Deciding Satisfiability for Overlaid Symbolic Heaps

Nicolas Peltier, Quentin Petitjean, Mihaela Sighireanu

Deciding Satisfiability for Overlaid Symbolic Heaps ⋆

Nicolas Peltier[1][0000-0002-8943-7000], Quentin Petitjean[2]✉[0009-0004-6504-8336], and Mihaela Sighireanu[2][0000-0002-1925-089X]

[1] Univ. Grenoble Alpes, CNRS, LIG, 38000 Grenoble France
[2] Univ. Paris-Saclay, ENS Paris-Saclay, CNRS, LMF, 91190 Gif-sur-Yvette France

Abstract. Separation logic (SL) is a widely used formalism for verifying programs that manipulate dynamically allocated memory, relying on the separating conjunction ⋆ to combine disjoint heap structures. The standard approach in SL lacks the expressive power to handle overlaid data structures where multiple structures share some locations. We consider a logic that extends SL with a new separating conjunction operator ⊛, enabling the composition of heaps with shared locations that allocate distinct fields. Our fragment supports generic inductive definitions and introduces set variables to constrain the locations shared by overlapping structures. We prove that the satisfiability problem for this fragment is in NEXPTIME, by reducing it to the satisfiability problem in BAPA [13], a decidable logic combining Boolean algebra of sets and Presburger arithmetic.

Keywords: Separation logic · Satisfiability problem · Overlaid structures · BAPA

1 Introduction

Separation logic (SL) [10,23] is widely employed in program verification. It extends Hoare logic to enable reasoning about programs that manipulate dynamically allocated memory. In SL, formulæ are constructed using two types of atoms: the first, denoted $x \to (y_1, \ldots, y_k)$, indicates that a memory block containing the tuple formed by the values of $y_1, \ldots, y_k$ is allocated at (and only at) location x; the second, `emp`, represents the empty heap, i.e., a heap with no allocated locations. These atoms are combined into formulæ using logical connectives, including a distinctive operator known as the *separating conjunction* written ⋆. The formula $\varphi_1 \star \varphi_2$ asserts that φ_1 and φ_2 hold on disjoint portions of the heap. This connective enables *local reasoning*, which enhances the scalability by allowing program properties to be asserted and verified solely with reference to the heap regions it affects. Additionally, to support the specification of recursive data structures, SL incorporates predicate atoms defined by inductive rules with fixpoint semantics. For example, list segments from x to y may be defined by the

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

following rules[3]:

$$\texttt{lseg}(x,y) \Leftarrow \texttt{emp} \star x \approx y\,,\ \ \texttt{lseg}(x,y) \Leftarrow x \to (z) \star \texttt{lseg}(z,y)\,. \tag{1}$$

The satisfiability and entailment problems have been extensively studied for various fragments of SL, particularly the so-called symbolic heaps fragment, which can be defined as (disjunctions of) existentially quantified separating conjunctions of atoms. It has been established that satisfiability in this fragment is EXPTIME-complete [2], while entailment is undecidable. For the latter problem, decidability can be achieved by restricting the form of the inductive rules defining predicate symbols (see, e.g., [9,4,19]).

However, this standard fragment of SL has limited expressive power. The connective $\star$ only permits the combination of disjoint structures, which precludes reasoning about structures that share elements—a common scenario in practical applications. For instance, consider a collection of pairwise disjoint sublists alongside an additional list containing all elements from these sublists in some arbitrary order; here, the sublists share nodes with the encompassing list. Another example is a tree structure whose leaves are linked in a list in an arbitrary order. Modelling an operation such as tree expansion would require removing a node α from the list of leaves and adding new nodes both to the list and as successors of α in the tree. In this case, the order of leaves in the list is arbitrary, so encoding the structure with a single inductive predicate (e.g., defining a tree with leaves chained from left to right) would not capture the intended class of structures, as the described operation would not preserve this invariant. Notably, in this example, only some nodes (the leaves) are shared, while inner nodes remain distinct. Such data structures, where locations may be shared across multiple structures, are termed "overlaid".

To address this limitation, various fragments of SL have been explored. Some offer sound decision procedures for program analysis [17,18,6,3], while others eschew inductive definitions in favour of an indexed separation conjunction combined with flow equations [11,20]. Notably, the logic NOLL [6] can describe nested linked lists with shared nodes, with satisfiability and entailment being NP-complete and co-NP-complete, respectively. To our knowledge, however, no fragment of SL fully addresses generic overlaid data structures while providing decidability results.

Building on prior work investigating decidable fragments of SL [19,2,4,16,22,7], we propose the logic OSL, inspired by [9,6], that supports overlaid data structures. Specifically, we model the heaps as partial finite functions from pairs (ℓ, f) (where ℓ is a location and f is a field) to tuples of locations. One field is thus allowed to make one location points to a tuple of locations. The use of different fields allow to associate several tuples of locations to a same location. As a result, a unique field is enough to support standard data-structures of SL. Heaps are combined using the weak separating conjunction operator $\circledast$ (introduced in [6]), which allows shared locations to allocate distinct record fields. OSL extends the simple inductive definitions of [6] to more general inductive definitions of [9], with the restriction that each inductive predicate al-

[3] We distinguish the equality in formula $\approx$ from the mathematical equality $=$. Moreover, in this paper, we adopt the convention that equations $x \approx y$ are satisfied only if the heap is empty. This entails no loss of generality in our context and simplifies the input language, as standard conjunction becomes dispensable.

locates only one field. However, the value referenced by a field is a tuple of locations, enabling straightforward descriptions of tree-shaped structures; the key constraint is that a predicate cannot allocate multiple distinct fields. Additionally, the fragment incorporates set variables within inductive predicates to collect subsets of allocated locations. We use these variables in set constraints to impose conditions on the overlap between inductive structures.

To summarize, the classic conjunction $\wedge$ imposes that a same heap verify two properties, the standard separating conjunction $\star$ imposes that a heap can be divided in two sub-heaps, each one verifying one side of the separating conjunction, and at last, the overlaid separating conjunction $\circledast$ allows the two sub-heaps to overlap if the overlapping locations are allocated using distinct fields.

Example 1. The tree-and-list example can be encoded in OSL as the formula $\mathtt{tree}(x, Y) \circledast \mathtt{ls}(y, Y)$, where the predicate atoms are defined by the following rules:

$$\mathtt{tree}(x, Y) \Leftarrow x.t \to () \star Y \approx \{x\}\,, \tag{2}$$

$$\mathtt{tree}(x, Y) \Leftarrow x.t \to (x_1, x_2) \star \mathtt{tree}(x_1, Y_1) \star \mathtt{tree}(x_2, Y_2) \star Y \approx Y_1 \sqcup Y_2\,, \tag{3}$$

$$\mathtt{ls}(y, Y) \Leftarrow y.l \to () \star Y \approx \{y\}\,, \tag{4}$$

$$\mathtt{ls}(y, Y) \Leftarrow y.l \to (y') \star \mathtt{ls}(y', Y') \star Y \approx \{y\} \sqcup Y'\,, \tag{5}$$

where the rules employ two distinct fields, t and l, to encode trees and lists respectively[4]. The symbols x, y, y', x_1, x_2 denote location variables, while Y, Y', Y_1, Y_2 represent sets of locations. The symbol $\sqcup$ stands for set union.

In this paper, we investigate the satisfiability problem for OSL and we demonstrate that it is in NEXPTIME. Reasoning about OSL formulæ requires combining standard spatial reasoning in SL (to handle predicate definitions, equalities, and constraints ensuring that a given field is allocated only once) with cardinality constraints arising from shared set variables. For instance, the formula $\mathtt{ls}_{\mathsf{even}}(x, Y) \circledast \mathtt{ls}_{\mathsf{odd}}(x, Y)$ is unsatisfiable if $\mathtt{ls}_{\mathsf{even}}$ and $\mathtt{ls}_{\mathsf{odd}}$ describe lists of even and odd lengths with set variables collecting all locations, respectively. The formula $\mathtt{ls}_1(x, X) \circledast \mathtt{ls}_2(x, X)$ is satisfiable if $\mathtt{ls}_1$ and $\mathtt{ls}_2$ both denote non-empty lists, provided these lists are chained using distinct fields, while $\mathtt{ls}_1(x, X) \star \mathtt{ls}_2(x, X)$ is always unsatisfiable (as the lists are non-empty each structure must allocate x). We thus reduce the satisfiability problem for OSL to a satisfiability problem in the logic BAPA [14], which combines the Boolean algebra of sets of uninterpreted elements with Presburger arithmetic to capture cardinality constraints on sets.

All proofs are given in the appendix.

2 Separation Logic with Overlaid Inductive Definitions

We introduce OSL, the variant of SL we propose for the specification of overlaid data structures. After presenting its syntax and semantics, we illustrate its specification power and introduce some of its fragments that will be used in the following sections.

[4] Both the lists and trees end at the empty tuple().

2.1 Syntax

In the definition below, the (possibly indexed) symbols φ, L, A, t, B and T denote *formulæ*, *equational atoms*, *arithmetic atoms*, *arithmetic terms*, *set atoms* and *set terms*, respectively (the symbol $\approx$ is overloaded). The symbols Φ, $\mathfrak{L}$, $\mathfrak{A}$, $\mathfrak{t}$, $\mathfrak{B}$ and $\mathfrak{T}$ denote the corresponding sets of objects of each sort.

Definition 1 (OSL formulæ). *Let $\mathcal{F}$ be a finite set of* field *symbols. Let $\mathcal{V}$ be a countably infinite set of* location variables*; let $\mathcal{S}$ be a countably infinite set of* set variables*, and let $\mathcal{P}$ be a set of* spatial predicate symbols*, where each symbol $p \in \mathcal{P}$ is associated with a unique arity $\#(p)$ (with countably infinite sets of predicate symbols of each arity). The set of* OSL formulæ *(or simply formulæ) φ is inductively defined as follows:*

$$\begin{aligned}\Phi \ni \varphi := {} & \mathtt{emp} \mid x.f \to (y_1, \ldots, y_d) \mid L \mid B \mid A \mid \varphi_1 \vee \varphi_2 \\ & \mid \varphi_1 \star \varphi_2 \mid \varphi_1 \circledast \varphi_2 \mid p(x_1, \ldots, x_{\#(p)-1}, X)\end{aligned}$$

with:

$$\begin{aligned}&\mathfrak{L} \ni L := x \approx y \mid x \not\approx y \qquad \mathfrak{t} \ni t := K \mid t_1 \oplus t_2 \mid K \odot t \mid |T| \\ &\mathfrak{A} \ni A := t_1 \approx t_2 \mid t_1 \not\approx t_2 \mid t_1 \prec t_2 \mid t_1 \not\prec t_2 \mid K \operatorname{div} t \mid K \operatorname{ndiv} t \\ &\mathfrak{B} \ni B := T_1 \approx T_2 \mid T_1 \not\approx T_2 \mid T_1 \sqsubseteq T_2 \mid T_1 \not\sqsubseteq T_2 \\ &\mathfrak{T} \ni T := \{x\} \mid X \mid \emptyset \mid T_1 \sqcup T_2 \mid T_1 \sqcap T_2\end{aligned}$$

where $\varphi_1, \varphi_2 \in \Phi$ are formulæ, $p \in \mathcal{P}$, $d \in \mathbb{N}$, $K \in \mathbb{Z}$, $x, y, x_1, \ldots, x_{\#(p)-1}, y_1, \ldots, y_d \in \mathcal{V}$, $f \in \mathcal{F}$ and $X \in \mathcal{S}$.

The definition above adheres to the standard SL fragment known as (disjunctive) *symbolic heap*, while incorporating additional features to enable the specification of overlaid data structures. Note that negations are not supported; thus, negated versions of some operators are added. To allow for overlaid data structures, the points-to atom $x.f \to (y_1, \ldots, y_d)$ identifies a portion of the memory block allocated at location x by a field f (for convenience, the arity of fields (d) is not fixed). Another OSL-specific feature is the inclusion of set variables (in $\mathcal{S}$), denoting finite sets of locations, and set constraints in formulæ. We write set variables in capital letters to distinguish them from location variables in $\mathcal{V}$. Set terms are composed from set variables and basic term $\emptyset$ and $\{x\}$ using union and intersection of sets. The set constraints $\mathfrak{B}$ and the arithmetic constraints $\mathfrak{A}$ are a subset of those in the logic BAPA [14]. The integer terms $\mathfrak{t}$ include arithmetic operations (addition $\oplus$ and multiplication $\odot$) and set cardinality denoted by $|T|$. For simplicity, we do not consider arithmetic variables (this is not restrictive, as they can be encoded by terms $|X|$ where X is a fresh set variable). The link between set variables and the heap is established using predicate atoms. Thus, a predicate atom $p(x_1, \ldots, x_{\#(p)-1}, X)$ has *exactly one* set argument, in addition to other arguments denoting locations. Intuitively, the set argument is intended to denote *a subset of the locations allocated in the data structure* specified by the predicate; it is defined in the predicate's definition. For readability, we assume that the set argument always occurs at the last position in the argument list. The last feature of OSL is the composition of formulæ using the "overlaid" separating conjunction operator $\circledast$, in addition to the classic separating

conjunction $\star$. Intuitively, $\circledast$ composes formulæ specifying portions of memory defined by different fields inside the allocated memory regions.

To introduce predicate definitions and the OSL semantics, we use the following notations. The cardinality of a set X is denoted by $\mathrm{card}(X)$. The set $\{x \in \mathbb{Z} \mid i \leq x \leq j\}$ is denoted by $[\![i, j]\!]$. The domain of a function f is written $\mathrm{dom}(f)$. The powerset of a set S is written $\mathscr{P}(S)$. The set of free (location and set) variables occurring in φ is denoted by $fv(\varphi)$. A *substitution* σ is a function from $\mathcal{V}$ to $\mathcal{V}$, and from $\mathcal{S}$ to $\mathfrak{T}$. Its domain, $\mathrm{dom}(\sigma)$, is the set of variables x such that $\sigma(x) \neq x$, and its image is $\mathrm{img}(\sigma) = \sigma(\mathrm{dom}(\sigma))$. For any expression e (a variable, tuple, term or formula), we denote by $e\sigma$ the expression obtained from e by replacing each free occurrence of a variable x with $\sigma(x)$. We denote by $\{x_i \mapsto t_i \mid 1 \leq i \leq n\}$ the substitution that maps each variable x_i to t_i, where $x_i \in \mathcal{V} \cup \mathcal{S}$ and t_i is either a variable or a set term.

An *inductive rule associated with the predicate* $p \in \mathcal{P}$ has the following form in OSL (up to associativity and commutativity of $\star$):

$$p(x_1, \ldots, x_n, X) \Leftarrow x_r.f \to (\vec{z}) \star \bigstar_{i=1}^{m} q_i(\vec{y_i}, Y_i) \star \varphi \star \left(X \approx E \sqcup \bigsqcup_{u \in U} Y_u \right), \qquad (6)$$

where $r \in [\![1, n]\!]$ and the following syntactic restrictions are satisfied:

- the set variables Y_i are pairwise distinct set variables and distinct from X;
- the set term E is either the constant $\emptyset$ or the term $\{x_r\}$;
- $U \subseteq [\![1, m]\!]$ (the set U may be $\emptyset$, in which case $\bigsqcup_{u \in U} Y_u$ is $\emptyset$);
- the formula φ is a $\star$-conjunction of pure equational atoms $L \in \mathfrak{Q}$.

The variables $x_1, \ldots, x_n, X$ are called *parameters* of p. We refer to the variables that appear in the right-hand side of the rule but are not among the predicate's parameters as the rule's *auxiliary variables*. These auxiliary variables include, in particular, the set variables Y_i. Formally, they are defined as $fv(x_r.f \to (\vec{z}) \star \bigstar_{i=1}^{m} q_i(\vec{y_i}, Y_i) \star \varphi \star (X \approx E \cup \bigsqcup_{u \in U} Y_u)) \setminus \{x_1, \ldots, x_n, X\}$. Our form of predicate rule extends classical inductive definitions of SL—such as those in [9]—by incorporating set parameters and set constraints. In the following, we shall assume that $U = [\![1, m]\!]$. This assumption simplifies the presentation, but does not affect the expressive power of the predicate definitions.

Remark 1. The conditions provided are less restrictive than those imposed in [9] to ensure the decidability of the entailment problem. No constraint is imposed on the spatial part of the formula, except that exactly one location x_r must be allocated in every rule[5]. However, the use of set constraints and arithmetic terms inside the rules is strictly limited: the only allowed constraint (besides equational atoms) is the equation $X \approx E \sqcup \bigsqcup_{u \in U} Y_u$, which relates the value of the set parameter X to the sets computed by recursive calls. It is essential that E contains only locations allocated by the rule, as this guarantees that the same location is added at most once to the computed set (this ensures that E contains only allocated locations and facilitates the computation of the

[5] This condition is imposed for technical convenience only and does not limit the expressive power of predicate definitions.

cardinality of X in Sect. 3.2). Note also that the rules do not contain the connective $\circledast$: as we will restrict the set of rules defining a predicate to use only one field (see Sect. 2.3), using this connective inside of a rule is pointless.

A *set of inductive definitions* (SID) $\mathcal{R}$ contains finitely many rules associated with predicate symbols in $\mathcal{P}$. We write $p(y_1, \ldots, y_n, Y) \Leftarrow_{\mathcal{R}} \psi$ if the SID $\mathcal{R}$ contains a rule $p(x_1, \ldots, x_n, X) \Leftarrow \varphi$, with $\psi = \varphi\{X \mapsto Y, x_i \mapsto y_i \mid i \in [\![1, n]\!]\}$.

Example 2. The following OSL formula specifies a heap where a (non-empty) binary tree $\mathtt{treel}(r, h, N)$ and a list segment $\mathtt{ls}(h, l, S)$ are overlaid such that the list starts in the left-most leaf of the tree h and contains some of this leaf's ancestors ($S \subseteq N$); the list ends into an empty memory block (l):

$$(\mathtt{treel}(r, h, N) \circledast \mathtt{ls}(h, l, S)) \star S \sqsubseteq N \star l.next \rightarrow () \tag{7}$$

The two predicates `treel` and `ls` are defined by the following rules:

$$\mathtt{treel}(x_r, x_l, X) \Leftarrow x_r.sons \rightarrow () \star x_l \approx x_r \star X \approx \{x_r\}\,, \tag{8}$$

$$\mathtt{treel}(x_r, x_l, X) \Leftarrow x_r.sons \rightarrow (y_l, y_r) \star x_l \not\approx x_r \star \mathtt{treel}(y_l, x_l, Y_l) \star \mathtt{tree}(y_r, Y_r) \star X \approx \{x_r\} \sqcup Y_l \sqcup Y_r\,, \tag{9}$$

$$\mathtt{tree}(x_r, X) \Leftarrow x_r.sons \rightarrow () \star X \approx \emptyset\,, \tag{10}$$

$$\mathtt{tree}(x_r, X) \Leftarrow x_r.sons \rightarrow (y_l, y_r) \star \mathtt{tree}(y_l, Y_l) \star \mathtt{tree}(y_r, Y_r) \star X \approx Y_l \sqcup Y_r\,, \tag{11}$$

$$\mathtt{ls}(x, y, X) \Leftarrow x.next \rightarrow (y) \star x \not\approx y \star X \approx \{x\}\,, \tag{12}$$

$$\mathtt{ls}(x, y, X) \Leftarrow x.next \rightarrow (z) \star x \not\approx y \star \mathtt{ls}(z, y, Y) \star X \approx \{x\} \sqcup Y\,. \tag{13}$$

The auxiliary variables in rule 9 are y_l, y_r, Y_l and Y_r.

2.2 Semantics

Definition 2 (OSL structure). *Let $\mathcal{L}$ be a countably infinite set of so-called* locations*. An OSL structure is a tuple $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ where $\mathfrak{s}$ is a* store*, i.e., a partial function from $\mathcal{V}$ to $\mathcal{L}$, $\mathfrak{h}$ is a* heap*, i.e., a partial finite function mapping pairs (ℓ, f) (where $\ell \in \mathcal{L}$ and $f \in \mathcal{F}$) to tuples of locations in $\mathcal{L}^*$ and Σ is a* set interpretation*, i.e., a partial function mapping set variables in $\mathcal{S}$ to finite subsets of $\mathcal{L}$.*

Thus $\mathrm{dom}(\mathfrak{h}) \subseteq \mathcal{L} \times \mathcal{F}$ for all heaps $\mathfrak{h}$. We denote by $\mathrm{dom}_f(\mathfrak{h})$ the set $\{\ell \mid (\ell, f) \in \mathrm{dom}(\mathfrak{h})\}$ and by $\mathrm{allocated}(\mathfrak{h})$ the set $\{\ell \mid \exists f \in \mathcal{F}.\, (\ell, f) \in \mathrm{dom}(\mathfrak{h})\}$. For conciseness, we omit in the following the semantics of arithmetic terms t and arithmetic constraints A; their semantics is the expected one and is detailed in Defs. 22 and 23 (Sect. D).

Definition 3 (Semantics of set terms). *Let T be a set term, $\mathfrak{s}$ be a store, and Σ be a set interpretation. The interpretation of set term denoted by $[\![T]\!]_{(\mathfrak{s},\Sigma)}$ is a set of locations in $\mathcal{L}$ and it is inductively defined by the following rules:*

- $[\![\emptyset]\!]_{(\mathfrak{s},\Sigma)} = \emptyset$;
- $[\![\{x\}]\!]_{(\mathfrak{s},\Sigma)} = \{\mathfrak{s}(x)\}$;
- $[\![X]\!]_{(\mathfrak{s},\Sigma)} = \Sigma(X)$;
- $[\![T_1 \sqcup T_2]\!]_{(\mathfrak{s},\Sigma)} = [\![T_1]\!]_{(\mathfrak{s},\Sigma)} \cup [\![T_2]\!]_{(\mathfrak{s},\Sigma)}$;
- $[\![T_1 \sqcap T_2]\!]_{(\mathfrak{s},\Sigma)} = [\![T_1]\!]_{(\mathfrak{s},\Sigma)} \cap [\![T_2]\!]_{(\mathfrak{s},\Sigma)}$.

Definition 4 (OSL semantics). *Given a formula φ, an SID $\mathcal{R}$ and a structure $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ with $fv(\varphi) \subseteq \mathrm{dom}(\mathfrak{s}) \cup \mathrm{dom}(\Sigma)$, the satisfaction relation $\models_{\mathcal{R}}$ is inductively defined as the least relation such that $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi$ iff one of the following conditions holds:*

- $\varphi = \mathtt{emp}$ *and* $\mathfrak{h} = \emptyset$;
- $\varphi = (x.f \to (y_1, \ldots, y_d))$ *and* $\mathfrak{h} = [(\mathfrak{s}(x), f) \mapsto (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_d))]$;
- $\varphi = (x \approx y)$, $\mathfrak{s}(x) = \mathfrak{s}(y)$ *and* $\mathfrak{h} = \emptyset$; *or* $\varphi = (x \not\approx y)$, $\mathfrak{s}(x) \neq \mathfrak{s}(y)$ *and* $\mathfrak{h} = \emptyset$;
- $\varphi = \varphi_1 \vee \varphi_2$ *and* $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi_i$, *for some* $i \in \{1, 2\}$;
- $\varphi = \varphi_1 \star \varphi_2$ *and there exist heaps* $\mathfrak{h}_1, \mathfrak{h}_2$ *such that* $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, *allocated*$(\mathfrak{h}_1) \cap$ *allocated*$(\mathfrak{h}_2) = \emptyset$ *and* $(\mathfrak{s}, \mathfrak{h}_i, \Sigma) \models_{\mathcal{R}} \varphi_i$, *for all* $i \in \{1, 2\}$;
- $\varphi = \varphi_1 \circledast \varphi_2$ *and there exist heaps* $\mathfrak{h}_1, \mathfrak{h}_2$ *such that* $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, $\mathrm{dom}(\mathfrak{h}_1) \cap \mathrm{dom}(\mathfrak{h}_2) = \emptyset$ *and* $(\mathfrak{s}, \mathfrak{h}_i, \Sigma) \models_{\mathcal{R}} \varphi_i$, *for all* $i \in \{1, 2\}$;
- $\varphi = p(x_1, \ldots, x_{\#(p)-1}, X)$, $p \in \mathcal{P}$ *and* $(\mathfrak{s}', \mathfrak{h}, \Sigma') \models_{\mathcal{R}} \psi$, *for some* $\mathfrak{s}'$ *matching* $\mathfrak{s}$ *on* $x_1, \ldots, x_{\#(p)-1}$, *some* Σ' *matching* Σ *on* X *and for some* ψ *such that* $\varphi \Leftarrow_{\mathcal{R}} \psi$;
- $\varphi = (T_1 \approx T_2)$, $[\![T_1]\!]_{(\mathfrak{s},\Sigma)} = [\![T_2]\!]_{(\mathfrak{s},\Sigma)}$ *and* $\mathfrak{h} = \emptyset$; *the definition is similar for* $\not\approx$, $\sqsubseteq$, $\not\sqsubseteq$.

We write $\varphi \models_{\mathcal{R}} \psi$ if for every structure $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ we have $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi \implies (\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \psi$. If both $\varphi \models_{\mathcal{R}} \psi$ and $\psi \models_{\mathcal{R}} \varphi$ hold, then we write $\varphi \equiv_{\mathcal{R}} \psi$.

We emphasize that the atoms $x \approx y$ or $x \not\approx y$ only hold for empty heaps (this convention simplifies notations as it avoids the use of standard conjunction). The same convention applies to arithmetic and set constraints. Formulæ are taken modulo the usual properties of SL connectives: associativity and commutativity of $\star, \circledast$ and $\vee$, neutrality of $\mathtt{emp}$ for $\star$ and $\circledast$, and commutativity of $\approx, \not\approx$. Intuitively, the interpretation of $\varphi_1 \star \varphi_2$ is stronger than that of $\varphi_1 \circledast \varphi_2$, as the latter allows for the allocation of the block at the same (starting) location in both φ_1 and φ_2, provided the considered offsets (represented by fields) are different. For instance, $x.f \to (y) \circledast x.g \to (y)$ is satisfiable, but not $x.f \to (y) \star x.g \to (y)$ or $x.f \to (y) \circledast x.f \to (y)$. Note that $\mathfrak{h}_1 \cup \mathfrak{h}_2$ is well defined if $\mathrm{dom}(\mathfrak{h}_1) \cap \mathrm{dom}(\mathfrak{h}_2) = \emptyset$ and that $\mathrm{allocated}(\mathfrak{h}_1) \cap \mathrm{allocated}(\mathfrak{h}_2) = \emptyset \implies \mathrm{dom}(\mathfrak{h}_1) \cap \mathrm{dom}(\mathfrak{h}_2) = \emptyset$. Thus $\varphi_1 \circledast \varphi_2$ is a logical consequence of $\varphi_1 \star \varphi_2$.

Definition 5 (OSL model). *An* $\mathcal{R}$-model *of a formula φ is a structure $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ such that $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi$. A formula ψ admitting an $\mathcal{R}$-model is* $\mathcal{R}$-satisfiable *(or simply* satisfiable *if $\mathcal{R}$ is clear from the context).*

2.3 1-Predicates

In the remainder of the paper, we assume that each predicate allocates a single field, in the sense that for every predicate $p \in \mathcal{P}$, there exists a unique field $f \in \mathcal{F}$ such that every points-to atom appearing in any unfolding of any atom $p(\vec{x}, X)$ is of the form $y.f \to (\vec{z})$, where $\vec{z}$ is a tuple of arbitrary length. More formally:

Definition 6 (F-formulæ). *Let $F \subseteq \mathcal{F}$. The set of* F-formulæ *and* F-predicates *are defined co-inductively as the biggest sets satisfying the following conditions:*

– *if* $x.f \to (\vec{y})$ *is an* F*-formula then* $f \in F$*;*
– *if* $\varphi_1 \bullet \varphi_2$ *is an* F*-formula (for some binary connective* $\bullet$*) then* φ_1, φ_2 *are* F*-formulæ;*
– *if* $p(\vec{x}, X)$ *is an* F*-formula then* p *is an* F*-predicate;*
– *if* p *is an* F*-predicate and* $p(\vec{x}, X) \Leftarrow \varphi \in \mathcal{R}$ *then* φ *is an* F*-formula.*

A 1-predicate *is a* $\{f\}$*-predicate, for some* $f \in \mathcal{F}$*.*

In particular, pure formulæ are always F-formulæ for all F. If φ is an $\{f\}$-formulæ, then it is clear that allocated($\mathfrak{h}$) $\subseteq \mathrm{dom}_f(\mathfrak{h})$ holds for all models $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ of φ. Consequently, $\varphi_1 \star \varphi_2$ and $\varphi_1 \circledast \varphi_2$ are equivalent if φ_1 and φ_2 are both $\{f\}$-formulæ (with the same field f). Notice also that, due to the form of the rule in Eq. 6, we also have $\Sigma(X) \subseteq \mathrm{dom}_f(\mathfrak{h})$.

2.4 Normal Form

The decision procedure proposed in Sect. 3 applies to a pair $(\varphi, \mathcal{R})$ in which both components are in *normal form*. This form can be obtained from general OSL formulæ and SID through a syntactic translation, which may cause an exponential increase in their respective sizes. We define this normal form below by gradually introducing its components.

Symbolic heap formula. As in classical SL, a *symbolic heap formula* [1] is a (separating) conjunction of atoms. This fragment is especially interesting for program analysis. A symbolic heap formula does not contain the $\vee$ connective. By distributivity of $\star$ and $\circledast$ over $\vee$, any OSL formula φ can be transformed into an equivalent formula, denoted $dnf(\varphi)$, which is a disjunction of symbolic heap formulæ. Then, the satisfiability of $dnf(\varphi)$ can be tested by checking whether at least one of its disjuncts is satisfiable.

Injective store models. For proof's readability, we focus on models $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ where $\mathfrak{s}$ is injective on location variables. This is enforced by adding a disequation $x \not\approx y$ for every pair of distinct location variables x, y.

Definition 7 (Normal form). *A pair* $(\varphi, \mathcal{R})$ *where* V *is the set of location variables in* φ *is in* normal form *if:*

1. φ *is a symbolic heap and contains a disequation* $x \not\approx y$ *for every pair of distinct variables in* V*.*
2. *If* $p(\vec{x}, X) \Leftarrow \psi$ *is a rule in* $\mathcal{R}$*,* $y, z \in \mathcal{V}$*,* y *is an auxiliary variable in* ψ *(i.e., a variable occurring in* ψ *but not in* $\vec{x}$*) and* z *is a variable distinct from* y *occurring in* $\vec{x}$ *or* ψ*, then* ψ *contains the atom* $y \not\approx z$*.*
3. *Each predicate* p *of arity* $n+1$ *is associated with a subset* $I_p \subseteq [\![1, n]\!]$ *(called the* main parameters *of* p*), which satisfies the following properties:*
   – *For every atom* $p(x_1, \ldots, x_n, X)$ *occurring in* φ *and every location variable* y *in* V*, there exists* $i \in I_p$ *such that* $x_i = y$*;*
   – *For every rule* $p(x_1, \ldots, x_n, X) \Leftarrow \psi$ *and every atom* $q(y_1, \ldots, y_m, Y)$ *in* ψ*, if* $i \in I_p$*, then there exists* $j \in I_q$ *such that* $y_j = x_i$*.*

   These conditions are designed to ensure that all location variables in φ *are passed as parameters to every predicate invoked during the evaluation of* φ*.*

Condition 1 is not restrictive. Indeed, it is possible to enumerate all the equality relations on V and test satisfiability of φ for each of these relations, by replacing in φ all the variables in the same equivalence class by the same representative and by adding disequations between distinct representatives. Condition 2 is easy to enforce: a rule of the form $p(\vec{x}, X) \Leftarrow \psi$ that includes an auxiliary variable y and another (arbitrary) variable z can be replaced by two rules: $p(\vec{x}, X) \Leftarrow \psi\{y \mapsto z\}$ and $p(\vec{x}, X) \Leftarrow \psi \star y \not\approx z$, thus distinguishing the cases where y is equal to or different from z. Condition 3 is enforced by adding $|V|$ parameters to each predicate atom, corresponding to the variables in the set V. These additional parameters are simply propagated unchanged through recursive calls in predicate's rules. Sect. F gives an example of transformation into normal form.

Remark 2. As stated for all the components of the normal form, a general OSL formula and a SID can always be translated in normal form at the cost of a potential exponential increase in their respective sizes.

3 Satisfiability Problem

The satisfiability problem for OSL is defined as: "Given a formula $\varphi \in \Phi$ and a SID $\mathcal{R}$, does a structure $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ exist such that $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi$?" Our main contribution is stated by the following theorem, which identifies a sufficient condition to decide this problem.

Theorem 1. *The satisfiability problem for OSL is decidable in exponential space if the predicates defined by the SID are* 1*-predicates.*

The result is obtained in two main steps summarised in the remainder of this section. We assume that the pair $(\varphi, \mathcal{R})$ is in normal form. As show in Sect. 2.4, this does not affect the expressive power of OSL.

The *first step* of the proof, presented in Sect. 3.1, introduces *decorations* of OSL formulæ. A decoration of a formula φ partitions the models of φ in such a way that each partition is characterized by (i) some aliasing and non-aliasing relations between the location variables, (ii) an assignment of location variables to the set variables, and (iii) an assignment of location variables to the allocated heap. Decorations are used to decorate predicate atoms and their defining rules in $\mathcal{R}$.
This results in a decorated SID $\mathcal{R}_{dec}$, in which only *consistent* rules are retained, namely, those where the partition induced by the decoration of the defining formula (i.e., the rule's right-hand side) is identical to the partition associated with the predicate being defined (i.e., the rule's left-hand side). This condition may be checked syntactically. Therefore, the initial question is split into N questions for the pairs $(\varphi_d, \mathcal{R}_d)$ obtained by decorating $(\varphi, \mathcal{R})$, where N is exponential in the size of the initial pair. Intuitively, this exponential blow-up is due to enumerating all decorations, which involves enumerating equivalence relations between free location variables, etc. The soundness and completeness of this step are stated by Thm. 2. Thus, if a formula φ has a model m, then there exists a decoration d such that the corresponding decorated formula φ_d also has m as a model. Conversely, if φ_d has m as a model, then m is also a model of φ.

This first step serves two purposes: (i) it ensures that all constraints induced by unfolding a predicate atom can be identified solely by inspecting predicate's decoration; and (ii) it enables the description of the cardinality of set variables as a context-free grammar (which in turn can be transformed into a Presburger formula). To illustrate the second point, consider the rules $p(x, y, X) \Leftarrow x.f \to () \star X \approx \emptyset$ and $p(x, y, X) \Leftarrow x.f \to () \star x \not\approx y \star X \approx \{x\}$. If $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models p(x, y, X)$, then $\mathrm{card}(\Sigma(X)) = 0$ when $\mathfrak{s}(x) = \mathfrak{s}(y)$ (i.e., x and y alias), and $\mathrm{card}(\Sigma(X)) \in \{0, 1\}$ otherwise. In contrast, if $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models p_d(x, y, X)$ with d a decoration, then the set of possible cardinalities of $\Sigma(X)$ depends only on the aliasing and non-aliasing relations in d: if d imposes a non-aliasing of x and y, then only the second rule can be unfolded and thus $\mathrm{card}(\Sigma(X)) = 1$; if d does not impose any condition on x and y, then the first rule must be unfolded and $\mathrm{card}(\Sigma(X)) = 0$; finally a decoration cannot impose an aliasing of x and y as there is no rule that requires such a condition. The second decoration admits models where $\mathfrak{s}(x) = \mathfrak{s}(y)$, but the cardinality of X does not depend on $\mathfrak{s}$.

The *second step*, presented in Sects. 3.2 and 3.3, translates a decorated pair $(\varphi_d, \mathcal{R}_d)$ into an equi-satisfiable formula in the logic BAPA [13]. The translation preserves all the set constraints of φ_d and it adds to them (a) Presburger formulæ that characterize the possible cardinalities of all set variables in φ_d; and (b) some set constraints encoding decorations, i.e., aliasing, non-aliasing, or membership constraints. Thm. 3 states that this translation is correct, i.e., it preserves satisfiability.

3.1 Decorations

Definition 8 (Decorated predicate symbol). *Let $p \in \mathcal{P}$ be a predicate symbol of arity $n+1$. A* decoration *of p is a symbol $p_{I,J,\sim,\not\approx}$ such that I, J are subsets of $[\![1, n]\!]$ with $I \subseteq J$, $\sim$ is an equivalence relation over (i.e., partition of) $[\![1, n]\!]$ and $\not\approx$ is a symmetric binary relation on $[\![1, n]\!]$. The set of all decorations for symbols in $\mathcal{P}$ is denoted by $\mathcal{P}_{dec}$.*

Intuitively, the decoration of a predicate symbol fixes certain aspects of the model of an atom $p(x_1, \ldots, x_n, X)$. For instance, $p_{I,J,\sim,\not\approx}$ specifies that I is the set of indices i (i.e., parameter positions) such that x_i occurs in X, and that J is the set of indices j such that x_j is allocated. The relations $\sim$ and $\not\approx$ encode aliasing and non-aliasing constraints, respectively: if $i \sim j$, then the store must map x_i and x_j to the same location; if $i \not\approx j$, then the locations of x_i and x_j must differ. Note that $i \not\approx j$ is *not* the negation of $i \sim j$, as both relations may be false in cases where no constraint is imposed on x_i and x_j. If $i(\not\approx \cap \sim) \neq \emptyset$, i.e., both $i \not\approx j$ and $i \sim j$ hold in a decoration, then the constraints resulting from the decoration are inconsistent; such decorations will be detected and eliminated in a later step (see Def. 14).

Definition 9 (Decorated formula). *A formula ψ is a* decoration of a formula φ *if it is obtained from φ by replacing every occurrence of a predicate symbol by a decoration of this symbol. A formula ψ is a* decoration *if there exists a formula φ such that ψ is a decoration of φ.*

Note that a predicate p symbol may appear in a formula with different decorations.

Definition 10 (Aliasing of a decorated formula). *Let ψ be a decoration of a symbolic heap formula with location variables V. The* aliasing *of ψ, denoted by $\equiv_\psi$, is the least reflexive, symmetric and transitive relation in V^2 such that $x \equiv_\psi y$ if one of the following conditions holds:*

(a) ψ contains an atom $x' \approx y'$ with $x \equiv_\psi x'$ and $y \equiv_\psi y'$;
(b) ψ contains an atom $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ with $x_i \equiv_\psi x$, $x_j \equiv_\psi y$ and $i \sim j$.

Intuitively, $\equiv_\psi$ is the aliasing relation between the location variables of ψ induced by the equality atoms and the decorations of predicate symbols.

Definition 11 (Allocated variables). *Let ψ be a decoration of a symbolic heap formula with location variables V. The set of variables* allocated *by ψ, denoted by $alloc(\psi)$, is a subset of V inductively defined as the set of location variables y such that one of the following conditions holds:*

(a) ψ contains an atom of the form $y.f \rightarrow (\vec{z})$;
(b) ψ contains an atom $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ such that $y = x_j$ with $j \in J$;
(c) $y \equiv_\psi y'$ with $y' \in alloc(\psi)$.

Intuitively, $alloc(\psi)$ denotes the set of variables in V that must be allocated in every model of ψ, given the decoration of predicates in ψ and the aliasing constraints.

Definition 12 (Distinguishing of a decorated formula). *Let ψ be a decoration of a symbolic heap formula with location variables V. The distinguishing of ψ, denoted $\not\equiv_\psi$, is the least symmetric relation in V^2 such that $x \not\equiv_\psi y$ if one of the following conditions holds:*

(a) ψ contains the atom $x \not\approx y$;
(b) ψ contains the atom $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ with $x_i = x$, $x_j = y$ and $i \not\sim j$;
(c) ψ contains $\psi_1 \star \psi_2$, $x \in alloc(\psi_1)$ and $y \in alloc(\psi_2)$;
(d) $x \equiv_\psi x'$, $y \equiv_\psi y'$ and $x' \not\equiv_\psi y'$.

Intuitively, $\not\equiv_\psi$ is the disequality relation between the location variables of ψ induced by the disequality atoms, the decorations of predicate symbols and the separating conjunctions. The notion $\not\equiv_\psi$ is only used in (and thus defined for) decorated formulæ that appear in a right hand side of a predicate rule. Note that if ψ is in normal form, then $x \not\equiv_\psi y$ holds for all distinct variables x and y occurring in ψ. However, this property does not necessarily hold for formulæ obtained through unfolding, as parameter instantiation may introduce aliasing.

Definition 13 (Decorated set variable). *Let ψ be a decoration of a symbolic heap formula with location variables V and let Y be a set variable. The* set of location variables attached to *Y by ψ, denoted by $[Y]_\psi$, is inductively defined as the set of location variables $y \in V$ such that one of the following conditions holds:*

(a) ψ contains a set constraint $Y \approx E \sqcup \bigsqcup_{u \in U} Z_u$ (see Eq. 6) with $y \in E$ or $y \in [Z_u]_\psi$ for some $u \in U$;
(b) ψ contains an atom $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ such that $X = Y$ and $y = x_i$ for some $i \in I$;

(c) $y \equiv_\psi y'$ *with* $y' \in [Y]_\psi$.

Intuitively, $[Y]_\psi$ is the subset of V, the location variables of ψ, that must be contained within the set variable Y given the decoration of predicates in ψ and the aliasing and set constraints.

Definition 14 (Consistent decoration). *The decoration* ψ *is* consistent *if the following condition holds: if* $x \not\simeq_\psi y$ *then* $x \not\equiv_\psi y$.

Definition 15 (Decorated rule). *Let* p *be a predicate symbol,* $p_{I,J,\sim,\not\simeq}$ *one of its decorations, and* $p(x_1, \ldots, x_n, X) \Leftarrow \varphi$ *a rule defining* p *in* $\mathcal{R}$. *A* decorated rule *for* $p_{I,J,\sim,\not\simeq}$ *is a rule* $p_{I,J,\sim,\not\simeq}(x_1, \ldots, x_n, X) \Leftarrow \psi$ *satisfying all the conditions below:*

- ψ *is a consistent decoration of* φ;
- $i \sim j$ *iff* $x_i \equiv_\psi x_j$ *for all* $i, j \in [\![1, n]\!]$; *and* $i \not\simeq j$ *iff* $x_i \not\equiv_\psi x_j$ *for all* $i, j \in [\![1, n]\!]$;
- $I = \{i \in [\![1, n]\!] \mid x_i \in [X]_\psi\}$; *and* $J = \{j \in [\![1, n]\!] \mid x_j \in alloc(\psi)\}$.

The set of decorated rules obtained from $\mathcal{R}$ *and the decorated predicate symbols is called a* decorated SID *and it is denoted by* $\mathcal{R}_{dec}$. *The* decoration of a pair $(\varphi, \mathcal{R})$ *is a pair* $(\psi, \mathcal{R}_{dec})$, *where* ψ *is a decoration of* φ *and* $\mathcal{R}_{dec}$ *is the set of decorated rules obtained from* $\mathcal{R}$.

Example 3. Consider the following predicate p defined by the following rules in $\mathcal{R}$:

$$p(x, y, z, X) \Leftarrow x.f \to (y) \star x \not\simeq y \star X \approx \emptyset\,, \quad p(x, y, z, X) \Leftarrow y.f \to (x) \star X \approx \{y\}\,,$$
$$p(x, y, z, X) \Leftarrow z.f \to () \star X \approx \emptyset\,.$$

Let $\varphi = p(x, y, z, X) \star y \not\simeq z \star z.f \to (x, y)$. The formula $\psi = p_{I,J,\sim,\not\simeq}(x, y, z, X) \star y \not\simeq z \star z.f \to (x, y)$ is a decoration of φ with $y \not\equiv_\psi z$. If $\sim$ is the identity, then $\equiv_\psi$ is the identity. Moreover:

- If $I = \emptyset$, $J = \{1\}$ and $1 \not\simeq 2$ then $x \not\equiv_\psi y$, $alloc(\psi) = \{x, z\}$ and $[X]_\psi = \emptyset$. The only rule associated with $p_{I,J,\sim,\not\simeq}$ is: $p_{I,J,\sim,\not\simeq}(x, y, z, X) \Leftarrow x.f \to (y) \star x \not\simeq y \star X \approx \emptyset$.
- If $I = \{2\}$, $J = \{2\}$ and $\not\simeq$ is empty then $alloc(\psi) = \{y, z\}$ and $[X]_\psi = \{y\}$. The only rule associated with $p_{I,J,\sim,\not\simeq}$ is $p_{I,J,\sim,\not\simeq}(x, y, z, X) \Leftarrow y.f \to (x) \star X \approx \{y\}$.
- If $I = \emptyset$, $J = \{3\}$ then the decoration is inconsistent because we obtain $z \not\equiv_\psi z$. Indeed, the only rule associated with $p_{I,J,\sim,\not\simeq}$ is: $p_{I,J,\sim,\not\simeq}(x, y, z, X) \Leftarrow z \to () \star X \approx \emptyset$; this rule allocates z yielding a contradiction with the fact that z is already allocated by $z.f \to (x, y)$ in ψ.
- If $I = J = \emptyset$ then $p_{I,J,\sim,\not\simeq}$ has no rule, hence $p_{I,J,\sim,\not\simeq}(x, y, z, X)$ (hence also ψ) is (trivially) unsatisfiable.

The proof of the following theorem is given in Sect. B.

Theorem 2. *Let* $(\varphi, \mathcal{R})$ *be a pair in normal form. For each structure* $(\mathfrak{s}, \mathfrak{h}, \Sigma)$, *we have* $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi$ *if and only if there exists a decoration* $(\psi, \mathcal{R}_{dec})$ *of* $(\varphi, \mathcal{R})$ *such that* $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$.

3.2 Cardinality Constraints

We now begin the second step of the proof, which translates a decorated pair $(\varphi_d, \mathcal{R}_d)$ in normal form into a Presburger formula. We first consider the cardinality constraints on the sets of locations $[\![X]\!]_{(\mathfrak{s},\Sigma)}$ that appear in decorated predicate atoms $p_{I,J,\sim,\neq}(\vec{x}, X)$. The cardinalities of these sets are constrained by the (decorated) predicate's rules. These constraints are essential for satisfiability checking. For instance, as observed in the introduction, the formula $\mathtt{ls}_{\mathtt{even}}(x, Y) \circledast \mathtt{ls}_{\mathtt{odd}}(x, Y)$ is unsatisfiable if $\mathtt{ls}_{\mathtt{even}}$ and $\mathtt{ls}_{\mathtt{odd}}$ describe lists of even and odd lengths respectively, and Y is defined in both cases to collect the set of locations allocated by each list. We show that the cardinality constraints induced by each decorated predicate atom can be encoded, in linear time, as an existential Presburger formula. To achieve this, we rely on known results [25] (see also Sect. C) about the Presburger encoding of Parikh images of context-free languages.

Definition 16 (Cardinality of a decorated atom). *Let $p_{I,J,\sim,\neq}(\vec{x}, X)$ be a decorated predicate atom of $(\psi, \mathcal{R}_{dec})$. The* cardinality set *of $p_{I,J,\sim,\neq}(\vec{x}, X)$, which is denoted by $Sp(p_{I,J,\sim,\neq}(\vec{x}, X))$, is defined by:*

$$Sp(p_{I,J,\sim,\neq}(\vec{x}, X)) = \{\mathrm{card}\,(\Sigma(X)) \mid \exists(\mathfrak{s}, \mathfrak{h}, \Sigma).\ (\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\neq}(\vec{x}, X)\}\,, \tag{14}$$

i.e., it is a (possible unbounded) subset of $\mathbb{N}$, which collects the cardinalities of X for each structure satisfying $p_{I,J,\sim,\neq}(\vec{x}, X)$.

It is straightforward to show that $Sp(p_{I,J,\sim,\neq}(\vec{x}, X)) = Sp(p_{I,J,\sim,\neq}(\vec{x}, X)\sigma)$, for every renaming σ. In the following, we show that $Sp(p_{I,J,\sim,\neq}(\vec{x}, X))$ is a semi-linear set that can be encoded using an existential Presburger formula with one free variable. To this end, we define a context-free grammar for the predicate atom $p_{I,J,\sim,\neq}(\vec{x}, X)$ using the rules of $\mathcal{R}_{dec}$, such that the language of this grammar encodes the elements of $Sp(p_{I,J,\sim,\neq}(\vec{x}, X))$ in unary. The idea is to associate each predicate atom with a non-terminal of the grammar, and to associate each rule used in the unfolding with a production rule of the grammar. The set of non-terminals is finite because the set of predicate atoms is finite up to a renaming of parameters. The context-free grammar for $p_{I,J,\sim,\neq}(\vec{x}, X)$ is defined as follows.

Definition 17 (Grammar of cardinalities). *Let $\mathcal{R}_{dec}$ be a decorated SID. Let $a \mapsto N_a$ be any mapping from decorated atoms to symbols, such that $N_a = N_b$ iff a and b are identical up to a renaming. The* grammar of cardinalities *for the decorated atom $p_{I,J,\sim,\neq}(\vec{x}, X)$ is a context-free grammar $\mathcal{G}_{p_{I,J,\sim,\neq}(\vec{x},X)} = (\mathcal{N}, \mathcal{T}, \mathcal{R}, N_0)$ where*

- *the set of terminals $\mathcal{T}$ is the singleton set $\{1\}$;*
- *the set of non-terminals $\mathcal{N}$ contains a non-terminal $N_{q_{K,L,\sim,\neq}(\vec{y},Y)}$ for each decorated predicate atom $q_{K,L,\sim,\neq}(\vec{y}, Y)$ (up to a renaming);*
- *the start symbol N_0 is $N_{p_{I,J,\sim,\neq}(\vec{x},X)}$;*
- *the set of derivation rules $\mathcal{R}$ contains a rule of the form*

$$N_{q_{K,L,\sim,\neq}(\vec{y},Y)} \to \omega\ N_{r^1_{K_1,L_1,\sim_1,\neq_1}(\vec{y}_1,Y_1)}\ \ldots\ N_{r^m_{K_m,L_m,\sim_m,\neq_m}(\vec{y}_m,Y_m)}\,, \tag{15}$$

for all formulæ ψ such that $q_{K,L,\sim,\neq}(\vec{y}, Y) \Leftarrow_{\mathcal{R}_{dec}} \psi$ where:

- ψ *is of the form* $y_j.f \to (\vec{z}) \star \bigstar_{i=1}^{m} r^i_{K_i,L_i,\sim_i,\neq_i}(\vec{y_i}, Y_i) \star \varphi \star Y \approx E \sqcup \bigsqcup_{i=1}^{m} Y_i$;
- $\omega = 1$ *if* $E = \{y_j\}$, *otherwise (i.e., if* $E = \emptyset$*)* ω *is the empty word* ε.

We denote by $L(G)$ *the language of words generated by the grammar* G.

The Parikh image [21] of a word $\omega = \omega_1 \cdots \omega_k \in \mathcal{T}^*$ is the assignment $\sigma \in \mathbb{N}^{\mathcal{T}}$ that maps each symbol $a \in \mathcal{T}$ to the number of its occurrences in ω, i.e., $\sigma(a) = \text{card}(j \mid a = \omega_j)$. The Parikh image of a set $L \subseteq \mathcal{T}^\star$ is the set of Parikh images of all $\omega \in L$. If a language L is generated by a context-free grammar G, then the Parikh image of L can be represented by an existential Presburger formula ξ_G, which can be computed in linear time, by [25, Th. 4] (also given in Sect. C). The formula ξ_G contains one free variable for each terminal symbol of G. We shall prove later (see in particular Prop. 5 in Sect. A) that the Parikh image of the language defined by $\mathcal{G}_{p_{I,J,\sim,\neq}(\vec{x},X)}$ is *exactly* the cardinality set $Sp(p_{I,J,\sim,\neq}(\vec{x}, X))$. Since the grammar of cardinalities $G = \mathcal{G}_{p_{I,J,\sim,\neq}(\vec{x},X)}$ has only one terminal, the formula ξ_G contains exactly one free variable and its size is in $O(|\mathcal{N}| + |\mathcal{N}| \cdot |\mathcal{R}|)$ where $\mathcal{N}$ and $\mathcal{R}$ are (respectively) the set of non-terminals and productions of $\mathcal{G}_{p_{I,J,\sim,\neq}(\vec{x},X)}$. To simplify notations, we shall denote by $\xi_{p_{I,J,\sim,\neq}(\vec{x},X)}$ the formula ξ_G with $G = \mathcal{G}_{p_{I,J,\sim,\neq}(\vec{x},X)}$, and by $\xi_{p_{I,J,\sim,\neq}(\vec{x},X)}(y)$ the formula obtained by replacing the unique free variable in ξ_G by y.

Remark 3. A smaller Presburger formula for $Sp(p_{I,J,\sim,\neq}(\vec{x}, X))$ may be obtained by translating $\mathcal{R}_{dec}$ into a NFA over a singleton alphabet and by using the result of [24]. The resulting formula, obtained in polynomial time, does not contain universal quantifiers and its size is linear in the size of the NFA. We choose to use context-free grammars and the result in [25] in order to simplify the proof of correctness.

3.3 Translation from SL to BAPA

The syntax of BAPA formulæ is recalled below, with definitions slightly adapted for consistency with those of OSL[6].

Definition 18 (BAPA formulæ [13]). *The set of* BAPA-formulæ φ^{BP} *is inductively defined as follows:*

$$\begin{array}{l} \varphi^{BP} := A^{BP} \mid B^{BP} \mid \varphi_1^{BP} \wedge \varphi_2^{BP} \mid \varphi_1^{BP} \vee \varphi_2^{BP} \mid \neg\varphi^{BP} \mid \exists X.\varphi^{BP} \mid \forall X.\varphi^{BP} \mid \exists k.\varphi^{BP} \mid \forall k.\varphi^{BP} \\ A^{BP} := t_1^{BP} \approx_{BP} t_2^{BP} \mid t_1^{BP} \prec_{BP} t_2^{BP} \mid K \,\text{div}\, t^{BP} \quad t^{BP} := k \mid K \mid t_1^{BP} \oplus_{BP} t_2^{BP} \mid K \odot_{BP} t^{BP} \mid |T^{BP}| \\ B^{BP} := T_1^{BP} \approx_{BP} T_2^{BP} \mid T_1^{BP} \sqsubseteq_{BP} T_2^{BP} \qquad T^{BP} := X \mid \emptyset \mid T_1^{BP} \sqcup_{BP} T_2^{BP} \mid T_1^{BP} \sqcap_{BP} T_2^{BP} \end{array} \tag{16}$$

where X *is a set variable,* K *is an integer constant, and* k *is an integer variable.*

Definition 19 (BAPA structure). *A* BAPA structure *is a tuple* $(\mathbb{S}, \mathbb{I})$ *where* $\mathbb{S}$ *is an interpretation of the set variables, i.e., a partial function from set variables to finite subsets* $\mathcal{L}$*, and* $\mathbb{I}$ *is a interpretation of the integer variables, i.e., a partial function from integer variables to* $\mathbb{Z}$.

[6] Specifically, instead of assuming a finite universe, we interpret set variables as finite subsets of $\mathcal{L}$, and omit all BAPA terms that depend on the universe, such as the complement operation.

The semantics of BAPA terms and formulæ are the expected ones (see Sect. A).

Definition 20. *The function $\mathcal{T}^f$ maps a decorated symbolic heap formula in OSL to a BAPA term containing all (named) locations allocated by a field $f \in \mathcal{F}$. It is defined inductively as follows (where L, A and B denote pure atoms, defined as in Def. 1):*

- $\mathcal{T}^f(\mathtt{emp}) = \mathcal{T}^f(L) = \mathcal{T}^f(A) = \mathcal{T}^f(B) = \emptyset$;
- $\mathcal{T}^f(x.g \to (y_1, \ldots, y_d)) = \begin{cases} V_x \text{ if } g = f \\ \emptyset \quad \text{otherwise} \end{cases}$;
- $\mathcal{T}^f(p_{I,J,\sim,\not\approx}(x_1, \ldots, x_n, X)) = \begin{cases} X \sqcup_{BP} \left(\bigsqcup_{j\in J}^{BP} V_{x_j}\right) \text{ if } p \text{ is a } \{f\}\text{-predicate} \\ \emptyset \qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$;
- $\mathcal{T}^f(\psi_1 \bullet \psi_2) = \mathcal{T}^f(\psi_1) \cup \mathcal{T}^f(\psi_2)$ *for* $\bullet \in \{\star, \circledast\}$.

Definition 21 (BAPA translation function). *Let $X \mapsto i_X$ be an injective function mapping every set variable to an integer variable (which is intended to denote the cardinality of the set). The function $\mathcal{T}$ translates a set term in $\mathfrak{T}$ into a BAPA set term and a decorated symbolic heap formula into a BAPA formula, given a decorated SID $\mathcal{R}_{dec}$ (which, to simplify notations, is considered fixed in the context):*

- $\mathcal{T}(\{x\}) = V_x$; $\mathcal{T}(X) = X$ *if* $X \in \mathcal{S}$; $\mathcal{T}(\emptyset) = \emptyset$; $\mathcal{T}(t) = t$ *if* t *is an arithmetic term;*
- $\mathcal{T}(\mathtt{emp}) = \text{true}$; $\mathcal{T}(x.f \to (y_1, \ldots, y_d)) = \text{true}$;
- $\mathcal{T}(x \approx y) = (V_x \approx_{BP} V_y)$; $\mathcal{T}(x \not\approx y) = (V_x \sqcap_{BP} V_y \approx_{BP} \emptyset)$;
- $\mathcal{T}(T_1 \approx T_2) = (\mathcal{T}(T_1) \approx_{BP} \mathcal{T}(T_2))$; *the definition is similar for* $\sqcup$, $\sqcap$, $\sqsubseteq$ *or* $<$;
- $\mathcal{T}(T_1 \not\approx T_2) = \neg(\mathcal{T}(T_1) \approx_{BP} \mathcal{T}(T_2))$; *the definition is similar for* $\not\sqsubseteq$ *or* $\not<$;
- $\mathcal{T}(K \operatorname{div} t) = (K \operatorname{div} \mathcal{T}(t))$; $\mathcal{T}(K \operatorname{ndiv} t) = \neg(K \operatorname{div} \mathcal{T}(t))$;
- $\mathcal{T}(p_{I,J,\sim,\not\approx}(x_1, \ldots, x_n, X)) = |X| \approx_{BP} i_X \wedge \xi_{p_{I,J,\sim,\not\approx}(x_1,\ldots,x_n,X)}(i_X)$
  $\wedge \left(\bigsqcup_{i\in I}^{BP} V_{x_i}\right) \sqsubseteq_{BP} X$
  $\wedge \left(\bigsqcup_{x\in\{x_1,\ldots,x_n\}\setminus\{x_j \mid j\in I\}}^{BP} V_x\right) \sqcap_{BP} X \approx_{BP} \emptyset$;
  where the formula $\xi_{p_{I,J,\sim,\not\approx}(x_1,\ldots,x_n,X)}$ *is defined as in Sect. 3.2.*
- $\mathcal{T}(\psi_1 \star \psi_2) = \mathcal{T}(\psi_1) \wedge \mathcal{T}(\psi_2) \wedge \left(\bigsqcup_{f\in\mathcal{F}}^{BP} \mathcal{T}^f(\psi_1)\right) \sqcap_{BP} \left(\bigsqcup_{f\in\mathcal{F}}^{BP} \mathcal{T}^f(\psi_2)\right) \approx_{BP} \emptyset$;
- $\mathcal{T}(\psi_1 \circledast \psi_2) = \mathcal{T}(\psi_1) \wedge \mathcal{T}(\psi_2) \wedge \bigwedge_{f\in\mathcal{F}} \left(\mathcal{T}^f(\psi_1) \sqcap_{BP} \mathcal{T}^f(\psi_2) \approx_{BP} \emptyset\right)$.

Proposition 1. *Let $T \in \mathfrak{T} \cup \mathfrak{t}$ be a set term or an arithmetic term. Let $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ be an OSL structure. Let $\mathbb{S}$ be defined by $\mathbb{S}(V_x) = \{\mathfrak{s}(x)\}$ for all location variables x, and $\mathbb{S}(X) = \Sigma(X)$ for all set variables X. Then, for every interpretation $\mathbb{I}$, $[\![T^{SL}]\!]_{(\mathfrak{s},\Sigma)} = [\![\mathcal{T}(T)]\!]_{(\mathbb{S},\mathbb{I})}$.*

Proof. The proof is by induction on T and t (see Sect. A).

For each decorated symbolic heap ψ in normal form containing location variables $x_1, \ldots, x_n$, we associate the following BAPA formula called $C(\psi)$:

$$C(\psi) = |V_{x_1}| \approx_{BP} 1 \wedge \cdots \wedge |V_{x_n}| \approx_{BP} 1 \wedge \mathcal{T}(\psi)\,. \tag{17}$$

Example 4. Let us consider the decorated formula $\psi = p_{I,J,\sim,\neq}(x,y,z,X) \star y \not\approx z \star z.f \to (x,y)$ taken from Ex. 3 with $I = \{2\}$, $J = \{2\}$, $\sim =$ Id, and $\neq = \emptyset$.

Then the BAPA formula obtained is:

$$\begin{aligned} C(\psi) = {} & |V_x| \approx_{BP} 1 \wedge |V_y| \approx_{BP} 1 \wedge |V_z| \approx_{BP} 1 \wedge |X| \approx_{BP} i_X \wedge \xi_{p_{I,J,\sim,\neq}(x,y,z,X)}(i_X) \\ & \wedge V_y \sqsubseteq_{BP} X \wedge (V_x \sqcup_{BP} V_z) \sqcap_{BP} X \approx_{BP} \emptyset \wedge V_y \sqcap_{BP} V_z \approx_{BP} \emptyset \wedge \text{true} \\ & \wedge \left(X \sqcup_{BP} V_y\right) \sqcap_{BP} V_Z \approx_{BP} \emptyset \,. \end{aligned} \tag{18}$$

It is straightforward to verify that, for all decorations ψ of a formula φ, $C(\psi)$ is an existentially quantified BAPA formula. This translation allows us to establish the decidability of the satisfiability problem for a decorated symbolic heap (see Sect. E):

Theorem 3. *Let $(\varphi, \mathcal{R})$ an OSL-formula and a SID in normal form. For all decorations $(\psi, \mathcal{R}_{dec})$ of $(\varphi, \mathcal{R})$, ψ has an $\mathcal{R}_{dec}$-model iff $C(\psi)$ is satisfiable (in BAPA).*

Theorem 1 then follows directly from Theorems 2 and 3. The NEXPTIME upper bound is obtained by observing that the size of the decorated, normalized set of rules is exponential in the size of the original SID, since the number of possible decorations and instantiations is exponential. Moreover, it is shown in [15] that satisfiability for existentially quantified BAPA formulæ is in NP.

4 Conclusion

We have developed a decision procedure to test the satisfiability of formulæ in a variant of separation logic (SL) that integrates inductively defined predicate symbols, overlaid data structures, set constraints, and Presburger arithmetic constraints on set cardinalities. This fragment, designed to address the limitations of standard SL in reasoning about overlaid data structures, uses a weaker form of separating conjunction and supports single-field inductive predicates augmented with set variables. Our procedure operates in exponential space, leveraging established results on the computation of the Parikh image of context-free languages over a unary alphabet and the satisfiability problem in the logic BAPA. This reduction yields a robust framework for handling complex memory models occurring in concurrent systems and advanced algorithmic designs, where locations may be shared across multiple data structures.

A natural extension of this work is to explore the entailment problem. Given the undecidability of entailment in many expressive SL variants (see for instance [5,22]), additional restrictions will be required to ensure decidability. Several questions also remain unresolved regarding satisfiability. First, the optimality of our procedure is uncertain: while satisfiability is clearly EXPTIME-hard, it is not clear whether NEXPTIME represents a tight upper bound. Second, it would be valuable to investigate whether the systematic enumeration of all decorations could be circumvented. This process currently incurs an exponential computational overhead, which does not affect the theoretical complexity analysis but poses practical challenges. Finally, it would be interesting to determine whether the conditions on the inductive rules could be relaxed. For instance, is satisfiability still decidable if a single predicate is permitted to allocate distinct fields?

References

1. Josh Berdine, Cristiano Calcagno, and Peter W. O'Hearn. Symbolic Execution with Separation Logic. In Kwangkeun Yi, editor, *Programming Languages and Systems*, volume 3780 of *Lecture Notes in Computer Science*, pages 52–68, Berlin, Heidelberg, 2005. Springer. doi:10.1007/11575467_5.
2. James Brotherston, Carsten Fuhs, Juan A. Navarro Pérez, and Nikos Gorogiannis. A decision procedure for satisfiability in separation logic with inductive predicates. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, CSL-LICS '14, pages 1–10, New York, NY, USA, July 2014. Association for Computing Machinery. doi:10.1145/2603088.2603091.
3. Cezara Drăgoi, Constantin Enea, and Mihaela Sighireanu. Local Shape Analysis for Overlaid Data Structures. In Francesco Logozzo and Manuel Fähndrich, editors, *Static Analysis*, volume 7935 of *Lecture Notes in Computer Science*, pages 150–171, Berlin, Heidelberg, 2013. Springer. doi:10.1007/978-3-642-38856-9_10.
4. Mnacho Echenim, Radu Iosif, and Nicolas Peltier. Decidable Entailments in Separation Logic with Inductive Definitions: Beyond Establishment. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2021.20.
5. Mnacho Echenim, Radu Iosif, and Nicolas Peltier. Entailment is Undecidable for Symbolic Heap Separation Logic Formulæ with Non-Established Inductive Rules. *Information Processing Letters*, 173:106169, January 2022. doi:10.1016/j.ipl.2021.106169.
6. Constantin Enea, Vlad Saveluc, and Mihaela Sighireanu. Compositional Invariant Checking for Overlaid and Nested Linked Lists. In Matthias Felleisen and Philippa Gardner, editors, *Programming Languages and Systems*, volume 7792 of *Lecture Notes in Computer Science*, pages 129–148, Berlin, Heidelberg, 2013. Springer. doi:10.1007/978-3-642-37036-6_9.
7. Xincai Gu, Taolue Chen, and Zhilin Wu. A Complete Decision Procedure for Linearly Compositional Separation Logic with Data Constraints. In Nicola Olivetti and Ashish Tiwari, editors, *Automated Reasoning*, volume 9706 of *Lecture Notes in Computer Science*, pages 532–549, Cham, 2016. Springer International Publishing. doi:10.1007/978-3-319-40229-1_36.
8. Christoph Haase. A survival guide to presburger arithmetic. *ACM SIGLOG News*, 5(3):67–82, July 2018. doi:10.1145/3242953.3242964.
9. Radu Iosif, Adam Rogalewicz, and Jiri Simacek. The Tree Width of Separation Logic with Recursive Definitions. In Maria Paola Bonacina, editor, *Automated Deduction – CADE-24*, volume 7898 of *Lecture Notes in Computer Science*, pages 21–38, Berlin, Heidelberg, 2013. Springer. ISSN: 1611-3349. doi:10.1007/978-3-642-38574-2_2.
10. Samin S. Ishtiaq and Peter W. O'Hearn. BI as an assertion language for mutable data structures. In *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '01, pages 14–26, New York, NY, USA, January 2001. Association for Computing Machinery. doi:10.1145/360204.375719.
11. Siddharth Krishna, Dennis Shasha, and Thomas Wies. Go with the flow: compositional abstractions for concurrent data structures. *Proceedings of the ACM on Programming Languages*, 2(POPL):37 : 1 – 31, December 2017. doi:10.1145/3158125.
12. Manfred Kufleitner. Yet another proof of Parikh's Theorem, October 2022. doi:10.48550/arXiv.2210.02925.

13. Viktor Kuncak, Huu Hai Nguyen, and Martin Rinard. An Algorithm for Deciding BAPA: Boolean Algebra with Presburger Arithmetic. In Robert Nieuwenhuis, editor, *Automated Deduction – CADE-20*, volume 3632 of *Lecture Notes in Computer Science*, pages 260–277, Berlin, Heidelberg, 2005. Springer. doi:10.1007/11532231_20.
14. Viktor Kuncak, Huu Hai Nguyen, and Martin Rinard. Deciding Boolean Algebra with Presburger Arithmetic. *Journal of Automated Reasoning*, 36(3):213–239, April 2006. doi:10.1007/s10817-006-9042-1.
15. Viktor Kuncak and Martin Rinard. Towards Efficient Satisfiability Checking for Boolean Algebra with Presburger Arithmetic. In Frank Pfenning, editor, *Automated Deduction – CADE-21*, volume 4603 of *Lecture Notes in Computer Science*, pages 215–230, Berlin, Heidelberg, 2007. Springer. doi:10.1007/978-3-540-73595-3_15.
16. Quang Loc Le and Xuan-Bach D. Le. An Efficient Cyclic Entailment Procedure in a Fragment of Separation Logic. In Orna Kupferman and Pawel Sobocinski, editors, *Foundations of Software Science and Computation Structures*, volume 13992 of *Lecture Notes in Computer Science*, pages 477–497, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-30829-1_23.
17. Oukseh Lee, Hongseok Yang, and Rasmus Petersen. Program Analysis for Overlaid Data Structures. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 592–608, Berlin, Heidelberg, 2011. Springer. doi:10.1007/978-3-642-22110-1_48.
18. Oukseh Lee, Hongseok Yang, and Rasmus Petersen. A divide-and-conquer approach for analysing overlaid data structures. *Formal Methods in System Design*, 41(1):4–24, August 2012. doi:10.1007/s10703-012-0151-7.
19. Christoph Matheja, Jens Pagel, and Florian Zuleger. A Decision Procedure for Guarded Separation Logic Complete Entailment Checking for Separation Logic with Inductive Definitions. *ACM Transactions on Computational Logic*, 24(1):1 : 1–76, January 2023. doi:10.1145/3534927.
20. Roland Meyer, Thomas Wies, and Sebastian Wolff. Make Flows Small Again: Revisiting the Flow Framework. In Sriram Sankaranarayanan and Natasha Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 13993 of *Lecture Notes in Computer Science*, pages 628–646, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-30823-9_32.
21. Rohit J. Parikh. On Context-Free Languages. *J. ACM*, 13(4):570–581, October 1966. doi:10.1145/321356.321364.
22. Nicolas Peltier. Testing the Satisfiability of Formulas in Separation Logic with Permissions. In Revantha Ramanayake and Josef Urban, editors, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 14278 of *Lecture Notes in Computer Science*, pages 427–445, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-43513-3_23.
23. J.C. Reynolds. Separation logic: a logic for shared mutable data structures. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*, pages 55–74, Copenhagen, Denmark, July 2002. ISSN: 1043-6871. doi:10.1109/LICS.2002.1029817.
24. Zdeněk Sawa. Efficient Construction of Semilinear Representations of Languages Accepted by Unary Nondeterministic Finite Automata. *Fundamenta Informaticae*, 123(1):97–106, April 2013. Publisher: SAGE Publications. doi:10.3233/FI-2013-802.
25. Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. On the Complexity of Equational Horn Clauses. In Robert Nieuwenhuis, editor, *Automated Deduction – CADE-20*, volume 3632 of *Lecture Notes in Computer Science*, pages 337–352, Berlin, Heidelberg, 2005. Springer. doi:10.1007/11532231_25.

A Properties of Decorations

We establish a number of basic results that will be useful in subsequent proofs.

Proposition 2. *Let p be a $\{f\}$-predicate. Let $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ be a model of $p(x_1, \ldots, x_n, X)$. Then, $\Sigma(X) \subseteq \mathrm{dom}_f(\mathfrak{h})$.*

Proof. The proof is done by induction on the size of $\mathfrak{h}$. Let $p(x_1, \ldots, x_n, X)$ be a predicate and let $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ be one of its models. There exist a rule $p(z_1, \ldots, z_n, Z) \Leftarrow \varphi'$ in $\mathcal{R}$ and a substitution σ such that $(z_1, \ldots, z_n, Z)\sigma = (x_1, \ldots, x_n, X)$, and $(\mathfrak{s}', \mathfrak{h}, \Sigma') \models_{\mathcal{R}} p(z_1, \ldots, z_n, Z)$ with $\mathfrak{s}' = \mathfrak{s} \circ \sigma$ and $\Sigma' = \Sigma \circ \sigma$. According to Def. 4, there exist a store $\mathfrak{s}''$ and a set interpretation Σ'' such that $(\mathfrak{s}'', \mathfrak{h}, \Sigma'') \models_{\mathcal{R}} \varphi'$, and $\mathfrak{s}''$ matches with $\mathfrak{s}'$ on $z_1, \ldots, z_n$, and $\Sigma''(Z) = \Sigma'(Z)$. According to Eq. 6 and Def. 4, $Z \approx E \sqcup \bigsqcup_{u \in U} Y_u$ and thus $\Sigma''(Z) = [\![E]\!]_{(\mathfrak{s},\Sigma)} \cup \bigcup_{u \in U} \Sigma''(Y_u)$. By the induction hypothesis, for all $u \in U$, $\Sigma''(Y_u) \subseteq \mathrm{dom}_f(\mathfrak{h}_u)$ with $\mathfrak{h}_u$ being the sub-heap of $\mathfrak{h}$ such that $(\mathfrak{s}'', \mathfrak{h}_u, \Sigma'') \models_{\mathcal{R}} q_u(\vec{y}_u, Y_u)$. Moreover, as $E = \{x\}$ or $E = \emptyset$, $[\![E]\!]_{(\mathfrak{s},\Sigma)} \subseteq \mathrm{dom}_f(\mathfrak{h})$. Finally, as for all $u \in U, \mathrm{dom}_f(\mathfrak{h}_u) \subseteq \mathrm{dom}_f(\mathfrak{h})$, $\Sigma(X) = \Sigma''(Z) \subseteq \mathrm{dom}_f(\mathfrak{h})$.

Proposition 3. *Let $p_{I,J,\sim,\not\approx}(x_1, \ldots, x_n, X) \Leftarrow \psi$ be a rule in $\mathcal{R}_{dec}$. Then, $[X]_\psi \subseteq alloc(\psi)$.*

Proof. The result follows from the form of the rules in Eq. 6, which ensures that all the variables that are added in X are allocated. More formally, consider any variable $x \in [X]_\psi$. By definition, there exists a location variable y such that $x \equiv_\psi y$ and by Def. 10 one of the following conditions hold:

- ψ contains an equation $X \approx (E \sqcup \bigsqcup_{u \in U} Y_u)$ and $y \in E$. Then, since the rules in $\mathcal{R}_{dec}$ are of the form of Eq. 6, ψ also contains a points-to atom $y.f \to (\vec{z})$ thus $y \in alloc(\psi)$ and consequently $x \in alloc(\psi)$.
- ψ contains an atom $q_{I',J',\sim',\not\approx'}(y_1, \ldots y_{n'}, Y)$ and $y = y_i$ with $i \in I'$. Then $i \in J'$ (as $I' \subseteq J'$ by Def. 8) so that $y \in alloc(\psi)$ and therefore $x \in alloc(\psi)$.

Proposition 4. *Let $\psi = p_{I,J,\sim,\not\approx}(x_1, \ldots, x_n, X)$ be a decorated predicate atom. If $\omega \in L(G_{\psi\sigma})$, for some substitution σ, then $|\omega| \geq \mathrm{card}(\{[x]_{\equiv_\psi} \mid x \in [X]_\psi\})$, where $[x]_{\equiv_\psi}$ denotes the equivalence class of x w.r.t. the relation $\equiv_\psi$.*

Proof. The proof is by induction on the length of the derivation yielding the word 1^ω in the grammar associated with $Sp(\psi\sigma)$. By definition of the grammar (see Def. 17), $\omega = \mathrm{card}(E) + \sum_{i \in U} \omega_i$, where $\mathcal{R}_{dec}$ contains a rule of the form (up to a renaming of the set variables) $p(x'_1, \ldots, x'_n, X) \Leftarrow \psi'$, $\psi'\theta\sigma$ is consistent, $\theta = \{x'_i \mapsto x_i \mid 1 \leq i \leq n\}$, ψ' is of the form $y_r.f \to (\vec{z}) \star \bigstar_{u=1}^{m} q^u_{K_u,L_u,\sim_u,\not\approx_u}(\vec{y}_u, Y_u) \star \varphi \star X \approx E \sqcup \bigsqcup_{u \in U} Y_u$ and $\omega_u \in L(G_{q^u_{K_u,L_u,\sim_u,\not\approx_u}(\vec{y}_u,Y_u)\theta\sigma})$. Let $\psi_u = q^u_{K_u,L_u,\sim_u,\not\approx_u}(\vec{y}_u, Y_u)\theta$. By the induction hypothesis, $\omega_u \geq \mathrm{card}(\{[x]_{\equiv_{\psi_u}} \mid x \in [Y_u]_{\psi_u}\})$. Furthermore, by definition of decorated rules (Def. 15), $\{[x]_{\equiv_\psi} \mid x \in [X]_\psi\} = \{[x]_{\equiv_\psi} \mid x \in [X]_{\psi'\theta}\}$. By Def. 13, we deduce that $\{[x]_{\equiv_\psi} \mid x \in [X]_\psi\} \subseteq E \cup \bigcup_{u \in U} \{[x]_{\equiv_\psi} \mid x \in [Y_u]_{\psi_u}\}$, hence $\mathrm{card}(\{[x]_{\equiv_\psi} \mid x \in [X]_\psi\}) \leq \mathrm{card}(E) + \sum_{u \in U} \mathrm{card}(\{[x]_{\equiv_\psi} \mid x \in [Y_u]_{\psi_u}\})$. Furthermore, $\equiv_{\psi_u}$ is included in $\equiv_\psi$, so that $\mathrm{card}(\{[x]_{\equiv_\psi} \mid x \in [X]_{\psi_u}\}) \leq \mathrm{card}(\{[x]_{\equiv_{\psi_u}} \mid x \in [X]_{\psi_u}\}) \leq \omega_u$. Thus $\mathrm{card}(\{[x]_{\equiv_\psi} \mid x \in [X]_\psi\} \leq \mathrm{card}(E) + \sum_{u \in U} \omega_u = \omega$.

Proposition 5. *Let $\mathcal{R}_{dec}$ be a decorated SID, let G be the grammar of cardinalities for the decorated atom $p_{I,J,\sim,\neq}(\vec{x}, X)$ and let $\xi_G(i)$ be the Presburger formula encoding the Parikh image of $L(G)$ (where i is the unique free variable in ξ_G). If $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\neq}(\vec{x}, X)$ and $K = \mathrm{card}(\Sigma(X))$, then $\xi_G\{i \mapsto k\}$ is satisfiable.*

Proof. Let $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ be a $\mathcal{R}_{dec}$-model of $p_{I,J,\sim,\neq}(\vec{x}, X)$ and let $K = \mathrm{card}\,\Sigma(X)$. By definition of the rules in G (and using the fact that $\Sigma(X) \subseteq \mathrm{dom}(\mathfrak{h})$), it is easy to check that there exists a derivation in $\mathcal{G}_X$ ensuring that $1^K \in Sp(p_{I,J,\sim,\neq}(\vec{x}, X))$. Thus, by definition, the Parikh image of 1^K is K. As a result, according to [25], $\xi_{p_{I,J,\sim,\neq}(\vec{x},X)}(K)$ is satisfiable.

B Proof of Theorem 2

We recall Theorem 2:

Theorem 2. *Let $(\varphi, \mathcal{R})$ be a pair in normal form. For each structure $(\mathfrak{s}, \mathfrak{h}, \Sigma)$, we have $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi$ if and only if there exists a decoration $(\psi, \mathcal{R}_{dec})$ of $(\varphi, \mathcal{R})$ such that $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$.*

Proof. The completeness is established by Thm. 4 and the correctness is established by Thm. 5 (see below).

B.1 Completeness of decorations

We first establish some lemmata relating the relations $\equiv_\psi$ and $\not\equiv_\psi$ to the aliasing and non-aliasing relations between variables, respectively; the set $alloc(\psi)$ to the set of allocated variables; and the set $[X]_\psi$ to the image of X.

Lemma 1. *For every decorated quantifier-free symbolic heap ψ, for all x, y, if $x \equiv_\psi y$, then for all models $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ of ψ, $\mathfrak{s}(x) = \mathfrak{s}(y)$.*

Proof. If $x = y$, then $\mathfrak{s}(x) = \mathfrak{s}(y)$ follows immediately. We consider $x \neq y$. By induction on the definition of $\models_{\mathcal{R}_{dec}}$:

- $\psi = \mathtt{emp}$ **or** $\psi = x_0 \not\approx y_0$ **or** $\psi = y_0.f \to (y_1, \ldots, y_d)$ **or** $\psi = A$ **or** $\psi = B$**:** In these cases, there are no distinct x, y such as $x \equiv_\psi y$ because ψ does not contain any predicate or $\approx$.
- $\psi = x_0 \approx y_0$**:** Only x_0 and y_0 satisfy $x_0 \equiv_\psi y_0$ and according to Def. 4 for all models $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ of ψ, $\mathfrak{s}(x_0) = \mathfrak{s}(y_0)$.
- $\psi = \psi_1 \star \psi_2$ **or** $\psi = \psi_1 \circledast \psi_2$**:** Because $x \equiv_\psi y$, there exists a sequence $(x_k)_{k \in [\![1,l]\!]}$ such that $x_1 = x$, $x_l = y$ and for all $k \in [\![1, l]\!]$, ψ contains either an atom $x_k \approx x_{k+1}$ or a decorated predicate atom $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ with $x_i = x_k$, $x_j = x_{k+1}$ and $i \sim j$. Let us take $k \in [\![1, l]\!]$. In both previously presented cases, the atom $x_k \approx x_{k+1}$ or $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ appears either in ψ_1 or in ψ_2, we call j the corresponding index. According to Def. 4 there exist heaps $\mathfrak{h}_1$, $\mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, and $(\mathfrak{s}, \mathfrak{h}_i, \Sigma) \models_{\mathcal{R}_{dec}} \psi_i$, for all $i \in \{1, 2\}$. By definition of $\equiv$, $x_k \equiv_{\psi_j} x_{k+1}$, and thus, by the induction hypothesis, $\mathfrak{s}(x_k) = \mathfrak{s}(x_{k+1})$. As a conclusion, $\mathfrak{s}(x_1) = \mathfrak{s}(x_2) = \cdots = \mathfrak{s}(x_{l-1}) = \mathfrak{s}(x_l)$, hence, $\mathfrak{s}(x) = \mathfrak{s}(y)$.

$\psi = p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$**:** By definition of $\equiv_\psi$, there exist i, j such that $x_i = x$, $x_j = y$, and $i \sim j$. Let $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ be a model of ψ. There exist a rule[7] $p_{I,J,\sim,\neq}(z_1, \ldots, z_n, X) \Leftarrow \psi'$ in $\mathcal{R}_{dec}$ and a substitution σ such that $(z_1, \ldots, z_n)\sigma = (x_1, \ldots, x_n)$, and, with $\mathfrak{s}' = \mathfrak{s} \circ \sigma$, $(\mathfrak{s}', \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\neq}(z_1, \ldots, z_n, X)$. According to Def. 4, there exist a store $\mathfrak{s}''$ and a set interpretation Σ'' such that $(\mathfrak{s}'', \mathfrak{h}, \Sigma'') \models_{\mathcal{R}_{dec}} \psi'$, and $\mathfrak{s}''$ matches with $\mathfrak{s}'$ on $z_1, \ldots, z_n$, and $\Sigma''(X) = \Sigma(X)$. Because the rule is in $\mathcal{R}_{dec}$ and $i \sim j$, $z_i \equiv_{\psi'} z_j$, and thus, by the induction hypothesis, $\mathfrak{s}''(z_i) = \mathfrak{s}''(z_j)$, i.e. $\mathfrak{s}'(z_i) = \mathfrak{s}'(z_j)$, i.e., $\mathfrak{s}(\sigma(z_i)) = \mathfrak{s}(\sigma(z_j))$. As a consequence, $\mathfrak{s}(x_i) = \mathfrak{s}(x_j)$, and thus, $\mathfrak{s}(x) = \mathfrak{s}(y)$.

Lemma 2. *Let $(\psi, \mathcal{R}_{dec})$ be in normal form. If $x \in alloc(\psi)$ then for all models $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ such that $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$, $\mathfrak{s}(x) \in allocated(\mathfrak{h})$.*

Proof. As a consequence of Lem. 1, if $x \in alloc(\psi)$ and $x \equiv_\psi y$, then for all models $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ of ψ, $\mathfrak{s}(x) = \mathfrak{s}(y)$, and thus, if $\mathfrak{s}(x) \in$ allocated$(\mathfrak{h})$, then $\mathfrak{s}(y) \in$ allocated$(\mathfrak{h})$. Therefore, we have to prove the lemma for the two other cases of the construction of $alloc(\psi)$ (see Def. 11). The proof is by induction on the definition of $\models_{\mathcal{R}_{dec}}$.

$\psi = \mathtt{emp}$ **or** $\psi = x_0 \not\approx y_0$ **or** $\psi = x_0 \approx y_0$ **or** $\psi = A$ **or** $\psi = B$**:** There is no location variable in $alloc(\psi)$.

$\psi = y_0.f \rightarrow (y_1, \ldots, y_d)$**:** The set $alloc(\psi)$ is the singleton $\{y_0\}$. All the models of ψ are such that $\mathfrak{h} = [(\mathfrak{s}(y_0), f) \rightarrow (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_d))]$. And thus for all models, $\mathfrak{s}(y_0) \in$ allocated$(\mathfrak{h})$.

$\psi = \psi_1 \star \psi_2$ **or** $\psi = \psi_1 \circledast \psi_2$**:** Because $x \in alloc(\psi)$, we may assume (by the above remark concerning the construction of $alloc(\psi)$) that ψ contains either a points-to allocating x or a decorated predicate atom $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ with $x_j = x$ and $j \in J$. In both cases, this atom appears either in ψ_1 or in ψ_2, we call j the corresponding index. According to Def. 4 there exist heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$ and $(\mathfrak{s}, \mathfrak{h}_i, \Sigma) \models_{\mathcal{R}_{dec}} \psi_i$, for all $i \in \{1, 2\}$. By definition of $alloc$, $x \in alloc(\psi_j)$, and thus, by the induction hypothesis, $\mathfrak{s}(x) \in$ allocated$(\mathfrak{h}_j)$. Because $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, $\mathfrak{s}(x) \in$ allocated$(\mathfrak{h})$.

$\psi = p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$**:** By the same remark concerning the construction of $alloc(\psi)$, we may assume that there exists j such that $x_j = x$ and $j \in J$. Let $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ be a model of ψ. There exist a rule[8] $p_{I,J,\sim,\neq}(z_1, \ldots, z_n, X) \Leftarrow \psi'$ in $\mathcal{R}_{dec}$ and a substitution σ such that $(z_1, \ldots, z_n)\sigma = (x_1, \ldots, x_n)$, and $(\mathfrak{s}', \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\neq}(z_1, \ldots, z_n, X)$ with $\mathfrak{s}' = \mathfrak{s} \circ \sigma$. According to Def. 4, there exist a store $\mathfrak{s}''$ and a set interpretation Σ'' such that $(\mathfrak{s}'', \mathfrak{h}, \Sigma'') \models_{\mathcal{R}_{dec}} \psi'$, and $\mathfrak{s}''$ matches with $\mathfrak{s}'$ on $z_1, \ldots, z_n$, and $\Sigma''(X) = \Sigma(X)$. Because the rule is in $\mathcal{R}_{dec}$ and $j \in J$, $z_j \in alloc(\psi')$, and thus, by the induction hypothesis, $\mathfrak{s}''(z_j) \in$ allocated$(\mathfrak{h})$, i.e., $\mathfrak{s}'(z_j) \in$ allocated$(\mathfrak{h})$. As a consequence, $\mathfrak{s}(x) = \mathfrak{s}(x_j) = \mathfrak{s}(\sigma(z_j)) \in$ allocated$(\mathfrak{h})$.

Lemma 3. *Let $(\psi, \mathcal{R}_{dec})$ be in normal form. For all x and y, if $x \not\equiv_\psi y$, then for all models $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ of ψ, $\mathfrak{s}(x) \neq \mathfrak{s}(y)$.*

Proof. The proof is by induction on $\not\equiv_\psi$ and $\models_{\mathcal{R}_{dec}}$.

[7] We may assume by renaming that the set variable occurring in the left-hand side of the rule is X.

[8] Same assumption as previously.

(a) If $x \not\approx y$ occurs in ψ, then, as $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$, necessarily $\mathfrak{s}(x) \neq \mathfrak{s}(y)$.

(b) Consider an atom $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ with $x_i = x$, $x_j = y$ and $i \neq j$. There exist a rule[9] $p_{I,J,\sim,\neq}(z_1, \ldots, z_n, X) \Leftarrow \psi'$ in $\mathcal{R}_{dec}$ and a substitution σ such that $(z_1, \ldots, z_n)\sigma = (x_1, \ldots, x_n)$, and $(\mathfrak{s}', \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\neq}(z_1, \ldots, z_n, X)$ with $\mathfrak{s}' = \mathfrak{s} \circ \sigma$. According to Def. 4, there exist a store $\mathfrak{s}''$ and a set interpretation Σ'' such that $(\mathfrak{s}'', \mathfrak{h}, \Sigma'') \models_{\mathcal{R}_{dec}} \psi'$, and $\mathfrak{s}''$ matches with $\mathfrak{s}'$ on $z_1, \ldots, z_n$, and $\Sigma''(X) = \Sigma(X)$. Because the rule is in $\mathcal{R}_{dec}$ and $i \neq j$, $z_i \not\approx_{\psi'} z_j$, and thus, by the induction hypothesis, $\mathfrak{s}''(z_i) \neq \mathfrak{s}''(z_j)$, i.e. $\mathfrak{s}'(z_i) = \mathfrak{s}'(z_j)$, i.e., $\mathfrak{s}(\sigma(z_i)) \neq \mathfrak{s}(\sigma(z_j))$. As a consequence, $\mathfrak{s}(x_i) \neq \mathfrak{s}(x_j)$, and thus, $\mathfrak{s}(x) \neq \mathfrak{s}(y)$.

(c) Consider formulæ ψ_1, ψ_2 such that $\psi_1 \star \psi_2$ appears in ψ, with $x \in \mathit{alloc}(\psi_1)$ and $y \in \mathit{alloc}(\psi_2)$. As $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$, necessarily there exist subheaps $\mathfrak{h}_1, \mathfrak{h}_2$ of $\mathfrak{h}$ such that $\mathrm{allocated}(\mathfrak{h}_1) \cap \mathrm{allocated}(\mathfrak{h}_2) = \emptyset$ and $(\mathfrak{s}, \mathfrak{h}_i, \Sigma) \models_{\mathcal{R}_{dec}} \psi_i$. By Lem. 2, we deduce that $\mathfrak{s}(x) \in \mathrm{allocated}(\mathfrak{h}_1)$ and $\mathfrak{s}(y) \in \mathrm{allocated}(\mathfrak{h}_2)$, so that $\mathfrak{s}(x) \neq \mathfrak{s}(y)$.

(d) Assume that $x \equiv_\psi x'$, $y \equiv_\psi y'$ and $x' \not\equiv_\psi y'$. By the induction hypothesis we get $\mathfrak{s}(x') \neq \mathfrak{s}(y')$ and by Lem. 1 we have $\mathfrak{s}(x) = \mathfrak{s}(x')$ and $\mathfrak{s}(y) = \mathfrak{s}(y')$ so that $\mathfrak{s}(x) \neq \mathfrak{s}(y)$.

Theorem 4. *Let $(\varphi, \mathcal{R})$ be in normal form. For each structure $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ that verifies $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi$, there exists a decoration $(\psi, \mathcal{R}_{dec})$ of $(\varphi, \mathcal{R})$ such that $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$.*

Proof. The proof proceeds by structural induction on the construction of $\models_{\mathcal{R}}$. Let φ be a formula and $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ an $\mathcal{R}$-model of φ.

$\varphi = \mathtt{emp}$ **or** $\varphi = x \approx y$ **or** $\varphi = x \not\approx y$ **or** $\varphi = y_0.f \to (y_1, \ldots, y_d)$ **or** $\varphi = A$ **or** $\varphi = B$**:** In these cases, there is no predicate in the formula φ, hence, it is also a decorated formula, and thus, a decoration of itself. As the semantics of the formula does not depend on the SID, $\models_{\mathcal{R}}$ is equivalent to $\models_{\mathcal{R}_{dec}}$ here.

$\varphi = \varphi_1 \star \varphi_2$**:** According to Def. 4 there exist heaps $\mathfrak{h}_1$, $\mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, $\mathrm{allocated}(\mathfrak{h}_1) \cap \mathrm{allocated}(\mathfrak{h}_2) = \emptyset$ and $(\mathfrak{s}, \mathfrak{h}_i, \Sigma) \models_{\mathcal{R}} \varphi_i$, for all $i \in \{1, 2\}$. By the induction hypothesis, for all $i \in \{1, 2\}$, there exists a decoration ψ_i of φ_i such that $(\mathfrak{s}, \mathfrak{h}_i, \Sigma) \models_{\mathcal{R}_{dec}} \psi_i$. The formula $\psi_1 \star \psi_2$ is a decoration of φ, and thus, by Def. 4, we have $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$.

$\varphi = \varphi_1 \circledast \varphi_2$**:** The proof is similar to that of the previous item.

$\varphi = p(x_1, \ldots, x_n, X)$**:** There exist a rule[10] $p(z_1, \ldots, z_n, X) \Leftarrow \varphi'$ in $\mathcal{R}$ and a substitution σ such that $(z_1, \ldots, z_n)\sigma = (x_1, \ldots, x_n)$, and $(\mathfrak{s}', \mathfrak{h}, \Sigma) \models_{\mathcal{R}} p(z_1, \ldots, z_n, X)$ with $\mathfrak{s}' = \mathfrak{s} \circ \sigma$. According to Def. 4, there exist a store $\mathfrak{s}''$ and a set interpretation Σ'' such that $(\mathfrak{s}'', \mathfrak{h}, \Sigma'') \models_{\mathcal{R}} \varphi'$, and $\mathfrak{s}''$ matches with $\mathfrak{s}'$ on $z_1, \ldots, z_n$, and $\Sigma''(X) = \Sigma(X)$. By the induction hypothesis, there exists a decoration ψ' of φ' such that $(\mathfrak{s}'', \mathfrak{h}, \Sigma'') \models_{\mathcal{R}_{dec}} \psi'$. We define $\sim, \not\approx, I$ and J such that $i \sim j$ iff $z_i \equiv_{\psi'} z_j$, $i \not\approx j$ iff $z_i \not\equiv_{\psi'} z_j$, $I = \{i \in [\![1, n]\!] \mid z_i \in [X]_{\psi'}\}$ and $J = \{j \in [\![1, n]\!] \mid z_j \in \mathit{alloc}(\psi')\}$.
Note that $I \subseteq J$, by Prop. 3. Thus $p_{I,J,\sim,\neq}$ is a decorated predicate obtained from p such that $p_{I,J,\sim,\neq}(z_1, \ldots, z_n, X)$ is a decoration of $p(z_1, \ldots, z_n, X)$. Furthermore, by Lems. 1 and 3, ψ' is necessarily consistent, as ψ' is satisfiable. Therefore, the rule $p_{I,J,\sim,\neq}(z_1, \ldots, z_n, X) \Leftarrow \psi'$ is included in $\mathcal{R}_{dec}$. As a result, by Def. 4, we have $(\mathfrak{s}', \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\neq}(z_1, \ldots, z_n, X)$, and thus, $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$.

[9] Same assumption as previously.

[10] Same assumption as previously.

B.2 Correctness of decorations

Theorem 5. *Let $(\varphi, \mathcal{R})$ be in normal form and let $(\psi, \mathcal{R}_{dec})$ be a decoration of $(\varphi, \mathcal{R})$. For each structure $(\mathfrak{s}, \mathfrak{h}, \Sigma)$, if $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$ then $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi$.*

Proof. The proof proceeds by structural induction on the construction of $\models_{\mathcal{R}_{dec}}$. Let φ be a formula, ψ a decoration of φ, and $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ an $\mathcal{R}_{dec}$-model of ψ.

- $\varphi = \mathtt{emp}$ **or** $\varphi = x \approx y$ **or** $\varphi = x \not\approx y$ **or** $\varphi = y_0.f \to (y_1, \ldots, y_d)$ **or** $\varphi = A$ **or** $\varphi = B$**:** In these cases, the formula φ is the only decoration of itself, and thus, the result follows immediately. As explained in the proof of Thm. 4, $\models_{\mathcal{R}}$ and $\models_{\mathcal{R}_{dec}}$ agree on φ.
- $\varphi = \varphi_1 \star \varphi_2$**:** The formula ψ must be of the form $\psi = \psi_1 \star \psi_2$ where ψ_i is a decoration of φ_i for all $i \in \{1, 2\}$. As $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$, by Def. 4, there exist heaps $\mathfrak{h}_1$, $\mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$, $\mathrm{allocated}(\mathfrak{h}_1) \cap \mathrm{allocated}(\mathfrak{h}_2) = \emptyset$ and $(\mathfrak{s}, \mathfrak{h}_i, \Sigma) \models_{\mathcal{R}_{dec}} \psi_i$, for all $i \in \{1, 2\}$. By the induction hypothesis, $(\mathfrak{s}, \mathfrak{h}_i, \Sigma) \models_{\mathcal{R}} \varphi_i$ for all $i \in \{1, 2\}$, and thus, $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi$.
- $\varphi = \varphi_1 \circledast \varphi_2$**:** The proof is similar to that of the previous item.
- $\varphi = p(x_1, \ldots, x_n, X)$**:** The formula ψ is of the form $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ where $p_{I,J,\sim,\neq}$ is a decoration of p. As $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$, there exist a rule[11] $p_{I,J,\sim,\neq}(z_1, \ldots, z_n, X) \Leftarrow \psi'$ in $\mathcal{R}_{dec}$ and a substitution σ such that $(z_1, \ldots, z_n)\sigma = (x_1, \ldots, x_n)$, and $(\mathfrak{s}', \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\neq}(z_1, \ldots, z_n, X)$ with $\mathfrak{s}' = \mathfrak{s} \circ \sigma$. According to Def. 4, there exist a store $\mathfrak{s}''$ and a set interpretation Σ'' such that $(\mathfrak{s}'', \mathfrak{h}, \Sigma'') \models_{\mathcal{R}_{dec}} \psi'$, and $\mathfrak{s}''$ matches with $\mathfrak{s}'$ on $z_1, \ldots, z_n$, and $\Sigma''(X) = \Sigma(X)$.
  By definition of $\mathcal{R}_{dec}$, there exists a rule $p(z_1, \ldots, z_n, X) \Leftarrow \varphi'$ where ψ' is a decoration of φ'. By the induction hypothesis, $(\mathfrak{s}'', \mathfrak{h}, \Sigma'') \models_{\mathcal{R}} \varphi'$, thus $(\mathfrak{s}', \mathfrak{h}, \Sigma) \models_{\mathcal{R}} p(z_1, \ldots, z_n, X)$ and finally $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi$.

C Parikh Image of Context-Free Languages

If L is a context-free language (CFL) then its Parikh image is a semilinear set over variables counting the number of occurrences of letters in Σ [21]. The semilinear sets are exactly the sets definable by existential Presburger formulæ [8], i.e., formulæ defined by the grammar below where variables en interpreted over natural numbers:

$$t ::= 0 \mid 1 \mid x \mid t_1 + t_2 \qquad \varphi ::= t_1 = t_2 \mid t_1 > t_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \exists x.\, \varphi$$

The above results are refined in [25] (see also [12]):

Theorem 6 (Theorem 4 in [25]). *Given a context-free grammar G, one can compute an existential Presburger formula φ_G for the Parikh image of the language of G in linear time.*

We provide here the formula built in the proof of this result. Let $G = (\mathcal{N}, \mathcal{T}, \mathcal{R}, N_0)$ be a context-free grammar, with non-terminals in $\mathcal{N}$, terminals in $\mathcal{T}$, production rules in $\mathcal{R}$, and the start symbol N_0.

The formula φ_G uses tree vectors of variables:

[11] Same assumption as previously.

- $\vec{x}$ has the size $|\mathcal{N}| + |\mathcal{T}|$ and it is indexed by non-terminals and terminals of G; x_{N_i} and x_{a_i} encode the Parikh image for the non-terminal N_i resp. terminal a_i in a word derived from the start symbol. We also use $\vec{x}_{\mathcal{N}}$ and $\vec{x}_{\mathcal{T}}$ for the vectors of variables associated to non-terminals resp. terminals.
- $\vec{y}$ has the size $|\mathcal{R}|$, it is indexed by the productions in $\mathcal{R}$; y_r encodes the number of applications of the rule r to obtain a derivation.
- $\vec{z}$ has the size $|\mathcal{N}| + |\mathcal{T}|$ and it is indexed by non-terminals and terminals of G; z_A ($A \in \mathcal{N} \cup \mathcal{T}$) reflects the distance of A from N_0 in a spanning tree on the subgraph of the communication-free Petri nets built from G where the edges are productions r for which $y_r > 0$.

Then, $\varphi_G(\vec{x}_{\mathcal{T}}) = \exists \vec{x}_{\mathcal{N}}.\ \exists \vec{y}.\ \exists \vec{z}.\ \psi_G$ where:

$$\psi_G = \bigwedge_{N \in \mathcal{N}} ite(N = N_0, 1, 0) + (\Sigma_{r:N' \to w} \mathcal{P}(w)(N) \cdot y_r) - (\Sigma_{r:N \to w} y_r) = 0 \tag{19}$$

$$\wedge \bigwedge_{a \in \mathcal{T}} x_a = \Sigma_{r:N' \to w} \mathcal{P}(w)(a) \cdot y_r \quad \wedge \bigwedge_{A \in \mathcal{N}} (z_a = 0 \vee z_a > 0) \tag{20}$$

$$\wedge \bigwedge_{A \in \mathcal{N}} \bigvee_{r:N \to w, N \neq N_0, \mathcal{P}(w)(A) > 0} (z_A = z_N + 1 \wedge y_r > 0 \wedge z_N > 0) \vee (z_A = 0 \wedge \bigwedge_{r:A \to w} y_r = 0) \tag{21}$$

$$\wedge z_{N_0} = 1 \tag{22}$$

D Semantic of BAPA and Proof of Prop. 1

We provide below the semantic of arithmetic terms and constraints in OSL from Def. 1.

Definition 22 (Semantic of arithmetic terms in OSL). *Let t be an arithmetic term, T be a set term, $\mathfrak{s}$ be a store, and Σ be a set interpretation. The interpretation of arithmetic term denoted by $[\![t]\!]_{(\mathfrak{s},\Sigma)}$ is an integer in $\mathbb{N}$ and it is inductively defined by the following rules:*

- $[\![K]\!]_{(\mathfrak{s},\Sigma)} = K$;
- $[\![t_1 \oplus t_2]\!]_{(\mathfrak{s},\Sigma)} = [\![t_1]\!]_{(\mathfrak{s},\Sigma)} + [\![t_2]\!]_{(\mathfrak{s},\Sigma)}$;
- $[\![K \odot t]\!]_{(\mathfrak{s},\Sigma)} = K \times [\![t]\!]_{(\mathfrak{s},\Sigma)}$;
- $[\![|T|]\!]_{(\mathfrak{s},\Sigma)} = \text{card}(T)$.

Definition 23 (OSL semantics extension). *Given a formula φ, a SID $\mathcal{R}$ and a structure $(\mathfrak{s}, \mathfrak{h}, \Sigma)$, $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}} \varphi$ iff one of the following conditions holds:*

- $\varphi = (t_1 \approx t_2)$ *and* $[\![t_1]\!]_{(\mathfrak{s},\Sigma)} = [\![t_2]\!]_{(\mathfrak{s},\Sigma)}$ *and* $\mathfrak{h} = \emptyset$;
- *For* $\varphi = (t_1 \not\approx t_2)$, $\varphi = (t_1 \prec t_2)$ *or* $\varphi = (t_1 \not\prec t_2)$ *the definition is similar to the previous one;*
- $\varphi = (K \text{ div } t)$ *and* $[\![t]\!]_{(\mathfrak{s},\Sigma)} = 0 \pmod K$ *and* $\mathfrak{h} = \emptyset$;
- $\varphi = (K \text{ ndiv } t)$ *and* $[\![t]\!]_{(\mathfrak{s},\Sigma)} \neq 0 \pmod K$ *and* $\mathfrak{h} = \emptyset$.

We recall the semantic of BAPA from [14]

Definition 24 (Semantic of BAPA terms). *Let T^{BP} be a set term and $(\mathbb{S}, \mathbb{I})$ be a BAPA structure. The interpretation of a BAPA set term $[\![T^{BP}]\!]_{(\mathbb{S},\mathbb{I})}$ is a set of locations in $\mathcal{L}$ and the interpretation of a BAPA arithmetic term $[\![t^{BP}]\!]_{(\mathbb{S},\mathbb{I})}$ is a element of $\mathbb{Z}$ inductively defined by the following rules:*

- $[\![X]\!]_{(\mathbb{S},\mathbb{I})} = \mathbb{S}(X)$; $[\![\emptyset]\!]_{(\mathbb{S},\mathbb{I})} = \emptyset$;
- $[\![T_1^{BP} \sqcap_{BP} T_2^{BP}]\!]_{(\mathbb{S},\mathbb{I})} = [\![T_1^{BP}]\!]_{(\mathbb{S},\mathbb{I})} \cap [\![T_2^{BP}]\!]_{(\mathbb{S},\mathbb{I})}$;
- $[\![T_1^{BP} \sqcup_{BP} T_2^{BP}]\!]_{(\mathbb{S},\mathbb{I})} = [\![T_1^{BP}]\!]_{(\mathbb{S},\mathbb{I})} \cup [\![T_2^{BP}]\!]_{(\mathbb{S},\mathbb{I})}$;
- $[\![k]\!]_{(\mathbb{S},\mathbb{I})} = \mathbb{I}(k)$; $[\![K]\!]_{(\mathbb{S},\mathbb{I})} = K$;
- $[\![t_1^{BP} \oplus_{BP} t_2^{BP}]\!]_{(\mathbb{S},\mathbb{I})} = [\![t_1^{BP}]\!]_{(\mathbb{S},\mathbb{I})} + [\![t_2^{BP}]\!]_{(\mathbb{S},\mathbb{I})}$;
- $[\![K \odot_{BP} t^{BP}]\!]_{(\mathbb{S},\mathbb{I})} = K \times [\![t^{BP}]\!]_{(\mathbb{S},\mathbb{I})}$;
- $[\![|T^{BP}|]\!]_{(\mathbb{S},\mathbb{I})} = \mathrm{card}([\![T^{BP}]\!]_{(\mathbb{S},\mathbb{I})})$.

Definition 25 (BAPA semantics). *Given a formula φ^{BP} and a structure $(\mathbb{S},\mathbb{I})$ the satisfaction relation $\models_{BP}$ is inductively defined as the least relation such that $(\mathbb{S},\mathbb{I}) \models_{BP} \varphi^{BP}$ iff one of the following conditions holds:*

- $\varphi^{BP} = T_1^{BP} \approx_{BP} T_2^{BP}$ *and* $[\![T_1^{BP}]\!]_{(\mathbb{S},\mathbb{I})} = [\![T_2^{BP}]\!]_{(\mathbb{S},\mathbb{I})}$;
- $\varphi^{BP} = T_1^{BP} \sqsubseteq_{BP} T_2^{BP}$ *and* $[\![T_1^{BP}]\!]_{(\mathbb{S},\mathbb{I})} \subseteq [\![T_2^{BP}]\!]_{(\mathbb{S},\mathbb{I})}$;
- $\varphi^{BP} = t_1^{BP} \approx_{BP} t_2^{BP}$ *and* $[\![t_1^{BP}]\!]_{(\mathbb{S},\mathbb{I})} = [\![t_2^{BP}]\!]_{(\mathbb{S},\mathbb{I})}$;
- $\varphi^{BP} = t_1^{BP} \prec_{BP} t_2^{BP}$ *and* $[\![t_1^{BP}]\!]_{(\mathbb{S},\mathbb{I})} < [\![t_2^{BP}]\!]_{(\mathbb{S},\mathbb{I})}$;
- $\varphi^{BP} = K \operatorname{div} t^{BP}$ *and* $[\![t^{BP}]\!]_{(\mathbb{S},\mathbb{I})} = 0 \pmod K$;
- $\varphi^{BP} = \varphi_1^{BP} \wedge \varphi_2^{BP}$ *and* $(\mathbb{S},\mathbb{I}) \models_{BP} \varphi_i^{BP}$, *for all* $i \in \{1,2\}$;
- $\varphi^{BP} = \varphi_1^{BP} \vee \varphi_2^{BP}$ *and* $(\mathbb{S},\mathbb{I}) \models_{BP} \varphi_i^{BP}$, *for some* $i \in \{1,2\}$;
- $\varphi^{BP} = \neg\varphi_1^{BP}$ *and* $(\mathbb{S},\mathbb{I}) \not\models_{BP} \varphi^{BP}$;
- $\varphi^{BP} = \exists X.\varphi_1^{BP}$ *and* $(\mathbb{S}',\mathbb{I}) \models_{BP} \varphi_1^{BP}$, *for some* $\mathbb{S}'$ *matching* $\mathbb{S}$ *on all set variables distinct from* X;
- $\varphi^{BP} = \forall X.\varphi_1^{BP}$ *and* $(\mathbb{S}',\mathbb{I}) \models_{BP} \varphi_1^{BP}$, *for all* $\mathbb{S}'$ *matching* $\mathbb{S}$ *on all set variables distinct from* X;
- $\varphi^{BP} = \exists k.\varphi_1^{BP}$ *and* $(\mathbb{S},\mathbb{I}') \models_{BP} \varphi_1^{BP}$, *for some* $\mathbb{I}'$ *matching* $\mathbb{I}$ *on all integer variables distinct from* k;
- $\varphi^{BP} = \forall x.\varphi_1^{BP}$ *and* $(\mathbb{S},\mathbb{I}') \models_{BP} \varphi_1^{BP}$, *for all* $\mathbb{I}'$ *matching* $\mathbb{I}$ *on all integer variables distinct from* k.

Proposition 1. *Let $T \in \mathfrak{T} \cup \mathfrak{t}$ be a set term or an arithmetic term. Let $(\mathfrak{s},\mathfrak{h},\Sigma)$ be an OSL structure. Let $\mathbb{S}$ be defined by $\mathbb{S}(V_x) = \{\mathfrak{s}(x)\}$ for all location variables x, and $\mathbb{S}(X) = \Sigma(X)$ for all set variables X. Then, for every interpretation $\mathbb{I}$, $[\![T^{SL}]\!]_{(\mathfrak{s},\Sigma)} = [\![\mathcal{T}(T)]\!]_{(\mathbb{S},\mathbb{I})}$.*

Proof. The proof is by induction on T:

$T = \{x\}$: $[\![\{x\}]\!]_{(\mathfrak{s},\Sigma)} = \{\mathfrak{s}(x)\} = \mathbb{S}(V_x) = [\![V_x]\!]_{(\mathbb{S},\mathbb{I})} = [\![\mathcal{T}(\{x\})]\!]_{(\mathbb{S},\mathbb{I})}$.
$T = \emptyset$: $[\![\emptyset]\!]_{(\mathfrak{s},\Sigma)} = \emptyset = [\![\mathbf{0}]\!]_{(\mathbb{S},\mathbb{I})} = [\![\mathcal{T}(\emptyset)]\!]_{(\mathbb{S},\mathbb{I})}$.
$T = X$: $[\![X]\!]_{(\mathfrak{s},\Sigma)} = \Sigma(X) = \mathbb{S}(X) = [\![X]\!]_{(\mathbb{S},\mathbb{I})} = [\![\mathcal{T}(X)]\!]_{(\mathbb{S},\mathbb{I})}$.
$T = T_1 \sqcup T_2$: By the induction hypothesis, $[\![T_i]\!]_{(\mathfrak{s},\Sigma)} = [\![T_i]\!]_{(\mathbb{S},\mathbb{I})}$ for all $i \in \{1,2\}$. Then, $[\![T]\!]_{(\mathfrak{s},\Sigma)} = [\![T_1]\!]_{(\mathfrak{s},\Sigma)} \cup [\![T_2]\!]_{(\mathfrak{s},\Sigma)} = [\![T_1]\!]_{(\mathbb{S},\mathbb{I})} \cup [\![T_2]\!]_{(\mathbb{S},\mathbb{I})} = [\![T_1 \sqcup T_2]\!]_{(\mathbb{S},\mathbb{I})} = \mathbb{S}(X) = [\![\mathcal{T}(T)]\!]_{(\mathbb{S},\mathbb{I})}$.
$T = T_1 \sqcap T_2$: As previously, $[\![T]\!]_{(\mathfrak{s},\Sigma)} = [\![\mathcal{T}(T)]\!]_{(\mathbb{S},\mathbb{I})}$.

Similarly, by induction on t^{SL}:

$t = K$: $[\![K]\!]_{(\mathfrak{s},\Sigma)} = K = [\![K]\!]_{(\mathbb{S},\mathbb{I})}$.
$t = t_1 \oplus t_2$: By the induction hypothesis, $[\![t_i]\!]_{(\mathfrak{s},\Sigma)} = [\![t_i]\!]_{(\mathbb{S},\mathbb{I})}$ for all $i \in \{1,2\}$. $[\![t]\!]_{(\mathfrak{s},\Sigma)} = [\![t_1]\!]_{(\mathfrak{s},\Sigma)} + [\![t_2]\!]_{(\mathfrak{s},\Sigma)} = [\![t_1]\!]_{(\mathbb{S},\mathbb{I})} + [\![t_2]\!]_{(\mathbb{S},\mathbb{I})} = [\![t]\!]_{(\mathbb{S},\mathbb{I})}$.
$t = K \odot t_1$: By the induction hypothesis, $[\![t_1]\!]_{(\mathfrak{s},\Sigma)} = [\![t_1]\!]_{(\mathbb{S},\mathbb{I})}$. $[\![t]\!]_{(\mathfrak{s},\Sigma)} = K \times [\![t_1]\!]_{(\mathfrak{s},\Sigma)} = K \times [\![t_1]\!]_{(\mathbb{S},\mathbb{I})} = [\![t]\!]_{(\mathbb{S},\mathbb{I})}$.
$t = |T|$: We know that $[\![T]\!]_{(\mathfrak{s},\Sigma)} = [\![T]\!]_{(\mathbb{S},\mathbb{I})}$. $[\![|T|]\!]_{(\mathfrak{s},\Sigma)} = \mathrm{card}([\![T]\!]_{(\mathfrak{s},\Sigma)}) = \mathrm{card}([\![T]\!]_{(\mathbb{S},\mathbb{I})}) = [\![|T|]\!]_{(\mathbb{S},\mathbb{I})}$.

E Proof of Theorem 3

We recall Thm. 3:

Theorem 3. *Let $(\varphi, \mathcal{R})$ an OSL-formula and a SID in normal form. For all decorations $(\psi, \mathcal{R}_{dec})$ of $(\varphi, \mathcal{R})$, ψ has an $\mathcal{R}_{dec}$-model iff $C(\psi)$ is satisfiable (in BAPA).*

E.1 Proof of "If ψ is satisfiable then $C(\psi)$ is satisfiable"

If ψ is satisfiable, there exists an $\mathcal{R}_{dec}$-model $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ of ψ, we define $(\mathbb{S}, \mathbb{I})$ as follows:

- let $\mathbb{I}(i_X) = \mathrm{card}(\Sigma(X))$ and
- let $\mathbb{S}(X) = \Sigma(X)$ for all set variables X of ψ and let $\mathbb{S}(V_x) = \{\mathfrak{s}(x)\}$ for all location variables x of ψ.

In the remainder of this sub-section, we show that $(\mathbb{S}, \mathbb{I})$ is a model of $C(\psi)$.

For all free location variables x of ψ, by construction of $\mathbb{S}$ and $\mathbb{I}$ and by definition of $\mathfrak{s}$, $\mathrm{card}(\mathbb{S}(V_x)) = 1$ and thus $(\mathbb{S}, \mathbb{I}) \models_{BP} |V_x| =_{BP} 1$. Therefore, we only need to show that $(\mathbb{S}, \mathbb{I}) \models_{BP} \mathcal{T}(\psi)$. The proof is done by induction on the form of formula ψ:

$\psi = \mathtt{emp}$ **or** $\psi = x.f \to (y_1, \ldots, y_d)$**:** $\mathcal{T}(\psi) = \mathrm{true}$ and the result is immediate.

$\psi = x \approx y$**:** $\mathcal{T}(\psi) = (V_x =_{BP} V_y)$ and by Def. 4, $\mathfrak{s}(x) = \mathfrak{s}(y)$, therefore $\mathbb{S}(V_x) = \mathbb{S}(V_y)$ and the result follows.

$\psi = x \not\approx y$**:** $\mathcal{T}(\psi) = (V_x \cap V_y =_{BP} \emptyset)$ and by Def. 4, $\mathfrak{s}(x) \neq \mathfrak{s}(y)$. Therefore $\{\mathfrak{s}(x)\} \cap \{\mathfrak{s}(y)\} = \emptyset$, i.e., $\mathbb{S}(V_x) \cap \mathbb{S}(V_y) = \emptyset$ and the result follows.

$\psi = (T_1 \diamond T_2)$ **for** $\diamond \in \{\approx, \sqsubseteq, \not\approx, \not\sqsubseteq\}$**:** By Prop. 1, for all $i \in \{1, 2\}$, we have $[\![T_i]\!]_{(\mathfrak{s},\Sigma)} = [\![\mathcal{T}(T_i)]\!]_{(\mathbb{S},\mathbb{I})}$ and thus $(\mathbb{S}, \mathbb{I}) \models_{BP} \mathcal{T}(\psi)$.

$\psi = (t_1 \diamond t_2)$ **for** $\diamond \in \{\approx, \prec, \not\approx, \not\prec\}$ **or** $\psi = (K \,\mathrm{div}\, t_1)$ **or** $\psi = (K \,\mathrm{ndiv}\, t_1)$**:** By Prop. 1, for all $i \in \{1, 2\}$, we have $[\![t_i]\!]_{(\mathfrak{s},\Sigma)} = [\![\mathcal{T}(t_i)]\!]_{(\mathbb{S},\mathbb{I})}$ and thus $(\mathbb{S}, \mathbb{I}) \models_{BP} \mathcal{T}(\psi)$.

$\psi = p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$**:** By hypothesis, $(\mathbb{S}, \mathbb{I}) \models_{BP} |X| =_{BP} i_X$ and by Prop. 5, $(\mathbb{S}, \mathbb{I}) \models_{BP} \xi_{p_{I,J,\sim,\neq}(x_1,\ldots,x_n,X)}(i_X)$.

We show below that, for any atom $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$, the set I corresponds exactly to the indices i such that x_i occurs in X. However, this property holds only for the predicate atoms appearing in ψ and not for those introduced during recursive calls. Therefore, we refine the property by focusing on the main parameters of the predicate atom (as defined in Sect. 2.4).

> **Lemma 4.** *For all decorated predicate atoms $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$, for all $\mathcal{R}_{dec}$-models $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ of $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ such that $\mathfrak{s}$ is injective and for all $i \in [\![1, n]\!]$:*
> - *$i \in I \implies \mathfrak{s}(x_i) \in \Sigma(X)$;*
> - *if x is a main parameter of $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ then $\mathfrak{s}(x) \in \Sigma(X) \implies x \in \{x_i \mid i \in I\}$.*
>
> *Proof.* We reason by induction on the size of $\mathfrak{h}$. Let $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ be a decorated predicate atom and $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ one of its $\mathcal{R}_{dec}$-models, where $\mathfrak{s}$ is injective. There exist a rule[12] $p_{I,J,\sim,\neq}(x'_1, \ldots, x'_n, X) \Leftarrow \psi$ in $\mathcal{R}_{dec}$ and a substitution σ such that $(x'_1, \ldots, x'_n)\sigma = (x_1, \ldots, x_n)$, and $(\mathfrak{s}', \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\neq}(x'_1, \ldots, x'_n, X)$ with $\mathfrak{s}' = \mathfrak{s} \circ \sigma$. According to Def. 4, there exist a store $\mathfrak{s}''$ and a set interpreta-

tion Σ'' such that $(\mathfrak{s}'', \mathfrak{h}, \Sigma'') \models_{\mathcal{R}_{dec}} \psi$, and $\mathfrak{s}''$ matches with $\mathfrak{s}'$ on $x'_1, \ldots, x'_n$, and $\Sigma''(X) = \Sigma(X)$.
Observe that $\mathfrak{s}''$ must be injective, as (by the assumption in Sect. 2.4) ψ contains a disequation $z \not\approx z'$ for all auxiliary location variables z and for all location variables $z' \in fv(\psi) \cap \mathcal{V}$ other than z. We recall that, by Eq. 6, ψ must be of the form $x_r.f \to (\vec{z}) \star \bigstar_{u=1}^{m} q^u_{I_u, J_u, \sim_u, \not\approx_u}(y^u_1, \ldots, y^u_{n_u}, Y_u) \star \varphi \star \left(X \approx E \sqcup \bigsqcup_{u=1}^{m} Y_u\right)$. We suppose that $E = \{x'_j\}$ (the case where $E = \emptyset$ is similar). Then $(\mathfrak{s}'', \mathfrak{h}, \Sigma'') \models_{\mathcal{R}_{dec}} (X \approx E \sqcup \bigsqcup_{u=1}^{m} Y_u)$ and thus $[\![X]\!]_{\mathfrak{s}'', \Sigma''} = [\![E \cup \bigcup_{u=1}^{m} Y_u]\!]_{\mathfrak{s}'', \Sigma''}$, i.e., $\Sigma(X) = \Sigma''(X) = \{\mathfrak{s}''(x'_j)\} \cup \bigcup_{u=1}^{m} \Sigma''(Y_u) = \{\mathfrak{s}(x_j)\} \cup \bigcup_{u=1}^{m} \Sigma''(Y_u)$.

If $i \in I$: By Def. 15, $x'_i \in [X]_\psi$. Because, by Lem. 1, for all x, y, $x \equiv_\psi y \implies \mathfrak{s}''(x) = \mathfrak{s}''(y)$ we can suppose that we are in the case (a) of Def. 13 (case (b) is impossible because the set variables Y_i are distinct from X). If $x'_i \in E$, then we immediately get $\mathfrak{s}''(x'_i) = \mathfrak{s}(x_i) \in \Sigma(X)$. If $x'_i \in [Y_u]_\psi$ for some $u \in [\![1, m]\!]$, then necessarily, by definition of $[Y_u]_\psi$, there exists a parameter $i' \in I_u$ such that $y_{i'} \equiv_\psi x'_i$ (so that $\mathfrak{s}''(y_{i'}) = \mathfrak{s}''(x'_i) = \mathfrak{s}(x_i)$). As there exists $\mathfrak{h}_u$ such as $(\mathfrak{s}'', \mathfrak{h}_u, \Sigma'') \models q_u(y^u_1, \ldots, y^u_{n_u}, Y_u)$, the induction hypothesis ensures that $\mathfrak{s}''(y_{i'}) \in \Sigma''(Y_u)$ and thus $\mathfrak{s}(x_i) \in \Sigma''(X) = \Sigma(X)$.

If $\mathfrak{s}(x) \in \Sigma(X)$ and x is a main parameter: If $\mathfrak{s}(x) \in \{\mathfrak{s}(x_j)\}$ then $\mathfrak{s}(x) = \mathfrak{s}(x_j)$ and because the store is injective on $x_1 \ldots, x_n$, $x = x_j$, with $j \in I$. Now, assume that $\mathfrak{s}(x) \in \Sigma''(Y_u)$ for some $u \in [\![1, m]\!]$. Note that x'_i (where $\sigma(x'_i) = x$) must occur in $y^u_1, \ldots, y^u_{n_u}$ (since x is a main parameter). As there exists $\mathfrak{h}_u$ such as $(\mathfrak{s}'', \mathfrak{h}_u, \Sigma'') \models q_u(y^u_1, \ldots, y^u_{n_u}, Y_u)$, the induction hypothesis ensures (as $\mathfrak{s}''(x'_i) = \mathfrak{s}(x)$) that there exists $j \in I_u$ such that $x'_i = y^u_j$. Therefore, the case (b) of Def. 13 ensure that $x \in \{x_j \mid j \in I\}$.

This ends the proof of Lem. 4. □

We go back now to the proof of Thm. 3.
According to Lem. 4, for all $i \in [\![1, n]\!]$, if $i \in I$, then $\mathfrak{s}(x_i) \in \Sigma(X)$, and if $\mathfrak{s}(x_i) \in \Sigma(X)$, then $x_i \in \{x_j \mid j \in I\}$. Therefore, for all $i \in I$, $[\![V_{x_i}]\!]_{(\mathbb{S},\mathbb{I})} = \{\mathfrak{s}(x_i)\} \subseteq \Sigma(X) = [\![X]\!]_{(\mathbb{S},\mathbb{I})}$, and, for all $i \in [\![1, n]\!]$ such that $x_i \notin \{x_j \mid j \in I\}$, $\{\mathfrak{s}(x_i)\} \cap \Sigma(X) = \emptyset$. As a consequence, $(\mathbb{S}, \mathbb{I}) \models_{BP} ((\bigcup_{i \in I} V_{x_i}) \subseteq X) \wedge \left(\left(\bigcup_{x_i \notin \{x_j \mid j \in I\}} V_{x_i}\right) \cap X =_{BP} \emptyset\right)$.

$\psi = \psi_1 \star \psi_2$: By Def. 4, there exist $\mathfrak{h}_1, \mathfrak{h}_2$ such that $(\mathfrak{s}, \mathfrak{h}_i, \Sigma) \models_{\mathcal{R}_{dec}} \psi_i$ for all $i \in \{1, 2\}$, $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$ and $\text{allocated}(\mathfrak{h}_1) \cap \text{allocated}(\mathfrak{h}_2) = \emptyset$. By induction hypothesis, and because $(\mathbb{S}, \mathbb{I})$ does not depend of $\mathfrak{h}$, $(\mathbb{S}, \mathbb{I}) \models_{BP} \mathcal{T}(\psi_i)$.
We need to show how that all the terms in $\mathcal{T}^f(\psi)$ are indeed allocated.

Lemma 5. *Let $(\psi, \mathcal{R}_{dec})$ a decoration. For all structures $(\mathfrak{s}, \mathfrak{h}, \Sigma)$, if $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$, then $[\![\mathcal{T}^f(\psi)]\!]_{(\mathbb{S},\mathbb{I})} \subseteq \text{dom}_f(\mathfrak{h})$ for all $f \in \mathcal{F}$ with $\mathbb{S}$ defined by $\mathbb{S}(X) = \Sigma(X)$ for all set variables X of ψ and $\mathbb{S}(V_x) = \{\mathfrak{s}(x)\}$ for all location variables x of ψ.*

Proof. By induction on ψ with $f, g \in \mathcal{F}$:

$\psi = \texttt{emp}$ or $\psi = x \approx y$ or $\psi = x \not\approx y$ or $\psi = A$ or $\psi = B$: Here, $[\![\mathcal{T}^f(\psi)]\!]_{(\mathbb{S},\mathbb{I})} = \emptyset$, and, by Def. 4, $\mathfrak{h} = \emptyset$.

[12] We may assume by renaming that the set variable occurring in the left-hand side of the rule is X.

$\psi = x.g \to (y_1, \ldots, y_d)$ **with** $f \neq g$**:** By Def. 4, the heap is $\mathfrak{h} = [(\mathfrak{s}(x), g) \mapsto (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_d))]$, and thus, $\mathrm{dom}_f(\mathfrak{h}) = \emptyset$. By Def. 20, $\mathcal{T}^f(\psi) = \emptyset$, and thus, $[\![\mathcal{T}^f(\psi)]\!]_{(\mathbb{S},\mathbb{I})} = \emptyset$.

$\psi = x.f \to (y_1, \ldots, y_d)$**:** By Def. 4, $\mathfrak{h} = [(\mathfrak{s}(x), f) \mapsto (\mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_d))]$, and as a result, $\mathrm{dom}_f(\mathfrak{h}) = \{\mathfrak{s}(x)\}$. By Def. 20, $\mathcal{T}^f(\psi) = V_x$, and thus, $[\![\mathcal{T}^f(\psi)]\!]_{(\mathbb{S},\mathbb{I})} = \{\mathfrak{s}(x)\}$.

$\psi = \psi_1 \star \psi_2$ **or** $\psi = \psi_1 \circledast \psi_2$**:** By Def. 20, $\mathcal{T}^f(\psi) = \mathcal{T}^f(\psi_1) \cup \mathcal{T}^f(\psi_2)$, and as a result, $[\![\mathcal{T}^f(\psi)]\!]_{(\mathbb{S},\mathbb{I})} = [\![\mathcal{T}^f(\psi_1)]\!]_{(\mathbb{S},\mathbb{I})} \cup [\![\mathcal{T}^f(\psi_2)]\!]_{(\mathbb{S},\mathbb{I})}$. By Def. 4, there exist $\mathfrak{h}_1, \mathfrak{h}_2$ such that $(\mathfrak{s}, \mathfrak{h}_i) \models_{\mathcal{R}_{dec}} \psi_i$ for all $i \in \{1, 2\}$, $\mathfrak{h} = \mathfrak{h}_1 \cup \mathfrak{h}_2$ and $\mathrm{allocated}(\mathfrak{h}_1) \cap \mathrm{allocated}(\mathfrak{h}_2) = \emptyset$ (resp. $\mathrm{dom}(\mathfrak{h}_1) \cap \mathrm{dom}(\mathfrak{h}_2) = \emptyset$). By the induction hypothesis $[\![\mathcal{T}^f(\psi_i)]\!]_{(\mathbb{S},\mathbb{I})} \subseteq \mathrm{dom}_f(\mathfrak{h}_i)$. As a result, $\mathrm{dom}_f(\mathfrak{h}) = \mathrm{dom}_f(\mathfrak{h}_1) \cup \mathrm{dom}_f(\mathfrak{h}_2) \supseteq [\![\mathcal{T}^f(\psi_1)]\!]_{(\mathbb{S},\mathbb{I})} \cup [\![\mathcal{T}^f(\psi_2)]\!]_{(\mathbb{S},\mathbb{I})} = [\![\mathcal{T}^f(\psi)]\!]_{(\mathbb{S},\mathbb{I})}$.

$\psi = p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ **with** p **a** $\{g\}$**-predicate and** $f \neq g$**:** Here, by Def. 20, $\mathcal{T}^f(\psi) = \emptyset$. And because p is a $\{g\}$-predicate with $f \neq g$, $\mathrm{dom}_f(\mathfrak{h}) = \emptyset$.

$\psi = p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ **with** p **a** $\{f\}$**-predicate:** By Def. 20, we have $\mathcal{T}^f(\psi) = X \cup \bigcup_{j \in J} V_{x_j}$, and thus, $[\![\mathcal{T}^f(\psi)]\!]_{(\mathbb{S},\mathbb{I})} = \Sigma(X) \cup \{\mathfrak{s}(x_j) \mid j \in J\}$. Because p is a $\{f\}$-predicate, $\mathrm{allocated}(\mathfrak{h}) = \mathrm{dom}_f(\mathfrak{h})$, and by Lem. 2, $alloc(\psi) \subseteq \mathrm{allocated}(\mathfrak{h})$, so that $\{\mathfrak{s}(x_j) \mid j \in J\} \subseteq \mathrm{dom}_f(\mathfrak{h})$. Moreover, by Prop. 2, $\Sigma(X) \subseteq \mathrm{dom}_f(\mathfrak{h})$. As a result, $[\![\mathcal{T}^f(\psi)]\!]_{(\mathbb{S},\mathbb{I})} \subseteq \mathrm{dom}_f(\mathfrak{h})$.

This ends the proof of Lem. 5. □

We go back now to the proof of Thm. 3.

Now, by Lem. 5, $[\![\left(\bigcup_{f \in \mathcal{F}} \mathcal{T}^f(\psi_i)\right)]\!]_{(\mathbb{S},\mathbb{I})} = \left(\bigcup_{f \in \mathcal{F}} [\![\mathcal{T}^f(\psi_i)]\!]_{(\mathbb{S},\mathbb{I})}\right) \subseteq \left(\bigcup_{f \in \mathcal{F}} \mathrm{dom}_f(\mathfrak{h})\right) = \mathrm{allocated}(\mathfrak{h}_i)$. As $\mathrm{allocated}(\mathfrak{h}_1) \cap \mathrm{allocated}(\mathfrak{h}_2) = \emptyset$, we obtain $[\![\left(\bigcup_{f \in \mathcal{F}} \mathcal{T}^f(\psi_1)\right)]\!]_{(\mathbb{S},\mathbb{I})} \cap [\![\left(\bigcup_{f \in \mathcal{F}} \mathcal{T}^f(\psi_2)\right)]\!]_{(\mathbb{S},\mathbb{I})} = \emptyset$ and $(\mathbb{S}, \mathbb{I}) \models_{BP} \left(\bigcup_{f \in \mathcal{F}} \mathcal{T}^f(\psi_1)\right) \cap \left(\bigcup_{f \in \mathcal{F}} \mathcal{T}^f(\psi_2)\right) = \emptyset$.

$\psi = \psi_1 \circledast \psi_2$**:** This case is done similarly to the previous one but the intersection is done field by field because the heaps are disjoint field by field.

E.2 Proof of "If $C(\psi)$ is satisfiable then ψ is satisfiable"

If $C(\psi)$ is satisfiable, there exists a model $(\mathbb{S}, \mathbb{I})$ of $C(\psi)$.

By definition of $C(\psi)$ (see Eq. 17), $\mathbb{S}(V_x)$ must be a singleton. Let $\mathfrak{s}$ and Σ be the store and set interpretation defined by $\{\mathfrak{s}(x)\} = \mathbb{S}(V_x)$ (for all location variables x in ψ) and $\Sigma(X) = \mathbb{S}(X)$ (for all set variables X in ψ).

By the assumption in Sect. 2.4, ψ contains an equation $x \not\approx y$ for all distinct x, y, $C(\psi)$ contains the sub-formula $V_x \sqcap_{BP} V_y = \emptyset$, and therefore $\mathfrak{s}$ is injective. Furthermore, ψ cannot contain any atom of the form $x \approx y$ with $x \neq y$, as otherwise ψ would not be consistent. Similarly, atoms of the form $x \not\approx x$ cannot occur in ψ. Moreover, all the set constraints in $\mathfrak{A}$ and $\mathfrak{B}$ occurring in ψ must be true in $(\mathfrak{s}, \emptyset, \Sigma)$, as they are validated by $(\mathbb{S}, \mathbb{I})$, by definition of $C(\psi)$. The transformation of negative version of operators into negative formulæ does not introduce any issue. This entails that $(\mathfrak{s}, \emptyset, \Sigma)$ is a model of the non-spatial part of ψ.

Now we handle the spatial part of ψ and show how to construct the corresponding heap. For this, we need a lemma stating that if $\mathfrak{s}$ and Σ satisfy all the conditions asserted by the decoration of an atom $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ (concerning the aliasing

and non-aliasing of variables and the fact that I denotes the set of parameters occurring in X) as well as the cardinality constraints extracted from the rules of $p_{I,J,\sim,\neq}$ (as defined in Sect. 3.2) then one can construct a heap $\mathfrak{h}$ such that $(\mathfrak{s}, \mathfrak{h}, \Sigma)$ is a model of $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$, and moreover, J is exactly the set of allocated parameters:

Lemma 6. *Let $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ be a decorated predicate atom. Let $\mathfrak{s}$ be an injective store and let Σ be a set interpretation. Assume that the following conditions are satisfied:*

1. *for all $i, j \in [\![1, n]\!]$, if $i \sim j$, then $\mathfrak{s}(x_i) = \mathfrak{s}(x_j)$ (hence $x_i = x_j$, as $\mathfrak{s}$ is injective);*
2. *for all $i, j \in [\![1, n]\!]$, if $i \not\sim j$, then $\mathfrak{s}(x_i) \neq \mathfrak{s}(x_j)$;*
3. *For all $i \in [\![1, n]\!]$, $\mathfrak{s}(x_i) \in \Sigma(X) \iff \exists j \in I,\ x_i = x_j$;*
4. *$\mathrm{card}(\Sigma(X)) \in Sp(p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X))$;*
5. *$\mathcal{L}$ is an infinite set of locations containing $\mathfrak{s}(\{x_i \mid i \in [\![1, n]\!]\})$ and $\Sigma(X)$.*

Then there exists a heap $\mathfrak{h}$ such that $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ and $\mathfrak{h} \in \mathcal{L} \times \mathcal{F} \times \mathcal{L}^*$. Moreover, for all $i \in [\![1, n]\!]$, $\mathfrak{s}(x_i) \in$ allocated$(\mathfrak{h}) \iff \exists j \in J,\ x_i = x_j$.

Proof. By Prop. 5, as $\mathrm{card}(\Sigma(X)) \in Sp(p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X))$, there exists a derivation yielding $\mathrm{card}(\Sigma(X))$ using the grammar of cardinalities (see Def. 17). The proof is by induction on the length of this derivation. Consider the derivation rule yielding the word $1^{\mathrm{card}(\Sigma(X))}$, as defined in Eq. 15 (see Def. 15). Let $p_{I,J,\sim,\neq}(x'_1, \ldots, x'_n, X) \Leftarrow \psi$ be its associated rule in $\mathcal{R}_{dec}$ (assuming by renaming that the set variable in the left-hand side is X). The rule satisfies the conditions Def. 15, and the formula ψ is of the form of Eq. 6:

$$x'_r \to (z'_1, \ldots, z'_d) \star \bigstar_{u=1}^{m} \psi_u \star \varphi \star \left(X \approx E \sqcup \bigsqcup_{u=1}^{m} Y_u \right),$$

where φ is a $\star$-conjunction of equational atoms, $x'_r \to (z'_1, \ldots, z'_d)$ is the (unique) points-to atom of ψ, E is either $\emptyset$ or $\{x'_r\}$ and $\{\psi_u \mid 1 \leq u \leq m\}$ is the set of predicate atoms in ψ. Let $\sigma = \{x'_i \mapsto x_i \mid i \in [\![1, n]\!]\}$ and let $\psi_u\sigma = q^u_{I_u,J_u,\sim_u,\neq_u}(y^u_1, \ldots, y^u_{n_u}, Y_u)$. The conditions in Def. 17 (see Eq. 15) ensure that $\psi\sigma$ is consistent (see Def. 14) and that there exist $\omega_u \in \mathbb{N}$ (for all $u \in [\![1, m]\!]$) and $\omega \in \{0, 1\}$ such that $\mathrm{card}(\Sigma(X)) = \omega + \sum_{u=1}^{m} \omega_u$ with $\omega_u \in Sp(\psi_u\sigma)$ and $\omega = \mathrm{card}(E)$. We suppose, w.l.o.g., that $\omega = 1$, i.e., $E = \{x'_r\}$ (the case where $\omega = 0$ is similar). We need to define a store $\mathfrak{s}'$ matching $\mathfrak{s}$ on $x_1, \ldots, x_n$, a set interpretation Σ' with $\Sigma'(X) = \Sigma(X)$ and a heap $\mathfrak{h}$ such that $(\mathfrak{s}', \mathfrak{h}, \Sigma') \models_{\mathcal{R}_{dec}} \psi\sigma$.

Observe that, by the condition on the auxiliary location variables in Sect. 2.4, ψ must contain a disequation of the form $z \not\approx z'$, for all auxiliary location variables z and for all location variables z' other than z. This entails that if $y^u_j \neq y^u_{j'}$ and y^u_j is auxiliary, then, $j \not\sim_u j'$ (as otherwise we would have $y^u_j \equiv_{\psi\sigma} y^u_{j'}$ and $\psi\sigma$ would be inconsistent). Moreover, by Defs. 11 and 12, if $u, u' \in [\![1, m]\!]$, $u \neq u'$, $j \in J_u$ and $j' \in J_{u'}$, then $y^u_j \not\equiv_{\psi\sigma} y^{u'}_{j'}$ so that $y^u_j \neq y^{u'}_{j'}$ (as $\psi\sigma$ is consistent). Consequently,

as $\mathfrak{s}$ is injective, if moreover, $y^u_j, y^{u'}_{j'} \in \{x_1, \ldots, x_n\}$, then $\mathfrak{s}(y^u_j) \neq \mathfrak{s}(y^{u'}_{j'})$. By a similar argument, we can prove that $\mathfrak{s}(y^u_j) \neq \mathfrak{s}(x_r)$, if $u \in [\![1, m]\!]$, $j \in J_u$ and $y^u_j \in \{x_1, \ldots, x_n\}$.

We must ensure that $\mathrm{card}(\Sigma'(Y_u)) = \omega_u$ for all $u \in [\![1, m]\!]$. By Prop. 3, only locations associated to allocated variables can be added to set variables. As a result, Σ' must be defined in such a way that the sets $\Sigma'(Y_u)$ are pairwise disjoint.

Consider the sets $S_u = \{\mathfrak{s}(y^u_i) \mid i \in I_u \wedge y^u_i \in \{x_1, \ldots, x_n\}\}$. Observe that these sets are pairwise disjoint. Indeed, if $i \in I_u$ and $i' \in I_{u'}$ with $u \neq u'$ and $y^u_i, y^{u'}_{i'} \in \{x_1, \ldots, x_n\}$ then necessarily, by the remark above, (as $I_u \subseteq J_u$ and $I_{u'} \subseteq J_{u'}$ by Def. 8) $\mathfrak{s}(y^u_i) \neq \mathfrak{s}(y^{u'}_{i'})$. Similarly, these sets cannot contain $\{\mathfrak{s}(x_r)\}$. We now prove that $\bigcup_{u=1}^m S_u \cup \{\mathfrak{s}(x_r)\} = \Sigma(X) \cap \{\mathfrak{s}(x_1), \ldots, \mathfrak{s}(x_n)\}$:

$\subseteq$ By definition of decorated rules, if $y^u_i = x_j$ with $j \in [\![1, n]\!]$ and $i \in I_u$ then (by Def. 13) $y^u_i \in [Y_u]_{\psi\sigma}$, and (by Def. 15) there must exist $j' \in I$ such that $y^u_i = x_{j'}$. By item 3, we get $\mathfrak{s}(y^u_i) \in \Sigma(X)$. We prove in the same way that $\{\mathfrak{s}(x_j)\} \subseteq \Sigma(X)$.

$\supseteq$ If $\mathfrak{s}(x_i) \in \Sigma(X)$ and $x_r \neq x_i$, then, using again item 3, we deduce that $\mathfrak{s}(x_i) = \mathfrak{s}(x_{i'})$ with $i' \in I$, whence $x_i = x_{i'}$. By definition of the decorated rules and using the fact that $x_i \neq x_r$, we obtain that $x_i \equiv_{\psi\sigma} y^u_j$ for some $u \in [\![1, m]\!]$ and $j \in I_u$ which entails that $y^u_j \in \{x_1, \ldots, x_n\}$, $\mathfrak{s}(x_i) = \mathfrak{s}(y^u_j)$ and therefore $\mathfrak{s}(x_i) \in S_u$.

Consequently, we can define, for all $u \in [\![1, m]\!]$: $\Sigma'(Y_u) = S_u \cup L'_u$ where the sets L'_u are pairwise disjoint subsets of $\mathcal{L}$ such that $\mathrm{card}(\Sigma'(Y_u)) = \omega_u$, $L'_u \subseteq \Sigma(X)$, and $L'_u \cap \{\mathfrak{s}(x_1), \ldots, \mathfrak{s}(x_n)\} = \emptyset$. This is possible because $\mathrm{card}(\Sigma(X)) = 1 + \sum_{u=1}^m \omega_u$, thus, by the remarks above, $\Sigma(X)$ contains exactly $\sum_{u=1}^m (\omega_u - \mathrm{card}(S_u))$ elements not occurring in $\{\mathfrak{s}(x_1), \ldots, \mathfrak{s}(x_n)\}$. As a result, $\Sigma(X)$ is the disjoint union of $\{\mathfrak{s}(x_r)\}$ and of all $\Sigma'(Y_u)$ for $u \in [\![1, m]\!]$.

As already observed, if $i \in J_u$ and $i' \in J_{u'}$ with $u \neq u'$ then $y^u_i \neq y^{u'}_{i'}$. Consequently, we may define $\mathfrak{s}'$ as follows:

- All auxiliary location variables z such that $z = y^u_i$ for some $i \in I_u$ are mapped to pairwise distinct locations in $\Sigma'(Y_u)$. This is always possible because, by Prop. 4, $\mathrm{card}(\{[x]_{\equiv_{\psi\sigma}} \mid x \in [Y_u]_{\psi\sigma}\}) \leq \omega_u$. Using item 1 of the lemma, and the fact that $y^u_j \neq y^u_{j'} \implies y^u_j \not\equiv_{\psi\sigma} y^u_{j'} \implies j \not\sim_u j'$ if y^u_j or $y^u_{j'}$ is auxiliary, we deduce that $\mathrm{card}(S_u) + \mathrm{card}(\{y^u_i \mid i \in I_u, y^u_i \notin \{x_1, \ldots, x_n\}\}) \leq \omega_u$, so that $\mathrm{card}(L'_u) \geq \mathrm{card}(\{y^u_i \mid i \in I_u, y^u_i \notin \{x_1, \ldots, x_n\}\})$.
- All other auxiliary location variables z, i.e., not satisfying the previous condition, are mapped to pairwise distinct arbitrary locations in $\mathcal{L} \setminus L'_u$, also distinct from every location $\mathfrak{s}'(z')$ defined in the previous items.

By construction, $\mathfrak{s}'$ is injective. As $\mathcal{L}$ is infinite, we can associate to all $u \in [\![1, m]\!]$ infinite subsets L_u of $\mathcal{L}$ containing $\Sigma'(Y_u)$ and $\{\mathfrak{s}'(y^u_i) \mid 1 \leq i \leq n_u\}$. We may assume that L_u contains no element of the image of $\mathfrak{s}'$, except those in $\Sigma'(Y_u)$ or $\{\mathfrak{s}'(y^u_i) \mid 1 \leq i \leq n_u\}$. By construction, it is clear that the only locations $\ell \in L_u \cap L_{u'}$ (for $u \neq u'$) are such that $\ell = \mathfrak{s}'(y^u_i) = \mathfrak{s}'(y^{u'}_{i'})$ for some $i \in [\![1, n_u]\!]$ and $i' \in [\![1, n_{u'}]\!]$. Let $u \in m$. We show that $\mathfrak{s}'$, Σ', ψ_σ, ω_u and L_u satisfy all the hypothesis of the lemma.

1. By construction, for all $i, j \in [\![1, n_u]\!]$, if $i \sim_u j$, then $y^u_i \equiv_{\psi\sigma} y^u_j$. Assume (for the sake of contradiction) that, $y^u_i \neq y^u_j$, then, necessarily $y^u_i = x_k$ and $y^u_j = x_l$ for some $k, l \in [\![1, n]\!]$ (as $\psi\sigma$ is consistent and contains a disequation $z \not\approx z'$ for all auxiliary location variables z and for all location variables z' other than z). Thus we must have (by definition of decorated rules, and using item 1) $k \sim l$ so that $\mathfrak{s}(x_k) = \mathfrak{s}(x_l)$, which contradicts the above assumption (as $\mathfrak{s}$ is injective).
2. For all $i, j \in [\![1, n_u]\!]$, if $i \not\sim_u j$, then $y^u_i \neq y^u_j$ since $\psi\sigma$ is consistent, thus $\mathfrak{s}'(y^u_i) \neq \mathfrak{s}'(y^u_j)$ as $\mathfrak{s}'$ is injective.
3. For all $i \in [\![1, n_u]\!]$, if $i \in I_u$, either $y^u_i \in \{x_1, \ldots, x_n\}$ and by construction of $\Sigma'(Y_u)$, $\mathfrak{s}'(y^u_i) \in S_u \subseteq \Sigma'(Y_u)$, or y^u_i is an auxiliary location variable and then by construction of $\mathfrak{s}'(y^u_i)$, $\mathfrak{s}'(y^u_i) \in \Sigma'(Y_u)$. Conversely, if $\mathfrak{s}'(y^u_i) \in \Sigma'(Y_u)$, either $\mathfrak{s}'(y^u_i) \in S_u$ hence $y^u_i \in \{y^u_j \mid j \in I_u\}$, or $\mathfrak{s}'(y^u_i) \in L'_u$ and by construction of $\mathfrak{s}'(y^u_i)$, $y^u_i \in \{y^u_j \mid j \in I_u\}$.
4. By construction of $\Sigma'(Y_u)$, $\mathrm{card}(\Sigma'(Y_u)) = \omega_u$, with $\omega_u \in Sp(\psi_u\sigma)$.
5. By definition, L_u is infinite and contains $\mathfrak{s}'(\{y^u_i \mid i \in [\![1, n_u]\!]\})$ and $\Sigma'(Y_u)$.

By the induction hypothesis there exists a heap $\mathfrak{h}_u$ such that $(\mathfrak{s}', \mathfrak{h}_u, \Sigma') \models_{\mathcal{R}_{dec}} \psi_u\sigma$, $\mathfrak{h} \in L_u \times \mathcal{F} \times L_u^*$ and for all $i \in [\![1, n_u]\!]$, $\mathfrak{s}(y^u_i) \in \mathrm{allocated}(\mathfrak{h}_u) \iff \exists j \in J_u \; y^u_i = y^u_j$.

As shown above, if $u \neq u'$, then $\mathfrak{s}'(y_j) \neq \mathfrak{s}'(y'_{j'})$, for all $j \in J_u$ and $j' \in J_{u'}$. If $\ell \in \mathrm{allocated}(\mathfrak{h}_u) \cap \mathrm{allocated}(\mathfrak{h}_{u'})$, then $\ell \in L_u \cap L_{u'}$, thus, by definition of the sets L_u, there exist two location variables $y^u_j, y^{u'}_{j'}$ such that $\ell = \mathfrak{s}'(y^u_j) = \mathfrak{s}'(y^{u'}_{j'})$. By the above property we may assume that $j \in J_u$ and $j' \in J_{u'}$, which yields a contradiction.

As a result, all heaps $\mathfrak{h}_u$ are pairwise disjoint. Let us define $\mathfrak{h}' = [(\mathfrak{s}(x_r), f) \mapsto (\mathfrak{s}(z'_1\sigma), \ldots, \mathfrak{s}(z'_d\sigma))]$. We may prove in a similar way that $\mathfrak{h}_u$ and $\mathfrak{h}'$ are disjoint, and we can define $\mathfrak{h} = \mathfrak{h}' \cup \bigcup_{u=1}^m \mathfrak{h}_u$. By construction, $\mathfrak{h} \in \mathcal{L} \times \mathcal{F} \times \mathcal{L}^*$.

We must show that $(\mathfrak{s}', \mathfrak{h}, \Sigma') \models_{\mathcal{R}_{dec}} \psi$. By construction, $(\mathfrak{s}', \mathfrak{h}', \Sigma') \models_{\mathcal{R}_{dec}} (x_r \rightarrow (z'_1, \ldots, z'_d))\sigma$ and for all $u \in [\![1, m]\!]$ $(\mathfrak{s}', \mathfrak{h}_u, \Sigma') \models_{\mathcal{R}_{dec}} \psi_u\sigma$. Then, we need to prove that $(\mathfrak{s}', \emptyset, \Sigma') \models_{\mathcal{R}_{dec}} \varphi\sigma$. Consider, for the sake of contradiction, an atom $y \bowtie z$ in $\psi\sigma$ such that $(\mathfrak{s}', \emptyset, \Sigma')$ is not a model of $y \bowtie z$. As $\varphi\sigma$ is consistent, $y \bowtie z$ cannot be of the form $x \not\approx x$ or $y \approx z$, where $y \neq z$ and at least one of the location variables y, z is auxiliary (indeed, by the assumption in Sect. 2.4, $\varphi\sigma$ contains a disequation $y \not\approx z$ for all such y, z). As $\mathfrak{s}'$ is injective, we cannot have both $\bowtie = \not\approx$ and $y \neq z$. Consequently, $y \bowtie z$ must be of the form $(x'_i \approx x'_j)\sigma$, with $i, j \in [\![1, n]\!]$, so that $x'_i \equiv_\psi x'_j$ (by Def. 10) and thus $i \sim j$ (by Def. 15), which is impossible as this would entail that $x_i = x_j$ (by item 1 of the lemma).

Therefore, the only thing left to prove is that $(\mathfrak{s}', \emptyset, \Sigma') \models_{\mathcal{R}_{dec}} X \approx (E \sqcup \bigsqcup_{u=1}^m Y_u)\sigma$, i.e., $\Sigma'(X) = \{\mathfrak{s}'(x_r)\} \cup \bigcup_{u=1}^m \Sigma'(Y_u)$, which is true by construction of Σ' (as we have proven above that $\Sigma(X) = \Sigma'(X)$ is the union of $\{\mathfrak{s}(x_r)\}$ and of all $\Sigma'(Y_u)$ for $u \in [\![1, m]\!]$).

As a result, $(\mathfrak{s}', \mathfrak{h}, \Sigma') \models_{\mathcal{R}_{dec}} \psi\sigma$, and thus, as $\mathfrak{s}'$ and $\mathfrak{s}$ match on $x_1, \ldots, x_n$ and $\Sigma('X) = \Sigma(X)$, $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\not\approx}(x_1, \ldots, x_n, X)$.

Now we establish the second part of the lemma, i.e., that for all $i \in [\![1, n]\!]$, $\mathfrak{s}(x_i) \in \mathrm{allocated}(\mathfrak{h}) \iff x_i \in \{x_j \mid j \in J\}$.

$\Leftarrow$ If $j \in J$ then by Def. 15, $x'_j \in alloc(\psi)$, so that $x_j \in alloc(\psi\sigma)$. By Def. 11 and by injectivity of $\mathfrak{s}'$ this entails that either $x_j = x_r$ and $\mathfrak{s}'(x_j) \in \text{allocated}(\mathfrak{h}') \subseteq \text{allocated}(\mathfrak{h})$, or there exists $u \in [\![1, m]\!]$ and $j' \in J_u$ such that $x_j = y^u_{j'}$ and thus $\mathfrak{s}'(x_j) \in \text{allocated}(\mathfrak{h}_u) \subseteq \text{allocated}(\mathfrak{h})$.

$\Rightarrow$ Conversely, if $\mathfrak{s}'(x_j) \in \text{allocated}(\mathfrak{h})$, two cases must be distinguished. Either $\mathfrak{s}'(x_j) \in \text{allocated}(\mathfrak{h}')$ and $\mathfrak{s}(x_j) = \mathfrak{s}(x_r)$, so that $x_j = x_r$, with $l \in J$ (by Def. 11). Or there exists $u \in [\![1, m]\!]$ such that $\mathfrak{s}'(x_j) \in \text{allocated}(\mathfrak{h}_u)$. This entails that $\mathfrak{s}'(x_j) \in L_u$. By definition of L_u, $L_u \cap \{\mathfrak{s}(x_1), \ldots, \mathfrak{s}(x_n)\}$ may only contain elements in $\{\mathfrak{s}'(y^u_1), \ldots, \mathfrak{s}'(y^u_{n_u})\}$ (as $S_u \subseteq \{\mathfrak{s}'(y^u_1), \ldots, \mathfrak{s}'(y^u_{n_u})\}$, $L'_u \cap \{\mathfrak{s}(x_1), \ldots, \mathfrak{s}(x_n)\} = \emptyset$ and the other elements in L_u cannot occur in the image of $\mathfrak{s}'$). Thus there exists $j' \in [\![1, n_u]\!]$ verifying $\mathfrak{s}(x_j) = \mathfrak{s}'(y^u_{j'})$ and by the induction hypothesis we may assume that $j' \in J_u$. By Defs. 11 and 15, this is possible only if $y^u_{j'} = x_{j''}$ for some $j'' \in J$.

This ends the proof of Lem. 6. □

We go back now to the proof of Thm. 3.

For simplicity, we assume that the formula φ (and therefore also ψ) contains no points-to atom[13]. We will now apply Lem. 6 to construct structures satisfying each of these atoms individually, and then demonstrate how these structures can be combined to form a model of the entire formula. We first check that all the hypotheses of Lem. 6 are satisfied. As ψ is consistent and $x_i \not\simeq_\psi x_j$ for all $x_i \neq x_j$, we must have, for all $i, j \in [\![1, n]\!]$: $i \sim j \implies x_i = x_j$ and $i \neq j \implies x_i \neq x_j$. As $(\mathbb{S}, \mathbb{I}) \models_{BP} C(\psi)$, $(\mathbb{S}, \mathbb{I}) \models_{BP} \left(\bigsqcup^{BP}_{i \in I} V_{x_i}\right) \sqsubseteq_{BP} X \wedge \left(\bigsqcup^{BP}_{x \in \{x_1, \ldots, x_n\} \setminus \{x_j \mid j \in I\}} V_x\right) \sqcap_{BP} X \approx_{BP} \emptyset$ and thus, $\mathfrak{s}(x_i) \in \Sigma(X) \iff \exists j \in I,\ x_i = x_j$. Moreover, as $\mathcal{L}$ is infinite, to each decorated predicate atom $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ occurring in ψ, we can assign a infinite set $L_{p_{I,J,\sim,\neq}(x_1,\ldots,x_n,X)} \subseteq \mathcal{L}$ containing $\mathfrak{s}(\{x_i \mid i \in [\![1, n]\!]\})$ and $\Sigma(X)$. We also assume, w.l.o.g., that these sets share no locations other than those in $\mathfrak{s}(\{x_i \mid i \in [\![1, n]\!]\})$ or $\Sigma(X)$. Thus, by Lem. 6, for all decorated predicate atoms $p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$ of ψ there exists a heap $\mathfrak{h}_{p_{I,J,\sim,\neq}(x_1,\ldots,x_n,X)}$ such that for all $i \in [\![1, n]\!]$, $\mathfrak{s}(x_i) \in \text{allocated}(\mathfrak{h}) \iff x_i \in \{x_j \mid j \in J\}$, and $(\mathfrak{s}, \mathfrak{h}_{p_{I,J,\sim,\neq}(x_1,\ldots,x_n,X)}, \Sigma) \models_{\mathcal{R}_{dec}} p_{I,J,\sim,\neq}(x_1, \ldots, x_n, X)$.

Let ψ_1 and ψ_2 be two formulas such that $\psi_1 \bullet \psi_2$ is a subformula of ψ (with $\bullet \in \{\star, \circledast\}$). Let $p^i_{I_i,J_i,\sim_i,\neq_i}(x^i_1, \ldots, x^i_{n_i}, X_i)$ be a decorated predicate atom of ψ_i (for all $i \in \{1, 2\}$). Let f_i be the field in $\mathcal{F}$ such that p^i is an $\{f_i\}$-predicate. If $\bullet = \star$, then, according to Def. 21, $(\mathbb{S}, \mathbb{I}) \models_{BP} \left(\bigcup_{f \in \mathcal{F}} \mathcal{T}^f(\psi_1)\right) \cap \left(\bigcup_{f \in \mathcal{F}} \mathcal{T}^f(\psi_2)\right) = \emptyset$. By Def. 20, X_1 appears in $\left(\bigcup_{f \in \mathcal{F}} \mathcal{T}^f(\psi_1)\right)$ and X_2 in $\left(\bigcup_{f \in \mathcal{F}} \mathcal{T}^f(\psi_2)\right)$. Moreover, if $i \in J_1$ and $j \in J_2$ then $V_{x^1_i}$ and $V_{x^2_j}$ appear in $\left(\bigcup_{f \in \mathcal{F}} \mathcal{T}^f(\psi_1)\right)$ and $\left(\bigcup_{f \in \mathcal{F}} \mathcal{T}^f(\psi_2)\right)$, respectively. As a result, we get $(\Sigma(X_1) \cup \{\mathfrak{s}(x^1_i) \mid i \in J_1\}) \cap (\Sigma(X_2) \cup \{\mathfrak{s}(x^2_i) \mid i \in J_2\}) = \emptyset$.

Similarly, if $\bullet = \circledast$, then $(\mathbb{S}, \mathbb{I}) \models_{BP} \bigwedge_{f \in \mathcal{F}} \left(\mathcal{T}^f(\psi_1) \cap \mathcal{T}^f(\psi_2) = \emptyset\right)$ and thus if $f_1 = f_2$ then we also have $(\Sigma(X_1) \cup \{\mathfrak{s}(x^1_i) \mid i \in J_1\}) \cap (\Sigma(X_2) \cup \{\mathfrak{s}(x^2_i) \mid i \in J_2\}) = \emptyset$.

Let $L_i = L_{p^i_{I_i,J_i,\sim_i,\neq_i}(x^1_i,\ldots,x^i_{n_i},X_i)}$ and $\mathfrak{h}_i = \mathfrak{h}_{p^i_{I_i,J_i,\sim_i,\neq_i}(x^1_i,\ldots,x^i_{n_i},X_i)}$ (for $i \in \{1, 2\}$) be the sets of locations and heaps associated with $p^i_{I_i,J_i,\sim_i,\neq_i}(x^i_1, \ldots, x^i_{n_i}, X_i)$. Consider any location

[13] This assumption does not reduce expressiveness, as points-to atoms can be encoded as predicate atoms using a single inductive rule that allocates one location.

$\ell \in \text{allocated}(\mathfrak{h}_1) \cap \text{allocated}(\mathfrak{h}_2)$. By definition of the sets L_1, L_2, as $\text{allocated}(\mathfrak{h}_i) \subseteq L_i$, we get $\ell \in \{\mathfrak{s}(x_1^1), \ldots, \mathfrak{s}(x_{n_i}^i)\} \cup \Sigma(X_i)$, i.e., (using the above property of $\mathfrak{h}_i$) $\ell \in \{\mathfrak{s}(x_j^i) \mid j \in J_i\} \cup \Sigma(X_i)$. This entails that $f_1 \neq f_2$ and that $\bullet = \circledast$, since otherwise we have proved that $(\Sigma(X_1) \cup \{\mathfrak{s}(x_i^1) \mid i \in J_1\}) \cap (\Sigma(X_2) \cup \{\mathfrak{s}(x_i^2) \mid i \in J_2\}) = \emptyset$. Therefore $\mathfrak{h}_1 \cup \mathfrak{h}_2$ is well-defined, and if $\bullet = \star$ then $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are disjoint.

Consequently, all the heaps $\mathfrak{h}_{p_{I,J,\sim,\neq}(x_1,\ldots,x_n,X)}$ can be merged in one $\mathfrak{h}$ and this heap verifies $(\mathfrak{s}, \mathfrak{h}, \Sigma) \models_{\mathcal{R}_{dec}} \psi$, and thus, ψ is satisfiable.

F Example

Specification. The following formula in normal form specifies a heap configuration including two overlapping lists:

$$(\mathtt{ls}^{\mathtt{n}}(y, \text{nil}, Y) \circledast \mathtt{ls}^{\mathtt{f}}(x, \text{nil}, F)) \star \underbrace{F \sqsubseteq Y \star x \not\approx y \star \text{nil} \not\approx x \star \text{nil} \not\approx y}_{\Pi} . \tag{23}$$

The first list uses the field $\mathtt{n}$ to link the memory cells reachable from the location y; this list is specified by the atom $\mathtt{ls}^{\mathtt{n}}(y, \text{nil}, Y)$, where nil is a location variable and Y is a set variable. The second list collects the memory cells linked by the field $\mathtt{f}$ and reachable from location x; the atom specifying this list is $\mathtt{ls}^{\mathtt{f}}(x, \text{nil}, F)$, where F is a set variable. The constraint Π states the fact that the store is injective on locations and specifies the relation between set variables.

Consider the following set of inductive definitions $\mathcal{R}$:

$$\mathtt{ls}^{\mathtt{n}}(u_1, u_2, U) \Leftarrow u_1.\mathtt{n} \rightarrow (u_2) \star u_1 \not\approx u_2 \star U \approx \{u_1\}\,, \tag{24}$$

$$\mathtt{ls}^{\mathtt{n}}(u_1, u_2, U) \Leftarrow u_1.\mathtt{n} \rightarrow (z) \star u_1 \not\approx u_2 \star \mathtt{ls}^{\mathtt{n}}(z, u_2, U') \star U \approx \{u_1\} \sqcup U'\,, \tag{25}$$

$$\mathtt{ls}^{\mathtt{f}}(u_1, u_2, U) \Leftarrow u_1.\mathtt{f} \rightarrow (u_2) \star u_1 \not\approx u_2 \star U \approx \{u_1\}\,, \tag{26}$$

$$\mathtt{ls}^{\mathtt{f}}(u_1, u_2, U) \Leftarrow u_1.\mathtt{f} \rightarrow (z) \star u_1 \not\approx u_2 \star \mathtt{ls}^{\mathtt{f}}(z, u_2, U') \star U \approx \{u_1\} \sqcup U'\,. \tag{27}$$

In these rules, the set variables associated to each predicate collect all the locations allocated by the predicate. Therefore, in the formula given by Eq. 23, the constraint $F \sqsubseteq Y$ implies that the cells reachable from x are included in those reachable from y.

Tranformation in normal form. First, all free variables of the initial formula are added as additional parameters of predicate atoms:

$$(\mathtt{ls}^{\mathtt{n}}(y, \text{nil}, x, y, \text{nil}, Y) \circledast \mathtt{ls}^{\mathtt{f}}(x, \text{nil}, x, y, \text{nil}, F)) \star F \sqsubseteq Y \star x \not\approx y \star \text{nil} \not\approx x \star \text{nil} \not\approx y\,. \tag{28}$$

This addition changes the rules as follows:

$$\mathtt{ls}^{\mathtt{n}}(u_1, u_2, u_3, u_4, u_5, U) \Leftarrow u_1.\mathtt{n} \rightarrow (u_2) \star u_1 \not\approx u_2 \star U \approx \{u_1\}\,, \tag{29}$$

$$\mathtt{ls}^{\mathtt{n}}(u_1, u_2, u_3, u_4, u_5, U) \Leftarrow u_1.\mathtt{n} \rightarrow (z) \star u_1 \not\approx u_2 \star \mathtt{ls}^{\mathtt{n}}(z, u_2, u_3, u_4, u_5, U') \star U \approx \{u_1\} \sqcup U'\,, \tag{30}$$

$$\mathtt{ls}^{\mathtt{f}}(u_1, u_2, u_3, u_4, u_5, U) \Leftarrow u_1.\mathtt{f} \rightarrow (u_2) \star u_1 \not\approx u_2 \star U \approx \{u_1\}\,, \tag{31}$$

$$\mathtt{ls}^{\mathtt{f}}(u_1, u_2, u_3, u_4, u_5, U) \Leftarrow u_1.\mathtt{f} \rightarrow (z) \star u_1 \not\approx u_2 \star \mathtt{ls}^{\mathtt{f}}(z, u_2, u_3, u_4, u_5, U') \star U \approx \{u_1\} \sqcup U'\,. \tag{32}$$

Second, we transform the rules above to ensure that Condition 2 (in the definition of normal form) holds for variable z. During this process, we can eliminate the rules for which the right hand side formula is (trivially) unsatisfiable. For instance, the set of rules for $\mathtt{ls}^{\mathtt{n}}$ becomes:

$$\mathtt{ls}^{\mathtt{n}}(u_1,u_2,u_3,u_4,u_5,U) \Leftarrow u_1.\mathtt{n} \rightarrow (u_2) \star u_1 \not\approx u_2 \star U \approx \{u_1\}\,, \tag{33}$$

$$\mathtt{ls}^{\mathtt{n}}(u_1,u_2,u_3,u_4,u_5,U) \Leftarrow u_1.\mathtt{n} \rightarrow (z) \star u_1 \not\approx u_2 \star_{i=1..5} z \not\approx u_i \star \mathtt{ls}^{\mathtt{n}}(z,u_2,u_3,u_4,u_5,U') \star U \approx \{u_1\} \sqcup U'\,, \tag{34}$$

$$\mathtt{ls}^{\mathtt{n}}(u_1,u_2,u_3,u_4,u_5,U) \Leftarrow u_1.\mathtt{n} \rightarrow (u_2) \star u_1 \not\approx u_2 \star \mathtt{ls}^{\mathtt{n}}(u_2,u_2,u_3,u_4,u_5,U') \star U \approx \{u_1\} \sqcup U'\,, \tag{35}$$

$$\mathtt{ls}^{\mathtt{n}}(u_1,u_2,u_3,u_4,u_5,U) \Leftarrow u_1.\mathtt{n} \rightarrow (u_3) \star u_1 \not\approx u_2 \star \mathtt{ls}^{\mathtt{n}}(u_3,u_2,u_3,u_4,u_5,U') \star U \approx \{u_1\} \sqcup U'\,, \tag{36}$$

$$\mathtt{ls}^{\mathtt{n}}(u_1,u_2,u_3,u_4,u_5,U) \Leftarrow u_1.\mathtt{n} \rightarrow (u_4) \star u_1 \not\approx u_2 \star \mathtt{ls}^{\mathtt{n}}(u_4,u_2,u_3,u_4,u_5,U') \star U \approx \{u_1\} \sqcup U'\,, \tag{37}$$

$$\mathtt{ls}^{\mathtt{n}}(u_1,u_2,u_3,u_4,u_5,U) \Leftarrow u_1.\mathtt{n} \rightarrow (u_5) \star u_1 \not\approx u_2 \star \mathtt{ls}^{\mathtt{n}}(u_5,u_2,u_3,u_4,u_5,U') \star U \approx \{u_1\} \sqcup U'\,. \tag{38}$$

In the rule 34, the auxiliary variable z is different from parameters. We omit the case when z is aliased by the parameter u_1 because the formula obtained by unfolding $\mathtt{ls}^{\mathtt{n}}(u_1,u_2,u_3,u_4,u_5,U')$ is inconsistent (u_1 is allocated twice). The rule 35 has also an inconsistent definition because any unfolding of $\mathtt{ls}^{\mathtt{n}}(u_2,u_2,u_3,u_4,u_5,U')$ leads to an inconsistent formula.

Decorate predicate symbols. The decorations of predicate symbols that are consistent with the formula and the rules are the following:

$\mathtt{ls}^{\mathtt{n}}_{d_1}$: $I = \{1\}$, $J = \{1\}$, $\sim$ is the identity and $\not\approx$ is $\{(1,2),(2,1)\}$;
$\mathtt{ls}^{\mathtt{n}}_{d_2}$: $I = \{1,3\}$, $J = \{1,3\}$, $\sim$ is the identity and $\not\approx$ is $\{(1,2),(2,1)\}$.

We obtain the following rules for the decorations of $\mathtt{ls}^{\mathtt{n}}$ in the decorated SID, $\mathcal{R}_{dec}$:

$$\mathtt{ls}^{\mathtt{n}}_{d_1}(u_1,u_2,u_3,u_4,u_5,U) \Leftarrow u_1.\mathtt{n} \rightarrow (u_2) \star u_1 \not\approx u_2 \star U \approx \{u_1\}\,, \tag{39}$$

$$\mathtt{ls}^{\mathtt{n}}_{d_1}(u_1,u_2,u_3,u_4,u_5,U) \Leftarrow u_1.\mathtt{n} \rightarrow (z) \star u_1 \not\approx u_2 \star_{i=1..5} z \not\approx u_i \star \mathtt{ls}^{\mathtt{n}}_{d_1}(z,u_2,u_3,u_4,u_5,U') \star U \approx \{u_1\} \sqcup U'\,, \tag{40}$$

$$\mathtt{ls}^{\mathtt{n}}_{d_2}(u_1,u_2,u_3,u_4,u_5,U) \Leftarrow u_1.\mathtt{n} \rightarrow (u_3) \star u_1 \not\approx u_2 \star \mathtt{ls}^{\mathtt{n}}_{d_2}(u_3,u_2,u_3,u_4,u_5,U') \star U \approx \{u_1\} \sqcup U'\,. \tag{41}$$

The case where $z = u_2$ (rule 35) is not considered as the right-hand side would be inconsistent (due to the constraint $1 \not\approx 2$, parameters 1 and 2 cannot be instantiated by the same variable). Similarly, the case where $z = u_4$ (rule 37) is irrelevant because the decoration would induce the constraints $1 \not\approx 4$ (because the locations corresponding to parameters 1 and 4 would be allocated twice), which makes the decoration of the initial formula inconsistent. Finally, the case where $z = u_5$ (rule 38) is discarded because the decoration would yield the constraint $2 \not\approx 5$, which, again, would be inconsistent in the initial formula.