h}}_1)$. By Proposition 4.6, since $(\mathfrak{s}, \mathfrak{h}_1) \models_{\mathfrak{R}} \psi_1$ and $\ell \in \mathit{dom}(\mathfrak{h}_1) \subseteq \mathit{ref}(\mathfrak{h}_1)$, there exists $y \in \mathit{alloc}(\psi_1)$ such that $\mathfrak{s}(y) \rightarrow_{\mathfrak{h}_1}^{*} \ell$, thus there is a sequence of locations $\ell_0, \ldots, \ell_n$ with $\ell_0 = \mathfrak{s}(y)$, $\ell = \ell_n$ and $\ell_i \rightarrow_{\mathfrak{h}_1} \ell_{i+1}$, for all $i = 0, \ldots, n-1$. Let k be the smallest index such that $\ell_k \notin \mathit{dom}(\widehat{\mathfrak{h}}_1)$. Note that $\ell_i \in \mathit{dom}(\mathfrak{h}_1)$, for all $i \leq k$ and that $y \in \mathit{alloc}(\phi_1)$. Indeed, $y \in \mathit{alloc}(\psi_1) \subseteq \mathit{alloc}(\phi_1) \cup \mathit{alloc}(\phi_2)$ by the hypothesis of the lemma, and if $y \in \mathit{alloc}(\phi_2)$, then we get (again by the hypothesis of the lemma) $\mathfrak{s}(y) \in \mathit{dom}(\mathfrak{h}_2)$, which contradicts the fact that $\ell_0 = \mathfrak{s}(y) \in \mathit{dom}(\mathfrak{h}_1)$, as $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are disjoint. By

Lemma 4.2, we deduce that $\ell_0 = \mathfrak{s}(y) \in dom(\widehat{\mathfrak{h}_1})$ since $y \in alloc(\phi_1)$ and $(\mathfrak{s}, \widehat{\mathfrak{h}_1}) \models_{\mathfrak{R}} \phi_1$. Thus $k > 0$, and necessarily $\ell_k \in ref(\widehat{\mathfrak{h}_1})$ (since $\ell_{k-1} \in dom(\widehat{\mathfrak{h}_1})$, by minimality of k). We deduce that $\ell_k \in ref(\widehat{\mathfrak{h}_1}) \setminus dom(\widehat{\mathfrak{h}_1})$. By Lemma 4.7, this entails that $\ell_k = \mathfrak{s}(x)$ for some $x \in \mathcal{V}(\phi_1)$, as $(\mathfrak{s}, \widehat{\mathfrak{h}_1}) \models \phi_1$. By the hypothesis of the lemma, we may assume that $x \in alloc(\phi_1 * \phi_2)$, since $\ell_k \in dom(\mathfrak{h}_1)$ and $(dom(\mathfrak{h}_1) \cup dom(\mathfrak{h}_2)) \cap \mathfrak{s}(\mathcal{V}) \subseteq \mathfrak{s}(alloc(\phi_1 * \phi_2))$. However, we cannot have $x \in alloc(\phi_1)$ because otherwise we would have $\mathfrak{s}(x) \in dom(\widehat{\mathfrak{h}_1})$ by Lemma 4.2, as $(\mathfrak{s}, \widehat{\mathfrak{h}_1}) \models_{\mathfrak{R}} \phi_1$. We cannot have $x \in alloc(\phi_2)$ either, since otherwise we would have $\mathfrak{s}(x) \in dom(\mathfrak{h}_2)$, as $\mathfrak{s}(alloc(\phi_2)) \subseteq dom(\mathfrak{h}_2)$ by the hypothesis of the lemma, hence $\mathfrak{s}(x) \notin dom(\mathfrak{h}_1)$ because $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are disjoint. Thus we obtain a contradiction. □

We then derive the result about rule S. The main issue is that we have to take into account the fact that the strategy "blocks" some applications of S, if the sequent is not narrow (see Condition 5 in Definition 4.31). We show that blocked applications do not yield valid premises. To illustrate the difficulties that arise when applying S, we provide the examples below. The first one illustrates the importance of the fact that each predicate allocates at most one parameter.

Example 5.18. Consider the sequent $p(x, y) \vdash^{\emptyset}_{\mathfrak{R}} q(x, y) * r(y)$, with the rules:

$$\begin{array}{rcl@{\qquad\qquad}rcl}
p(x,y) & \Leftarrow & x \mapsto (y) * p'(y) & q(x,y) & \Leftarrow & x \mapsto (y) \\
p'(y) & \Leftarrow & y \mapsto () & r(y) & \Leftarrow & y \mapsto ()
\end{array}$$

Note that the rules of p are *not* P-rules, as both x and y are allocated by $p(x, y)$. Here, although the sequent is indeed valid, no application of S yields valid premises: to show that the sequent is valid, one has first to unfold $p(x, y)$ once, before applying S. In our context, such a situation cannot arise as each predicate atom allocates only one of its parameters, namely its root. As we shall see, this entails that every decomposition of the right-hand side of the sequent necessarily corresponds to some (purely syntactic) decomposition of the left-hand side.

The next example illustrates the importance of determinism.

Example 5.19. Consider the sequent $\mathcal{S} : p(x, y) * q(y) * p(z, y) \vdash^{\emptyset}_{\mathfrak{R}} p'(x, y) * q'(z, y)$, with the rules:

$$\begin{array}{rcl@{\qquad}rcl}
p(x,y) & \Leftarrow & x \mapsto (y) & q(y) & \Leftarrow & y \mapsto (z) * q(z) \\
q(y) & \Leftarrow & y \mapsto () & q(y) & \Leftarrow & y \mapsto (y) \\
p'(x,y) & \Leftarrow & x \mapsto (z) * p'(z,y) & q'(x,y) & \Leftarrow & x \mapsto (z) * q'(z,y) \\
p'(x,y) & \Leftarrow & x \mapsto (y) & q'(x,y) & \Leftarrow & x \mapsto (y) \\
p'(x,y) & \Leftarrow & x \mapsto () & q'(x,y) & \Leftarrow & x \mapsto (x)
\end{array}$$

Intuitively, $q(y)$ denotes a list starting at y and ending either with an empty tuple or with a loop on its last element, whereas $p'(x, y)$ (resp. $q'(x, y)$) denotes a list starting at x and ending either with y or with an empty tuple (resp. with a loop on the last element). These rules are not deterministic (for instance there is an overlap between the rules $p'(x, y) \Leftarrow x \mapsto (z) * p'(z, y)$ and $p'(x, y) \Leftarrow x \mapsto (y)$). Although the sequent $\mathcal{S}$ is valid, there is no application of S that yields valid premises. Indeed, none of the sequents $p(x,y) * q(y) \vdash_{\mathfrak{R}}^{\emptyset} p'(x,y)$, or $q(y) * p(z,y) \vdash_{\mathfrak{R}}^{\emptyset} q'(z,y)$ are valid. Here S cannot be applied before the entire list $q(y)$ is unfolded, as the decision to group $q(y)$ with $p(x,y)$ or $p(y,z)$ cannot be made before the last cell in $q(y)$ is known. This would yield an infinite proof tree (with infinitely many branches). The fact that the rules are deterministic prevents such a behavior to occur.

Lemma 5.20. Let $\mathcal{S} : \phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} (\psi_1 * \psi_2) \curlywedge \top$ be a valid sequent where $\psi_i \neq \mathit{emp}$ for $i = 1,2$ and ξ is a conjunction of disequations. If $\mathcal{S}$ is not an axiom, then there exists an application of S with conclusion $\mathcal{S}$ for which all the premises are valid.

Proof:
To prove that S applies on $\mathcal{S}$, it is necessary to show that ϕ is of the form $\phi_1 * \phi_2$. However ϕ_i cannot be chosen arbitrarily, since the premises have to be valid. To construct ϕ_i, we shall construct a model of ϕ of a particular form, get a decomposition of the heap of this model from the formula $\psi_1 * \psi_2$ on the right-hand side of the sequent, and use this decomposition to compute suitable formulas ϕ_1 and ϕ_2. We first notice that, since $\mathcal{S}$ is not an axiom, ϕ must be heap-satisfiable. Furthermore, since $\mathcal{S}$ is valid, it cannot be an anti-axiom by Lemma 4.27, thus $\mathit{alloc}(\psi_i) \subseteq \mathit{alloc}(\phi)$ for $i = 1,2$. Since $\psi_i \neq \mathit{emp}$, this entails that $\mathit{alloc}(\phi)$ is not empty. Let U_1, U_2 be disjoint infinite subsets of $\mathfrak{U}_{\mathtt{loc}}$ and let $\mathfrak{s}$ be an injective store such that $\mathfrak{s}(\mathcal{V}) \cap U_i = \emptyset$, for $i = 1,2$. Note that such U_1, U_2 and $\mathfrak{s}$ exist since $\mathfrak{U}_{\mathtt{s}}$ is infinite, for all $\mathtt{s} \in \mathfrak{S}$. By Lemma 4.26 applied with $U = U_1$ and with any variable $x \in \mathit{alloc}(\phi)$, there exists an $\mathfrak{R}$-model $(\mathfrak{s}, \mathfrak{h})$ of ϕ such that $\mathit{dom}(\mathfrak{h}) \subseteq U_1 \cup \mathfrak{s}(\mathit{alloc}(\phi))$. Since $\mathfrak{s}$ is injective and ξ is a conjunction of disequations, necessarily $\mathfrak{s} \models \xi$ (observe that ξ contains no disequation of the form $u \not\approx u$ as otherwise $\mathcal{S}$ would be an axiom). If there exists $x \in V$ such that $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h})$, then since $\mathit{dom}(\mathfrak{h}) \subseteq U_1 \cup \mathfrak{s}(\mathit{alloc}(\phi))$ and $\mathfrak{s}(\mathcal{V}) \cap U_1 = \emptyset$, there must exist $x' \in \mathit{alloc}(\phi)$ such that $\mathfrak{s}(x) = \mathfrak{s}(x')$. Since $\mathfrak{s}$ is injective, necessarily $x = x'$, which entails that $\mathcal{S}$ is an axiom, contradicting the hypotheses of the lemma. We deduce that $\mathfrak{s}(V) \cap \mathit{dom}(\mathfrak{h}) = \emptyset$. Since $\mathfrak{s}$ is injective, it is injective on V, as otherwise V would contain two occurrences of the same variable and $\mathcal{S}$ would be an axiom. Because $\mathcal{S}$ is valid, we deduce that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} (\psi_1 * \psi_2)$, hence there exist disjoint heaps $\mathfrak{h}_1$ and $\mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$ and $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \psi_i$, for $i = 1,2$. Let ϕ_i be the separating conjunction of all the spatial atoms α occurring in ϕ such that $\mathfrak{s}(\mathit{alloc}(\alpha)) \subseteq \mathit{dom}(\mathfrak{h}_i)$. By Lemma 4.2, $\mathfrak{s}(\mathit{alloc}(\phi)) \subseteq \mathit{dom}(\mathfrak{h}) = \mathit{dom}(\mathfrak{h}_1) \cup \mathit{dom}(\mathfrak{h}_2)$, hence every atom α occurs in either ϕ_1 or ϕ_2, and since $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are disjoint, no atom α can occur both in ϕ_1 and ϕ_2. Therefore, $\phi = \phi_1 * \phi_2$.

We now prove that $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \phi_i$ for $i = 1,2$. To this aim we use Lemma 5.17, thus we verify that all the hypotheses of the lemma are satisfied. Since $\mathit{dom}(\mathfrak{h}) \subseteq U_1 \cup \mathfrak{s}(\mathit{alloc}(\phi))$ and $U_1 \cap \mathfrak{s}(\mathcal{V}) = \emptyset$, we have $\mathit{dom}(\mathfrak{h}) \cap \mathfrak{s}(\mathcal{V}) \subseteq \mathfrak{s}(\mathit{alloc}(\phi))$, i.e. $(\mathit{dom}(\mathfrak{h}_1) \cup \mathit{dom}(\mathfrak{h}_2)) \cap \mathfrak{s}(\mathcal{V}) = \mathfrak{s}(\mathit{alloc}(\phi_1 * \phi_2))$. Furthermore, $\mathfrak{s}(\mathit{alloc}(\phi_i)) \subseteq \mathit{dom}(\mathfrak{h}_i)$ for $i = 1,2$ by definition of ϕ_i, $(\mathfrak{s}, \mathfrak{h}_1 \uplus \mathfrak{h}_2) = (\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi = \phi_1 * \phi_2$, $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \psi_i$ (for all $i = 1,2$) and $\mathit{alloc}(\psi_1 * \psi_2) \subseteq \mathit{alloc}(\phi_1 * \phi_2)$. By Lemma 5.17, we deduce that $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathfrak{R}} \phi_i$ for $i = 1,2$.

It is clear that there is an application of S on $\mathcal{S} = (\phi_1 * \phi_2) \curlywedge \xi \models_{\mathfrak{R}} (\psi_1 * \psi_2) \curlywedge \top$ yielding the premises $\phi_i \curlywedge \xi \vdash_{\mathfrak{R}}^{V_i} \psi_i$ for $i = 1,2$, where $V_i = V \cup \mathit{alloc}(\phi_{3-i})$. There remains to prove that these premises are valid and that the application of the rule is allowed by the strategy. Assume that one of these premises, say $\phi_1 \curlywedge \xi \vdash_{\mathfrak{R}}^{V_1} \psi_1$, is not valid. By Corollary 5.16 applied with $U = U_2$, $\phi_1 \curlywedge \xi \vdash_{\mathfrak{R}}^{V_1} \psi_1$ admits a counter-model $(\mathfrak{s}', \mathfrak{h}'_1)$, where $\mathit{dom}(\mathfrak{h}'_1) \cup \mathfrak{s}'(\mathcal{V}_{\mathtt{loc}}) \subseteq U_2$. By construction we have $\mathfrak{s}'(V_1) \cap \mathit{dom}(\mathfrak{h}'_1) = \emptyset$, thus $\mathfrak{s}'(V) \cap \mathit{dom}(\mathfrak{h}'_1) = \emptyset$ since $V \subseteq V_1$. Moreover, $\mathfrak{s}'$ is injective on $V_1 = V \cup alloc(\phi_2)$. By Lemma 4.26, ϕ_2 therefore admits a model $(\mathfrak{s}', \mathfrak{h}'_2)$ with $dom(\mathfrak{h}'_2) \subseteq U_1 \cup \mathfrak{s}'(alloc(\phi_2))$. We show that $\mathfrak{h}'_1$ and $\mathfrak{h}'_2$ are disjoint. Assume for the sake of contradiction that $\ell \in dom(\mathfrak{h}'_1) \cap dom(\mathfrak{h}'_2)$. Since $U_1 \cap U_2 = \emptyset$, we have $dom(\mathfrak{h}'_1) \cap dom(\mathfrak{h}'_2) \subseteq \mathfrak{s}'(alloc(\phi_2))$, thus $\ell \in \mathfrak{s}'(alloc(\phi_2))$. But since $\mathfrak{s}'(V_1) \cap dom(\mathfrak{h}'_1) = \emptyset$ and $alloc(\phi_2) \subseteq V_1$ we also have $\ell \notin \mathfrak{s}'(alloc(\phi_2))$, a contradiction. Thus $\mathfrak{h}'_1$ and $\mathfrak{h}'_2$ are disjoint and we have $(\mathfrak{s}', \mathfrak{h}'_1 \uplus \mathfrak{h}'_2) \models_{\mathfrak{R}} (\phi_1 * \phi_2) \curlywedge \xi = \phi \curlywedge \xi$. If $\mathfrak{s}'(V) \cap dom(\mathfrak{h}'_2)$ contains a location ℓ then since $U_1 \cap U_2 = \emptyset$, necessarily $\ell = \mathfrak{s}'(x)$, for some $x \in alloc(\phi_2)$, and since $\mathfrak{s}'$ is injective on $V_1 = V \cup alloc(\phi_2)$ and $\ell \in \mathfrak{s}'(V)$, necessarily, $x \in V$ and $\mathcal{S}$ is an axiom, which contradicts the hypotheses of the lemma. We deduce that $\mathfrak{s}'(V) \cap dom(\mathfrak{h}'_2) = \emptyset$, and since $\mathfrak{s}'(V) \cap dom(\mathfrak{h}'_1) = \emptyset$, that $\mathfrak{s}'(V) \cap dom(\mathfrak{h}'_1 \uplus \mathfrak{h}'_2) = \emptyset$. Furthermore, by construction, $\mathfrak{s}'$ is injective on V. Since $\mathcal{S}$ is valid, this entails that $(\mathfrak{s}', \mathfrak{h}'_1 \uplus \mathfrak{h}'_2) \models_{\mathfrak{R}} (\psi_1 * \psi_2)$, hence $\mathfrak{h}'_1 \uplus \mathfrak{h}'_2 = \mathfrak{h}''_1 \uplus \mathfrak{h}''_2$, where $(\mathfrak{s}', \mathfrak{h}''_i) \models_{\mathfrak{R}} \psi_i$. By Lemma 3.6, since $(\mathfrak{s}', \mathfrak{h}'_2) \models_{\mathfrak{R}} \psi_2$ and $\mathfrak{R}$ is deterministic, we get $\mathfrak{h}''_2 = \mathfrak{h}'_2$, and therefore $\mathfrak{h}''_1 = \mathfrak{h}'_1$, thus $(\mathfrak{s}', \mathfrak{h}'_1) \models_{\mathfrak{R}} \psi_1$. This contradicts the fact that $(\mathfrak{s}', \mathfrak{h}'_1)$ is a counter-model of $\phi_1 \curlywedge \xi \vdash_{\mathfrak{R}}^{V_2} \psi_1$.

If $\mathcal{S}$ is narrow, then the proof is completed, since all the applications of rule S are considered by the strategy in this case. If $\mathcal{S}$ is not narrow, then the application of the rule may be blocked by Condition 7 of Definition 4.31. In this case, this means that there exist ϕ'_1, ϕ'_2 such that $\phi = \phi'_1 * \phi'_2$ and $\phi'_1 \curlywedge \xi \vdash_{\mathfrak{R}}^{V'_1} \psi_1$ is valid, where $V'_1 = V \cup alloc(\phi'_2)$. We assume that the other premise, $\phi'_2 \curlywedge \xi \vdash_{\mathfrak{R}}^{V'_2} \psi_2$ (with $V'_2 = V \cup alloc(\phi'_1)$) is not valid, and we derive a contradiction. By Corollary 5.16, this premise admits a counter-model $(\mathfrak{s}'', \mathfrak{h}''_2)$ such that $dom(\mathfrak{h}''_2) \cup \mathfrak{s}''(\mathcal{V}_{\mathtt{loc}}) \subseteq U_1$. We have $(\mathfrak{s}'', \mathfrak{h}''_2) \models_{\mathfrak{R}} \phi'_2$, $\mathfrak{s}'' \models \xi$, $\mathfrak{s}''(V'_2) \cap dom(\mathfrak{h}''_2) = \emptyset$, $\mathfrak{s}''$ is injective on V'_2 and $(\mathfrak{s}, \mathfrak{h}''_2) \not\models_{\mathfrak{R}} \psi_2$. By Lemma 4.26, ϕ'_1 admits a model $(\mathfrak{s}'', \mathfrak{h}''_1)$ such that $dom(\mathfrak{h}''_1) \subseteq U_2 \cup \mathfrak{s}''(alloc(\phi'_1))$.

Note that the heaps $\mathfrak{h}''_1$ and $\mathfrak{h}''_2$ are disjoint. Indeed, if $\ell \in dom(\mathfrak{h}''_1) \cap dom(\mathfrak{h}''_2)$ then, since $dom(\mathfrak{h}''_1) \subseteq U_2 \cup \mathfrak{s}''(alloc(\phi'_1))$ and $dom(\mathfrak{h}''_2) \cup \mathfrak{s}''(\mathcal{V}_{\mathtt{loc}}) \subseteq U_1$, with $U_1 \cap U_2 = \emptyset$, we obtain $\ell \in \mathfrak{s}''(alloc(\phi'_1))$, so that $\ell \in \mathfrak{s}''(V'_2)$, contradicting the fact that $\mathfrak{s}''(V'_2) \cap dom(\mathfrak{h}''_2) = \emptyset$.

We prove that $\mathfrak{s}''$ is injective on V'_1, and that $dom(\mathfrak{h}''_1) \cap \mathfrak{s}''(V_1) = \emptyset$, which entails that $(\mathfrak{s}'', \mathfrak{h}''_1) \models_{\mathfrak{R}} \psi_1$, since $\phi'_1 \curlywedge \xi \vdash_{\mathfrak{R}}^{V'_1} \psi_1$ is valid. If there exist x_1, x_2 such that $\{x_1, x_2\} \subseteq_m V'_1 = V \cup alloc(\phi'_2)$ and $\mathfrak{s}''(x_1) = \mathfrak{s}''(x_2)$ then, as $\mathfrak{s}''$ is injective on $V'_2 = V \cup alloc(\phi'_1)$, necessarily at least one of the variables x_1, x_2 – say x_1 – is in $alloc(\phi'_2)$. As $(\mathfrak{s}'', \mathfrak{h}''_2) \models \phi'_2$, we get $\mathfrak{s}''(x_1) \in dom(\mathfrak{h}''_2)$ (by Lemma 4.2), and $\{x_1, x_2\} \not\subseteq_m alloc(\phi'_2)$ (by Corollary 4.3). This entails that $\mathfrak{s}''(x_1) \notin \mathfrak{s}''(V'_2)$ (as $\mathfrak{s}''(V'_2) \cap dom(\mathfrak{h}''_2) = \emptyset$) and $x_2 \in V \subseteq V'_2$, yielding a contradiction, since $\mathfrak{s}''(x_1) = \mathfrak{s}''(x_2)$. Now assume that there exists $y \in V'_1 = V \cup alloc(\phi'_2)$ such that $\mathfrak{s}''(y) \in dom(\mathfrak{h}''_1)$. As $dom(\mathfrak{h}''_1) \subseteq U_2 \cup \mathfrak{s}''(alloc(\phi'_1))$ and $dom(\mathfrak{h}''_2) \cup \mathfrak{s}''(\mathcal{V}_{\mathtt{loc}}) \subseteq U_1$, we get $\mathfrak{s}''(y) \in \mathfrak{s}''(alloc(\phi'_1))$, i.e., there exists $y' \in alloc(\phi'_1)$ such that $\mathfrak{s}''(y) = \mathfrak{s}''(y')$. If $y \in V$, then $\{y, y'\} \subseteq_m V'_2$, contradicting the fact that $\mathfrak{s}''$ is injective on V'_2. Thus $y \in alloc(\phi'_2)$, so that $\mathfrak{s}(y) \in dom(\mathfrak{h}''_2)$ (by Lemma 4.2), contradicting the fact that $\mathfrak{h}''_1$ and $\mathfrak{h}''_2$ are disjoint.

Thus $(\mathfrak{s}'', \mathfrak{h}''_1 \uplus \mathfrak{h}''_2) \models_{\mathfrak{R}} \phi'_1 * \phi'_2 = \phi$. As $\mathfrak{s}'' \models \xi$ and $\mathfrak{s}''(V) \cap dom(\mathfrak{h}''_i) \subseteq \mathfrak{s}(V_i) \cap dom(\mathfrak{h}''_i)$ is empty, $\mathfrak{s}''$ is injective on V, and $\mathcal{S}$ is valid, we get $(\mathfrak{s}'', \mathfrak{h}''_1 \uplus \mathfrak{h}''_2) \models_{\mathfrak{R}} \psi_1 * \psi_2$. Therefore, there exist disjoint heaps $\widehat{\mathfrak{h}}''_i$ (for $i = 1, 2$), such that $(\mathfrak{s}'', \widehat{\mathfrak{h}}''_i) \models_{\mathfrak{R}} \psi_i$ and $\mathfrak{h}''_1 \uplus \mathfrak{h}''_2 = \widehat{\mathfrak{h}}''_1 \uplus \widehat{\mathfrak{h}}''_2$. By Lemma 3.6, since $(\mathfrak{s}'', \mathfrak{h}''_1) \models_{\mathfrak{R}} \psi_1$, we deduce that $\mathfrak{h}''_1 = \widehat{\mathfrak{h}}''_1$, thus $\mathfrak{h}''_2 = \widehat{\mathfrak{h}}''_2$, hence and $(\mathfrak{s}'', \mathfrak{h}''_2) \models_{\mathfrak{R}} \psi_2$, which contradicts the fact that $(\mathfrak{s}'', \mathfrak{h}''_2)$ is a counter-model of $\phi'_2 \curlywedge \xi \vdash_{\mathfrak{R}}^{V'_2} \psi_2$. □

We handle the case of the other rules, more precisely we show that there is always an inference rule that applies on a valid sequent (if it is not an axiom):

Lemma 5.21. For any valid sequent $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ that is not an axiom, there exists an application of the inference rules that yields valid premises.

Proof:
Since $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ is not an axiom, ϕ is heap-satisfiable, $V \cap \mathit{alloc}(\phi) = \emptyset$ and V contains at most one occurrence of each variable. Note that if an invertible rule applies on $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ then the result holds, since the premises are necessarily valid. Thus we may assume that $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ is irreducible by all invertible rules, hence (by Lemma 4.19) by all the rules except S.

Let $\mathfrak{s}$ be an injective store such that the set $U = \mathfrak{U}_{\mathtt{loc}} \setminus \mathfrak{s}(\mathcal{V})$ is infinite. If $\phi \neq \mathit{emp}$, then ϕ contains at least one variable of sort loc, and by Lemma 4.26, we deduce that ϕ admits a model $(\mathfrak{s}, \mathfrak{h})$ such that $\mathit{dom}(\mathfrak{h}) \subseteq U \cup \mathfrak{s}(\mathit{alloc}(\phi))$. If $\phi = \mathit{emp}$ then the same property trivially holds, with $\mathfrak{h} = \emptyset$. Note in particular that the assertion $\mathit{dom}(\mathfrak{h}) \subseteq U \cup \mathfrak{s}(\mathit{alloc}(\phi))$ entails that $\mathfrak{s}(V) \cap \mathit{dom}(\mathfrak{h}) = \emptyset$ since $\mathfrak{s}(V) \cap U = \emptyset$, $V \cap \mathit{alloc}(\phi) = \emptyset$ and $\mathfrak{s}$ is injective.

If ξ contains an equation then rule R applies, which contradicts our assumption that the sequent is irreducible by all invertible rules; thus ξ is a conjunction of disequations. Furthermore, none of these disequations may be of the form $x \not\approx x$ because otherwise $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ would be an axiom. This entails that $\mathfrak{s} \models \xi$. Since $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ is valid, $\mathfrak{s}(V) \cap \mathit{dom}(\mathfrak{h}) = \emptyset$ and $\mathfrak{s}$ is injective (hence injective on V because V contains at most one occurrence of each variable), we deduce that

$$(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \psi \curlywedge \zeta. \qquad (\dagger)$$

Assume that ζ contains an equation $x \approx y$. If $x = y$ then Rule E applies, and otherwise, we necessarily have $\mathfrak{s}(x) \neq \mathfrak{s}(y)$, hence $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \psi \curlywedge \zeta$ which contradicts (†). Thus ζ contains no equation. Now assume that ζ contains a disequation $x \not\approx y$. If $\{x, y\} \subseteq \mathcal{C}$ (i.e., both x and y are constants) or $\{x, y\} \subseteq_m \mathit{alloc}(\phi) \cup V$ then $\phi \curlywedge \xi \rhd_V x \not\approx y$, and E applies, which is impossible by hypothesis. If $x = y$ then $x \not\approx y$ is unsatisfiable, which contradicts (†). Let $\mathfrak{s}'$ be a store verifying $\mathfrak{s}'(x) = \mathfrak{s}'(y)$, that maps all other variables to pairwise distinct elements of the appropriate sort, also distinct from $\mathfrak{s}(x)$, and such that $U' = \mathfrak{U}_{\mathtt{loc}} \setminus \mathfrak{s}'(\mathcal{V})$ is infinite. Such a store exists since $\mathfrak{U}_{\mathtt{loc}}$ is infinite and $\{x, y\} \not\subseteq \mathcal{C}$. If $\mathfrak{s}' \not\models \xi$, then since ξ contains no equation, $x \not\approx y$ must occur in ξ. Then the rule E applies, which contradicts our assumption that the sequent is irreducible by all invertible rules. Thus $\mathfrak{s}' \models \xi$. Since $\{x, y\} \not\subseteq_m \mathit{alloc}(\phi)$, $\mathfrak{s}'$ is injective on $\mathit{alloc}(\phi)$. By Lemma 4.26, we deduce that ϕ admits a model of the form $(\mathfrak{s}', \mathfrak{h}')$ with $\mathit{dom}(\mathfrak{h}') \subseteq U' \cup \mathfrak{s}'(\mathit{alloc}(\phi))$. If $\mathfrak{s}'(V) \cap \mathit{dom}(\mathfrak{h}')$ contains a location ℓ, then $\ell = \mathfrak{s}'(z) = \mathfrak{s}'(z')$, for some $z \in \mathit{alloc}(\phi)$ and $z' \in V$. Since $\mathit{alloc}(\phi) \cap V = \emptyset$ and $\mathfrak{s}'$ is injective on all pair of variables except on $\{x, y\}$, necessarily we have either $x = z$ and $y = z'$, or $x = z'$ and $y = z$, which means that $\{x, y\} \subseteq_m \mathit{alloc}(\phi) \cup V$, hence E applies, contradicting our assumption that the sequent is irreducible by all invertible rules. We deduce that $\mathfrak{s}'(V) \cap \mathit{dom}(\mathfrak{h}') = \emptyset$. Since $\{x, y\} \not\subseteq_m V$, $\mathfrak{s}'$ is necessarily injective on V. As $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ is valid, we deduce that $(\mathfrak{s}', \mathfrak{h}') \models_{\mathfrak{R}} \psi \curlywedge \zeta$ hence $\mathfrak{s}'(x) \neq \mathfrak{s}'(y)$, which contradicts the definition of $\mathfrak{s}'$. Therefore, $\zeta = \top$.

If $\psi = \mathit{emp}$ then necessarily $\phi = \mathit{emp}$, since otherwise by Corollary 4.4 we would have $\mathfrak{h} \neq \emptyset$, which contradicts (†). Hence in this case, $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ is of the form $\mathit{emp} \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \mathit{emp}$ and is an axiom.

If ψ is of the form $\psi_1 * \psi_2$, where $\psi_i \neq \mathit{emp}$ for $i = 1, 2$, then the proof follows from Lemma 5.20. Thus, we may assume that ψ is a spatial atom. Let x be the root of ψ. If $x \not\in \mathit{roots}(\phi)$, then we have $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathfrak{R}} \psi$ by Lemma 4.2, which contradicts (†). Therefore, ϕ contains a spatial atom with root x. If this spatial atom is a predicate atom then $\mathtt{U}$ can be applied. Now assume that ϕ contains a points-to atom $x \mapsto (y_1, \ldots, y_n)$, i.e., is of the form $x \mapsto (y_1, \ldots, y_n) * \phi'$ (modulo AC). Since ψ is a spatial atom with root x, there are two cases to consider:

1. ψ is of the form $x \mapsto (y'_1, \ldots, y'_n)$. We have $\mathfrak{h} = \{(\mathfrak{s}(x), \mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_n))\} \uplus \mathfrak{h}'$, where $\mathfrak{h}'$ is a heap such that $(\mathfrak{s}, \mathfrak{h}') \models_{\mathfrak{R}} \phi'$ and $\mathfrak{s}(x) \not\in \mathit{dom}(\mathfrak{h}')$. By (†), we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \psi$, thus $\mathfrak{h} = \{(\mathfrak{s}(x), \mathfrak{s}(y'_1), \ldots, \mathfrak{s}(y'_n))\}$. This entails that $\mathfrak{h}' = \emptyset$ (hence $\phi' = \mathit{emp}$ by Corollary 4.4), and that $\mathfrak{s}(y_i) = \mathfrak{s}(y'_i)$ for all $i = 1, \ldots, n$. Since $\mathfrak{s}$ is injective, we have $y_i = y'_i$ for all $i = 1, \ldots, n$. Therefore, $\psi = \phi$, and $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ is an axiom (because $\zeta = \top$), which contradicts the hypothesis of the lemma.

2. ψ is a predicate atom. By (†), we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \psi$ and by definition there exists a formula γ' such that $\psi \Leftarrow_{\mathfrak{R}} \gamma'$, where $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \gamma'$ for some associate $\mathfrak{s}'$ of $\mathfrak{s}$ w.r.t. $\mathcal{V}(\gamma') \setminus \mathcal{V}(\psi)$, and in particular we have $\mathfrak{s}(x) = \mathfrak{s}'(x)$. As the rules in $\mathfrak{R}$ are $\mathtt{P}$-rules, γ' is of the form $(x \mapsto (u_1, \ldots, u_m) * \psi') \curlywedge \zeta'$, with $\mathcal{V}(\gamma') \subseteq \mathcal{V}(\psi) \cup \{u_1, \ldots, u_m\}$, and we have $\mathfrak{h}(\mathfrak{s}(x)) = \mathfrak{h}(\mathfrak{s}'(x)) = (\mathfrak{s}'(u_1), \ldots, \mathfrak{s}'(u_m))$. Since ϕ contains a points-to atom $x \mapsto (y_1, \ldots, y_n)$ and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{R}} \phi$, necessarily $\mathfrak{h}(\mathfrak{s}(x)) = (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_n))$. Consequently $n = m$ and $\mathfrak{s}(y_i) = \mathfrak{s}'(u_i)$ for all $i = 1, \ldots, n$. Let σ be the substitution mapping every variable u_i not occurring in $\mathcal{V}(\psi)$ to y_i. Note that σ is well-defined: if $u_i = u_j$ then $\mathfrak{s}(y_i) = \mathfrak{s}(y_j)$ and $y_i = y_j$ since $\mathfrak{s}$ is injective. It is straightforward to check that $\mathfrak{s}'(z) = \mathfrak{s}(z\sigma)$ for all variables $z \in \mathcal{V}$, using the definition of σ and the fact that $\mathfrak{s}'$ and $\mathfrak{s}$ coincide on all variables not occurring in $u_1, \ldots, u_n$. If $\mathtt{I}$ applies with this substitution σ then the proof is completed. Otherwise, we must have $\phi \curlywedge \xi \not\triangleright_V \zeta'\sigma$, i.e., ζ' must contain a disequation $u' \not\approx v'$ such that $\phi \curlywedge \xi \not\triangleright_V u'\sigma \not\approx v'\sigma$. Note that this entails that $u'\sigma$ and $v'\sigma$ cannot both be constants. Consider a store $\mathfrak{s}''$ verifying $\mathfrak{s}''(u'\sigma) = \mathfrak{s}''(v'\sigma)$, and mapping all other variables to pairwise distinct elements of the appropriate sort. We may assume that $\mathfrak{s}''$ is chosen in such a way that the set $U' = \mathfrak{U} \setminus \mathfrak{s}''(\mathcal{V})$ is infinite. By definition, $\mathfrak{s}'' \not\models \zeta'\sigma$, and since $\phi \curlywedge \xi \not\triangleright_V u'\sigma \not\approx v'\sigma$, we have $\{u'\sigma, v'\sigma\} \not\subseteq_m V \cup \mathit{alloc}(\phi)$, thus $\mathfrak{s}''$ is injective on $\mathit{alloc}(\phi) \cup V$ and $\mathfrak{s}'' \models \xi$ (since ξ is a conjunction of disequations not containing $u'\sigma \not\approx v'\sigma$). By Lemma 4.26, ϕ admits a model $(\mathfrak{s}'', \mathfrak{h}'')$ such that $\mathit{dom}(\mathfrak{h}'') \subseteq U'' \cup \mathfrak{s}''(\mathit{alloc}(\phi))$. This entails that $\mathfrak{s}''(V) \cap \mathit{dom}(\mathfrak{h}'') = \emptyset$. Since $\mathfrak{s}''$ is injective on V and $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ is valid, we deduce that $(\mathfrak{s}'', \mathfrak{h}'') \models_{\mathfrak{R}} \psi$, thus $\psi \Leftarrow_{\mathfrak{R}} \gamma''$, where $(\hat{\mathfrak{s}}, \mathfrak{h}'') \models_{\mathfrak{R}} \gamma''$, for some formula γ'' and associate $\hat{\mathfrak{s}}$ of $\mathfrak{s}''$ w.r.t. $\mathcal{V}(\gamma'') \setminus \mathcal{V}(\psi)$. Since the rules in $\mathfrak{R}$ are $\mathtt{P}$-rules, γ'' is of the form $(x \mapsto (u'_1, \ldots, u'_k) * \psi'') \curlywedge \zeta''$, and we have $\mathfrak{h}''(\mathfrak{s}''(x)) = (\hat{\mathfrak{s}}(u'_1), \ldots, \hat{\mathfrak{s}}(u'_k))$. We may assume, by renaming, that $\{u'_1, \ldots, u'_k\} \cap \{u_1, \ldots, u_m\} \subseteq \mathcal{V}(\psi)$. Since ϕ contains a points-to atom $x \mapsto (y_1, \ldots, y_n)$ and $(\mathfrak{s}'', \mathfrak{h}'') \models_{\mathfrak{R}} \phi$, necessarily $\mathfrak{h}''(\mathfrak{s}''(x)) = (\mathfrak{s}''(y_1), \ldots, \mathfrak{s}''(y_n))$, and thus we must have $k = n = m$ and for all $i = 1, \ldots, n$, $\mathfrak{s}''(y_i) = \hat{\mathfrak{s}}(u'_i)$. Let σ' be the substitution mapping every variable u'_i not occurring in $\mathcal{V}(\psi)$ to the first variable y_j in $y_1, \ldots, y_n$ such that $\mathfrak{s}''(y_j) = \hat{\mathfrak{s}}(u'_i)$, and let θ be the substitution mapping every variable y_i to y_j, where j is the smallest index in $1, \ldots, n$ such that $\mathfrak{s}'' \models y_i \approx y_j$. Let $\hat{\sigma} = \sigma \cup \sigma'$ and $\theta' = \hat{\sigma}\theta$; note that $\hat{\sigma}$ is well-defined since $\mathit{dom}(\sigma) \cap \mathit{dom}(\sigma') = \emptyset$, as $\{u'_1, \ldots, u'_k\} \cap \{u_1, \ldots, u_m\} \subseteq \mathcal{V}(\psi)$. By

construction, θ' is a unifier of $(u_1, \ldots, u_n)$ and $(u'_1, \ldots, u'_n)$. Since $\mathfrak{R}$ is $\mathtt{loc}$-deterministic, one of the two formulas ζ' or ζ'' contains a disequation $v \not\approx w$ with $v\theta' = w\theta'$ and $v, w \in \mathcal{V}_{\mathtt{loc}}$. Note that $v\hat{\sigma} \neq w\hat{\sigma}$ since otherwise $\zeta'\sigma$ or $\zeta''\sigma'$ would be unsatisfiable. We show that $v \not\approx w$ occurs in ζ'. Assume, for the sake of contradiction, that $v \not\approx w$ occurs in ζ''. Then $v\hat{\sigma} = v\sigma'$ and $w\hat{\sigma} = w\sigma'$. By definition of σ' and θ', we have $z\sigma' = z\theta'$ for all $z \in dom(\sigma')$, thus we get $v\sigma' = w\sigma'$, hence (by definition of σ') $\hat{\mathfrak{s}}(v) = \hat{\mathfrak{s}}(w)$, which contradicts the fact that $(\hat{\mathfrak{s}}, \mathfrak{h}'') \models_{\mathfrak{R}} \gamma'' = (x \mapsto (u'_1, \ldots, u'_k) * \psi'') \curlywedge \zeta''$. Thus necessarily $v \not\approx w$ occurs in ζ' and we have $v\hat{\sigma} = v\sigma$ and $w\hat{\sigma} = w\sigma$. As all rules are $\mathtt{P}$-rules, necessarily at least one of the two variables v, w (say v) is an existential variable, and we must have $v \in alloc(\psi')$. Since $(\mathfrak{s}', \mathfrak{h}) \models_{\mathfrak{R}} \gamma' = (x \mapsto (u_1, \ldots, u_m) * \psi') \curlywedge \zeta'$, we deduce by Lemma 4.2 that $\mathfrak{s}'(v) \in dom(\mathfrak{h})$, hence $\mathfrak{s}(v\sigma) \in dom(\mathfrak{h})$. By definition of $(\mathfrak{s}, \mathfrak{h})$, this entails that $\mathfrak{s}(v\sigma) \in \mathfrak{s}(alloc(\phi))$, and since $\mathfrak{s}$ is injective we get $v\sigma \in alloc(\phi)$.

Assume that $w\sigma \in alloc(\phi)$. Then $alloc(\phi)$ contains spatial atoms with respective roots $w\sigma$ and $v\sigma$. These atoms must be distinct since $v\sigma \neq w\sigma$. By Lemma 4.2 these locations are allocated in disjoint part of the heap $\mathfrak{h}''$, hence necessarily $\mathfrak{s}''(w\sigma) \neq \mathfrak{s}''(v\sigma)$. We deduce that $v\theta' \neq w\theta'$, which contradicts our assumption. Thus $w\sigma \notin alloc(\phi)$, and since $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ cannot be an anti-axiom, necessarily $w\sigma \in \mathcal{V}(\psi)$. We also have $w\sigma \notin V$, since $dom(\mathfrak{h}'') \cap \mathfrak{s}''(V) = \emptyset$. We may assume that the sequent $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$ is irreducible by the rule C, so that, as ψ is a predicate atom, ξ contains a disequation $v\sigma \not\approx w\sigma$ (since $v\sigma \in alloc(\phi)$, $w\sigma \in \mathcal{V}(\psi) \setminus (alloc(\phi) \cup V)$). But then we have $\mathfrak{s}'' \not\models \xi$, contradicting the definition of $\mathfrak{s}''$.

□

The completeness result follows immediately:

Theorem 5.22. If $\mathcal{S}$ is a valid sequent, then it admits a fully expanded (possibly infinite) proof tree.

Proof:
The theorem follows from Lemma 5.21. □

5.3. Termination and Complexity

We now show that the proof procedure runs in polynomial time. To this purpose we provide (Definition 5.23 and Lemma 5.26) a characterization of the I-reducible sequents that may occur in a proof tree.

Definition 5.23. Let $\kappa \in \mathbb{N}$. A sequent $\mathcal{S}$ is a *κ-companion* of a symbolic heap $\phi \curlywedge \xi$ if $\mathcal{S}$ is of the form $(\phi_1\sigma * \phi_2 * \phi_3) \curlywedge (\xi_0 \wedge \xi_1 \wedge \xi_2) \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \top$, where:

1. ξ_0 is the set of disequations occurring in ξ that are of the form $x \not\approx t$ (up to commutativity), with $x \in (\mathcal{V}(\phi_1\sigma * \phi_2 * \phi_3) \cup \mathcal{V}(\psi)) \setminus \mathcal{V}_{\mathtt{loc}}$ and $t \neq x$.

2. $\xi_1 = \bigwedge_{x \in \mathcal{V}^{\star}(\mathcal{S}), y \in A} x \not\approx y$, where $A = alloc(\phi_1\sigma * \phi_2 * \phi_3) \setminus \mathcal{V}(\psi)$.

3. ξ_2 is a conjunction of disequations of the form $u \not\approx v$, where $u, v \in \mathcal{V}(\psi)$.

4. ψ is a predicate atom.

5. $V \subseteq \mathcal{V}(\psi)$.

6. Either $\phi_1 = emp$, or there exists a predicate atom α such that $root(\alpha) = root(\psi)$ and $\alpha \Leftarrow_{\mathfrak{R}} \phi_1$.

7. $len(\phi_2) \leq \kappa + 1$.

8. σ is a substitution such that $card(dom(\sigma)) \leq \kappa$.

9. $\phi\sigma$ is of the form $\phi_3 * \eta_1 * \eta_2$ (modulo AC), where $len(\eta_1) \leq 2 \cdot \kappa$ and:

   (a) no atom β occurring in ϕ_3 is such that $root(\psi) \not\to^*_{\phi_1\sigma*\phi_2*\phi_3*\eta_2} root(\beta)$; and

   (b) no atom β occurring in η_2 is such that $root(\psi) \to^*_{\phi_1\sigma*\phi_2*\phi_3*\eta_2} root(\beta)$.

Intuitively, we aim at describing sequents $\mathcal{S}$ that are `I`-reducible and occur in some proof tree, where the formula $\phi \curlywedge \xi$ is meant to denote the left-hand side of the root sequent. It is clear that the left-hand side of $\mathcal{S}$ will contain some parts that are inherited from the formula $\phi \curlywedge \xi$, together with some other parts, that are introduced by the inference rules applied along the branch of $\mathcal{S}$ in the proof tree. First, observe that Conditions 4 (stating that the right-hand side of $\mathcal{S}$ is a predicate atom ψ) and 5 (stating that the variables in V must occur in ψ) are easy consequences of the fact that $\mathcal{S}$ is `I`-reducible: rules `S` and `V` cannot be applied. The substitution σ is intended to encode the effect of the (cumulative) applications of rule `C` (in the left branch): some variables initially occurring in $\phi \curlywedge \xi$ are replaced by other variables. Note that Condition 8 bounds the size of this substitution, which is essential to avoid any exponential blow-up. As we shall see, it stems from the fact that rule `C` only applies, by Definition 4.31 (4), to a variable in $\mathcal{V}^\star(\mathcal{S})$ which restricts the number of applications of the rule. The formula ξ_0 denotes the set of disequations between variables of a sort distinct from `loc`, which are all inherited from ξ (as the rules contain no disequations of this form). On the other hand, ξ_1 denotes the set of disequations that were introduced by previous applications of `C` (in the right branch). Note that since $\mathcal{S}$ is `I`-reducible, we may assume that all the possible applications of `C` have been applied, leading either to the elimination of a variable (encoded in σ) or to the addition of some disequation, this is why ξ_1 contains the set of *all* possible disequations. The formula ξ_2 denotes the set of disequations between variables in ψ, which are arbitrary (this is not problematic as the number of such disequations is bounded, as ψ is an atom). The formula ϕ_1 denotes the part of the formula added in the left-hand side of the sequent by the *last* application of `U`. The formula may be empty, because such application does not always exist (for instance if $\mathcal{S}$ is the root sequent). The atom α then denotes the predicate atom on which `U` was previously applied. Observe that the root of this atom is always the same as that of ψ, by Definition 4.31 (3). Note also that the formula contains no atom obtained from any application of `U` occurring *before* the last one (i.e., on atoms other than α): indeed, we shall prove that all such atoms must have been eliminated by previous decomposition steps when `I` was applied. The formula $\phi_1\sigma$ contains all spatial atoms inherited from the initial formula ϕ. Note that some atoms in ϕ may have been deleted, by previous applications of `S`. The difficulty here is that this could lead to an exponential blow-up, since in principle one could consider all possible subsets of ϕ. To overcome this issue, we refine the criterion by taking into account all the additional reachability conditions that prevent $\mathcal{S}$ from being an axiom or an anti-axiom. We thus further split the conjunction of atoms that were deleted from ϕ into two parts $\eta_1 * \eta_2$, where η_1 contains a *bounded* number of atoms that can be chosen in an *arbitrary* way, and η_2 may contain an *arbitrary* number of atoms, but is fully determined by the choice of ϕ_1, ϕ_2 and η_1.

The following lemma shows that a formula has only polynomially many κ-companions, up to a renaming of variables.

Lemma 5.24. Let $\phi \curlywedge \xi$ be a spatial formula and E be a set of variables. For every fixed number κ, if $ar_{max}(\mathfrak{R}) \leq \kappa$ (i.e., if the maximal arity of the predicate symbols in bounded by κ), then the number of sequents $\mathcal{S}$ that are κ-companions of $\phi \curlywedge \xi$ and such that $\mathcal{V}(\mathcal{S}) \subseteq E$ is polynomial w.r.t. $|\phi| + |\mathfrak{R}| + card(E)$ (up to AC), and the size of $\mathcal{S}$ is polynomial w.r.t. $|\phi| + |\mathfrak{R}|$.

Proof:
By Definition 5.23, $\mathcal{S}$ is of the form $(\phi_1\sigma * \phi_2 * \phi_3) \curlywedge (\xi_1 \wedge \xi_2) \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \top$, and satisfies Conditions 1-9. Note that since ψ is a predicate atom, we have $|\psi| \leq 1 + ar_{max}(\mathfrak{R}) \leq \kappa + 1$ and $card(\mathcal{V}(\psi)) \leq ar_{max}(\mathfrak{R}) \leq \kappa$.

We have $len(\phi_1 * \phi_2 * \phi_3) = len(\phi_1) + len(\phi_2) + len(\phi_3) \leq len(\phi_1) + \kappa + 1 + len(\phi) \leq |\mathfrak{R}| + |\phi| + \kappa + 1$, thus $|\phi_1\sigma * \phi_2 * \phi_3| \leq (\kappa + 1) \cdot (|\mathfrak{R}| + |\phi| + \kappa + 1) = O(|\phi| + |\mathfrak{R}|)$. Since $\mathcal{V}(\zeta_1) \cup \mathcal{V}(\zeta_2) \subseteq \mathcal{V}(\phi_1\sigma * \phi_2 * \phi_3) \cup \mathcal{V}(\psi)$, we deduce that $|\xi_1 \wedge \xi_2| = O((|\phi| + |\mathfrak{R}|)^2)$ hence $|\mathcal{S}| = O((|\phi| + |\mathfrak{R}|)^2)$.

Let $n = card(E) + |\phi| + card(\mathcal{C}) + 1$. Note that $n \leq card(E) + |\phi| + |\mathfrak{R}| + 1$. The number of possible predicate atoms ψ and α is at most $card(\mathcal{P}) \cdot n^{\kappa} \leq |\mathfrak{R}| \cdot n^{\kappa}$. Similarly, the number of formulas ϕ_2 is at most $|\mathfrak{R}|^{\kappa+1} \cdot n^{(\kappa+1)^2}$. Once α is fixed, the number of formulas ϕ_1 cannot be greater than the number of rules in $\mathfrak{R}$, up to a renaming of variables. Since $card(dom(\sigma)) \leq \kappa$, there are at most $n^{2\cdot\kappa}$ possible substitutions σ we only have to choose the set $dom(\sigma)$, i.e., at most κ elements among the variables occurring in ϕ or ϕ_1, and the image of each variable in $dom(\sigma)$ among the variables in E. The number of formulas ϕ_3 is at most $len(\phi)^{2\cdot\kappa}$ (since we only have to choose the formula η_1 in ϕ, as η_2 is entirely determined by the choice of ϕ_1, ϕ_2 and η_1). We have $\mathcal{V}(\xi_2) \subseteq \mathcal{V}(\psi)$ thus $card(\mathcal{V}(\xi_2)) \leq \kappa$, and there are 2^{κ^2} possible formulas ξ_2. Since $V \subseteq \mathcal{V}(\psi)$, we have $card(V) \leq \kappa$ and the number of possible sets V is 2^{κ}. Finally, the formulas ξ_0 and ξ_1 are fixed once ϕ and ϕ_1, ϕ_2, ϕ_3 and σ are fixed. □

Lemma 5.26 below is used to prove that every sequent that is $\mathtt{I}$-reducible and occurs in a proof tree is a companion of the left-hand side of the root of the proof tree. Together with Lemma 5.24 and the result stated in Lemma 5.12 on the number of $\mathtt{I}$-free descendants of a sequent, this will entail that the size of the proof tree is polynomial w.r.t. the size of the root. We need the following proposition, which is an easy consequence of the definition of the rules.

Proposition 5.25. Let $\mathcal{S}_i = \lambda_i \vdash_{\mathfrak{R}}^{V_i} \gamma_i$ (for $i = 1, 2$) be sequents. If $\mathcal{S}_2$ is a successor of $\mathcal{S}_1$, then $alloc(\lambda_1) \cap alloc(\gamma_1) \cap alloc(\lambda_2) \subseteq alloc(\gamma_2)$.

Proof:
Assume that $x \in alloc(\lambda_1) \cap alloc(\gamma_1) \cap alloc(\lambda_2)$ and $x \notin alloc(\gamma_2)$. It is easy to check, by inspection

of the rules, that the only rule that can remove the variable x from $alloc(\gamma_1)$ is S (indeed, I deletes a predicate atom but introduce a points-to atom with the same root, and the rules R and C cannot replace x as otherwise it would not occur in λ_2). By definition of the rule, $\mathcal{S}_1$ also has a successor $\mathcal{S}_2' = \lambda_2' \vdash_{\mathfrak{R}}^{V_2'} \gamma_2'$ such that $x \in V_2'$ (since $x \in alloc(\lambda_2)$) and $x \in alloc(\gamma_2')$ (since $x \notin alloc(\gamma_2)$). This entails that $\mathcal{S}_2'$ is an anti-axiom, contradicting Condition 1 in Definition 4.31. □

Lemma 5.26. Assume that $ar_{max}(\mathfrak{R}) \leq \kappa$ for $\kappa \in \mathbb{N}$, and let $\mathcal{S} = \phi \curlywedge \xi \vdash \psi \curlywedge \zeta$ be an equality-free sequent. Every I-reducible sequent occurring in a proof tree with root $\mathcal{S}$ is a κ-companion of the left-hand side $\phi \curlywedge \xi$ of $\mathcal{S}$.

Proof:
Let $\mathcal{S}' = \phi' \curlywedge \xi' \vdash_{\mathfrak{R}}^{V'} \psi' \curlywedge \zeta'$ be an I-reducible sequent occurring in a proof tree with root $\mathcal{S}$. Let $\mathcal{S}_1, \dots, \mathcal{S}_n$, be a path from $\mathcal{S}$ to $\mathcal{S}'$, where $\mathcal{S}_1 = \mathcal{S}$, $\mathcal{S}_n = \mathcal{S}'$ and $\mathcal{S}_i = \lambda_i \vdash_{\mathfrak{R}}^{V_i} \gamma_i$ for $i = 1, \dots, n$. We check that all the conditions in Definition 5.23 are satisfied. Note that by Condition 1 in Definition 4.31, $\mathcal{S}'$ cannot be an axiom or an anti-axiom.

- **Right-Hand Side.** We first show that the right-hand side ψ' is indeed a predicate atom. By Condition 2 in Definition 4.31, $\zeta' = \top$ and ψ' is a predicate atom, thus Condition 4 of Definition 5.23 holds.

- **Pure Formulas.** We then prove that the pure formulas ξ' fulfill Conditions 1, 2 and 3. If ξ' contains an equality then the rule R applies, hence $\mathcal{S}'$ cannot be I-reducible because R has priority over I (Condition 6 in Definition 4.31). Thus ξ' is a conjunction of disequations, and it can be written as $\xi_0 \curlywedge \xi_1 \wedge \xi_2$ where ξ_0 is the set of disequations in ξ' between terms of a sort distinct from `loc`, $\mathcal{V}(\xi_2) \subseteq \mathcal{V}(\psi') \cap \mathcal{V}_{\texttt{loc}}$ and ξ_1 only contains disequations of the form $u \not\approx v$ (up to commutativity of $\not\approx$) with $u, v \in \mathcal{V}_{\texttt{loc}}$ and $u \notin \mathcal{V}(\psi')$. Note that each disequation involves distinct terms because by hypothesis, $\mathcal{S}'$ is not an axiom. Consider one such disequation $u \not\approx v$ in ξ_1. If $u \notin \mathcal{V}(\phi')$ then rule W applies by the second application condition of the rule, which is impossible since this rule has priority over I and $\mathcal{S}'$ would not be I-reducible. We deduce that $u \in \mathcal{V}(\phi')$, and since $\mathcal{S}'$ is not an anti-axiom, we must have $u \in alloc(\phi')$. If $v \notin \mathcal{V}(\phi') \cup \mathcal{V}(\psi')$ or $v \in alloc(\phi') \cup V'$ then rule W also applies (in the latter case, we have $\phi' \rhd_{V_i} u \not\approx v$ by Condition 3 of Definition 4.8). Thus $v \in \mathcal{V}(\phi') \cup \mathcal{V}(\psi')$ and $v \notin alloc(\phi') \cup V$. This entails that $v \in \mathcal{V}(\psi')$ since otherwise $\mathcal{S}'$ would be an anti-axiom by Condition 5 of Definition 4.24 because $u \in alloc(\phi')$, hence v is necessarily of sort `loc`. We therefore have $v \in \mathcal{V}^\star(\mathcal{S}')$ and $u \in alloc(\phi') \setminus \mathcal{V}(\psi')$. Conversely, if a disequation $u \not\approx v$ with $v \in \mathcal{V}^\star(\mathcal{S}')$ and $u \in alloc(\phi') \setminus \mathcal{V}(\psi')$ does not occur in ξ_1 then by Condition 4 in Definition 4.31, C would be applicable, which is impossible. Thus Conditions 2 and 3 in Definition 5.23 are satisfied. By Proposition 4.17 (5) no rule introduces a disequation between terms of a sort distinct from `loc`, all disequations in ξ_0 must occur in ξ. Furthermore, such a disequation cannot be of the form $t \not\approx t$ (otherwise $\mathcal{S}'$ would be an axiom as explained above) and must contain a variable occurring in ϕ' or ψ' (otherwise W would apply). Conversely, no rule can delete a disequation containing a variable in $(\mathcal{V}(\phi') \cup \mathcal{V}(\psi')) \setminus \mathcal{V}_{\texttt{loc}}$ from the left-hand side of the sequent. Thus Condition 1 holds.

- **Non Allocated Variables.** We now check that Condition 5 applies, namely that $V' \subseteq \mathcal{V}(\psi')$. Assume, for the sake of contradiction, that V' contains a variable $x \notin \mathcal{V}(\psi')$. Since $\mathcal{S}'$ is not an anti-axiom, we have $V' \cap (\mathcal{V}(\phi') \setminus \mathcal{V}(\psi')) = \emptyset$, hence $x \notin \mathcal{V}(\phi')$. If $x \in \mathcal{V}(\xi')$ then W applies, and otherwise V applies. In both cases, $\mathcal{S}'$ cannot be I-reducible (since W and V have priority over I), which contradicts our assumption. Thus Condition 5 holds.

- **Substitution σ.** We now define the substitution σ, and we show that Condition 8 holds. Let i be the least number such that the right-hand side γ_i of $\mathcal{S}_i$ is a spatial atom (i must exist since n fulfills this condition, by Condition 2 in Definition 4.31), and let $V^\star = \mathcal{V}^\star(\mathcal{S}_i)$. Note that γ_i must be a predicate atom. Indeed, the right-hand side of $\gamma_n = \lambda_n \vdash_{\mathfrak{R}}^{V_n} \gamma_n$ is a predicate atom since $\mathcal{S}_n$ is I-reducible, and by Condition 6 of Proposition 4.17, the only rule that can introduce a predicate atom to the right-hand side of a premise is I, which by Condition 2 in Definition 4.31, applies only if the right-hand side of the conclusion is a predicate atom. Thus γ_j cannot be a points-to atom. By Proposition 5.4 we have $\mathcal{V}^\star(\mathcal{S}_j) \subseteq V^\star$, for all $j = i, \dots, n$, and $card(V^\star) \leq ar_{max}(\mathfrak{R}) \leq \kappa$ since $V^\star \subseteq \mathcal{V}(\mathcal{S}_i)$ and γ_i is a predicate atom. Let $I \subseteq \{1, \dots, n\}$ be the set of indices j such that rule C applies on $\mathcal{S}_j$, and replaces a variable x_j by a variable y_j (i.e., such that $\mathcal{S}_{j+1}$ is a left premise of an application of C on $\mathcal{S}_j$ with variables x_j and y_j). By Condition 4 of Definition 4.31, C applies only if the right-hand side of the conclusion is a predicate atom, thus we must have $I \subseteq \{i, \dots, n\}$. Let $\sigma = \sigma_n$, where $\sigma_0 = id$, $\sigma_{j+1} = \sigma_j\{x_{j+1} \leftarrow y_{j+1}\}$ if $j + 1 \in I$ and $\sigma_{j+1} = \sigma_j$ otherwise. By Condition 4 in Definition 4.31, for all $j \in I$, we have $y_j \in \mathcal{V}^\star(\mathcal{S}_j) \subseteq \mathcal{V}^\star(\mathcal{S}_i) = V^\star$. Also, $y_j \notin \mathcal{V}^\star(\mathcal{S}_{j+1})$ because x_j must be allocated in the left-hand side of $\mathcal{S}_j$, hence, after the replacement, y_j is allocated in the left-hand side of $\mathcal{S}_{j+1}$. Thus by Proposition 5.4, $y_j \notin \mathcal{V}^\star(\mathcal{S}_{j'})$, for all $j' > j$, which entails that the y_j for $j \in I$ are pairwise distinct. Therefore, $card(I) \leq card(V^\star) \leq \kappa$, hence $card(dom(\sigma)) \leq \kappa$ and Condition 8 is satisfied.

- **Unfolded Atom.** Now we define the unfolded predicate atom α and we prove that Condition 6 holds. If rule U is applied on a sequent $\mathcal{S}_k$ and $\mathcal{S}'$ is an I-free descendant of $\mathcal{S}_k$, then we denote by α the predicate atom on which U is applied, and by $\phi_1 \curlywedge \chi$ the formula occurring in $\mathcal{S}_{k+1}$ such that $\alpha \Leftarrow_{\mathfrak{R}} \phi_1 \curlywedge \chi$. If no such application of U exists then we simply set $\phi_1 = emp$. Note that if k and α exist then they must be unique, since we assumed that $\mathcal{S}'$ is an I-free descendant of $\mathcal{S}_k$ and by Corollary 5.8 there must be an application of I along the path between any two applications of U. Also, by Condition 3 in Definition 4.31, γ_k is a spatial atom and $root(\alpha) = root(\gamma_k)$. By Proposition 5.6, the only rules that can be applied on $\mathcal{S}_{k+1}, \dots, \mathcal{S}_{n-1}$ are W, C, or V. Each of these rules only affects the right-hand side of its premise by instantiating it with σ thus $root(\alpha)\sigma = root(\psi')$. First consider the case where k exists. Since the rules in $\mathfrak{R}$ are P-rules, λ_{k+1} must contain a points-to atom. By Condition 3 in Definition 4.31, the root of this atom must be the same as that of $\gamma_k = \gamma_{k+1}$. Furthermore, λ_{k+1} contains no equation, since otherwise this equation would occur in λ_k, rule R would be applicable on $\mathcal{S}_k$ and this rule has priority over U. Therefore, $\mathcal{S}_{k+1}$ is quasi-I-reducible. By Proposition 5.6, this entails that only the rules W, V or C may be applied along the path $\mathcal{S}_{k+1}, \dots, \mathcal{S}_n$. These rules either do not affect spatial formulas, or uniformly replace a variable x_j (for $j \in I$) by $y_j = x_j\sigma_j$ (hence eventually, every variable x is replaced by $x\sigma$). Thus ϕ' is of the form $\phi_1\sigma * \phi'_1$. It is clear that the previous assertion trivially holds in the case where k does not exist by letting $\phi_1 = emp$. Thus Condition 6 holds.

- **Spatial Formula.** We now prove that the spatial formula ϕ'_1 is of the required form and fulfills Condition 7. The formula ϕ'_1 may be written as $\phi_2 * \phi_3 * \phi_4$, where:

  - $alloc(\phi_2) \subseteq (V^\star \cup alloc(\psi'))$,
  - $alloc(\phi_3) \subseteq \mathcal{V}(\phi) \setminus (V^\star \cup alloc(\psi'))$,
  - $alloc(\phi_4) \cap (\mathcal{V}(\phi) \cup V^\star \cup alloc(\psi')) = \emptyset$.

  If ϕ_2 contains two atoms with the same root then ϕ' is heap-unsatisfiable and $\mathcal{S}'$ is an axiom. Thus $len(\phi_2) \leq card(V^\star) + 1 \leq \kappa + 1$ and Condition 7 holds.

  The only rule that can add new predicate symbols to a premise is U, replacing an atom α' by a formula λ such that $\alpha' \Leftarrow_{\mathfrak{R}} \lambda$. Since the rules in $\mathfrak{R}$ are P-rules, the root of every predicate atom in λ must be a fresh variable y, not occurring in $\mathcal{V}(\phi)$. If $y \in dom(\sigma)$ then y is eventually replaced by some variable in $V^\star$, hence the corresponding predicate atom from ϕ'_1 is in ϕ_2. If $y \notin dom(\sigma)$, then since this variable is never replaced, the corresponding atom must occur in ϕ_4 by construction. This entails that all the atoms in ϕ_3 must be obtained from atoms in ϕ after application of the substitution σ, i.e., that $\phi\sigma = \phi_3 * \eta$ for some formula η, up to AC. The formula η can be written as $\eta_1 * \eta_2$ where $alloc(\eta_1) \subseteq \mathcal{V}(\psi') \cup V^\star$ and $alloc(\eta_2) \cap (\mathcal{V}(\psi') \cup V^\star) = \emptyset$. This entails that $len(\eta_1) \leq 2 \cdot \kappa$, since $card(\mathcal{V}(\psi')) \leq \kappa$ and $card(V^\star) \leq \kappa$.

  We prove that necessarily $\phi_4 = emp$. Assume for the sake of contradiction that ϕ_4 contains an atom α', and let $u = root(\alpha')$. By definition of ϕ_4, $u \notin \mathcal{V}(\phi) \cup V^\star \cup alloc(\psi')$. The variable u must have been introduced by an application of rule U on some sequent $\mathcal{S}_m$, as U is the only rule that can introduce new variables to the left-hand side of a sequent. By Proposition 5.7, $\mathcal{S}_{m+1}$ is quasi-I-reducible, and we must have $m \neq k$ (if k exists), since otherwise α' would occur in $\phi_1\sigma$. Since the rules in $\mathfrak{R}$ are P-rules, we have $u \in \mathcal{V}_{\mapsto}(\mathcal{S}_{m+1})$. Since $m \neq k$, by Corollary 5.8, there exists m' such that $m < m'$ and $\mathcal{S}_{m'}$ is I-reducible, i.e., rule I is applied on $\mathcal{S}_{m'}$. Let m' be the minimal number satisfying this property. The path $\mathcal{S}_{m+1}, \dots, \mathcal{S}_{m'}$ only contains applications of W, V and C, and $u \in \mathcal{V}_{\mapsto}(\mathcal{S}_{m'})$ by Proposition 5.6. Note that if k exists then necessarily $m' < k$, moreover, none of these rules can introduce new variables to the right-hand side of a sequent. In particular, variable u, which is a fresh variable that does not occur in γ_m, cannot occur on the right-hand side $\gamma_{m'}$ of $\mathcal{S}_{m'}$. Let θ denote the substitution used in the application of I on $\mathcal{S}_{m'}$. As $u \in \mathcal{V}_{\mapsto}(\mathcal{S}_{m'})$, there exists a variable $z \in dom(\theta)$ such that $z\theta = u$. Since the rules in $\mathfrak{R}$ are P-rules, this entails that $u \in alloc(\gamma_{m'+1})$. Consequently, $u \in alloc(\lambda_{m'+1}) \cap alloc(\gamma_{m'+1})$ and by (repeatedly) applying Proposition 5.25, we deduce that $u \in alloc(\psi')$ (since by hypothesis $u \in alloc(\phi_4) \subseteq alloc(\phi')$), which contradicts the definition of ϕ_4.

- **Reachability Conditions.** Finally, we show that Condition 9 holds, i.e., that η_2 contains exactly the atoms β in $\phi_3 * \eta_2$ such that $root(\psi') \not\rightarrow_{\phi_1\sigma * \phi_2 * \phi_3 * \eta_2} root(\beta)$. Assume that ϕ_3 contains an atom β such that $root(\psi') \not\rightarrow_{\phi_1\sigma * \phi_2 * \phi_3 * \eta_2} root(\beta)$. Since $\phi' = \phi_1\sigma * \phi_2 * \phi_3$, we get $root(\psi') \not\rightarrow_{\phi'} root(\beta)$, $root(\beta) \in alloc(\phi')$ and $root(\beta) \notin alloc(\psi')$ (because $root(\beta) \neq root(\psi')$ and $alloc(\psi') = \{root(\psi')\}$), thus $\mathcal{S}'$ is an anti-axiom. Conversely, assume that η_2

contains an atom β such that $root(\psi') \rightarrow_{\phi_1\sigma * \phi_2 * \phi_3 * \eta_2} root(\beta)$. Note that $root(\beta) \not\in \mathcal{V}(\psi') \cup V^\star$, by definition of η_2, which entails that $root(\beta) \not\in codom(\sigma)$. If $root(\beta) \not\in alloc(\phi')$, then by considering the first variable $y \not\in alloc(\phi') = alloc(\phi_1\sigma * \phi_2 * \phi_3)$ along the path from $root(\psi')$ to $root(\beta)$, we have $y \in \mathcal{V}(\phi')$, $y \not\in alloc(\phi')$ and $y \in alloc(\eta_2)$, hence $y \not\in \mathcal{V}(\psi')$. Thus $\mathcal{S}'$ is an anti-axiom, contradicting our hypothesis. Therefore, $root(\beta) \in alloc(\phi') = alloc(\phi_1\sigma) \cup alloc(\phi_2) \cup alloc(\phi_3)$. We distinguish the three cases:

- Assume that $root(\beta) \in alloc(\phi_1\sigma)$ i.e., that k exists, $\phi_1 \neq emp$ and $\alpha \Leftarrow_{\mathfrak{R}} \phi_1 \curlywedge \chi$. Since $root(\beta) \not\in codom(\sigma)$, we also have $root(\beta) \in alloc(\phi_1)$. Since the rules in $\mathfrak{R}$ are $\mathtt{P}$-rules, the variables in $alloc(\phi_1)$ are either the root of α or fresh variables. Now $root(\alpha)\sigma$ must be distinct from $root(\beta)$, since $root(\beta) \not\in \mathcal{V}(\psi')$ and $root(\alpha)\sigma = root(\psi')$. Furthermore, all fresh variables are necessarily distinct from $root(\beta)$. Indeed, they cannot occur in ϕ, by definition, and $root(\beta) \subseteq alloc(\phi)$, as $root(\beta) \subseteq alloc(\eta_2) \subseteq alloc(\eta) \subseteq alloc(\phi\sigma)$ and $root(\beta) \not\in codom(\sigma)$. Thus this case cannot occur.
- If $root(\beta) \in alloc(\phi_2)$, then $root(\beta) \in V^\star \cup alloc(\psi')$ (by definition of ϕ_2), which contradicts the fact that $root(\beta) \not\in \mathcal{V}(\psi') \cup V^\star$.
- If $root(\beta) \in alloc(\phi_3)$, then $root(\beta)$ occurs twice in $alloc(\phi\sigma)$ because $\phi\sigma = \phi_3 * \eta_1 * \eta_2$; hence, it also occurs twice in $alloc(\phi)$ because $root(\beta) \not\in codom(\sigma)$. This entails that ϕ is heap-unsatisfiable, a contradiction.

Thus we get a contradiction, which entails that Condition 9 in Definition 4.31 is satisfied.

□

We now introduce a restricted form of variable renaming[4]. The definition is useful to straightforwardly detect cycles in a proof tree, which is essential for efficiency. For each sort $\mathtt{s}$, we consider infinite (fixed) sequences of pairwise distinct variables of sort $\mathtt{s}$: $x_i^{\mathtt{s}}$, with $i \in \mathbb{N}$ (not occurring in the root sequent). A sequent $\mathcal{S}'$ occurring in a proof tree with root $\mathcal{S}$ is *normalized* if, for every variable y of sort $\mathtt{s}$ occurring in $\mathcal{V}(\mathcal{S}') \setminus \mathcal{V}(\mathcal{S})$, there exists $i \leq |\mathcal{S}'|$ such that $y = x_i^{\mathtt{s}}$. It is clear that every sequent $\mathcal{S}'$ can be reduced in polynomial time to a normalized sequent. Indeed, it suffices to rename all the variables $y_1, \dots, y_n$ of sorts $\mathtt{s}_1, \dots, \mathtt{s}_n$ occurring in $\mathcal{S}'$ but not in $\mathcal{S}$ by the variables $x_1^{\mathtt{s}_1}, \dots, x_n^{\mathtt{s}_n}$, respectively. As n cannot be greater than the size of $\mathcal{S}'$, the obtained sequent is normalized. Such an operation is called a *normalization*.

Lemma 5.27. Assume that $ar_{max}(\mathfrak{R}) \leq \kappa$, for some fixed $\kappa \in: \mathbb{N}$. For every sequent $\mathcal{S}$, the number of sequents occurring in a proot tree with root $\mathcal{S}$ is polynomial w.r.t. $|\mathcal{S}| + |\mathfrak{R}|$, up to normalization.

Proof:
By definition, every descendant of $\mathcal{S}$ is either an $\mathtt{I}$-free descendant of $\mathcal{S}$ or an $\mathtt{I}$-free descendant of some $\mathtt{I}$-reducible sequent $\mathcal{S}'$. By Condition 6 in Definition 4.31, $\mathtt{R}$ applies with the highest priority on $\mathcal{S}$, eventually yielding a (unique) sequent equality-free $\mathcal{S}''$ of the form $\phi \curlywedge \xi \vdash_{\mathfrak{R}}^{V} \psi \curlywedge \zeta$, with $|\mathcal{S}''| \leq |\mathcal{S}|$ and $\mathcal{V}(\phi) \subseteq \mathcal{V}(\mathcal{S})$. By Lemma 5.26, every $\mathtt{I}$-free sequent $\mathcal{S}'$ is a companion of of $\phi \curlywedge \xi$.

[4] Note that we cannot assume that sequents are taken modulo variable renaming, since identity would then be difficult to test (it would be equivalent to the well-known graph isomorphism problem).

By Lemma 5.24, the number of such sequents $\mathcal{S}'$, after normalization (w.r.t. $\mathcal{S}''$), is polynomial w.r.t. $|\phi| + |\mathfrak{R}|$ (hence w.r.t. $|\mathcal{S}| + |\mathfrak{R}|$). Indeed, it is clear that the normalized form of $\mathcal{S}'$ is also a companion of $\phi \curlywedge \xi$ and by definition it only contains variables in $\mathcal{V}(\mathcal{S}'') \cup \{x_i^{\mathtt{s}} \mid i \leq |\mathcal{S}'|\}$. Furthermore, $|\mathcal{S}'|$ is polynomial w.r.t. $|\mathcal{S}| + |\mathfrak{R}|$, thus the same property holds for $card(\mathcal{V}(\mathcal{S}'))$. Then the proof follows immediately from Lemma 5.12. □

Putting everything together, we derive the main result of the paper:

Theorem 5.28. The validity problem for sequents $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$, where $\mathfrak{R}$ is a `loc`-deterministic set of `P`-rules with $ar_{max}(\mathfrak{R}) \leq \kappa$ and $record_{max}(\mathfrak{R}) \leq \kappa$, for some fixed number κ, is in PTIME.

Proof:
By Theorems 5.13 and 5.22, $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$ is valid iff it admits a fully expanded proof tree. We first apply inductively the inference rules in all possible ways, starting with the initial sequent $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$ and proceeding with all the descendants of $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$, until we obtain either an axiom or a sequent that has already been considered, up to normalization. By Lemma 5.27, the number of such descendants (up to normalization) is polynomial w.r.t. $|\mathcal{S}| + |\mathfrak{R}|$, thus it suffices to show that each inference step can be performed in polynomial time. This can be established by an inspection of the rules. We first observe that the set of all possible rule applications can be computed in polynomial time w.r.t. the size of the conclusion (in particular, both the number of premises and their size are always polynomial w.r.t. the size of the conclusion): this is straightforward to check for all rules except for `S`, for which the result stems from Proposition 4.34. Also, it is straightforward to verify that the application conditions of the rules can be tested in polynomial time.

Afterwards, it suffices to compute the set of provable sequents among those that are reachable from the initial one and check whether $\lambda \vdash_{\mathfrak{R}}^{V} \gamma$ occurs in this set. Since infinite proof trees are allowed, we actually compute the complement of this set, i.e., the set of sequents that are *not* provable. This can be done in polynomial time, using a straightforward fixpoint algorithm, by computing the least set of sequents $\mathcal{S}$ such that, for all rule applications with conclusion $\mathcal{S}$, at least one of the premises is not provable. More precisely, a sequent is not provable if it has no successor (no rule can be applied on it), or more generally, if for all possible rule applications, at least one of the premises is not provable.

Note that, in order to apply rule `S` on a sequent that is not narrow, it is necessary to check that the left premise is valid, and this must be done by recursively applying the proof procedure. This is feasible because the left premise is always narrow (since its right-hand side is a predicate atom). Consequently, the computation of valid sequents must be performed in two steps, first for narrow sequents, then for those that are not narrow, as the application of the rules on non narrow sequents depends on the validity of narrow sequents. By Proposition 4.32 all the descendants of narrow sequents are also narrow, hence the converse does not hold, and the procedure is well-founded. □

6. Conclusion

A proof procedure has been devised for checking entailments between Separation Logic formulas with spatial predicates denoting recursive data structures, for a new class of inductive rules, called `loc`-deterministic. The soundness and completeness of the calculus have been established. The conditions imposed on the data structures are stronger than those considered in [13, 14] but on the other hand the procedure terminates in polynomial time, provided the arity of the predicates is bounded. Both the considered fragment and the proof procedure are very different from previously known polynomial-time decision procedures for testing entailments [7, 5]. In particular, the considered formulas may contain several non-built-in and non-compositional predicates (in the sense of [11]), and the structures may contain unallocated nodes, denoting data (with no predicate other than equality). In addition, several lower bounds have been established for the entailment problem, showing that any relaxing of the proposed conditions makes the problem untractable.

Future work includes the implementation of the procedure and the evaluation of its practical performance. The combination with theory reasoning (to reason on data stored inside the structures) will also be considered. In particular, it would be interesting to define a fragment that encompasses both the `loc`-deterministic `P`-rules considered in the present paper and the SHLIDe fragment defined in [18], as the conditions defining both classes of rules are very different and non comparable, as evidenced by the examples and explanations given in the introduction of the paper.

References

[1] Josh Berdine, Cristiano Calcagno, and Peter W. O'Hearn. Smallfoot: Modular automatic assertion checking with separation logic. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem 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 *LNCS*, pages 115–137. Springer, 2005.

[2] Josh Berdine, Byron Cook, and Samin Ishtiaq. Slayer: Memory safety for systems-level code. In Ganesh Gopalakrishnan 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] Cristiano Calcagno and Dino Distefano. Infer: An automatic program verifier for memory safety of C programs. In Mihaela Gheorghiu Bobaru, Klaus Havelund, Gerard J. Holzmann, and Rajeev Joshi, editors, *NASA Formal Methods - Third International Symposium, NFM 2011, Pasadena, CA, USA, April 18-20, 2011. Proceedings*, volume 6617 of *LNCS*, pages 459–465. Springer, 2011.

[4] Cristiano Calcagno, Peter W. O'Hearn, and Hongseok 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. `doi:10.1109/LICS.2007.30`.

[5] Taolue Chen, Fu Song, and Zhilin Wu. Tractability of separation logic with inductive definitions: Beyond lists. In Roland Meyer and Uwe Nestmann, editors, *28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany*, volume 85 of *LIPIcs*, pages 37:1–37:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[6] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007. release October, 12th 2007.

[7] B. Cook, C. Haase, J. Ouaknine, M. J. Parkinson, and J. Worrell. Tractable reasoning in a fragment of separation logic. In *Proc. of CONCUR'11*, volume 6901 of *LNCS*. Springer, 2011.

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

[9] Mnacho Echenim, Radu Iosif, and Nicolas 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.

[10] Mnacho Echenim and Nicolas Peltier. A proof procedure for separation logic with inductive definitions and data. *J. Autom. Reason.*, 67(3):30, 2023.

[11] Constantin Enea, Ondrej Lengál, Mihaela Sighireanu, and Tomás Vojnar. Compositional entailment checking for a fragment of separation logic. *Formal Methods Syst. Des.*, 51(3):575–607, 2017.

[12] Constantin Enea, Mihaela Sighireanu, and Zhilin Wu. On automated lemma generation for separation logic with inductive definitions. In *ATVA 2015, Proceedings*, pages 80–96, 2015.

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

[14] Radu Iosif, Adam Rogalewicz, and Tomás Vojnar. Deciding entailments in inductive separation logic with tree automata. In Franck Cassez and Jean-François Raskin, editors, *ATVA 2014, Proceedings*, volume 8837 of *LNCS*, pages 201–218. Springer, 2014.

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

[16] Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Ales Bizjak, Lars Birkedal, and Derek Dreyer. Iris from the ground up: A modular foundation for higher-order concurrent separation logic. *J. Funct. Program.*, 28:e20, 2018.

[17] Jens Katelaan and Florian Zuleger. Beyond symbolic heaps: Deciding separation logic with inductive definitions. In Elvira Albert and Laura Kovács, editors, *LPAR 2020: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Alicante, Spain, May 22-27, 2020*, volume 73 of *EPiC Series in Computing*, pages 390–408. EasyChair, 2020.

[18] Quang Loc Le and Xuan-Bach D. Le. An efficient cyclic entailment procedure in a fragment of separation logic. In Orna Kupferman and Pawel Sobocinski, editors, *Foundations of Software Science and Computation Structures - 26th International Conference, FoSSaCS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings*, volume 13992 of *LNCS*, pages 477–497. Springer, 2023.

[19] Christoph Matheja, Jens Pagel, and Florian Zuleger. A decision procedure for guarded separation logic complete entailment checking for separation logic with inductive definitions. *ACM Trans. Comput. Log.*, 24(1):1:1–1:76, 2023. `doi:10.1145/3534927`.

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

[21] The Coq Development Team. *The Coq Proof Assistant Reference Manual V7.1.* INRIA-Rocquencourt, CNRS-ENS Lyon (France), October 2001. `http://coq.inria.fr/doc/main.html`.