Reasoning on Dynamic Transformations of Symbolic Heaps

Nicolas Peltier ✉
Univ. Grenoble Alpes, CNRS, LIG, Grenoble, France

Abstract

Building on previous results concerning the decidability of the satisfiability and entailment problems for separation logic formulas with inductively defined predicates, we devise a proof procedure to reason on dynamic transformations of memory heaps. The initial state of the system is described by a separation logic formula of some particular form, its evolution is modeled by a finite transition system and the expected property is given as a linear temporal logic formula built over assertions in separation logic.

2012 ACM Subject Classification Theory of computation → Separation logic; Theory of computation → Automated reasoning; Theory of computation → Modal and temporal logics

Keywords and phrases Separation Logic, Symbolic Heaps, Linear Temporal Logic

Digital Object Identifier 10.4230/LIPIcs.TIME.2022.9

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

1 Introduction

Separation logic (SL) [14] is a dialect of bunched logic [10], that was introduced in verification to reason on programs manipulating dynamically allocated memory. The logic uses a particular connective $*$ to assert that two formulas hold on disjoint parts of the memory, which allows for more concise specifications. It supports *local reasoning*, in the sense that the properties of a program can be asserted and proven by referring only to the part of the memory that is affected by the program, and not to the global state of the system. The expressive power of the logic may be enhanced by using inductively defined predicates, which can be used to define recursive data structures of unbounded sizes, such as lists or trees. For instance, the following rules define a predicate $\mathtt{lseg}(x,y)$ denoting a non empty list segment from x to y: $\{\mathtt{lseg}(x,y) \Leftarrow x \mapsto (y), \quad \mathtt{lseg}(x,y) \Leftarrow \exists z.(x \mapsto (z) * \mathtt{lseg}(z,y))\}$. Informally, x, y, z denotes locations (i.e., memory addresses), $x \mapsto (y)$ states that location x is allocated and points to location y and the *separating conjunction* $x \mapsto (z) * \mathtt{lseg}(z,y)$ states that the heap contains a list segment $\mathtt{lseg}(z,y)$ together with an additional memory cell x that points to z (it implicitly entails that x is distinct from all the memory locations allocated in the list segment from z to y). These predicates may be hard coded, but they may also be defined by the user, to tackle custom data structures. For the fragment of separation logic called *symbolic heaps* (formally defined later), satisfiability is decidable [3], but entailment is undecidable in general (entailment cannot be reduced to satisfiability since the fragment does not include negations). However, a general class of decidable entailment problems is described in [7], based on restrictions on the form of the inductive rules that define the semantics of the inductive predicates. More recently, it was shown that the entailment problem is 2-EXPTIME complete [11, 4] for such inductive rules. Building on these results, we devise in the present work a proof procedure to reason on dynamic transformations of data structures specified by SL formulas with inductively defined predicates. More precisely, we consider entailments of the form $\phi \models^{\mathcal{S}}_{\mathcal{R}} \Phi$, where ϕ is an SL formula (more precisely a

29th International Symposium on Temporal Representation and Reasoning (TIME 2022).
Editors: Alexander Artikis, Roberto Posenato, and Stefano Tonetta; Article No. 9; pp. 9:1–9:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

symbolic heap), $\mathcal{R}$ is a set of inductive rules, $\mathcal{S}$ is a transition system and Φ is a formula combining symbolic heaps with temporal connectives of linear temporal logic (LTL) [13]. Informally, such an entailment is valid if the formula Φ holds w.r.t. all the runs obtained by starting from a structure satisfying the formula ϕ and following the transition system $\mathcal{S}$. The symbolic heap ϕ describes the initial state of the system, $\mathcal{R}$ defines the semantics of the inductively defined predicate symbols, $\mathcal{S}$ describes how the system evolves along time and Φ gives the expected behavior of the system. The system $\mathcal{S}$ may affect the considered structure by changing the value of variables, by allocating or freeing memory locations, or by redirecting already allocated locations. For instance, we may check whether an entailment $\mathtt{lseg}(x, \mathtt{nil}) \models_{\mathcal{R}}^{\mathcal{S}} \boldsymbol{F}\, \mathtt{lseg}(x, x)$ holds, meaning that an initial list segment from x to $\mathtt{nil}$ is eventually transformed into a circular list, or that $\mathtt{lseg}(x, \mathtt{nil}) \models_{\mathcal{R}}^{\mathcal{S}} \boldsymbol{G}(q \Rightarrow \mathtt{lseg}(x, \mathtt{nil}))$ holds, meaning that each time the system reaches state q the heap contains a list from x to $\mathtt{nil}$. We show that the entailment problem is undecidable in general, but decidable if the considered transition system satisfies some conditions, which, intuitively, prevent actions affecting the value of the variables to occur inside loops (the other actions are not restricted). The proposed decision procedure is modular, and relies on a combination of the algorithm described in [12, 11] for checking the satisfiability of separation logic formulas with usual model checking and model construction procedures for LTL.

Related work

Dynamic transformations are usually tackled in SL using Hoare logic, with pre and post-conditions defined with the help of separating implications (see, e.g., [1]). Separating implication is not used in our approach due to the difficulty of reasoning automatically with this connective, especially in connection with inductive definitions (however, the so-called *context predicates* introduced in Section 7 can be viewed as a restricted form of separating implication). The combination of SL with temporal connectives is rather natural and has been considered in [2]. In [8, 6], temporal extensions of the related bunched logic are considered. Our approach departs from this work because the fragment of separation logic that we consider is very different: while the logic in [2] is based on quantifier-free separation logic formulas (with arbitrary combinations of boolean and separating connectives), we focus on symbolic heaps, i.e., on separating conjunctions of inductively defined atoms (with existential quantification). Thus on one hand our basic assertion language is more restricted because we strongly restricts the nesting of separation connectives, but on the other hand the addition of inductively defined predicates greatly increases the expressive power of the language and allows one to tackle richer data structures. In particular we emphasize that – without temporal connectives – entailment is 2-EXPTIME complete for the fragment that we consider, whereas satisfiability is PSPACE-complete for that considered in [2].

2 Separation Logic

We define the syntax and semantics of a fragment of separating logic called *symbolic heaps* and we recall the conditions on the inductive rules that ensure that the entailment problem is decidable. Most definitions are standard, see [14, 7] for additional explanations and examples.

▶ **Definition 1** (Symbolic Heaps). *Let $\mathcal{V}$ be a countably infinite set of* variables*. Let $\mathcal{P}$ be a finite set of* predicate symbols*. Each symbol p in $\mathcal{P}$ is associated with a unique natural number called the* arity *of p. Let κ be a fixed natural number, denoting the number of record fields. An* equational atom *is an expression of the form $x \simeq y$ or $x \not\simeq y$, where $x, y \in \mathcal{V}$. A* points-to atom *is an expression of the form* $x \mapsto (y_1, \dots, y_\kappa)$ *with* $x, y_1, \dots, y_\kappa \in \mathcal{V}$*. A* predicate atom *is an expression of the form* $p(x_1, \dots, x_n)$ *with* $p \in \mathcal{P}$*,* $n = arity(p)$ *and* $x_1, \dots, x_n \in \mathcal{V}$*. A* spatial atom *is either a points-to atom or a predicate atom. An* atom *is either an equational atom or a spatial atom. The set of* symbolic heaps *is the set of expressions of the form:* $\exists x_1 \dots \exists x_n.(\alpha_1 * \dots * \alpha_m)$ *where* $x_1, \dots, x_n$ *are variables and* $\alpha_1, \dots, \alpha_m$ *are atoms (with possibly* $n = 0$ *and/or* $m = 0$*). The connective* $*$ *is called* separating conjunction*. An empty separating conjunction is denoted by emp. For every symbolic heap* ϕ*, we denote by* $fv(\phi)$ *the set of variables freely occurring in* ϕ*.*

For all vectors $\boldsymbol{x} = (x_1, \dots, x_n)$ and $\boldsymbol{y} = (y_1, \dots, y_n)$ of the same length, we denote by $\boldsymbol{x} \simeq \boldsymbol{y}$ the separating conjunction $x_1 \simeq y_1 * \dots * x_n \simeq y_n$. If ϕ is a symbolic heap, then $\exists \boldsymbol{x}.\phi$ denotes the symbolic heap $\exists x_1 \dots \exists x_n.\phi$. For every symbolic heap ϕ we denote by $v_{\mapsto}(\phi)$ the set of free variables x such that ϕ contains a points-to atom of the form $x \mapsto (\boldsymbol{y})$.

▶ **Definition 2** (Substitutions). *A* substitution *is a function mapping every variable* x *to a variable. For every substitution* σ *and for every symbolic heap* ϕ*, we denote by* $\phi\sigma$ *the symbolic heap obtained from* ϕ *by replacing every free occurrence of a variable* x *by* $\sigma(x)$*. If* $x_1, \dots, x_n$ *are pairwise distinct variables, we denote by* $\{x_1 \leftarrow y_1, \dots, x_n \leftarrow y_n\}$ *the substitution* σ *such that* $\sigma(x_i) = y_i$ *for all* $i = 1, \dots, n$ *and* $\sigma(x) = x$ *if* $x \notin \{x_1, \dots, x_n\}$*.*

Symbolic heaps are interpreted in structures defined as follows.

▶ **Definition 3** (SL Structures). *Let* $\mathcal{L}$ *be a countably infinite set of so-called* locations*. An (SL)* structure *is a pair* $(\mathfrak{s}, \mathfrak{h})$ *where:*

- $\mathfrak{s}$ *is a* store*, i.e., a function mapping every variable to a location.*
- $\mathfrak{h}$ *is a* heap*, i.e., a finite partial function mapping locations to* κ*-tuples of locations. We denote by* $dom(\mathfrak{h})$ *the finite domain of* $\mathfrak{h}$*, by* $|\mathfrak{h}|$ *the cardinality of* $dom(\mathfrak{h})$ *and by* $locs(\mathfrak{h})$ *the set:* $\{\ell_i \mid \ell_0 \in dom(\mathfrak{h}), \mathfrak{h}(\ell_0) = (\ell_1, \dots, \ell_\kappa), 0 \leq i \leq \kappa\}$*.*

A location $\ell \in dom(\mathfrak{h})$ *is* allocated *in* $\mathfrak{h}$*. A variable* x *such that* $\mathfrak{s}(x) \in dom(\mathfrak{h})$ *is* allocated *in* $(\mathfrak{s}, \mathfrak{h})$*.*

Intuitively, $\mathfrak{s}$ gives the values of the variables and $\mathfrak{h}$ denotes the dynamically allocated memory. A heap will often be denoted as a set of tuples $\mathfrak{h} = \{(\ell_0, \dots, \ell_\kappa) \mid \ell_0 \in dom(\mathfrak{h}), \mathfrak{h}(\ell_0) = (\ell_1, \dots, \ell_n)\}$. In particular, $\emptyset$ denotes the heap that allocates no location. Two heaps $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are *disjoint* if $dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2) = \emptyset$. In this case $\mathfrak{h}_1 \uplus \mathfrak{h}_2$ denotes the union of $\mathfrak{h}_1$ and $\mathfrak{h}_2$ defined as follows: $dom(\mathfrak{h}_1 \uplus \mathfrak{h}_2) \overset{\text{def}}{=} dom(\mathfrak{h}_1) \cup dom(\mathfrak{h}_2)$, and $\mathfrak{h}(\ell) = \mathfrak{h}_i(\ell)$ for all $i = 1, 2$ and $\ell \in dom(\mathfrak{h}_i)$.

The semantics of the predicate symbols is defined by user-provided inductive rules:

▶ **Definition 4** (Inductive Rules). *A* set of inductive definitions *(SID) is a set of rules of the form* $p(x_1, \dots, x_n) \Leftarrow \phi$ *such that* $p \in \mathcal{P}$*,* $n = arity(p)$*,* $x_1, \dots, x_n$ *are pairwise distinct variables, and* ϕ *is a symbolic heap with* $fv(\phi) \subseteq \{x_1, \dots, x_n\}$*.*

For every symbolic heap ϕ*, we write* $\phi \Leftarrow_{\mathcal{R}} \phi'$ *if* ϕ *is of the form* $\exists \boldsymbol{u}.(p(y_1, \dots, y_n) * \phi')$*,* $\mathcal{R}$ *contains a rule* $p(x_1, \dots, x_n) \Leftarrow \exists \boldsymbol{v}.\psi$ *(where* ψ *contains no quantifier) and* $\phi' = \exists \boldsymbol{u} \exists \boldsymbol{v}.(\psi\{x_i \leftarrow y_i \mid i = 1, \dots, n\} * \phi')$*. We assume by* α*-renaming that the vector* $\boldsymbol{v}$ *contains no variable in* $\boldsymbol{u}$*,* $fv(\phi)$ *or* $(x_1, \dots, x_n)$*. As usual* $\Leftarrow_{\mathcal{R}}^*$ *is the reflexive and transitive closure of* $\Leftarrow_{\mathcal{R}}$*.*

The satisfiability relation is defined inductively as follows. We emphasize that equational atoms are valid only if the heap is empty; this convention allows us to simplify notations (it avoids having to use both the separating conjunction and the standard one).

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

- $\mathfrak{h} = \emptyset$ *and either* ($\phi = (x \simeq y)$ *and* $\mathfrak{s}(x) = \mathfrak{s}(y)$), *or* ($\phi = (x \not\simeq y)$ *and* $\mathfrak{s}(x) \neq \mathfrak{s}(y)$).
- $\phi = x \mapsto (y_1, \dots, y_\kappa)$ *and* $\mathfrak{h} = \{(\mathfrak{s}(x), \mathfrak{s}(y_1), \dots, \mathfrak{s}(y_\kappa))\}$.
- $\phi = \phi_1 * \phi_2$ *and there exist disjoint heaps* $\mathfrak{h}_1$ *and* $\mathfrak{h}_2$ *such that* $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$ *and* $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathcal{R}} \phi_i$, *for all* $i = 1, 2$.
- $\phi = p(x_1, \dots, x_n)$ *with* $p \in \mathcal{P}$, $p(x_1, \dots, x_n) \Leftarrow^*_{\mathcal{R}} \psi$, ψ *contains no predicate symbols and* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \psi$.
- $\phi = \exists x.\psi$, *and there exists a store* $\mathfrak{s}'$ *coinciding with* $\mathfrak{s}$ *on all variables distinct from* x *such that* $(\mathfrak{s}', \mathfrak{h}) \models_{\mathcal{R}} \psi$.

An $\mathcal{R}$-model *of* ϕ *is a structure* $(\mathfrak{s}, \mathfrak{h})$ *such that* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$. *If* ϕ, ϕ' *are symbolic heaps, we write* $\phi \models_{\mathcal{R}} \phi'$ *if the entailment* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \implies (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi'$ *for all SL structures* $(\mathfrak{s}, \mathfrak{h})$, *and* $\phi \equiv_{\mathcal{R}} \psi$ *if* $\phi \models_{\mathcal{R}} \psi$ *and* $\psi \models_{\mathcal{R}} \phi$.

Restricting Inductive Definitions

While the entailment problem is undecidable in general for symbolic heaps with inductively defined predicates, a very general decidable class is identified in [7]. This fragment is defined by restricting the form of the inductive rules, which must satisfy three conditions, recalled below (we use the slightly more general version of establishment given in [11]).

▶ **Definition 6** (Progress, Connectedness and Establishment (PCE)). *A rule* $p(x_1, \dots, x_n) \leftarrow \exists \boldsymbol{y}.\phi$ *(where* ϕ *contains no quantifier) is:*

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

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

▶ **Example 7.** The following set, defining a list segment ending at an arbitrary location, is progressing and connected, but *not* established:

$$\{\mathtt{lseg}'(x) \Leftarrow \exists y.x \mapsto (y), \quad \mathtt{lseg}'(x) \Leftarrow \exists z.(x \mapsto (z) * \mathtt{lseg}'(z))\}$$

In the remainder of the paper we assume that a set of inductive rules $\mathcal{R}$ is given, satisfying the PCE conditions. This is the case for the rules given in the Introduction for the predicate `lseg`. We now introduce a notion of heap constraints, which combine positive and negative assertions denoted by symbolic heaps, with constraints specifying that some variables are unallocated:

▶ **Definition 8** (Heap Constraint). *A* heap constraint *is a triple* (S^+, S^-, X), *where* S^+ *and* S^- *are sets of symbolic heaps,* $S^+ \neq \emptyset$ *and* $X \subseteq \mathcal{V}$. *We write* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} (S^+, S^-, X)$ *if for all* $\phi \in S^+$*:* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$*; for all* $\phi \in S^-$*:* $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathcal{R}} \phi$*; and for all* $x \in X$*:* $\mathfrak{s}(x) \notin dom(\mathfrak{h})$.

The decidability of the satisfiability problem for such constraints follows from [12, 11]:

▶ **Lemma 9.** *There exists an algorithm that, given a heap constraint* (S^+, S^-, X) *and a progressing, connected and established set of rules* $\mathcal{R}$, *checks whether there exists an SL structure* $(\mathfrak{s}, \mathfrak{h})$ *such that* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} (S^+, S^-, X)$.

3 Actions Operating on SL Structures

We define the basic actions that can occur in transition systems. The set of actions includes tests, affectations, redirections of allocated locations, as well as allocations and desallocations.

▶ **Definition 10** (Actions). *Let $\mathcal{V}^\star$ be a finite set of variables. A* term *is either an element of $\mathcal{V}^\star$ or an expression of the form $x.i$ where $x \in \mathcal{V}^\star$ and $i \in \{1, \dots, \kappa\}$. A* condition *is a boolean combination of* atomic conditions, *that are expressions of the form $t \approx s$ where t, s are terms. An* action *is an expression of one of the following forms:* `pass` *(null action); $t := s$, where t and s are terms (affectation or redirection);* `alloc`(x) *or* `free`(x)*, where $x \in \mathcal{V}^\star$ (allocation and desallocation); or* `test`(γ)*, where γ is a condition (test).*

The semantics of conditions is defined below.

▶ **Definition 11** (Semantics of Conditions). *For every structure $(\mathfrak{s}, \mathfrak{h})$ and for every term t we write $t \triangleright_{(\mathfrak{s},\mathfrak{h})} \ell$ (t* evaluates to *ℓ) if either $t \in \mathcal{V}^\star$ and $\mathfrak{s}(t) = \ell$, or $t = x.i$ with $x \in \mathcal{V}^\star$, $\mathfrak{s}(x) \in dom(\mathfrak{h})$, $\mathfrak{h}(\mathfrak{s}(x)) = (\ell_1, \dots, \ell_\kappa)$ and $\ell = \ell_i$. We write $(\mathfrak{s}, \mathfrak{h}) \models t \approx s$ if there exists $\ell \in \mathcal{L}$ such that $t \triangleright_{(\mathfrak{s},\mathfrak{h})} \ell$ and $s \triangleright_{(\mathfrak{s},\mathfrak{h})} \ell$. The relation $\models$ is extended to every boolean combination of atomic conditions inductively as usual.*

Observe that the semantics of $x \approx y$ is different from that of $x \simeq y$, which requires that the heap be empty. Furthermore, $(\mathfrak{s}, \mathfrak{h}) \models x.i \approx x.i$ (for $i = 1, \dots, \kappa$) holds iff x is allocated. We thus denote by $\mathtt{A}(x)$ (for "x is allocated") the formula $x.1 \approx x.1$. The semantics of actions is rather natural, and formally defined below (to make allocations deterministic we assume that the variable allocated by `alloc`(x) points to itself).

▶ **Definition 12** (Semantics of Actions). *For every SL structure $(\mathfrak{s}, \mathfrak{h})$ and action a, we denote by $(\mathfrak{s}, \mathfrak{h})[a]$ the result of the application of the action a on $(\mathfrak{s}, \mathfrak{h})$ defined as follows:*

- *If $a =$ `pass` then $(\mathfrak{s}, \mathfrak{h})[a] \overset{def}{=} (\mathfrak{s}, \mathfrak{h})$.*
- *If $a = (x := s)$ with $x \in \mathcal{V}^\star$ and $s \triangleright_{(\mathfrak{s},\mathfrak{h})} \ell$ then $(\mathfrak{s}, \mathfrak{h})[a] \overset{def}{=} (\mathfrak{s}', \mathfrak{h})$, where $\mathfrak{s}'(x) = \ell$ and $\mathfrak{s}'$ coincides with $\mathfrak{s}$ on all variables distinct from x.*
- *If $a = (x.i := s)$ with $x \in \mathcal{V}^\star$, $\mathfrak{s}(x) \in dom(\mathfrak{h})$, $\mathfrak{h}(\mathfrak{s}(x)) = (\ell_1, \dots, \ell_n)$ and $s \triangleright_{(\mathfrak{s},\mathfrak{h})} \ell$ then $(\mathfrak{s}, \mathfrak{h})[a] \overset{def}{=} (\mathfrak{s}, \mathfrak{h}')$, where $dom(\mathfrak{h}') = dom(\mathfrak{h})$, $\mathfrak{h}'(\mathfrak{s}(x)) = (\ell_1, \dots, \ell_{i-1}, \ell, \ell_{i+1}, \dots, \ell_\kappa)$, and $\mathfrak{h}'$ coincides with $\mathfrak{h}$ on all locations distinct from $\mathfrak{s}(x)$.*
- *If $a =$ `free`(x), $\mathfrak{s}(x) \in dom(\mathfrak{h})$ then $(\mathfrak{s}, \mathfrak{h})[a] \overset{def}{=} (\mathfrak{s}, \mathfrak{h}')$ where $dom(\mathfrak{h}') = dom(\mathfrak{h}) \setminus \{\mathfrak{s}(x)\}$ and $\mathfrak{h}'$ coincides with $\mathfrak{h}$ on all locations distinct from $\mathfrak{s}(x)$.*
- *If $a =$ `alloc`(x), $\mathfrak{s}(x) \notin dom(\mathfrak{h})$ then $(\mathfrak{s}, \mathfrak{h})[a] \overset{def}{=} (\mathfrak{s}, \mathfrak{h}')$ where $dom(\mathfrak{h}') = dom(\mathfrak{h}) \cup \{\mathfrak{s}(x)\}$, $\mathfrak{h}(\mathfrak{s}(x)) = (\mathfrak{s}(x), \dots, \mathfrak{s}(x))$ and $\mathfrak{h}'$ coincides with $\mathfrak{h}$ on all locations distinct from $\mathfrak{s}(x)$.*
- *If $a =$ `test`(γ) and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \gamma$ then $(\mathfrak{s}, \mathfrak{h})[a] \overset{def}{=} (\mathfrak{s}, \mathfrak{h})$.*

Otherwise $(\mathfrak{s}, \mathfrak{h})[a]$ is undefined.

▶ **Proposition 13.** *For all structures $(\mathfrak{s}, \mathfrak{h})$ and actions a, if $(\mathfrak{s}', \mathfrak{h}') = (\mathfrak{s}, \mathfrak{h})[a]$ then $\mathfrak{s}'(\mathcal{V}^\star) \cup locs(\mathfrak{h}') \subseteq \mathfrak{s}(\mathcal{V}^\star) \cup locs(\mathfrak{h})$.*

Proof. By an immediate case analysis on the set of actions. ◀

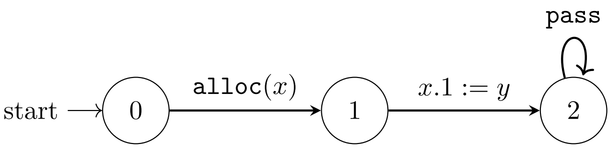
4 Transition Systems

Transition systems are finite state automata where the transitions are labeled by actions:

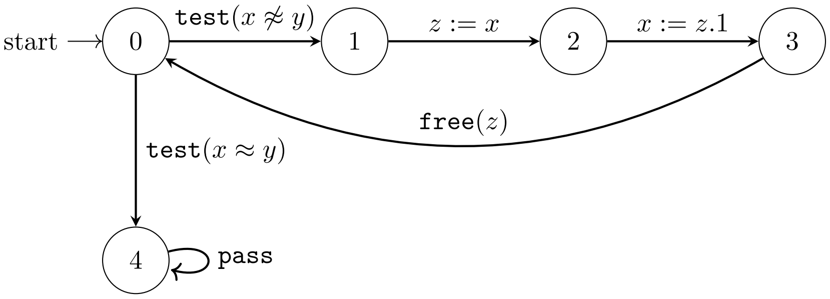
▶ **Definition 14** (Transition Systems). *Let $\mathbb{S}$ be a countably infinite set of* states. *A* transition system *is a triple $\mathcal{S} = (Q, R, q_I)$ where Q is a finite subset of $\mathbb{S}$, R is a finite set of* transition rules *of the form (q, a, q') where $q, q' \in Q$ and a is an action, and $q_I \in Q$ is the initial state. A* run *in $\mathcal{S}$* from *a structure $(\mathfrak{s}, \mathfrak{h})$ is an infinite sequence of tuples $(q_i, \mathfrak{s}_i, \mathfrak{h}_i, a_i)_{i\in\mathbb{N}}$ such that $q_0 = q_I$, $(\mathfrak{s}_0, \mathfrak{h}_0) = (\mathfrak{s}, \mathfrak{h})$, and for every $i \in \mathbb{N}$: $(q_i, a_i, q_{i+1}) \in R$ and $(\mathfrak{s}_{i+1}, \mathfrak{h}_{i+1}) = (\mathfrak{s}_i, \mathfrak{h}_i)[a_i]$. We denote by $\succeq_{\mathcal{S}}$ the smallest transitive relation such that $(q, a, q') \in R \implies q \succeq_{\mathcal{S}} q'$. We write $q \sim_{\mathcal{S}} q'$ iff $q \succeq_{\mathcal{S}} q'$ and $q' \succeq_{\mathcal{S}} q$ and $q \succ_{\mathcal{S}} q'$ if $q \succeq_{\mathcal{S}} q'$ and $q' \not\succeq_{\mathcal{S}} q$.*

Note that the above definition entails that $(\mathfrak{s}_i, \mathfrak{h}_i)[a_i]$ must be defined, for all $i \in \mathbb{N}$. For simplicity we assume that all runs are infinite (finite runs may be encoded if needed by adding a final state q_F with a transition $(q_F, \texttt{pass}, q_F)$).

▶ **Example 15.** The following transition system adds an element x to a list starting at y:



▶ **Example 16.** The following transition system desallocates a list segment from x to y.



5 Temporal Formulas

We now define temporal formulas built over a set of assertions containing symbolic heaps, states, actions and conditions, using the usual set of LTL connectives:

▶ **Definition 17** (Syntax of LTL Formulas). *The set $\mathbb{A}_{\mathcal{S}}$ of* LTL atoms *contains all symbolic heaps ϕ with $fv(\phi) \subseteq \mathcal{V}^\star$, all atomic conditions, all actions and all states in $\mathbb{S}$. The set of* LTL formulas *is the least set containing $\mathbb{A}_{\mathcal{S}}$ and such that for all LTL formulas Φ, Ψ: $\neg\Phi$, $\Phi \vee \Psi$, $\boldsymbol{X}\, \Phi$, $\Phi\, \boldsymbol{U}\, \Psi$ are LTL formulas.*

The additional connectives $\wedge$, $\boldsymbol{F}$, $\boldsymbol{R}$ etc. are defined as usual. The semantics of LTL formulas is recalled below. Note that LTL atoms are interpreted arbitrarily at this point.

▶ **Definition 18** (Semantics of LTL Formulas). *An* LTL interpretation *$\mathcal{I}$ is a mapping from $\mathbb{A}_{\mathcal{S}} \times \mathbb{N}$ to $\{true, false\}$. For any LTL formula Φ, we write $\mathcal{I} \models \Phi$ if $(\mathcal{I}, 0) \models \Phi$, and $(\mathcal{I}, i) \models \Phi$ iff one of the following conditions holds:*

- *$\Phi \in \mathbb{A}_{\mathcal{S}}$ and $\mathcal{I}(\Phi, i) = true$, or $\Phi = \neg\Psi$ and $(\mathcal{I}, i) \not\models \Psi$;*
- *$\Phi = \Phi_1 \vee \Phi_2$ and $(\mathcal{I}, i) \models \Phi_j$, for some $j = 1, 2$; or $\Phi = \boldsymbol{X}\,\Psi$ and $(\mathcal{I}, i+1) \models \Psi$;*
- *$\Phi = \Phi_1 \boldsymbol{U} \Phi_2$ and there exists $j \geq i$ such that $(\mathcal{I}, j) \models \Phi_2$ and $(\mathcal{I}, k) \models \Phi_1$ for all $k \in \{i, \ldots, j-1\}$.*

In the following, we will assume that all the considered LTL interpretations are *ultimately periodic* (so that they admit a finite representation) i.e., that there exist natural numbers k, l such that, for every $i \geq k$ and for every atom $\alpha \in \mathbb{A}_{\mathcal{S}}$: $\mathcal{I}(\alpha, i) = \mathcal{I}(\alpha, i+l)$. It is well-known that every satisfiable LTL formula admits an ultimately periodic model. Definition 19 relates the semantics of LTL atoms to SL structures and transition systems.

▶ **Definition 19** (Compatibility). *Let $\mathcal{S} = (Q, R, q_I)$ be a transition system. An LTL interpretation $\mathcal{I}$ is* compatible *with a run $(q_i, \mathfrak{s}_i, \mathfrak{h}_i, a_i)_{i\in\mathbb{N}}$ in $\mathcal{S}$ w.r.t. a formula Φ iff the following conditions holds, for all $i \in \mathbb{N}$:*

- *For every symbolic heap or condition ϕ occurring in Φ, $\mathcal{I}(\phi, i) = true \iff (\mathfrak{s}_i, \mathfrak{h}_i) \models_{\mathcal{R}} \phi$.*
- *For all actions a, $\mathcal{I}(a, i) = true \iff a = a_i$.*
- *For all states $q \in Q$, $\mathcal{I}(q, i) = true \iff q_i = q$.*

An LTL interpretation $\mathcal{I}$ is compatible *with an SL structure $(\mathfrak{s}, \mathfrak{h})$ and a transition system $\mathcal{S} = (Q, R, q_I)$, w.r.t. a formula Φ if it is compatible with some run $(q_i, \mathfrak{s}_i, \mathfrak{h}_i, a_i)_{i\in\mathbb{N}}$ in $\mathcal{S}$.*

We are now in the position to define the satisfiability relation that relates SL structures to LTL formulas, w.r.t. a given transition system.

▶ **Definition 20** (Entailment). *Let $\mathcal{S}$ be a transition system. For every structure $(\mathfrak{s}, \mathfrak{h})$ and LTL formula Φ, we write $(\mathfrak{s}, \mathfrak{h}) \models^{\mathcal{S}}_{\mathcal{R}} \Phi$ iff $\mathcal{I} \models \Phi$ holds for every LTL interpretation compatible with $(\mathfrak{s}, \mathfrak{h})$ and $\mathcal{S}$ w.r.t. Φ. We write $(\mathfrak{s}, \mathfrak{h}) \models^{\mathcal{S}/(q_i,a_i)_{i\in\mathbb{N}}}_{\mathcal{R}} \Phi$ if there exists a run $(q_i, \mathfrak{s}_i, \mathfrak{h}_i, a_i)_{i\in\mathbb{N}}$ in $\mathcal{S}$ with $\mathfrak{s}_0 = \mathfrak{s}$ and $\mathfrak{h}_0 = \mathfrak{h}$, and an LTL interpretation $\mathcal{I}$ that is compatible with $(q_i, \mathfrak{s}_i, \mathfrak{h}_i, a_i)_{i\in\mathbb{N}}$ such that $\mathcal{I} \models \Phi$.*

For every symbolic heap ϕ, we write $\phi \models^{\mathcal{S}}_{\mathcal{R}} \Phi$ if the entailment $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi \implies (\mathfrak{s}, \mathfrak{h}) \models^{\mathcal{S}}_{\mathcal{R}} \Phi$ holds for all structures $(\mathfrak{s}, \mathfrak{h})$.

▶ **Example 21.** If $\mathcal{S}$ is the transition system of Example 15, then the entailments $\mathtt{lseg}(y, z) \models^{\mathcal{S}}_{\mathcal{R}} \boldsymbol{F}\,\mathtt{lseg}(x, z)$ and $\mathtt{lseg}(y, z) \models^{\mathcal{S}}_{\mathcal{R}} \boldsymbol{X}\boldsymbol{X}\boldsymbol{G}\,\mathtt{lseg}(x, z)$ are valid. Note that the structures in which x is initially allocated are not considered for testing the entailment.

If $\mathcal{S}$ now denotes the transition system of Example 16, then the entailment $\mathtt{lseg}(x, y) \models^{\mathcal{S}}_{\mathcal{R}}$ $\boldsymbol{F}$ *emp* is *not* valid (because the initial list segment may be cyclic).

It is easy to see that model checking is decidable:

▶ **Lemma 22.** *The problem of checking whether $(\mathfrak{s}, \mathfrak{h}) \models^{\mathcal{S}/(q_i,a_i)_{i\in\mathbb{N}}}_{\mathcal{R}} \Phi$ is decidable (if the sequence $(q_i, a_i)_{i\in\mathbb{N}}$ is ultimately periodic).*

Proof. Since $\mathfrak{s}$, $\mathfrak{h}$, q_i and a_i are given, the run (if it exists) $(q_i, \mathfrak{s}_i, \mathfrak{h}_i, a_i)_{i\in\mathbb{N}}$ such that $\mathfrak{s}_0 = \mathfrak{s}$ and $\mathfrak{h}_0 = \mathfrak{h}$, and the compatible LTL interpretation $\mathcal{I}$ are easy to compute, using Definition 11. Using Proposition 13, we get $\mathfrak{s}_i(\mathcal{V}^\star) \cup \mathit{locs}(\mathfrak{h}_i) \subseteq \mathfrak{s}(\mathcal{V}^\star) \cup \mathit{locs}(\mathfrak{h})$ for all $i \in \mathbb{N}$, thus the set of structures $\{(\mathfrak{s}_i, \mathfrak{h}_i) \mid i \in \mathbb{N}\}$ is necessarily finite. Thus, the interpretation $\mathcal{I}$ is ultimately periodic and the test $\mathcal{I} \models \Phi$ can be performed using well-known algorithms for LTL. ◀

However, the entailment problem is undecidable in general:

▶ **Theorem 23.** *The problem of checking whether $\phi \models^{\mathcal{S}}_{\mathcal{R}} \Phi$ is undecidable (even if $\mathcal{R}$ is progressing, connected and established).*

Proof (Sketch). Turing machines (TM) may be simulated by transition systems: the elements of the alphabet are denoted by pairwise distinct free variables, a tape $(x_1, \dots, x_n)$ is encoded as a heap (denoting a doubly linked list): $\{(\ell_i, \ell'_i, \ell_{i-1}, \ell_{i+1}) \mid i = 1, \dots, n\}$ with $\ell'_i = \mathfrak{s}(x_i)$, and the position of the head is denoted by a variable x. Moves are encoded by actions $x := x.2$ (left move) or $x := x.3$ (right move). Tests are performed by actions of the form $\mathtt{test}(x.1 \approx y)$, where y is the variable associated with the considered symbol. The action $x.1 := y$ writes y at the current position in the tape. Note that if the initial heap does not contain enough allocated locations then the transition system may be "stuck" (because a right move cannot be applied, hence no run will exist). However, the following rules define a predicate p that allocates a tape of arbitrary size filled with a symbol u (which may be instantiated by a blank denoted by some free variable b). The variables y and z denote the start and the end of the tape, respectively: $\{p(x, y, z, u) \Leftarrow x \mapsto (u, y, z), \quad p(x, y, z, u) \Leftarrow \exists x'.(x \mapsto (u, y, x') * p(x', x, z, u))\}$. It is easy to check that the non termination of the considered TM (on an empty tape) can be checked by testing whether the entailment $p(x, y, z, b) \models^{\mathcal{S}}_{\mathcal{R}} \boldsymbol{G}(\neg \bigvee_{q \in Q_F} q)$ holds, where Q_F is the set of final states (note however that the entailment $p(x, y, z, b) \models^{\mathcal{S}}_{\mathcal{R}} \boldsymbol{F}(\bigvee_{q \in Q_F} q)$ does *not* encode termination, as it may have counter models in which $p(x, y, z, b)$ does not allocate enough memory cells to execute the TM). ◀

To overcome this issue we require that no action of the form $x := t$ occurs inside a loop:

▶ **Definition 24** (Oriented Transition System). *A transition system $\mathcal{S} = (Q, R, q_I)$ is* oriented *if for every transition (q, a, q') in R, if a is of the form $x := t$ then $q \succ_{\mathcal{S}} q'$.*

The transition system of Example 15 is oriented, but not that of Example 16.

6 Symbolic Execution of Actions

We now show how to execute actions symbolically on SL formulas. We first define an LTL formula encoding the conditions ensuring that an action can be performed:

▶ **Definition 25** (Precondition). *For all actions a, $pre(a)$ is defined as follows (with $x, y \in \mathcal{V}^\star$): $pre(\mathtt{alloc}(x)) \overset{def}{=} \neg \mathtt{A}(x)$, $pre(\mathtt{free}(x)) \overset{def}{=} \mathtt{A}(x)$, $pre(x.i := y) \overset{def}{=} \mathtt{A}(x)$, $pre(x := y.i) \overset{def}{=} \mathtt{A}(y)$, $pre(x.i := y.j) \overset{def}{=} \mathtt{A}(x) \wedge \mathtt{A}(y)$, $pre(\mathtt{test}(\gamma)) \overset{def}{=} \gamma$ and $pre(a) \overset{def}{=} \top$ otherwise.*

▶ **Proposition 26.** *For every action a and for every structure $(\mathfrak{s}, \mathfrak{h})$, $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} pre(a)$ iff $(\mathfrak{s}, \mathfrak{h})[a]$ is defined.*

Proof. Immediate. ◀

Given a symbolic heap ϕ and action a, it is sometimes possible to compute the strongest postcondition of ϕ w.r.t. to a, which describes the state of the memory after action a is performed on a structure satisfying ϕ:

▶ **Definition 27** (Strongest Postcondition). *For every symbolic heap ϕ and for every action a, we define a formula $spc(\phi, a)$ (strongest postcondition of α w.r.t. a) as follows (where x' denotes a fresh variable).*

- $spc(\phi, \mathtt{pass}) \overset{def}{=} \phi$.
- $spc(\exists \boldsymbol{y}.\phi, \mathtt{alloc}(x)) \overset{def}{=} \exists \boldsymbol{y}.(x \mapsto (x, \dots, x) * \phi)$.
- $spc(\exists \boldsymbol{y}.(x \mapsto (y_1, \dots, y_\kappa) * \phi), \mathtt{free}(x)) \overset{def}{=} \exists \boldsymbol{y}.\phi$.
- $spc(\phi, x := y) \overset{def}{=} \exists x'.(\phi\{x \leftarrow x'\} * x \simeq y)$ *(if $x, y \in \mathcal{V}^\star$).*
- $spc(\exists \boldsymbol{u}.(\phi * y \mapsto (y_1, \dots, y_\kappa)), x := y.i) \overset{def}{=} \exists \boldsymbol{u} \exists x'.((\phi * y \mapsto (y_1, \dots, y_\kappa))\{x \leftarrow x'\} * x \simeq y_i)$.

- *$spc(\exists \boldsymbol{u}.(x \mapsto (x_1,\dots,x_\kappa) * \phi), x.i := z) \overset{def}{=} \exists \boldsymbol{u}.(x \mapsto (x_1,\dots,x_{i-1},z,x_{i+1},\dots,x_\kappa) * \phi)$ (if $z \in \mathcal{V}^\star$).*
- *$spc(\exists \boldsymbol{u}.(x \mapsto (x_1,\dots,x_\kappa) * \phi), x.i := x.j) \overset{def}{=} \exists \boldsymbol{u}.(x \mapsto (x_1,\dots,x_{i-1},x_j,x_{i+1},\dots,x_\kappa) * \phi)$.*
- *$spc(\exists \boldsymbol{u}.(x \mapsto (x_1,\dots,x_\kappa) * y \mapsto (y_1,\dots,y_\kappa) * \phi), x.i := y.j) \overset{def}{=} \exists \boldsymbol{y}.(x \mapsto (x_1,\dots,x_{i-1},y_j, x_{i+1},\dots,x_\kappa) * y \mapsto (y_1,\dots,y_\kappa) * \phi)$ (if $x \neq y$).*
- *Otherwise $spc(\phi,a)$ is undefined.*

In all cases, ϕ may be emp.

▶ **Example 28.** For instance, we have:

$$\begin{array}{lcl} spc(x \mapsto (y,z), x.1 := x) & = & x \mapsto (x,z) \\ spc(x \mapsto (y,z), x := y) & = & \exists x'.(x' \mapsto (y,z) * x \simeq y) \\ spc(x \mapsto (y,z), \mathtt{free}(x)) & = & emp \end{array}$$

But both $spc(x \mapsto (y,z), y.1 := x)$ and $spc(\mathtt{lseg}(x,y), x.1 := y)$ are undefined.

▶ **Lemma 29.** *Let ϕ be a symbolic heap and let a be an action. If $spc(\phi,a)$ is defined then for every structure $(\mathfrak{s},\mathfrak{h})$ such that $(\mathfrak{s},\mathfrak{h})[a]$ is defined, we have $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi \implies (\mathfrak{s},\mathfrak{h})[a] \models_{\mathcal{R}} spc(\phi,a)$.*

Proof. By inspection of the different actions, using Definition 11. ◀

Similarly, it is possible in some cases to define the weakest precondition of a symbolic heap w.r.t. an action, asserting conditions that guarantee that the given formula is satisfied after the action is performed:

▶ **Definition 30** (Weakest Precondition). *For every symbolic heap ϕ and for every action a, the formula $wpc(\phi,a)$ is defined as follows (where x' denotes a fresh variable).*

- *$wpc(\phi, \mathtt{pass}) \overset{def}{=} \phi$.*
- *$wpc(\exists \boldsymbol{x}.(\phi * x \mapsto (y_1,\dots,y_\kappa)), \mathtt{alloc}(x)) \overset{def}{=} \exists \boldsymbol{x}.(\phi * y_1 \simeq x * \dots * y_\kappa \simeq x)$.*
- *$wpc(\exists \boldsymbol{x}.\phi, \mathtt{free}(x)) \overset{def}{=} \exists \boldsymbol{x} \exists y_1 \dots \exists y_\kappa.(\phi * x \mapsto (y_1,\dots,y_\kappa))$.*
- *$wpc(\phi, x := y) \overset{def}{=} \phi\{x \leftarrow y\}$ (if $x,y \in \mathcal{V}^\star$).*
- *$wpc(\exists \boldsymbol{x}.(\phi * x \mapsto (x_1,\dots,x_\kappa)), x.i := y) \overset{def}{=} \exists \boldsymbol{x} \exists x'.(\phi * x \mapsto (x_1,\dots,x_{i-1},x',x_{i+1},\dots,x_\kappa) * x_i \simeq y)$ (if $y \in \mathcal{V}^\star$).*
- *$wpc(\exists \boldsymbol{x}.(\phi * x \mapsto (x_1,\dots,x_\kappa) * y \mapsto (y_1,\dots,y_\kappa)), x.i := y.j) \overset{def}{=} \exists \boldsymbol{x} \exists x'.(\phi * x \mapsto (x_1,\dots, x_{i-1},x',x_{i+1},\dots,x_\kappa) * y \mapsto (y_1,\dots,y_\kappa) * x_i \simeq y_j)$.*
- *$wpc(\exists \boldsymbol{x}.(\phi * x \mapsto (x_1,\dots,x_\kappa)), x.i := x.j) \overset{def}{=} \exists \boldsymbol{x} \exists x'.(\phi * x \mapsto (x_1,\dots,x_{i-1},x',x_{i+1},\dots,x_\kappa) * x_i \simeq x_j)$.*
- *$wpc(\exists \boldsymbol{x}.(\phi * x \mapsto (x_1,\dots,x_\kappa)), y := x.i) \overset{def}{=} \exists \boldsymbol{x}.((\phi * x \mapsto (x_1,\dots,x_\kappa))\{y \leftarrow x_i\})$ if $x \neq y$. The case where $x = y$ is handled by encoding the action $x := x.i$ as the sequence $z := x; x := z.i$, where z is a special variable in $\mathcal{V}^\star$ not occurring in the considered transition system.*
- *Otherwise, $wpc(\alpha,a)$ is undefined.*

▶ **Example 31.** For instance, we have:

$$\begin{array}{lcl} wpc(x \mapsto (y,z), x.1 := x) & = & \exists x'.(x \mapsto (x',z) * y \simeq x) \\ wpc(x \mapsto (y,z), x := y) & = & y \mapsto (y,z) \\ wpc(x \mapsto (y,z), \mathtt{alloc}(x)) & = & y \simeq x * z \simeq x \end{array}$$

Both $wpc(x \mapsto (y,z), \mathtt{free}(y))$ and $wpc(\mathtt{lseg}(x,y), x.1 := y)$ are undefined.

▶ **Lemma 32.** *Let ϕ be a symbolic heap and let a be an action. If $wpc(\phi, a)$ is defined then for every structure $(\mathfrak{s}, \mathfrak{h})$ such that $(\mathfrak{s}, \mathfrak{h})[a]$ is defined, we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} wpc(\phi, a) \iff (\mathfrak{s}, \mathfrak{h})[a] \models_{\mathcal{R}} \phi$.*

Proof. By inspection of the different cases. ◀

Intuitively, the weakest pre-conditions will be used to propagate towards the initial time all the constraints occurring along the run, while strongest post-conditions will be used to ensure that, at any time, the shape of the heap can be described as a symbolic heap, so that all the conditions that hold along the run can be embedded in a heap constraint.

7 Context Predicates

As shown in the previous section, post and preconditions cannot be defined for all symbolic heaps. Indeed, in some cases, the conditions can be computed only if the consider formula contains some specific points-to atom(s) $x \mapsto (\dots)$, where x is some variable involved in the action (for instance for actions $x.i := y$). In this section we devise an algorithm that, given a symbolic heap ϕ and a variable x, returns a disjunction of symbolic heaps equivalent to ϕ (on structures that allocate x), and such that all symbolic heaps contain a points-to atom of the form $x \mapsto (\boldsymbol{y})$. The latter condition will enable the computation of post and preconditions. To this aim, we consider so-called *context predicates* (adapted from [5]). For every pair of predicates p, q with $arity(p) = n$ and $arity(q) = m$, we define a predicate $(q \multimap p)$ of arity $n + m$ in such a way that $(q \multimap p)(x_1, \dots, x_n, y_1, \dots, y_m)$ is satisfied by all (non empty) structures that will satisfy $p(x_1, \dots, x_n)$ after a disjoint heap satisfying $q(y_1, \dots, y_m)$ is added to the current heap. Intuitively, the rules of $(q \multimap p)$ are defined exactly as those of p, except that exactly one call to $q(y_1, \dots, y_m)$ is removed. More formally, for each rule $p(u_1, \dots, u_n) \Leftarrow \exists \boldsymbol{w}.(u_1 \mapsto (\boldsymbol{y}) * p'(\boldsymbol{z}) * \psi)$ in $\mathcal{R}$ we introduce two rules:

$$(q \multimap p)(u_1, \dots, u_n, v_1, \dots, v_m) \Leftarrow \exists \boldsymbol{w}.(u_1 \mapsto (\boldsymbol{y}) * (q \multimap p')(\boldsymbol{z}, v_1, \dots, v_m) * \psi)$$

$$(q \multimap p)(u_1, \dots, u_n, v_1, \dots, v_m) \Leftarrow \exists \boldsymbol{w}.(u_1 \mapsto (\boldsymbol{y}) * \boldsymbol{z} \simeq (v_1, \dots, v_m) * \psi) \quad \text{if } q = p'$$

It is easy to check that these rules fulfill the conditions of Definition 6. Note that the $\multimap$ operation may be nested, e.g., one may consider predicates such as $(\mathtt{lseg} \multimap (\mathtt{lseg} \multimap \mathtt{lseg}))$. Thus $\mathcal{R}$ is actually infinite, and the rules must be computed on demand.

▶ **Example 33.** For instance $(\mathtt{lseg} \multimap \mathtt{lseg})$ is defined by the rules:

$$(\mathtt{lseg} \multimap \mathtt{lseg})(x, y, u, v) \Leftarrow \exists z.(x \mapsto (z) * z \simeq u * v \simeq y)$$

and

$$(\mathtt{lseg} \multimap \mathtt{lseg})(x, y, u, v) \Leftarrow \exists z.x \mapsto (z) * (\mathtt{lseg} \multimap \mathtt{lseg})(x, z, u, v)$$

The proposed transformation algorithm relies on the use of these context predicates. The idea is that, by Definition 6, a variable x is allocated in a structure validating a predicate atom ϕ iff the corresponding unfolding of ϕ contains a predicate atom of the form $q(z_1, \dots, z_m)$, for some $q \in \mathcal{P}$, where z_1 has the same value as x. Using context predicates it is possible to transform the formula in a way that this atom occurs explicitly in it, since a predicate atom $p(\boldsymbol{y})$ calling $q(\boldsymbol{z})$ is equivalent to $q(\boldsymbol{z}) * (q \multimap p)(\boldsymbol{z}, \boldsymbol{y})$. Then, it suffices to unfold this atom once to get a points-to atom of the form $x \mapsto (\dots)$. More formally:

▶ **Definition 34** (Computation of $\langle\phi\rangle_x$). *Let ϕ be a symbolic heap and let $x \in \mathcal{V}^\star$. The set $\langle\phi\rangle_x$ is defined as follows:*

1. *If $x \in v_{\mapsto}(\phi)$ then $\langle\phi\rangle_x \overset{def}{=} \{\phi\}$.*
2. *Otherwise, $\langle\phi\rangle_x$ is the set of formulas that are of one of the following forms:*
   a. *$\exists \boldsymbol{u}.(x \simeq x' * x \mapsto (\boldsymbol{y}) * \psi)$ where ϕ is of the form $\exists \boldsymbol{u}.(x' \mapsto (\boldsymbol{y}) * \psi)$.*
   b. *$\exists \boldsymbol{u} \exists \boldsymbol{v}.(x \simeq x' * \psi' * \psi)$ where ϕ is of the form $\exists \boldsymbol{u}.(p(x', \boldsymbol{y}) * \psi)$ and $p(x, \boldsymbol{y}) \Leftarrow_{\mathcal{R}} \exists \boldsymbol{v}.\psi'$.*
   c. *$\exists \boldsymbol{u} \exists \boldsymbol{v} \exists z_1 \dots \exists z_m.((q \multimap p)(\boldsymbol{y}, z_1, \dots, z_m) * z_1 \simeq x * \psi' * \psi)$ where ϕ is of the form $\exists \boldsymbol{u}.(p(\boldsymbol{y}) * \psi)$, $q \in \mathcal{P}$, $m = arity(q)$, $z_1, \dots, z_m$ are pairwise distinct fresh variables and $q(x, z_2, \dots, z_m) \Leftarrow_{\mathcal{R}} \exists \boldsymbol{v}.\psi'$.*

Item 1 corresponds to the trivial case where ϕ already contains an atom $x \mapsto (\dots)$. Item 2a corresponds to the case where ϕ contains an atom $x' \mapsto (\dots)$ where $x \simeq x'$ holds. Item 2b handles the case where ϕ contains an atom $p(x', \boldsymbol{y})$ that (immediately) allocates x (by the progress condition this happens iff $x \simeq x'$ holds). Finally, Item 2c tackles the general case, where ϕ contains an atom $p(\boldsymbol{y})$ which (eventually) calls an atom $q(z_1, z_2, \dots, z_m)$ that allocates x. For instance $\langle \mathtt{lseg}(x,y)\rangle_z$ contains the symbolic heaps: $\exists u.(z \mapsto (u) * \mathtt{lseg}(u,y) * x \simeq z)$ and $\exists u, v, w.(\mathtt{lseg} \multimap \mathtt{lseg})(x, y, u, v) * z \mapsto (w) * \mathtt{lseg}(w, v) * u \simeq z)$. Note that both formulas contain a points-to atom of the form $z \mapsto (\dots)$. The following lemmata state that $\langle\phi\rangle_x$ fulfills all the expected properties.

▶ **Lemma 35.** *Let ϕ be a symbolic heap and let $x \in \mathcal{V}^\star$. For every formula $\phi' \in \langle\phi\rangle_x$, $x \in v_{\mapsto}(\phi')$. Thus if $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi'$ then $\mathfrak{s}(x) \in dom(\mathfrak{h})$.*

Proof. Let $\phi' \in \langle\phi\rangle_x$. If $x \in v_{\mapsto}(\phi)$ then $\langle\phi\rangle_x = \{\phi\}$ thus $\phi' = \phi$ and $x \in v_{\mapsto}(\phi')$. In all other cases in Definition 34, either ϕ' contains a points-to atom $x \mapsto (\boldsymbol{y})$, or ϕ' contains a formula ψ' such that there exists an atom α of root x (α is either $p(x, \boldsymbol{y})$ or $q(x, z_2, \dots, z_m)$) such that $\alpha \Leftarrow_{\mathcal{R}} \exists \boldsymbol{v}.\psi'$. By the progress condition necessarily $x \in v_{\mapsto}(\psi')$, so that $x \in v_{\mapsto}(\phi')$. The second part of the lemma follows immediately from the definition of the semantics. ◀

▶ **Lemma 36.** *Let ϕ be a symbolic heap and let $x \in \mathcal{V}^\star$. For every formula $\psi \in \langle\phi\rangle_x$ and for all SL structures $(\mathfrak{s}, \mathfrak{h})$: $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \psi \implies (\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$.*

▶ **Lemma 37.** *Let ϕ be a symbolic heap and let $x \in \mathcal{V}^\star$. For every SL structure $(\mathfrak{s}, \mathfrak{h})$ such that $\mathfrak{s}(x) \in dom(\mathfrak{h})$ and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$, we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \psi$, for some $\psi \in \langle\phi\rangle_x$.*

8 Axioms

Building on the previous results, we define LTL axioms ensuring that an LTL interpretation is compatible with some SL structure, for a given transition system $\mathcal{S} = (Q, R, q_I)$. The axioms are obtained by embedding all the previous definitions and properties in LTL (a, ϕ, γ and x range over the set of actions, symbolic heaps, conditions and variables in $\mathcal{V}^\star$, respectively and t, s are terms).

1. $\boldsymbol{G}(x.i \approx s \Rightarrow \mathtt{A}(x))$ for all $i \in \{1, \dots, \kappa\}$.
2. $\boldsymbol{G}(a \Rightarrow (\psi \Rightarrow \boldsymbol{X}\, spc(\psi, a)))$ (if $spc(\psi, a)$ is defined).
3. $\boldsymbol{G}(a \Rightarrow (wpc(\psi, a) \Leftrightarrow \boldsymbol{X}\, \psi))$ (if $wpc(\psi, a)$ is defined).
4. $\boldsymbol{G}(\mathtt{A}(x) \Rightarrow (\psi \Leftrightarrow \bigvee_{\xi \in \langle\psi\rangle_x} \xi)) \wedge \boldsymbol{G}(\psi \Rightarrow \bigwedge_{y \in v_{\mapsto}(\psi)} \mathtt{A}(y))$.
5. $\boldsymbol{G}(\exists \boldsymbol{u}.(x \mapsto (x_1, \dots, x_\kappa) * \psi) \Rightarrow (x.i \approx y \Leftrightarrow \exists \boldsymbol{u}.(x \mapsto (x_1, \dots, x_\kappa) * \psi * x_i \simeq y)))$, if $y \in \mathcal{V}^\star$.
6. $\boldsymbol{G}(\exists \boldsymbol{u}.(x \mapsto (x_1, \dots, x_\kappa) * y \mapsto (y_1, \dots, y_\kappa) * \psi) \Rightarrow (x.i \approx y.j \Leftrightarrow \exists \boldsymbol{u}.(x \mapsto (x_1, \dots, x_\kappa) * y \mapsto (y_1, \dots, y_\kappa) * \psi * x_i \simeq y_j)))$.
7. $\boldsymbol{G}((\exists \boldsymbol{u}.\psi) \Rightarrow (x \approx y \Leftrightarrow \exists \boldsymbol{u}.(\psi * x \simeq y)))$.

8. $\boldsymbol{G}((\mathtt{pass} \vee t := s \vee \mathtt{test}(\gamma)) \Rightarrow (\mathtt{A}(x) \Leftrightarrow \boldsymbol{X}\,\mathtt{A}(x)))$, where $t \neq x$.
9. $\boldsymbol{G}(\mathtt{free}(x) \Rightarrow \bigwedge_{y \in \mathcal{V}^\star}((x \approx y \Rightarrow \boldsymbol{X}\,\neg\mathtt{A}(y))) \wedge (x \not\approx y \Rightarrow (\mathtt{A}(y) \Leftrightarrow \boldsymbol{X}\,\mathtt{A}(y))))$.
10. $\boldsymbol{G}(\mathtt{alloc}(x) \Rightarrow \bigwedge_{y \in \mathcal{V}^\star}((x \approx y \Rightarrow \boldsymbol{X}\,\mathtt{A}(y))) \wedge (x \not\approx y \Rightarrow (\mathtt{A}(y) \Leftrightarrow \boldsymbol{X}\,\mathtt{A}(y))))$.
11. $\boldsymbol{G}(\neg x \vee \neg y)$, if $x \neq y$ and (either x, y are both actions, or $\{x, y\} \subseteq Q$).
12. $\boldsymbol{G}(q \Rightarrow \bigvee_{(q,a,q') \in R}(a \wedge \boldsymbol{X}\,q))$.
13. $\boldsymbol{G}(a \Rightarrow \mathit{pre}(a))$.
14. $\bigwedge_{\psi \in S^+} \phi \wedge \bigwedge_{x \in V} \neg\mathtt{A}(x) \Rightarrow \bigvee_{\xi \in S^-} \phi$, if (S^+, S^-, X) is a unsatisfiable heap constraint. This formula is denoted by $\Gamma(S^+, S^-, X)$ in the following.

This set of axioms is infinite, as the set of symbolic heaps is infinite. To ensure termination, we need to further restrict the axioms. To this aim, we define (given a symbolic heap ϕ) two sets $\mathtt{Fw}(\mathcal{S}, \phi)$ and $\mathtt{Bw}(\mathcal{S}, \phi, \Phi)$, which, informally, contain triples (ψ, q, X), where ψ denotes a symbolic heap obtained by (forward or backward) propagation along the runs in $\mathcal{S}$ (starting from formulas occurring in the initial entailment), and q is the corresponding state. The set X contains variables that either occur in $v_{\mapsto}(\phi)$ or are known to be non allocated at state q (this information is essential for finiteness because it allows one to "block" some generation rules). The sets are defined inductively as follows:

- $(\phi, q_I, \emptyset) \in \mathtt{Fw}(\mathcal{S}, \phi)$, and if $q \in Q$ and ψ occurs in Φ then $(\psi, q, \emptyset) \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$.
- If $(\psi, q, X) \in \mathtt{Fw}(\mathcal{S}, \phi)$, $(q, a, q') \in R$ and $\phi' = \mathit{spc}(\psi, a)$ then $(\psi', q', X') \in \mathtt{Fw}(\mathcal{S}, \phi)$, where $X' = X$ if a is not of the form $x := t$ with $x \in \mathcal{V}^\star$ and otherwise $X' = \emptyset$.
- If $(\psi, q', X) \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$, $(q, a, q') \in R$ and $\phi' = \mathit{wpc}(\psi, a)$ then $(\phi', q, X') \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$, where $X' = \emptyset$ if a is of the form $x := t$ with $x \in \mathcal{V}^\star$, and $X' = X$ otherwise.
- If $(\psi, q, X) \in \mathtt{Fw}(\mathcal{S}, \phi)$ (resp. $(\psi, q, X) \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$) and $\xi \in \langle\psi\rangle_x$ with $x \in \mathcal{V}^\star \setminus X$ then $(\xi, q, X \cup \{x\}) \in \mathtt{Fw}(\mathcal{S}, \phi)$ (resp. $(\xi, q, X \cup \{x\}) \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$).
- If $(\exists \boldsymbol{u}.\psi, q, X) \in \mathtt{Fw}(\mathcal{S}, \phi)$ then $(\exists \boldsymbol{u}.(\psi \wedge x \simeq y), q, X) \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$, for all $x, y \in \mathit{fv}(\psi) \cup \mathcal{V}^\star$.

The sets $\mathtt{Fw}(\mathcal{S}, \phi)$ and $\mathtt{Bw}(\mathcal{S}, \phi, \Phi)$ are finite (up to some simplifications) if $\mathcal{S}$ is oriented (see Lemma 41 in Appendix D). We denote by $\mathcal{A}(\mathcal{R}, \mathcal{S}, \phi)$ the set of axioms satisfying the following conditions. For Axiom 3 we require that the considered symbolic heap ψ occurs in some triple in $\mathtt{Bw}(\mathcal{S}, \phi, \Phi)$. For Axiom 14 all the symbolic heaps in S^+ and S^- must occur in $\mathtt{Bw}(\mathcal{S}, \phi, \Phi)$. For Axiom 4, ψ must occur in either $\mathtt{Fw}(\mathcal{S}, \phi)$ or $\mathtt{Bw}(\mathcal{S}, \phi, \Phi)$. For 5, 6 and 7, the symbolic heap at the left-hand side of $\Rightarrow$ must occur in $\mathtt{Fw}(\mathcal{S}, \phi)$ (which entails that the one occurring at the right-hand side occurs in $\mathtt{Bw}(\mathcal{S}, \phi, \Phi)$). The following theorems relate the considered entailment problem with standard LTL satisfiability.

▶ **Theorem 38.** *Every LTL model $\mathcal{I}$ that is compatible with $(\mathfrak{s}, \mathfrak{h})$ and $\mathcal{S}$ w.r.t. all symbolic heaps occurring in $\mathcal{A}(\mathcal{R}, \mathcal{S}, \phi) \cup \{\phi, q_I, \Phi\}$ satisfies $\mathcal{A}(\mathcal{R}, \mathcal{S}, \phi) \cup \{\phi, q_I, \Phi\}$.*

Proof (Sketch). The soundness of Axioms 2 and 3 stems from Lemmata 29 and 32, respectively. The soundness of Axiom 13 stems from Proposition 26. Axioms 12 and 11 encode the semantics of actions and states, according to the transition system $\mathcal{S}$. The soundness of Axiom 4 is a consequence of Lemmata 36 and 37. The soundness of Axioms 14 follows from the semantics of heap constraints. The soundness of Axioms 8,9, 10 is a consequence of Definition 11. Finally, the soundness of Axioms 1, 5, 6 and 7 stems from the semantics of atomic conditions (Axioms 5, 6 and 7 embed conditions of the form $t \simeq s$ into symbolic heaps). ◀

▶ **Theorem 39.** *If $\mathcal{A}(\mathcal{R}, \mathcal{S}, \phi) \cup \{\phi, q_I, \Phi\}$ admits an LTL model $\mathcal{I}$ then there exists a structure $(\mathfrak{s}, \mathfrak{h})$ such that $\mathcal{I}$ is compatible with $(\mathfrak{s}, \mathfrak{h})$ and $\mathcal{S}$, w.r.t. ϕ and all symbolic heaps in Φ.*

9 Proof Procedure

Algorithm 1 Entailment Checking Algorithm.

```
Require: A progressing, connected and established SID $\mathcal{R}$, an oriented transition system $\mathcal{S}$,
Require: a symbolic heap $\phi$ and an LTL formula $\Phi$
  $\mathcal{A} \leftarrow \{\phi, q_I, \neg\Phi\}$
  while $\mathcal{A}$ admits an LTL interpretation $\mathcal{I}$ do
    $S^+ \leftarrow \{\phi \in \mathbb{A}_{\mathcal{S}} \mid \mathcal{I}(\phi, 0) = true,\ \phi$ is a symbolic heap$\}$
    $S^- \leftarrow \{\phi \in \mathbb{A}_{\mathcal{S}} \mid \mathcal{I}(\phi, 0) = false,\ \phi$ is a symbolic heap$\}$
    $X \leftarrow \{x \in \mathcal{V}^\star \mid \mathcal{I}(\phi, 0) \not\models \mathtt{A}(x)\ \}$
    if $(S^+, S^-, X)$ is unsatisfiable {This test is decidable by Lemma 9} then
      $\mathcal{A} \leftarrow \mathcal{A} \cup \Gamma(S^+, S^-, X)$
    else
      Let $(\mathfrak{s}, \mathfrak{h})$ be an $\mathcal{R}$-model of $(S^+, S^-, X)$
      if $r_{\mathcal{I}}$ is defined and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{\mathcal{S}/r_{\mathcal{I}}} \neg\Phi$ {the test is decidable by Lemma 22} then
        Return $(\mathfrak{s}, \mathfrak{h})$
      else
        Let $\Psi$ be a formula in $\mathcal{A}(\mathcal{R}, \mathcal{S}, \phi)$ s.t. $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathcal{R}}^{\mathcal{S}} \Psi$ {$\Psi$ exists by Theorem 39}
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{\Psi\}$
      end if
    end if
  end while
  Return $\top$
```

Even if $\mathcal{S}$ is oriented, the set $\mathcal{A}(\mathcal{R}, \mathcal{S}, \phi)$ is exponential w.r.t. the size of $\mathcal{R}$, ϕ and $\mathcal{S}$, and only a small part of this set will be relevant, hence computing all axioms explicitly is not practical. Algorithm 1 computes these axioms on demand, in the spirit of the well-known *DPLL*($\mathcal{T}$) procedure (see, e.g., [9]) by calling external tools to solve LTL and SL satisfiability problems. The idea is to construct an LTL interpretation and to refine it incrementally by adding relevant axioms until we get either a model that is compatible with some SL structure, or a set of axioms that is unsatisfiable (in LTL). For all LTL interpretations $\mathcal{I}$, $r_{\mathcal{I}}$ is the sequence $(q_i, a_i)_{i \in \mathbb{N}}$ (if it exists) such that q_i is the unique state in Q (resp. the only action) with $\mathcal{I}(q_i, i) = true$ (resp. $\mathcal{I}(a_i, i) = true$).

▶ **Theorem 40.** *If Algorithm 1 returns $\top$ then the entailment $\phi \models_{\mathcal{R}}^{\mathcal{S}} \Phi$ holds. If it returns an SL structure $(\mathfrak{s}, \mathfrak{h})$ then $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathcal{R}}^{\mathcal{S}} \Phi$. Moreover, if $\mathcal{S}$ is oriented then the algorithm always terminates.*

Proof. Termination is immediate (if $\mathcal{S}$ is oriented) since at each iteration one new formula from $\mathcal{A}(\mathcal{R}, \mathcal{S}, \phi)$ is added in $\mathcal{A}$ and the set $\mathcal{A}(\mathcal{R}, \mathcal{S}, \phi)$ is finite (as $\mathtt{Bw}(\mathcal{S}, \phi, \Phi)$ and $\mathtt{Fw}(\mathcal{S}, \phi)$ are both finite). If $\top$ is returned then by definition of the algorithm $\mathcal{A}(\mathcal{R}, \mathcal{S}, \phi) \cup \{q_I, \phi, \neg\Phi\}$ is unsatisfiable thus the entailment $\phi \models_{\mathcal{R}}^{\mathcal{S}} \Phi$ is valid by Theorem 38. If the algorithm returns a structure $(\mathfrak{s}, \mathfrak{h})$ then by definition $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}}^{\mathcal{S}/(q_i, a_i)_{i \in \mathbb{N}}} \neg\Phi$ for some sequence $(q_i, a_i)_{i \in \mathbb{N}}$, thus there is a run $(q_i, \mathfrak{s}_i, \mathfrak{h}_i, a_i)_{i \in \mathbb{N}}$ and a compatible LTL interpretation $\mathcal{I}$ such that $\mathcal{I} \not\models \Phi$. ◀

10 Discussion

A natural issue is to determine whether Algorithm 1 is complete for refutation (when $\mathcal{S}$ is not oriented), i.e., whether it always returns a counter model if the entailment is not valid (by Theorem 23 it cannot be complete for validity). Another natural continuation is to extend the expressive power of the logic by considering more complex temporal connectives (to allow for quantification over paths). It would also be interesting to extend the language in order to handle more complex (possibly non deterministic) actions. For instance, it should be noticed that actions in our framework cannot create new locations (as evidenced by Proposition 13). This is important, because, otherwise, since universal quantification is not allowed, the corresponding pre/post-conditions could not be expressed in the language. This entails that C-like allocations for instance are not built-in: they must be performed by handling a stack of available locations, allocated in the symbolic heap describing the initial state of the system by an atom such as $\mathtt{lseg}(x, y)$ (an instruction such as $\mathtt{malloc}(z)$ can be simulated by two actions $z := x$ and $x := x.1$). The complexity of the entailment problem for oriented systems also deserves to be precisely identified (it is 2-EXPTIME hard by [4]).

References

1 Callum Bannister, Peter Höfner, and Gerwin Klein. Backwards and forwards with separation logic. In Jeremy Avigad and Assia Mahboubi, editors, *ITP 2018, FloC 2018, Oxford, UK, July 9–12, 2018, Proceedings*, volume 10895 of *LNCS*, pages 68–87. Springer, 2018.

2 Rémi Brochenin, Stéphane Demri, and Étienne Lozes. Reasoning about sequences of memory states. In Sergei N. Artëmov and Anil Nerode, editors, *LFCS 2007, New York, NY, USA, June 4–7, 2007, Proceedings*, volume 4514 of *LNCS*, pages 100–114. Springer, 2007.

3 James Brotherston, Carsten Fuhs, Juan Antonio Navarro Pérez, and Nikos Gorogiannis. A decision procedure for satisfiability in separation logic with inductive predicates. In Thomas A. Henzinger and Dale Miller, editors, *CSL-LICS '14, Vienna, Austria, July 14–18, 2014*, pages 25:1–25:10. ACM, 2014.

4 Mnacho Echenim, Radu Iosif, and Nicolas Peltier. Entailment checking in separation logic with inductive definitions is 2-exptime hard. In *LPAR 2020, Alicante, Spain, May 22–27, 2020*, volume 73 of *EPiC Series in Computing*, pages 191–211. EasyChair, 2020.

5 Mnacho Echenim, Radu Iosif, and Nicolas Peltier. Decidable entailments in separation logic with inductive definitions: Beyond establishment. In *CSL 2021*, EPiC Series in Computing. EasyChair, 2021.

6 Didier Galmiche and Daniel Méry. Labelled tableaux for linear time bunched implication logic. In *ASL 2022 (Workshop on Advancing Separation Logic)*, 2022.

7 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.

8 Norihiro Kamide. Temporal BI: proof system, semantics and translations. *Theor. Comput. Sci.*, 492:40–69, 2013.

9 Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *J. ACM*, 53(6):937–977, 2006.

10 Peter W. O'Hearn and David J. Pym. The logic of bunched implications. *Bull. Symb. Log.*, 5(2):215–244, 1999.

11 Jens Pagel, Christoph Matheja, and Florian Zuleger. Complete entailment checking for separation logic with inductive definitions, 2020. arXiv:2002.01202.

12 Jens Pagel and Florian Zuleger. Beyond symbolic heaps: Deciding separation logic with inductive definitions. In *LPAR-23*, volume 73 of *EPiC Series in Computing*, pages 390–408. EasyChair, 2020.

13 Amir Pnueli. The temporal logic of programs. In *FOCS, Providence, Rhode Island, USA*, pages 46–57. IEEE Computer Society, 1977.

14 J.C. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *Proc. of LICS'02*, 2002.

A Proof of Lemma 9

We use the algorithm developed in [5] to test the validity of entailments between SL formulas (if the considered SID is progressing, connected and established), combined with the technique devised in [12] to cope with conjunctions (see also [11]). Let $X = \{x_1, \dots, x_n\}$, where the order on the x_i is arbitrary. For every $i = 1, \dots, n$, we denote by Ψ_i the formula: $(\bigvee_{j=1}^{i-1} x_i \simeq x_j) \vee x_i \mapsto (x_i, \dots, x_i)$. Let $\Psi = \Psi_1 * \dots * \Psi_n$. By definition, if $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \Psi$ then $dom(\mathfrak{h}') = \{\mathfrak{s}(x) \mid x \in X\}$, hence, for every structure $(\mathfrak{s}, \mathfrak{h})$, there is at most one heap $\mathfrak{h}' \subseteq \mathfrak{h}$ with $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \Psi$. Moreover, for all stores $\mathfrak{s}$, we have $(\mathfrak{s}, \mathfrak{h}_{\mathfrak{s}}) \models_{\mathcal{R}} \Psi$, where $\mathfrak{h}_{\mathfrak{s}}$ denotes the heap: $\{(\ell, \dots, \ell) \mid \exists x \in X \text{ s.t. } \ell = \mathfrak{s}(x)\}$. Let $\Phi = \bigwedge_{\phi \in S^+} (\Psi * \phi) \wedge \neg(\bigvee_{\phi \in S^-} (\Psi * \phi))$. Note that since S^+ is not empty, Φ is a guarded formula (as defined in [12, Fig. 1]), except that it contains existential quantifiers (the fact that S^+ is non empty is essential, as otherwise the negation would not be guarded). The satisfiability of Φ can be tested by combining the techniques devised in [12] and [5]. The idea is to compute an abstraction of the possible models of Φ bottom-up. Points-to atoms, inductive predicates, separating conjunctions and existential quantifications can be handled as explained in [5], whereas conjunctions and guarded negations are handled as it is done in [12]. We prove that (S^+, S^-, X) is satisfiable iff Φ is satisfiable:

- Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} (S^+, S^-, X)$. Then $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ for all $\phi \in S^+$, $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathcal{R}} \phi$ for all $\phi \in S^-$, and $\mathfrak{s}(x) \notin dom(\mathfrak{h})$ for all $x \in X$. Then $\mathfrak{h}_{\mathfrak{s}}$ and $\mathfrak{h}$ are disjoint, thus we get $(\mathfrak{s}, \mathfrak{h} \uplus \mathfrak{h}_{\mathfrak{s}}) \models_{\mathcal{R}} \phi * \Psi$ for all $\phi \in S^+$. If $(\mathfrak{s}, \mathfrak{h} \uplus \mathfrak{h}_{\mathfrak{s}}) \models_{\mathcal{R}} \phi * \Psi$ for some $\phi \in S^-$ then since $\mathfrak{h}_{\mathfrak{s}}$ is the unique heap such that $(\mathfrak{s}, \mathfrak{h}_{\mathfrak{s}}) \models_{\mathcal{R}} \Psi$, we deduce that we must have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$, which contradicts our assumption. Thus $(\mathfrak{s}, \mathfrak{h} \uplus \mathfrak{h}_{\mathfrak{s}}) \models_{\mathcal{R}} \Phi$.
- Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \Phi$. Then, we get $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi * \Psi$, for all $\phi \in S^+$. Since $\mathfrak{h}_{\mathfrak{s}}$ is the unique heap such that $(\mathfrak{s}, \mathfrak{h}_{\mathfrak{s}}) \models_{\mathcal{R}} \Psi$, this entails that $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \phi$, with $\mathfrak{h}' = \mathfrak{h} \setminus \mathfrak{h}_{\mathfrak{s}}$. Since $dom(\mathfrak{h}_{\mathfrak{s}}) = \{\mathfrak{s}(x) \mid x \in X\}$ we get $\mathfrak{s}(x) \notin dom(\mathfrak{h}')$, for all $x \in X$. If $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} \phi$, for some $\phi \in S^-$ then we deduce $(\mathfrak{s}, \mathfrak{h}' \uplus \mathfrak{h}_{\mathfrak{s}}) \models_{\mathcal{R}} \phi * \Psi$, which contradicts our hypothesis. Thus $(\mathfrak{s}, \mathfrak{h}') \models_{\mathcal{R}} (S^+, S^-, X)$.

B Proof of Lemma 36

Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \psi$. We show, by induction on $|\mathfrak{h}|$, that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$. If $x \in v_{\mapsto}(\phi)$ then $\langle \phi \rangle_x = \{\phi\}$ thus $\phi = \psi$ and the proof is immediate. Otherwise, we distinguish the following cases, following Definition 34:

- $\phi' = \exists \boldsymbol{u}.(x \simeq x' * x \mapsto (\boldsymbol{y}) * \psi)$ and $\phi = \exists \boldsymbol{u}.(x' \mapsto (\boldsymbol{y}) * \psi)$. It is clear that $\phi' \models_{\mathcal{R}} \phi$.
- $\phi' = \exists \boldsymbol{u} \exists \boldsymbol{v}.(x \simeq x' * \psi' * \psi)$, and $\phi = \exists \boldsymbol{u}.(p(x', \boldsymbol{y}) * \psi)$ with $p(x, \boldsymbol{y}) \Leftarrow_{\mathcal{R}} \exists \boldsymbol{v}.\psi'$. In this case, we get $\phi' \models_{\mathcal{R}} \exists \boldsymbol{u}.(x \simeq x' * p(x, \boldsymbol{y}) * \psi)$, thus $\phi' \models_{\mathcal{R}} \exists \boldsymbol{u}.(p(x', \boldsymbol{y}) * \psi) = \phi$.
- $\phi' = \exists \boldsymbol{u} \exists \boldsymbol{v} \exists z_1 \dots \exists z_m.((q \mathrel{-\!\!\bullet} p)(\boldsymbol{y}, z_1, \dots, z_m) * z_1 \simeq x * \psi' * \psi)$ and $\phi = \exists \boldsymbol{u}.(p(\boldsymbol{y}) * \psi)$, with $q(x, z_2, \dots, z_m) \Leftarrow_{\mathcal{R}} \exists \boldsymbol{v}.\psi'$. Then we get $\phi' \models_{\mathcal{R}} \exists \boldsymbol{u} \exists z_1 \dots \exists z_m.((q \mathrel{-\!\!\bullet} p)(\boldsymbol{y}, z_1, \dots, z_m) * z_1 \simeq x * q(x, z_2, \dots, z_m) * \psi)$, and by definition of the rules associated with the predicate $(q \mathrel{-\!\!\bullet} p)$, one of the following conditions holds (with $\boldsymbol{y} = (y_1, \dots, y_n)$):
  - $\phi' \models_{\mathcal{R}} \exists \boldsymbol{u} \exists z_1 \dots \exists z_m \exists \boldsymbol{w} \exists \boldsymbol{v}.(y_1 \mapsto (\boldsymbol{y}') * (q \mathrel{-\!\!\bullet} p')(\boldsymbol{z}', z_1, \dots, z_m) * \psi'' * z_1 \simeq x * \psi' * \psi)$ with $p(\boldsymbol{y}) \Leftarrow_{\mathcal{R}} y_1 \mapsto (\boldsymbol{y}') * p'(\boldsymbol{z}') * \psi''$. This entails that there exists a store $\mathfrak{s}'$ coinciding with $\mathfrak{s}$ with all the variables not occurring in $\boldsymbol{u}, z_1, \dots, z_m, \boldsymbol{w}, \boldsymbol{v}$ and disjoint

heaps $\mathfrak{h}', \mathfrak{h}''$ such that $\mathfrak{h} = \mathfrak{h}' \uplus \mathfrak{h}''$, $(\mathfrak{s}', \mathfrak{h}') \models_{\mathcal{R}} y_1 \mapsto (\boldsymbol{y}') * \psi''$ and $(\mathfrak{s}', \mathfrak{h}'') \models_{\mathcal{R}} (q \multimap p')(\boldsymbol{z}', z_1, \dots, z_m) * z_1 \simeq x * \psi' * \psi$. This entails that $\mathfrak{h}' \neq \emptyset$ thus $|\mathfrak{h}| > |\mathfrak{h}''|$. By definition, $(q \multimap p')(\boldsymbol{z}', z_1, \dots, z_m) * z_1 \simeq x * \psi' * \psi \in \langle p'(\boldsymbol{z}') * \psi \rangle_x$, hence by the induction hypothesis we get $(\mathfrak{s}', \mathfrak{h}'') \models_{\mathcal{R}} p'(\boldsymbol{z}') * \psi$, so that $(\mathfrak{s}', \mathfrak{h}) \models_{\mathcal{R}} y_1 \mapsto (\boldsymbol{y}') * \psi'' * p'(\boldsymbol{z}') * \psi$, hence $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \exists \boldsymbol{u}.(p(\boldsymbol{y}) * \psi) = \phi$.

- $\phi' \models_{\mathcal{R}} \exists \boldsymbol{u} \exists z_1 \dots \exists z_m \exists \boldsymbol{v} \exists \boldsymbol{w}.(y_1 \mapsto (\boldsymbol{y}') * \boldsymbol{z}' \simeq (z_1, \dots, z_m) * \psi'' * z_1 \simeq x * \psi' * \psi)$ with $p(\boldsymbol{y}) \Leftarrow_{\mathcal{R}} \exists \boldsymbol{w}.(y_1 \mapsto (\boldsymbol{y}') * q(\boldsymbol{z}') * \psi'')$. Since $q(x, z_2, \dots, z_m) \Leftarrow_{\mathcal{R}} \exists \boldsymbol{v}.\psi'$, we deduce that $\phi' \models_{\mathcal{R}} \exists \boldsymbol{u} \exists z_1 \dots \exists z_m \exists \boldsymbol{w}.(y_1 \mapsto (\boldsymbol{y}') * \boldsymbol{z}' \simeq (z_1, \dots, z_m) * \psi'' * z_1 \simeq x * q(x, z_2, \dots, z_m) * \psi)$, thus $\phi' \models_{\mathcal{R}} \exists \boldsymbol{u} \exists \boldsymbol{w}.(y_1 \mapsto (\boldsymbol{y}') * \psi'' * q(\boldsymbol{z}') * \psi)$, hence $\phi' \models_{\mathcal{R}} \exists \boldsymbol{u}.(p(\boldsymbol{y}) * \psi) = \phi$.

C Proof of Lemma 37

Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ and that $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h})$. We show, by induction on $|\mathfrak{h}|$, that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \psi$, for some $\psi \in \langle \phi \rangle_x$. The symbolic heap ϕ is necessarily of the form $\exists \boldsymbol{u}.(\phi_1 * \dots * \phi_k)$, where the $\phi_1, \dots, \phi_k$ are atoms. We assume by α-renaming that x does not occur in $\boldsymbol{u}$. By definition of the semantics of SL, there exists a store $\mathfrak{s}'$ (coinciding with $\mathfrak{s}$ on all variables not occurring in $\boldsymbol{u}$) and disjoint heaps $\mathfrak{h}_i$ such that $(\mathfrak{s}', \mathfrak{h}_i) \models_{\mathcal{R}} \phi_i$ (for all $i = 1, \dots, n$) and $\mathfrak{h} = \mathfrak{h}_1 \uplus \dots \uplus \mathfrak{h}_k$. Since $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h})$, necessarily $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h}_i)$ for some $i = 1, \dots, k$, say $i = 1$. Let $\phi' = \phi_2 * \dots * \phi_k$. We distinguish several cases.

- Assume that ϕ_1 is a points-to atom $x' \mapsto (\boldsymbol{y})$. If $x = x'$, then $x \in v_{\mapsto}(\phi)$, so that $\langle \phi \rangle_x = \{\phi\}$, hence the proof is immediate. Otherwise, since (by definition of the semantics of SL) $\mathit{dom}(\mathfrak{h}_1) = \{\mathfrak{s}'(x')\}$ and $\mathfrak{s}(x) \in \mathfrak{h}_1$ we must have $\mathfrak{s}(x') = \mathfrak{s}(x) = \mathfrak{s}'(x)$. By Definition 34 (2a), $\langle \phi \rangle_x$ contains a formula of the form $\exists \boldsymbol{u}.(x \mapsto (\boldsymbol{y}) * x \simeq x' * \phi')$. We have $(\mathfrak{s}', \mathfrak{h}) \models_{\mathcal{R}} x \mapsto (\boldsymbol{y}) * x \simeq x' * \phi'$, so that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \exists \boldsymbol{u}.(x \mapsto (\boldsymbol{y}) * x \simeq x' * \phi')$.
- Assume that ϕ_1 is a predicate atom $p(x', \boldsymbol{y})$ and that $\mathfrak{s}'(x') = \mathfrak{s}(x)$. Then we get $(\mathfrak{s}', \mathfrak{h}) \models_{\mathcal{R}} p(x, \boldsymbol{y}) * x \simeq x' * \phi'$, so that $(\mathfrak{s}', \mathfrak{h}) \models_{\mathcal{R}} \exists \boldsymbol{v}.(\psi' * x \simeq x' * \phi')$ with $p(x, \boldsymbol{y}) \Leftarrow_{\mathcal{R}} \exists \boldsymbol{v}.\psi'$. Thus $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \exists \boldsymbol{u} \exists \boldsymbol{v}.(\psi' * x \simeq x' * \phi')$, and by Def. 34 (2b), this formula is in $\langle \phi \rangle_x$.
- Finally, assume that ϕ_1 is a predicate atom $p(x', \boldsymbol{y})$ and that $\mathfrak{s}'(y_1) \neq \mathfrak{s}(x)$. Necessarily, $p(x', \boldsymbol{y}) \Leftarrow_{\mathcal{R}} \exists \boldsymbol{v}.(x' \mapsto (\boldsymbol{y}') * \psi)$ with $(\mathfrak{s}'', \mathfrak{h}_1) \models_{\mathcal{R}} x' \mapsto (\boldsymbol{y}') * \psi$, for some store $\mathfrak{s}''$ coinciding with $\mathfrak{s}'$ on all variables not occurring in $\boldsymbol{v}$. Since $\mathfrak{s}'(y_1) \neq \mathfrak{s}(x)$ and $\mathfrak{s}(x) \in \mathit{dom}(\mathfrak{h}_1)$, ψ must be of the form $p'(\boldsymbol{z}) * \psi'$ (with possibly $\psi' = \mathit{emp}$) and there exist disjoint heaps $\mathfrak{h}', \mathfrak{h}''$ such that $\mathfrak{h}_1 = \mathfrak{h}' \uplus \mathfrak{h}''$, $(\mathfrak{s}'', \mathfrak{h}') \models_{\mathcal{R}} x' \mapsto (\boldsymbol{y}') * \psi'$ and $(\mathfrak{s}'', \mathfrak{h}'') \models_{\mathcal{R}} p'(\boldsymbol{z})$. This entails that $\mathfrak{h}' \neq \emptyset$, thus $|\mathfrak{h}''| < |\mathfrak{h}|$, and by the induction hypothesis, we deduce that $\langle p'(\boldsymbol{z}) \rangle_x$ contains a formula ψ'' such that $(\mathfrak{s}'', \mathfrak{h}'') \models_{\mathcal{R}} \psi''$. We get $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \exists \boldsymbol{u} \exists \boldsymbol{v}.(x' \mapsto (\boldsymbol{y}') * \psi'' * \psi' * \phi')$. By Definition 34, ψ'' is of one of the following forms:

2b $\psi'' = \exists \boldsymbol{w}.(\xi * z_1 \simeq x)$, with $\boldsymbol{z} = (z_1, \dots, z_m)$ and $p'(x, z_2, \dots, z_m) \Leftarrow_{\mathcal{R}} \exists \boldsymbol{w}.\xi$. Then we get $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \exists \boldsymbol{u} \exists \boldsymbol{v} \exists \boldsymbol{w}.(x' \mapsto (\boldsymbol{y}') * \xi * z_1 \simeq x * \psi' * \phi')$. By definition of the rules defining $(p' \multimap p)$, we have: $(p' \multimap p)(x', \boldsymbol{y}, \boldsymbol{z})) \Leftarrow_{\mathcal{R}} \exists \boldsymbol{v}.(x' \mapsto (\boldsymbol{y}') * \psi' * \boldsymbol{z} \simeq \boldsymbol{z})$, so that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \exists \boldsymbol{u} \exists \boldsymbol{w}.((p' \multimap p)(x', \boldsymbol{y}, \boldsymbol{z})) * \xi * z_1 \simeq x * \phi')$, hence $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \exists \boldsymbol{u} \exists \boldsymbol{w} \exists \boldsymbol{z}'.((p' \multimap p)(x', \boldsymbol{y}, \boldsymbol{z}')) * \xi * z_1' \simeq x * \phi')$, where $\boldsymbol{z}' = (z_1', \dots, z_m')$ is a vector of fresh pairwise distinct variables. By Definition 34 (2c), the latter formula occurs in $\langle \phi \rangle_x$.

2c $\psi'' = \exists z_1', \dots, z_m'.((q \multimap p')(\boldsymbol{z}, z_1', \dots, z_m') * z_1' \simeq x * \xi)$, with $q(x, z_2', \dots, z_m') \Leftarrow_{\mathcal{R}} \xi$. We get $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \exists \boldsymbol{u} \exists \boldsymbol{v} \exists z_1', \dots, z_m'.(x' \mapsto (\boldsymbol{y}') * (q \multimap p')(\boldsymbol{z}, z_1', \dots, z_m') * z_1' \simeq x * \xi * \psi' * \phi')$. By definition of the rules defining $(q \multimap p)$, we have $(q \multimap p)(x', \boldsymbol{y}, z_1', \dots, z_m') \Leftarrow_{\mathcal{R}} \exists \boldsymbol{v}.(x' \mapsto (\boldsymbol{y}') * (q \multimap p')(\boldsymbol{z}, z_1', \dots, z_m') * \psi')$, thus $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \exists \boldsymbol{u} \exists z_1', \dots, z_m'.((q \multimap p)(x', \boldsymbol{y}, z_1', \dots, z_m') * z_1' \simeq x * \xi * \phi')$. By Definition 34 (2c), this formula is in $\langle \phi \rangle_x$.

D Finiteness of $\mathtt{Bw}(\mathcal{S},\phi,\Phi)$ and $\mathtt{Fw}(\mathcal{S},\phi)$

We write $\phi \rightarrow_s \psi$ if ψ is obtained from ϕ by using one of the above simplification rules.

$$\mathtt{C}_{\simeq} : \exists \boldsymbol{u}.(x \not\simeq x * \xi) \rightarrow \bot \qquad \mathtt{E}_{\not\simeq} : \exists \boldsymbol{u} \exists x.(x \not\simeq x_1 * \dots x \not\simeq x_n * \xi) \rightarrow \exists \boldsymbol{u}.\xi \text{ if } x \not\in \mathit{fv}(\xi) \cup \{x_1,\dots,x_n\}$$

$$\mathtt{C}_{*} : \exists \boldsymbol{u}.(x \mapsto (\boldsymbol{y}) * x \mapsto (\boldsymbol{z}) * \xi) \rightarrow \bot \qquad \mathtt{E}_{\simeq} : \exists \boldsymbol{u} \exists x.(x \simeq y * \xi) \rightarrow \exists \boldsymbol{u}.\xi\{x \leftarrow y\}$$

It is easy to verify that $\rightarrow_s$ is well-founded, and that $\phi \rightarrow_s \psi \implies \phi \equiv_{\mathcal{R}} \psi$.

▶ **Lemma 41.** *If $\mathcal{S}$ is oriented then the sets $\mathtt{Bw}(\mathcal{S},\phi,\Phi)$ and $\mathtt{Fw}(\mathcal{S},\phi)$ are finite (up to associativity and commutativity of $*$, α-renaming and equivalence w.r.t. $\rightarrow_s$).*

Proof. We assume that all symbolic heaps are in normal form w.r.t. $\rightarrow_s$. Let $\mathcal{S} = (Q, R, q_I)$. We give the proof for $\mathtt{Bw}(\mathcal{S},\phi,\Phi)$, the set $\mathtt{Fw}(\mathcal{S},\phi)$ can be handled in a similar way (the only difference is that one must consider the order $\preceq_{\mathcal{S}}$ instead of $\succeq_{\mathcal{S}}$, and that $\mathtt{Fw}(\mathcal{S},\phi)$ does not depend on $\mathtt{Bw}(\mathcal{S},\phi,\Phi)$, while $\mathtt{Bw}(\mathcal{S},\phi,\Phi)$ depends on $\mathtt{Fw}(\mathcal{S},\phi)$). If $\mathtt{Bw}(\mathcal{S},\phi,\Phi)$ is infinite then by definition (assuming that $\mathtt{Fw}(\mathcal{S},\phi)$ is finite) by Köning's lemma there must exist an infinite sequence of pairwise distinct triples (ϕ_i, q_i, X_i) $(i \in \mathbb{N})$ such that $X_0 = \emptyset$ and for every $i \in \mathbb{N}$, one of the following conditions holds:

- there exists an action a_i such that $(q_{i+1}, a_i, q_i) \in R$ and $\phi_{i+1} = \mathit{wpc}(\phi_i, a_i)$, where $X_{i+1} = X_i \setminus \{x_i\}$ if a_i is of the form $x_i := t_i$ with $x_i \in \mathcal{V}^\star$, and $X_{i+1} = \emptyset$ otherwise, or;
- $\phi_{i+1} = \psi_i$ with $\psi_i \in \langle \phi_i \rangle_{x_i}$ for some variable $x \in \mathcal{V}^\star \setminus X_i$, $q_{i+1} = q_i$ and $X_{i+1} = X_i \cup \{x_i\}$.

In both cases we have $q_{i+1} \succeq_{\mathcal{S}} q_i$, by definition of $\succeq_{\mathcal{S}}$ (see Definition 14). Since the set of states Q is finite, necessarily there exists a natural number k such that, $q_{i+1} \not\succ_{\mathcal{S}} q_i$ holds for all $i \geq k$. Since by hypothesis $\mathcal{S}$ is oriented, this entails that R contains no transition of the form $(q_{i+1}, x_i := t_i, q_i)$ with $x_i \in \mathcal{V}^\star$ and $i \geq k$. Consequently, we must have $X_{i+1} \supseteq X_i$, for all $i \geq k$. By definition of $\mathtt{Bw}(\mathcal{S},\phi,\Phi)$, $X_i \subseteq \mathcal{V}^\star$ for all $i \in \mathbb{N}$, and since $\mathcal{V}^\star$ is finite we deduce that there exists $l \in \mathbb{N}$ such that $l \geq k$ and $X_i = X_{i+1}$ for all $i \geq l$. By definition of $\mathtt{Bw}(\mathcal{S},\phi,\Phi)$, this entails that ϕ_{i+1} must be of form $\mathit{wpc}(\phi_i, a_i)$, for all $i \geq l$, and a_i is not of the form $x_i := t_i$. Note that this implies that all the predicates symbols occurring in ϕ_i occur in ϕ_l (since all the predicates in $\mathit{wpc}(\phi_i, a_i)$ must occur in ϕ_i). For all $i \geq l$, we denote by n_i the number of atoms in ϕ_i that are not equational and not of the form $x \mapsto (\boldsymbol{y})$ with $x \in \mathcal{V}^\star$. By inspection of the different cases in Definition 30 (taking into account the fact that a_i is not of the form $x_i := t_i$), it is easy to check that $n_{i+1} = n_i$ holds for all $i \geq l$. Indeed, the only case in which $\mathit{wpc}(\phi_i, a_i)$ contains an atom that does not occur in ϕ_i is when this atom is either an equation or a points-to atom with a left-hand side in $\mathcal{V}^\star$ (furthermore, the simplification rules in $\rightarrow_s$ cannot add new atoms in the formula). By irreducibility w.r.t. the rule $\mathtt{C}_*$, this entails that the number of spatial atoms in ϕ_i (for $i \geq l$) is at most $\mathit{card}(\mathcal{V}^\star) + n_l$. Assume that ϕ_i contains an existential variable x that does not occur in a spatial atom. By irreducibility w.r.t. the rule $\mathtt{E}_{\simeq}$, x cannot occur in an equation. By irreducibility w.r.t. $\mathtt{C}_{\simeq}$, it cannot occur in a disequation $x \not\simeq x$. Thus the only atoms in which x occurs are of the form $x \not\simeq x_i$, with $x_i \neq x$, and the rule $\mathtt{E}_{\not\simeq}$ applies, which contradicts the fact that ϕ_i is in normal form w.r.t. $\rightarrow_s$. Consequently, all the existential variables in ϕ_i occur in a spatial atom. Since the number of such atoms is bounded, necessarily the number of existential variables is bounded. As both the set of free variables $\mathcal{V}^\star$ and the set of predicate symbols in ϕ_i is finite, this entails that there exist finitely many symbolic heaps ϕ_i (with $i \geq l$), which contradicts our assumption. ◀

E Proof of Theorem 39

We construct a run $(q_i, \mathfrak{s}_i, \mathfrak{h}_i, a_i)_{i \in \mathbb{N}}$, and a corresponding sequence of triples $(\phi'_i, q_i, X_i)_{i \in \mathbb{N}}$ by induction on i, with $(\mathfrak{s}_0, \mathfrak{h}_0) \models_{\mathcal{R}} \phi$. We simultaneously establish the following inductive invariant:

a The equivalence $\mathcal{I}(\xi, i) = \mathit{true} \iff (\mathfrak{s}_i, \mathfrak{h}_i) \models_{\mathcal{R}} \xi$ holds for all atomic conditions ξ, and also for all symbolic heaps ξ such that there exists $q \in Q$ and $X \subseteq \mathcal{V}^\star$ with $(\xi, q, X) \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$, for all $x \in X \setminus v_{\mapsto}(\xi)$.

b For all $q \in Q$ and for all actions a, $\mathcal{I}(q, i) = \mathit{true}$ iff $q = q_i$ and $\mathcal{I}(a, i) = \mathit{true}$ iff $a = a_i$.

c $(\phi'_i, q_i, X_i) \in \mathtt{Fw}(\mathcal{S}, \phi)$ with $\mathcal{I}(\phi'_i, i) = \mathit{true}$ and $\forall x \in X_i \setminus v_{\mapsto}(\phi'_i) : \mathfrak{s}_i(x) \notin \mathit{dom}(\mathfrak{h}) \wedge (\mathfrak{s}_i, \mathfrak{h}_i) \not\models \mathtt{A}(x)$.

Note that the invariant entails in particular that $\mathcal{I}$ is compatible with $(q_i, \mathfrak{s}_i, \mathfrak{h}_i, a_i)_{i \in \mathbb{N}}$, w.r.t. all symbolic heaps occurring in Φ. Indeed, by definition of $\mathtt{Bw}(\mathcal{S}, \phi, \Phi)$, $(\psi, q, \emptyset) \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$ for all symbolic heaps occurring in Φ and for all states q.

- **Base case $(i = 0)$.** Let $q_0 \stackrel{\text{def}}{=} q_I$, $\phi'_0 \stackrel{\text{def}}{=} \phi$ and $X_0 \stackrel{\text{def}}{=} \emptyset$. Let S^+ (resp. S^-) be the set of symbolic heaps ψ occurring in $\mathcal{A}(\mathcal{R}, \mathcal{S}, \phi)$, $\{\phi\}$ or Φ such that $\mathcal{I}(\psi, 0) = \mathit{true}$ (resp. $\mathcal{I}(\psi, 0) = \mathit{false}$). By hypothesis, $\Phi \in S^+$, thus $S^+ \neq \emptyset$. Let X be the set of variables x such that $\mathcal{I}(\mathtt{A}(x), 0) = \mathit{false}$. By definition, $\mathcal{I} \not\models \Gamma(S^+, S^-, X)$, thus, by Axiom 14, (S^+, S^-, X) cannot be unsatisfiable, and there exists a structure $(\mathfrak{s}_0, \mathfrak{h}_0)$ such that $(\mathfrak{s}_0, \mathfrak{h}_0) \models_{\mathcal{R}} (S^+, S^-, X)$. By construction, $\mathcal{I}(\xi, 0) = \mathit{true} \iff (\mathfrak{s}_0, \mathfrak{h}_0) \models_{\mathcal{R}} \xi$ holds for all symbolic heaps ξ occurring in $\mathcal{A}(\mathcal{R}, \mathcal{S}, \phi)$, $\{\phi\}$ or Φ, and in particular, $(\mathfrak{s}_0, \mathfrak{h}_0) \models_{\mathcal{R}} \phi$. Still by construction, $\mathcal{I}(\mathtt{A}(x), 0) = \mathit{false} \implies (\mathfrak{s}_0, \mathfrak{h}_0) \not\models_{\mathcal{R}} \mathtt{A}(x)$. Conversely, if $\mathcal{I}(\mathtt{A}(x), 0) = \mathit{true}$, then by Axiom 4, necessarily $(\mathcal{I}, 0) \models \xi$, for some $\xi \in \langle \phi \rangle_x$, thus $(\mathfrak{s}_0, \mathfrak{h}_0) \models_{\mathcal{R}} \xi$ and by Lemma 35, we get $(\mathfrak{s}_0, \mathfrak{h}_0) \models_{\mathcal{R}} \mathtt{A}(x)$. Thus Property a holds for all symbolic heaps and for all conditions of the form $\mathtt{A}(x)$.
  By hypothesis we have $(\mathcal{I}, 0) \models q_I$, and, by Axiom 11, $(\mathcal{I}, 0) \models \neg q$, for all states $q \neq q_I$. By Axioms 12 and 11, there exists a unique action a_0 such that $(\mathcal{I}, 0) \models a_0$. Thus Property b holds.
  By definition of $\mathtt{Fw}(\mathcal{S}, \phi)$ we have $(\phi, q_I, \emptyset) \in \mathtt{Fw}(\mathcal{S}, \phi)$ thus Property c holds.
  It only remains to prove that Property a holds for all atomic conditions (other than those of the form $\mathtt{A}(x)$). Consider any atomic condition α, and assume that $\mathcal{I}(\alpha, 0) = \mathit{true}$ (the case whether $\mathcal{I}(\alpha, 0) = \mathit{false}$ is handled in a similar way). We show that $(\mathfrak{s}_0, \mathfrak{h}_0) \models \alpha$. Assume that α is of the form $x \approx y$, with $x, y \in \mathcal{V}^\star$. By definition ϕ'_0 is of the form $\exists \boldsymbol{u}.\phi'$, for some symbolic heap ϕ' containing no quantifier. By definition of $\mathtt{Bw}(\mathcal{S}, \phi, \Phi)$ we have $(\exists \boldsymbol{u}.(\phi' * x \simeq y), q_0, X_0) \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$. By Axiom 7, since $\mathcal{I}(\phi'_0, 0) = \mathit{true}$, we get $\mathcal{I}(\exists \boldsymbol{u}.(\phi' * x \simeq y), 0) = \mathit{true}$, thus $(\mathfrak{s}_0, \mathfrak{h}_0) \models_{\mathcal{R}} \exists \boldsymbol{u}.(\phi' * x \simeq y)$ (by Property a, which has already been established for symbolic heaps). Thus $\mathfrak{s}_0(x) = \mathfrak{s}_0(y)$ and therefore $(\mathfrak{s}_0, \mathfrak{h}_0) \models_{\mathcal{R}} x \approx y$. Assume that α is of the form $x.i \approx y$, with $x, y \in \mathcal{V}^\star$. By Axiom 1 we must have $\mathcal{I}(\mathtt{A}(x), 0) = \mathit{true}$, hence by Axiom 4 we deduce that $\mathcal{I}(\psi, 0) = \mathit{true}$, for some $\psi \in \langle \phi'_0 \rangle_x$ (note that, by definition of $\mathtt{Fw}(\mathcal{S}, \phi)$, we have $(\psi, q_0, X_0 \cup \{x\}) \in \mathtt{Fw}(\mathcal{S}, \phi)$). By Lemma 35, ψ is of the form $\exists \boldsymbol{v}.(x \mapsto (x_1, \dots, x_\kappa) * \psi')$. We have $(\exists \boldsymbol{v}.(x \mapsto (x_1, \dots, x_\kappa) * \psi' * x_i \simeq y), q_0, X_0 \cup \{x\}) \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$, thus by Axiom 5 we deduce that $\mathcal{I}(\exists \boldsymbol{v}.(x \mapsto (x_1, \dots, x_\kappa) * \psi' * x_i \simeq y), 0) = \mathit{true}$, so that $(\mathfrak{s}_0, \mathfrak{h}_0) \models_{\mathcal{R}} x.i \approx y$. The proof is similar if α is of the form $x.i \approx y.j$ (using Axiom 6).

- **Inductive case.** Assume that $(q_i, \mathfrak{s}_i, \mathfrak{h}_i, a_i)$ has been constructed and that the invariant above holds for all $i \leq k$. As $\mathcal{I}(a_k, k) = \mathit{true}$, by Axiom 13, we have $(\mathcal{I}, k) \models \mathit{pre}(a_k)$, hence $(\mathfrak{s}_k, \mathfrak{h}_k) \models_{\mathcal{R}} \mathit{pre}(a_k)$ (by Property a). By Proposition 26, we deduce that $(\mathfrak{s}_k, \mathfrak{h}_k)[a_k]$

is defined. Let $(\mathfrak{s}_{k+1}, \mathfrak{h}_{k+1}) = (\mathfrak{s}_k, \mathfrak{h}_k)[a_k]$. By Axioms 12 and 11, there exist a unique action a_{k+1} and state q_{k+1} such that $\mathcal{I}(a_{k+1}, k+1) = \mathcal{I}(q_{k+1}, k+1) = \mathit{true}$, with $(q_k, a_k, q_{k+1}) \in R$.

We show that Property a is satisfied for $k+1$. We first observe that, if a_k is not of the form $x := s$, then, using Axioms 8, 9 and 10, it is easy to check that $\mathcal{I}(\mathtt{A}(x), k+1)$ holds iff $\mathfrak{s}_{k+1}(x) \in \mathit{dom}(\mathfrak{h}_{k+1})$, i.e., that Property a holds if $\xi = \mathtt{A}(x)$. Indeed, if a_k is of the form $\mathtt{free}(x)$ (resp. $\mathtt{alloc}(x)$) then we have by Axiom 9 (resp. Axiom 10), $\mathcal{I}(\mathtt{A}(y), k+1) = \mathit{false}$ (resp. $\mathcal{I}(\mathtt{A}(y), k+1) = \mathit{true}$) if $\mathcal{I}(x \simeq y, k) = \mathit{true}$ and $\mathcal{I}(\mathtt{A}(y), k+1) = \mathcal{I}(\mathtt{A}(y), k)$ otherwise. Furthermore, by Axiom 8, $\mathcal{I}(\mathtt{A}(y), k+1) = \mathcal{I}(\mathtt{A}(y), k)$ holds for all $y \in \mathcal{V}^\star$ if a_k is not of the above forms.

Consider a triple $(\psi, q, X) \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$ such that for all $x \in X \setminus v_{\mapsto}(\psi)$, $\mathfrak{s}_{k+1}(x) \notin \mathit{dom}(\mathfrak{h}_{k+1})$ (†). If a_k contains a term $x.i$ where $x \notin v_{\mapsto}(\psi)$, then (by definition of $(\mathfrak{s}_k, \mathfrak{h}_k)[a_k]$) $\mathfrak{s}_{k+1}(x) \in \mathit{dom}(\mathfrak{h}_{k+1})$, so that $x \notin X$. Thus, by definition of $\mathtt{Bw}(\mathcal{S}, \phi, \Phi)$, $(\xi, q, X \cup \{x\}) \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$, for all $\xi \in \langle\psi\rangle_x$. Note that (since actions of the form $x := x.i$ are forbidden) we must have $\mathcal{I}(\mathtt{A}(x), k+1) \Leftrightarrow \mathfrak{s}_{k+1}(x) \in \mathit{dom}(\mathfrak{h}_{k+1})$, hence $\mathcal{I}(\mathtt{A}(x), k+1) = \mathit{true}$ and by Axiom 4, necessarily $(\mathcal{I}, k+1) \models \psi \Leftrightarrow \bigvee_{\xi \in \langle\psi\rangle_x} \xi$. By Lemma 35, $x \in v_{\mapsto}(\xi)$, for all $\xi \in \langle\psi\rangle_x$. By repeating this process (if needed) on any other variable y such that that the condition above holds (in case a_k contains another occurrence of a term $y.j$), we eventually obtain a set of symbolic heaps S such that $(\mathcal{I}, k+1) \models \psi \Leftrightarrow \bigvee_{\xi \in S} \xi$, for all $\xi \in S$, $\mathit{wpc}(\xi, a_k)$ is defined, and there exists X' such that $(\xi, q_k, X') \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$, with $X' = X \cup Y$, for some set of variables $Y \subseteq v_{\mapsto}(\xi)$. This entails (by definition of $\mathtt{Bw}(\mathcal{S}, \phi, \Phi)$) that, for all $\xi \in S$, $(\mathit{wpc}(\xi, a_k), q_{k+1}, X'') \in \mathtt{Bw}(\mathcal{S}, \phi, \Phi)$, for some X'' that is either empty (if a_k is of the form $x := t$ with $x \in \mathcal{V}^\star$) or identical to X' (otherwise). Furthermore, we have $(\mathfrak{s}_{k+1}, \mathfrak{h}_{k+1}) \models_{\mathcal{R}} \psi \iff \exists \xi \in S$ s.t. $(\mathfrak{s}_{k+1}, \mathfrak{h}_{k+1}) \models_{\mathcal{R}} \xi$, by Lemmata 36 and 37. By Property a in the inductive invariant (at rank k) the equivalence $\mathcal{I}(\mathit{wpc}(\xi, a_k), k) = \mathit{true} \iff (\mathfrak{s}_k, \mathfrak{h}_k) \models_{\mathcal{R}} \mathit{wpc}(\xi, a_k)$ holds for all $\xi \in S$. Indeed, we have $\forall x \in X'' \setminus v_{\mapsto}(\mathit{wpc}(\xi, a_k))$, $\mathfrak{s}_k(x) \notin \mathit{dom}(\mathfrak{h}_k)$: if the condition is not fulfilled, then a_k cannot be of the form $x := t$ (otherwise $X'' = \emptyset$) and either a_k is of the form $\mathtt{free}(x)$ and x must occur in $v_{\mapsto}(\mathit{wpc}(\xi, a_k))$, by definition of $\mathit{wpc}(\xi, a_k)$; or x must be allocated in $(\mathfrak{s}_{k+1}, \mathfrak{h}_{k+1})$, and then (by †, since $X'' \subseteq X \cup Y \subseteq X \cup v_{\mapsto}(\xi)$) $x \in v_{\mapsto}(\psi) \subseteq v_{\mapsto}(\xi)$, so that $x \in v_{\mapsto}(\mathit{wpc}(\xi, a_k))$ by definition of $\mathit{wpc}(\xi, a_k)$ (as $a_k \neq \mathtt{alloc}(x)$, since $\mathfrak{s}_k(x) \in \mathit{dom}(\mathfrak{h}_k)$). By Lemma 32 we get $\mathcal{I}(\mathit{wpc}(\xi, a_k), k) = \mathit{true} \iff (\mathfrak{s}_{k+1}, \mathfrak{h}_{k+1}) \models_{\mathcal{R}} \xi$, and by Axiom 3, this yields: $\mathcal{I}(\xi, k+1) = \mathit{true} \iff (\mathfrak{s}_{k+1}, \mathfrak{h}_{k+1}) \models_{\mathcal{R}} \xi$, so that $\mathcal{I}(\psi, k+1) = \mathit{true} \iff (\mathfrak{s}_{k+1}, \mathfrak{h}_{k+1}) \models_{\mathcal{R}} \psi$.

We now show that $\mathtt{Fw}(\mathcal{S}, \phi)$ contains a tuple $(\phi'_{k+1}, q_{k+1}, X_{k+1})$ such that $\mathcal{I}(\phi'_{k+1}, k+1) = \mathit{true}$. Let Y be the set of variables y such that a_k contains a term of the form $y.i$ (for some $i \in \mathbb{N}$) and $y \notin v_{\mapsto}(\phi'_k)$. By applying the function $\langle\rangle_x$ on all variables in Y, we get a set S of symbolic heaps such that $(\mathcal{I}, k) \models \phi'_k \Leftrightarrow \bigvee_{\xi \in S} \xi$. Furthermore, for all variables $y \in Y$, we have $\mathfrak{s}(y) \in \mathit{dom}(\mathfrak{h}_k)$ (since $(\mathfrak{s}_k, \mathfrak{h}_k)[a'_k]$ is defined), thus $y \notin X_k$. By definition of $\mathtt{Fw}(\mathcal{S}, \phi)$, we deduce that for all $\xi \in S$, $(\xi, q_k, X_k \cup Y) \in \mathtt{Fw}(\mathcal{S}, \phi)$. Moreover, by Lemma 35, we have $Y \subseteq v_{\mapsto}(\xi)$, and $\mathit{spc}(\xi, a_k)$ is defined for all $\xi \in S$. Then we get by Axiom 2, $(\mathcal{I}, k+1) \models \mathit{spc}(\xi, a_k)$, for some $\xi \in S$. We define: $\phi'_{k+1} \overset{\text{def}}{=} \mathit{spc}(\xi, a_k)$. By definition of $\mathtt{Fw}(\mathcal{S}, \phi)$, $(\mathit{spc}(\xi, a_k), q_{k+1}, X_{k+1}) \in \mathtt{Fw}(\mathcal{S}, \phi)$ for some set X_{k+1}. Let $x \in x \in X_{k+1} \setminus v_{\mapsto}(\phi'_{k+1})$. Assume that $\mathfrak{s}_{k+1}(x) \in \mathit{dom}(\mathfrak{h})$. By definition of $\mathtt{Fw}(\mathcal{S}, \phi)$, a_k cannot be of the form $z := t$, where $z \in \mathcal{V}^\star$ (otherwise X_{k+1} would be empty). Thus $X_{k+1} = X_k \cup Y$. Since $x \in \mathit{dom}(\mathfrak{s}_{k+1})$, we have $a_k \neq \mathtt{free}(x)$. Since $x \notin v_{\mapsto}(\phi'_{k+1})$, we have $a_k \neq \mathtt{alloc}(x)$, by definition of $\mathit{spc}(\xi, a_k)$. Thus a_k is of the form $t := s$ where t is not a variable, and we must have $x \in \mathit{dom}(\mathfrak{h}_k)$, and $v_{\mapsto}(\phi'_{k+1}) = v_{\mapsto}(\xi) \supseteq Y$. This entails that $x \in X_k$, which contradicts Property c at rank k.

Finally, using the symbolic heap ϕ'_{k+1}, the equivalence $\mathcal{I}(\alpha, k+1) = \mathit{true} \iff (\mathfrak{s}_{k+1}, \mathfrak{h}_{k+1}) \models_{\mathcal{R}} \alpha$ can be established for all atomic conditions α exactly as for the base case. The case where $\alpha = \mathtt{A}(x)$ and a_k is of the form $x := t$ is handled by noting that we have both $(\mathfrak{s}_{k+1}, \mathfrak{h}_{k+1}) \models \mathtt{A}(x) \Leftrightarrow \bigvee_{\xi \in \langle \phi'_{k+1} \rangle_x} \xi$ (since $(\mathfrak{s}_{k+1}, \mathfrak{h}_{k+1}) \models \phi'_{k+1}$, using Lemmata 35, 36 and 37) and $(\mathcal{I}, k+1) \models \mathtt{A}(x) \Leftrightarrow \bigvee_{\xi \in \langle \phi'_{k+1} \rangle_x} \xi$ (by Axiom 4).