A Complete Decision Procedure for Linearly Compositional Separation Logic with Data Constraints

Xincai Gu[1,2], Taolue Chen[3], and Zhilin Wu[1](✉)

[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
wuzl@ios.ac.cn

[2] University of Chinese Academy of Sciences, Beijing, China

[3] Department of Computer Science, Middlesex University, London, UK

Abstract. Separation logic is a widely adopted formalism to verify programs manipulating dynamic data structures. Entailment checking of separation logic constitutes a crucial step for the verification of such programs. In general this problem is undecidable, hence only incomplete decision procedures are provided in most state-of-the-art tools. In this paper, we define a linearly compositional fragment of separation logic with inductive definitions, where traditional shape properties for linear data structures, as well as data constraints, e.g., the sortedness property and size constraints, can be specified in a unified framework. We provide complete decision procedures for both the satisfiability and the entailment problem, which are in NP and Π_3^{P} respectively.

1 Introduction

Program verification requires reasoning about complex, unbounded size data structures that may carry data ranging over infinite domains. Examples of such data structures are multi-linked lists, nested lists, trees, etc. Programs manipulating these data structures may modify their shape (due to dynamic creation and destructive updates) as well as the data attached to their elements.

Separation Logic (SL) is a well-established approach for deductive verification of programs that manipulate dynamic data structures [18,24]. Typically, SL is used in combination with inductive definitions, which provide a natural description of the data structures manipulated by a program.

In program verification, SL is normally used to express assertions about program configurations, for example in the style of Hoare logic. Checking the validity

Taolue Chen is partially supported by the ARC Discovery Project (DP160101652), the Singapore Ministry of Education AcRF Tier 2 grant (MOE2015-T2-1-137), and an oversea grant from the State Key Laboratory of Novel Software Technology, Nanjing Unviersity. Zhilin Wu is partially supported by the NSFC grants (No. 61100062, 61272135, 61472474, and 61572478).

N. Olivetti and A. Tiwari (Eds.): IJCAR 2016, LNAI 9706, pp. 532–549, 2016.
DOI: 10.1007/978-3-319-40229-1_36

of these assertions is naturally reduced to the *entailment* problem of the logic, i.e., given two SL formulae φ and ψ, to check whether $\varphi \models \psi$ holds.

Because of its importance, entailment checking has been explored extensively (see, e.g., [1,9,16]). In general, it is an undecidable problem, hence only *incomplete* decision procedures can be expected. This is especially the case when both shape properties and data (size) constraints are taken into consideration. Indeed, various separation logic based tools, e.g., INFER [8], SLEEK/HIP [9], DRYAD [19,23], and SPEN [13], only provide incomplete decision procedures.

Undoubtedly *complete* decision procedures are highly desirable: besides being theoretically appealing, they also have practical importance, for instance in tasks such as debugging of specification, counterexample generation, etc. The challenge is thus to find fragments of SL which are sufficiently expressive for writing program assertions while still feature a complete decision procedure for the entailment checking. This would enable efficient automated validation of the verification conditions.

Contributions. In this paper, we define a *linearly compositional* fragment of SL with inductive definitions (abbreviated as $\mathsf{SLID}_{\mathsf{LC}}$), where both shape properties, e.g., singly and doubly linked lists, linked lists with tail pointers, and data constraints, e.g., sortedness property and size constraints, can be expressed. The basic idea of $\mathsf{SLID}_{\mathsf{LC}}$ is to focus on the compositional predicates introduced in [14], while restricting to linear shapes (e.g., singly and doubly linked lists, or linked lists with tail pointers), and data constraints in the form of difference bound relations (which are sufficient to express sortedness properties and size constraints). Our main contribution is to provide complete decision procedures for the satisfiability and entailment problem of $\mathsf{SLID}_{\mathsf{LC}}$.

For the satisfiability problem, from each $\mathsf{SLID}_{\mathsf{LC}}$ formula φ we define an abstraction of φ, i.e., $\mathsf{Abs}(\varphi)$, where Boolean variables are introduced to encode the spatial part of φ, together with quantifier free Presburger formulae to represent the transitive closure of the data constraint in the inductive definitions. The satisfiability of φ is then reduced to the satisfiability of $\mathsf{Abs}(\varphi)$, which can be solved by the start-of-the-art SMT solvers (e.g., Z3 [25]), with an NP upper-bound.

For the entailment problem, from each $\mathsf{SLID}_{\mathsf{LC}}$ formula φ we first construct a graph representation $\mathcal{G}_\varphi$. We then demonstrate some nice properties of $\mathcal{G}_\varphi$, which enable us to extend and adapt the concept of homomorphisms introduced in [11], to obtain a decision procedure to perform entailment checking with a Π_3^{P} upper-bound. Compared to the logic in [11], the logic $\mathsf{SLID}_{\mathsf{LC}}$ is different in the following sense: (1) we adopt the classical semantics whereas [11] adopted the intuitionistic semantics, which can be considered as a special case, and is arguably less meaningful for program verification. (2) the logic in [11] only addresses singly linked list segments, the logic $\mathsf{SLID}_{\mathsf{LC}}$ is much more expressive: $\mathsf{SLID}_{\mathsf{LC}}$ allows specifying data constraints, as well as defining more shapes, e.g., doubly linked lists, linked lists with tail pointers; in addition, we allow different predicates to occur in φ and ψ for the entailment problem $\varphi \models \psi$. Because of these differences, we are not able to repeat the approach in [11] to transform the graphs into normal forms and then check graph homomorphism between the normal forms.

Instead our decision procedure introduces some new concepts e.g. allocating plans for φ and is considerably more involved than that in [11].

Related Work. We first discuss work on separation logic with inductive definitions where both shape properties and data constraints can be expressed. Various fragments have been explored and we focus on decision procedures for the entailment problem.

The most relevant work is [3], where data constraints, specified by universal quantifiers over index variables, were added to a fragment of separation logic with the *lseg* predicate (where *lseg* denotes list segments). Compared with the work in [3]: For the shape constraints, the logic there focused on singly linked lists, while in $\mathsf{SLID_{LC}}$, various linear data structures can be specified. For the data constraints, the logic there can specify set and multiset constraints, while $\mathsf{SLID_{LC}}$ does not. On the other hand, when restricted to arithmetic constraints over integer variables, the decision procedure in [3] is incomplete for fragments that can express list segments where the data values are consecutive, which can be easily expressed in $\mathsf{SLID_{LC}}$ (cf. *plseg* predicate in Example 1).

The tool SLEEK/HIP [9] provides a decision procedure which is incomplete in general and relies on the invariants of the inductive definitions. These invariants are essentially the transitive closures of the data constraints in the inductive definitions, and are supposed to be provided by the user. In comparison, we focus on a less expressive logic $\mathsf{SLID_{LC}}$, and our decision procedure can *automatically* compute the *precise* invariants of the inductive definitions.

The tool GRASSHOPPER [20–22] encoded separation logic with inductive definitions into a fragment of first-order logic with reachability predicates, whose satisfiability problem was shown in NP. The logic considered there includes both shape and data constraints and the decision procedure is complete. However the logic is unable to encode the size or multiset constraints. In contrast, our approach can fully handle the size constraints, and the multiset constraints on condition that their transitive closure can be computed (or provided as an oracle).

The tool DRYAD [19,23] reduces to the satisfiability problem in the theory of uninterpreted functions, which is sound, but incomplete. In addition, the decision procedure is *not* fully automatic since it relies on the users to provide lemmas, e.g., $\mathit{lseg}(E_1; E_2) * \mathit{lseg}(E_2; E_3) \models \mathit{lseg}(E_1; E_3)$.

Other work includes the cyclic-proof approach [6,10] which is based on induction on the paths of proof trees. The approach can deal with data constraints but the decision procedures there are incomplete. The work [14] considered the automated lemma generation, where the concept of compositional predicates was introduced. However, the decision procedure provided there is incomplete.

There have also been much work on the decision procedures for the fragments of SL with inductive definitions that contain no data constraints. To cite a few, the work [2,15] focused on the symbolic heap fragments where the shape constraints for list segments and binary trees can be specified and complete proof systems were given, the tool SLIDE [16,17] considered separation logic with general inductive definitions and reduced the entailment problem to the language inclusion problem of tree automata, tool SPEN [13] provided an incomplete decision procedure for a compositional fragment of separation logic with inductive definitions, and the paper [7] designed a complete decision procedure for the satisfiability problem of separation logic with general inductive definitions.

There are also other works on separation logic. The work [4] considered first-order separation logic over linked lists extended with length constraints where the decidability frontier was identified. However, neither data structures other than singly linked lists nor other forms of data constraints (e.g. sortedness) were addressed. The work [5,12] considered the fragments of first-order separation logic (without inductive definitions). The authors identified the decidability frontier and resolved some long-standing expressibility issues.

2 Linearly Compositional Separation Logic with Inductive Definitions

In this section, we introduce the *linearly compositional* fragment of separation logic with inductive definitions, denoted by $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$, where $\mathcal{P}$ is a finite set of *inductive predicates*. In $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$, both shape properties (e.g. doubly linked lists) and data constraints (e.g. sortedness and size constraints) can be specified.

We consider two data types, i.e., the *location* type $\mathbb{L}$ and the *integer* type $\mathbb{Z}$. As a convention, $l, l', \cdots \in \mathbb{L}$ denote locations and $n, n', \cdots \in \mathbb{Z}$ denote integers. Accordingly, variables in $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ comprise *location variables* of the location type and *data variables* of the integer type. Namely, we assume a set of location variables LVars ranged over by uppercase letters $E, F, X, Y, \cdots$ and a set of data variables DVars ranged over by lowercase letters $x, y, \cdots$. Note that in literature sometimes locations are treated simply as a subset of integers, which is not adopted here for the sake of clarity. We consider two kinds of *fields*, i.e., location fields from $\mathcal{F}$ and data fields from $\mathcal{D}$. Each field $f \in \mathcal{F}$ (resp. $d \in \mathcal{D}$) is associated with $\mathbb{L}$ (resp. $\mathbb{Z}$).

$\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ formulae may contain inductive predicates, each of which is of the form $P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$ and has an associated inductive definition. The parameters of an inductive predicate are classified into three groups: *source parameters* $\boldsymbol{\alpha}$, *destination parameters* $\boldsymbol{\beta}$, and *static parameters* $\boldsymbol{\xi}$. We require that the source parameters $\boldsymbol{\alpha}$ and the destination parameters $\boldsymbol{\beta}$ are *matched* in type, namely, the two tuples have the same length $\ell > 0$ and for each $i : 1 \leqslant i \leqslant \ell$, α_i and β_i have the same data type. Without loss of generality, it is assumed that the first components of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are a location variable. In the sequel, for clarity, we explicitly identify the first parameters of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and write $E, \boldsymbol{\alpha}$ and $F, \boldsymbol{\beta}$.

$\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ formulae comprise three types of formulae: *pure formulae* Π, *data formulae* Δ, and *spatial formulae* Σ, which are defined by the following rules,

$$\begin{array}{lr}
\Pi ::= E = F \mid E \neq F \mid \Pi \wedge \Pi & \text{(pure formulae)} \\
\Delta ::= \mathtt{true} \mid x \ \mathtt{o}\ c \mid x \ \mathtt{o}\ y + c \mid \Delta \wedge \Delta & \text{(data formulae)} \\
\Sigma ::= \mathtt{emp} \mid E \mapsto \rho \mid P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi}) \mid \Sigma * \Sigma & \text{(spatial formulae)} \\
\rho ::= (f, X) \mid (d, x) \mid \rho, \rho &
\end{array}$$

where $\mathfrak{o} \in \{=, \leqslant, \geqslant\}$, c is an integer constant, $P \in \mathcal{P}$, $f \in \mathcal{F}$, and $d \in \mathcal{D}$. For spatial formulae Σ, formulae of the form $\mathtt{emp}$, $E \mapsto \rho$, or $P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$ are called *spatial atoms*. In particular, formulae of the form $E \mapsto \rho$ and $P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$ are called *points-to atoms* and *predicate atoms* respectively. Moreover, we call E as *the root* of these points-to or predicate atoms.

We are now in a position to introduce the *linearly compositional* predicates, which are the main focus of the current paper. A predicate $P \in \mathcal{P}$ is *linearly compositional* if the inductive definition of P is given by the following two rules,

– base rule $R_0 : P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi}) ::= E = F \wedge \boldsymbol{\alpha} = \boldsymbol{\beta} \wedge \mathtt{emp}$,
– inductive rule $R_1 : P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi}) ::= \exists \boldsymbol{X} \exists \boldsymbol{x}.\ \Delta \wedge E \mapsto \rho * P(Y, \boldsymbol{\gamma}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$.

The left-hand (resp. right-hand) side of a rule is called the *head* (resp. *body*) of the rule. We note that the body of R_1 does not contain pure formulae.

In the sequel, we specify some constraints on the inductive rule R_1 which enable us to obtain *complete* decision procedures for the satisfiability and entailment problem later.

The first constraint (**C1**) is from [14] which guarantees that $P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$ enjoys the composition lemma (cf. Proposition 1). This lemma is the basis of our decision procedure for the entailment problem (cf. Sect. 4.2).

C1. None of the variables from $F, \boldsymbol{\beta}$ occur elsewhere in the body of R_1, that is, in Δ, or $E \mapsto \rho$.

The second (**C2**) and third (**C3**) constraint address the data constraint Δ in the body of R_1. Intuitively, the two constraints require that different data parameters of $P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$ do not interfere with each other and the value of each data source parameter α_i is determined either by ρ, or γ_i.

C2. Each conjunct of Δ is of the form $\alpha_i\ \mathfrak{o}\ c$, $\alpha_i\ \mathfrak{o}\ \xi_j$, or $\alpha_i\ \mathfrak{o}\ \gamma_i + c$ for $\mathfrak{o} \in \{=, \leqslant, \geqslant\}$, $1 \leqslant i \leqslant |\boldsymbol{\alpha}| = |\boldsymbol{\gamma}|$, $1 \leqslant j \leqslant |\boldsymbol{\xi}|$, and $c \in \mathbb{Z}$.

C3. For each $1 \leqslant i \leqslant |\boldsymbol{\alpha}|$ such that α_i is a data variable, either α_i occurs in ρ, or Δ contains $\alpha_i = \gamma_i + c$ for some $c \in \mathbb{Z}$.

Furthermore, we have **C4**–**C6**, which are self-explained.

C4. Each variable occurs in $P(Y, \boldsymbol{\gamma}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$ (resp. ρ) at most once.

C5. All location variables from $\boldsymbol{\alpha} \cup \boldsymbol{\xi} \cup \boldsymbol{X}$ occur in ρ.

C6. $Y \in \boldsymbol{X}$ and $\boldsymbol{\gamma} \subseteq \{E\} \cup \boldsymbol{X} \cup \boldsymbol{x}$.

Note that according to the constraint **C6**, none of the variables from $\boldsymbol{\alpha} \cup \boldsymbol{\xi}$ occur in $\boldsymbol{\gamma}$. Moreover, from the constraint **C5** and **C6**, we know that Y occurs in ρ. By the semantics defined later, this would guarantee that in each model of $P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$, the sub-heap represented by $P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$, seen as a directed graph, is connected.

We remark that these constraints are technical, and we leave as future work to make them as general as possible. However, in practice, inductive predicates satisfying these constraints are sufficient to model linear data structures with data and size constraints, cf. Example 1.

For a linearly compositional predicate $P \in \mathcal{P}$, let $\mathrm{Flds}(P)$ (resp. $\mathrm{LFlds}(P)$) denote the set of fields (resp. location fields) occurring in the inductive rules of P. Moreover, define the *principal* location field of P, denoted by $\mathrm{PLFld}(P)$, as the location field $f \in \mathrm{LFlds}(P)$ such that (f, Y) occurs in ρ. Note that the principal location field is unique. For a spatial atom a, let $\mathrm{Flds}(a)$ denote the set of fields that a refers to: if $a = E \mapsto \rho$, then $\mathrm{Flds}(a)$ is the set of fields occurring in ρ; if $a = P(-)$, then $\mathrm{Flds}(a) := \mathrm{Flds}(P)$.

We write $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ for the collection of separation logic formulae $\varphi = \Pi \wedge \Delta \wedge \Sigma$ satisfying the following constraints,

- **linearly compositional predicates**: all predicates from $\mathcal{P}$ are linearly compositional,
- **domination of principal location field**: for each pair of predicates $P_1, P_2 \in \mathcal{P}$, if $\mathrm{Flds}(P_1) = \mathrm{Flds}(P_2)$, then $\mathrm{PLFld}(P_1) = \mathrm{PLFld}(P_2)$,
- **uniqueness of predicates**: there is $P \in \mathcal{P}$ such that each predicate atom of Σ is of the form $P(-)$, and for each points-to atom occurring in Σ, the set of fields of this atom is $\mathrm{Flds}(P)$.

For an $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ formula φ, let $\mathsf{Vars}(\varphi)$ (resp. $\mathsf{LVars}(\varphi)$, resp. $\mathsf{DVars}(\varphi)$) denote the set of (resp. location, resp. data) variables occurring in φ. Moreover, we use $\varphi[\boldsymbol{\mu}/\boldsymbol{\alpha}]$ to denote the simultaneous replacement of the variables α_j by μ_j in φ.

For the semantics of $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$, each formula is interpreted on the states. Formally, a *state* is a pair (s, h), where

- s is an assignment function which is a partial function from $\mathsf{LVars} \cup \mathsf{DVars}$ to $\mathbb{L} \cup \mathbb{Z}$ such that $dom(s)$ is finite and s respects the data type,
- h is a *heap* which is a partial function from $\mathbb{L} \times (\mathcal{F} \cup \mathcal{D})$ to $\mathbb{L} \cup \mathbb{D}$ such that
  - h respects the data type of fields, that is, for each $l \in \mathbb{L}$ and $f \in \mathcal{F}$ (resp. $l \in \mathbb{L}$ and $d \in \mathcal{D}$), if $h(l, f)$ (resp. $h(l, d)$) is defined, then $h(l, f) \in \mathbb{L}$ (resp. $h(l, d) \in \mathbb{Z}$); and
  - h is field-consistent, i.e. every location in h possess the same set of fields.

For a heap h, we use $\mathsf{ldom}(h)$ to denote the set of locations $l \in \mathbb{L}$ such that $h(l, f)$ or $h(l, d)$ is defined for some $f \in \mathcal{F}$ and $d \in \mathcal{D}$. Moreover, we use $\mathrm{Flds}(h)$ to denote the set of fields $f \in \mathcal{F}$ or $d \in \mathcal{D}$ such that $h(l, f)$ or $h(l, d)$ is defined for some $l \in \mathbb{L}$.

Two heaps h_1 and h_2 are said to be *field-compatible* if $\mathrm{Flds}(h_1) = \mathrm{Flds}(h_2)$. We write $h_1 \# h_2$ if $\mathsf{ldom}(h_1) \cap \mathsf{ldom}(h_2) = \emptyset$. Moreover, we write $h_1 \uplus h_2$ for the disjoint union of two field-compatible fields h_1 and h_2 (this implies that $h_1 \# h_2$).

Let (s, h) be a state and φ be an $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ formula. The semantics of $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ formulae is defined as follows,

- $(s, h) \models E = F$ (resp. $(s, h) \models E \neq F$) if $s(E) = s(F)$ (resp. $s(E) \neq s(F)$),
- $(s, h) \models \Pi_1 \wedge \Pi_2$ if $(s, h) \models \Pi_1$ and $(s, h) \models \Pi_2$,
- $(s, h) \models x \ \mathtt{o}\ c$ (resp. $(s, h) \models x \ \mathtt{o}\ y + c$) if $s(x) \ \mathtt{o}\ c$ (resp. $s(x) \ \mathtt{o}\ s(y) + c$),
- $(s, h) \models \Delta_1 \wedge \Delta_2$ if $(s, h) \models \Delta_1$ and $(s, h) \models \Delta_2$,
- $(s, h) \models \mathtt{emp}$ if $\mathsf{ldom}(h) = \emptyset$,
- $(s, h) \models E \mapsto \rho$ if $\mathsf{ldom}(h) = s(E)$, and for each $(f, X) \in \rho$ (resp. $(d, x) \in \rho$), $h(s(E), f) = s(X)$ (resp. $h(s(E), d) = s(x)$),

- $(s,h) \models P(E,\boldsymbol{\alpha};F,\boldsymbol{\beta};\boldsymbol{\xi})$ if $(s,h) \in [\![P(E,\boldsymbol{\alpha};F,\boldsymbol{\beta};\boldsymbol{\xi})]\!]$,
- $(s,h) \models \Sigma_1 * \Sigma_2$ if there are h_1, h_2 such that $h = h_1 \uplus h_2$, $(s,h_1) \models \Sigma_1$ and $(s,h_2) \models \Sigma_2$.

where the semantics of predicates $[\![P(E,\boldsymbol{\alpha};F,\boldsymbol{\beta};\boldsymbol{\xi})]\!]$ is given by the least fixed point of a monotone operator constructed from the body of rules for P in a standard way as in [7].

Example 1. Below are a few examples of the data structures definable in $\mathsf{SLID_{LC}}[\mathcal{P}]$: *slseg* for sorted list segments, *dllseg* for doubly linked list segments, *tlseg* for list segments with tail pointers, *plseg* for list segments where the data values are consecutive, and *ldllseg* for doubly list segments with lengths.

$$\begin{aligned}
&\mathit{slseg}(E,x;F,x') ::= E = F \wedge x = x' \wedge \mathtt{emp},\\
&\mathit{slseg}(E,x;F,x') ::= \exists X, x''.\ x \leqslant x'' \wedge\\
&\qquad\qquad E \mapsto ((\mathtt{next},X),(\mathtt{data},x)) * \mathit{slseg}(X,x'';F,x').\\
&\mathit{dllseg}(E,P;F,L) ::= E = F \wedge P = L \wedge \mathtt{emp},\\
&\mathit{dllseg}(E,P;F,L) ::= \exists X.\ E \mapsto ((\mathtt{next},X),(\mathtt{prev},P)) * \mathit{dllseg}(X,E;F,L).\\
&\mathit{tlseg}(E;F;B) ::= E = F \wedge \mathtt{emp},\\
&\mathit{tlseg}(E;F;B) ::= \exists X.\ E \mapsto ((\mathtt{next},X),(\mathtt{tail},B)) * \mathit{tlseg}(X;F;B).\\
&\mathit{plseg}(E,x;F,x') ::= E = F \wedge x = x' \wedge \mathtt{emp},\\
&\mathit{plseg}(E,x;F,x') ::= \exists X, x''.\ x'' = x + 1 \wedge\\
&\qquad\qquad E \mapsto ((\mathtt{next},X),(\mathtt{data},x)) * \mathit{plseg}(X,x'';F,x').\\
&\mathit{ldllseg}(E,P,x;F,L,x') ::= E = F \wedge P = L \wedge x = x' \wedge \mathtt{emp},\\
&\mathit{ldllseg}(E,P,x;F,L,x') ::= \exists X, x''.\ x = x'' + 1 \wedge E \mapsto ((\mathtt{next},X),(\mathtt{prev},P))\\
&\qquad\qquad * \mathit{ldllseg}(X,E,x'';F,L,x').
\end{aligned}$$

On the other hand, the predicate *tlseg*2 defined below is *not* linearly compositional, since F occurs twice in the body of the inductive rule.

$$\begin{aligned}
&\mathit{tlseg}2(E;F) ::= E = F \wedge \mathtt{emp},\\
&\mathit{tlseg}2(E;F) ::= \exists X.\ E \mapsto ((\mathtt{next},X),(\mathtt{tail},F)) * \mathit{tlseg}2(X;F).
\end{aligned}$$

For a formula φ, let $[\![\varphi]\!]$ denote the set of states (s,h) such that $(s,h) \models \varphi$. Let φ, ψ be $\mathsf{SLID_{LC}}[\mathcal{P}]$ formulae, then define $\varphi \models \psi$ as $[\![\varphi]\!] \subseteq [\![\psi]\!]$.

Proposition 1 [14]. *For each linearly compositional predicate $P \in \mathcal{P}$, it holds that $P(E,\boldsymbol{\alpha};F,\boldsymbol{\beta};\boldsymbol{\xi}) * P(F,\boldsymbol{\beta};G,\boldsymbol{\gamma};\boldsymbol{\xi}) \models P(E,\boldsymbol{\alpha};G,\boldsymbol{\gamma};\boldsymbol{\xi})$.*

We focus on the following two decision problems.

- Satisfiability: Given an $\mathsf{SLID_{LC}}[\mathcal{P}]$ formula φ, decide whether $[\![\varphi]\!]$ is empty.
- Entailment: Given two $\mathsf{SLID_{LC}}[\mathcal{P}]$ formulae φ, ψ such that $\mathsf{Vars}(\psi) \subseteq \mathsf{Vars}(\varphi)$, decide whether $\varphi \models \psi$ holds.

The rest of this paper is devoted to sound and complete decision procedures for the satisfiability and entailment problem of $\mathsf{SLID_{LC}}[\mathcal{P}]$.

3 Satisfiability

To decide the satisfiability of a separation logic formula φ, in [13], a *Boolean* abstraction $\mathsf{BoolAbs}(\varphi)$ of φ was constructed such that φ is satisfiable iff $\mathsf{BoolAbs}(\varphi)$ is satisfiable. Our decision procedure for $\mathsf{SLID_{LC}}[\mathcal{P}]$ follows this general approach. However, $\mathsf{SLID_{LC}}[\mathcal{P}]$ admits data constrains (viz. difference bound constraints specified in the data formulae) which are considerably more involved. The following example shows these data constraints are somehow intertwined with the "shape" part of the logic and they should be taken into account simultaneously when the satisfiability is concerned.

Example 2. Suppose $\varphi = E_1 = E_4 \wedge x_1 > x_2 + 1 \wedge \mathit{ldllseg}(E_1, E_3, x_1; E_2, E_4, x_2)$. From the inductive definition of *ldllseg* and $x_1 > x_2 + 1$, we know that if φ is satisfiable, then for any state (s, h) such that $(s, h) \models \varphi$, it holds that $|\mathsf{ldom}(h)| \geqslant 2$. On the other hand, in any heap (s, h) such that $(s, h) \models \mathit{ldllseg}(E_1, E_3, x_1; E_2, E_4, x_2)$ and $|\mathsf{ldom}(h)| \geqslant 2$, we know that both $s(E_1)$ and $s(E_4)$ are allocated and $s(E_1) \neq s(E_4)$. This contradicts to the fact that $E_1 = E_4$ is a conjunct in φ. Therefore, φ is unsatisfiable.

In the rest of this section, we will show how to extend the abstraction of formulae in [13] to obtain an abstraction in the presence of data constraints. In this case, the abstraction is *not* a Boolean formula, but a formula involving Boolean variables, (in)equality constraints over location variables, and difference bounded constraints over data variables. The satisfiability of these formulae can be decided by off-the-shelf SMT solvers. We also remark that, compared to the logic in [13], predicates in $\mathsf{SLID_{LC}}[\mathcal{P}]$ may have more than one source or destination parameter which gives rises to further technical difficulties.

Let $\varphi = \Pi \wedge \Delta \wedge \Sigma$ be an $\mathsf{SLID_{LC}}[\mathcal{P}]$ formula. Suppose $\Sigma = a_1 * \cdots * a_n$, where each a_i is either a points-to atom or a predicate atom.

Assume $a_i = P(Z_1, \boldsymbol{\mu}; Z_2, \boldsymbol{\nu}; \boldsymbol{\chi})$ where the inductive rule for P is

$$R_1 : P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi}) ::= \exists \boldsymbol{X} \exists \boldsymbol{x}.\ \Delta' \wedge E \mapsto \rho * P(Y, \boldsymbol{\gamma}; F, \boldsymbol{\beta}; \boldsymbol{\xi}).$$

We extract the data constraint $\Delta_P(\boldsymbol{\alpha'}, \boldsymbol{\beta'})$ out of R_1. Formally, $\Delta_P(\boldsymbol{\alpha'}, \boldsymbol{\beta'}) := \Delta'[\boldsymbol{\beta'}/\boldsymbol{\gamma'}]$, where $\boldsymbol{\alpha'}$ (resp. $\boldsymbol{\gamma'}$, $\boldsymbol{\beta'}$) is the projection of $\boldsymbol{\alpha}$ (resp. $\boldsymbol{\gamma}$, $\boldsymbol{\beta}$) to data variables. For instance, $\Delta_{ldllseg}(x, x') := (x = x'' + 1)[x'/x''] = (x = x' + 1)$. Note that $\Delta_P(\boldsymbol{\alpha'}, \boldsymbol{\beta'})$ may contain data variables from $\boldsymbol{\xi}$.

Furthermore, by Proposition 2, a Presburger formula $\psi_P(k, \boldsymbol{\alpha'}, \boldsymbol{\beta'})$ where k occurs as a free variable, can be constructed to describe the composition of the relation corresponding to $\Delta_P(\boldsymbol{\alpha'}, \boldsymbol{\beta'})$ for k times. In the running example, $\psi_{ldllseg}(k, x, x') := x = x' + k$.

Proposition 2. *Suppose $P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi}) \in \mathcal{P}$. Then a quantifier free Presburger formula $\psi_P(k, \boldsymbol{\alpha'}, \boldsymbol{\beta'})$ where k occurs as a free variable, can be constructed in linear time to define, for each $k \geqslant 1$, the composition of the relation corresponding to $\Delta_P(\boldsymbol{\alpha'}, \boldsymbol{\beta'})$ for k times.*

As the next step, we define two formulae $\mathsf{Ufld}_1(a_i)$ and $\mathsf{Ufld}_{\geqslant 2}(a_i)$ obtained by unfolding the rule R_1 once and at least twice respectively. For each a_i, we introduce a fresh integer variable k_i. Before the definition of the two formulae, we introduce a notation first.

Definition 1 **($\mathsf{idx}_{(P,\gamma,E)}$).** *Let $P \in \mathcal{P}$ and R_1 be the inductive rule in the definition of P. If in the body of R_1, E occurs in $\boldsymbol{\gamma}$, then we use $\mathsf{idx}_{(P,\gamma,E)}$ to denote the unique index j such that $\gamma_j = E$ (The uniqueness follows from* **C4***).*

We define $\mathsf{Ufld}_1(a_i)$ and $\mathsf{Ufld}_{\geqslant 2}(a_i)$ by distinguishing the following two cases.

– If in the body of R_1, E occurs in $\boldsymbol{\gamma}$, then let

$$\mathsf{Ufld}_1(a_i) := \\ (E = \beta_{\mathsf{idx}_{(P,\gamma,E)}} \wedge k_i = 1 \wedge \psi_P(k_i, \boldsymbol{\alpha'}, \boldsymbol{\beta'}))[Z_1/E, \boldsymbol{\mu}/\boldsymbol{\alpha}, Z_2/F, \boldsymbol{\nu}/\boldsymbol{\beta}, \boldsymbol{\chi}/\boldsymbol{\xi}],$$

and

$$\mathsf{Ufld}_{\geqslant 2}(a_i) := \\ (E \neq \beta_{\mathsf{idx}_{(P,\gamma,E)}} \wedge k_i \geqslant 2 \wedge \psi_P(k_i, \boldsymbol{\alpha'}, \boldsymbol{\beta'}))[Z_1/E, \boldsymbol{\mu}/\boldsymbol{\alpha}, Z_2/F, \boldsymbol{\nu}/\boldsymbol{\beta}, \boldsymbol{\chi}/\boldsymbol{\xi}].$$

– Otherwise, let

$$\mathsf{Ufld}_1(a_i) := (k_i = 1 \wedge \psi_P(k_i, \boldsymbol{\alpha'}, \boldsymbol{\beta'}))[Z_1/E, \boldsymbol{\mu}/\boldsymbol{\alpha}, Z_2/F, \boldsymbol{\nu}/\boldsymbol{\beta}, \boldsymbol{\chi}/\boldsymbol{\xi}],$$

and

$$\mathsf{Ufld}_{\geqslant 2}(a_i) := (k_i \geqslant 2 \wedge \psi_P(k_i, \boldsymbol{\alpha'}, \boldsymbol{\beta'}))[Z_1/E, \boldsymbol{\mu}/\boldsymbol{\alpha}, Z_2/F, \boldsymbol{\nu}/\boldsymbol{\beta}, \boldsymbol{\chi}/\boldsymbol{\xi}].$$

Example 3. Let φ be the formula in Example 2 and a_1 be the (unique) spatial atom in φ. Since the atom $P(X, E, x''; F, L, x')$ occurs in body of the inductive rule of *ldllseg* (where we have $E = \gamma_1$), we deduce that $\mathsf{Ufld}_1(a_1) := E_1 = E_4 \wedge k_1 = 1 \wedge x_1 = x_2 + k_1$ and $\mathsf{Ufld}_{\geqslant 2}(a_1) := E_1 \neq E_4 \wedge k_1 \geqslant 2 \wedge x_1 = x_2 + k_1$.

For each atom $a_i = P(Z_1, \boldsymbol{\mu}; Z_2, \boldsymbol{\nu}; \boldsymbol{\chi})$ in Σ, we introduce a Boolean variable $[Z_1, i]$. Moreover, if in the body of the inductive rule of P, E occurs in $\boldsymbol{\gamma}$, then introduce a Boolean variable $[\nu_{\mathsf{idx}_{(P,\gamma,E)}}, i]$. Let $\mathsf{BVars}(\varphi)$ denote the set of introduced Boolean variables. We define *the abstraction of* φ to be $\mathsf{Abs}(\varphi) ::= \Pi \wedge \Delta \wedge \phi_\Sigma \wedge \phi_*$ over $\mathsf{BVars}(\varphi) \cup \{k_i \mid 1 \leqslant i \leqslant n\} \cup \mathsf{Vars}(\varphi)$, where ϕ_Σ and ϕ_* are defined as follows.

– $\phi_\Sigma = \bigwedge\limits_{1 \leqslant i \leqslant n} \mathsf{Abs}(a_i)$ is an abstraction of Σ where
  - if $a_i = E \mapsto \rho$, then $\mathsf{Abs}(a_i) = [E, i]$,
  - if $a_i = P(Z_1, \boldsymbol{\mu}; Z_2, \boldsymbol{\nu}; \boldsymbol{\chi})$ and in the body of the inductive rule of P, E occurs in $\boldsymbol{\gamma}$, then

$$\begin{aligned}\mathsf{Abs}(a_i) = {} & (\neg[Z_1, i] \wedge \neg[\nu_{\mathsf{idx}_{(P,\gamma,E)}}, i] \wedge Z_1 = Z_2 \wedge \boldsymbol{\mu} = \boldsymbol{\nu} \wedge k_i = 0) \vee \\ & ([Z_1, i] \wedge [\nu_{\mathsf{idx}_{(P,\gamma,E)}}, i] \wedge \mathsf{Ufld}_1(P(Z_1, \boldsymbol{\mu}; Z_2, \boldsymbol{\nu}; \boldsymbol{\chi}))) \vee \\ & ([Z_1, i] \wedge [\nu_{\mathsf{idx}_{(P,\gamma,E)}}, i] \wedge \mathsf{Ufld}_{\geqslant 2}(P(Z_1, \boldsymbol{\mu}; Z_2, \boldsymbol{\nu}; \boldsymbol{\chi}))),\end{aligned}$$

- if $a_i = P(Z_1, \boldsymbol{\mu}; Z_2, \boldsymbol{\nu}; \boldsymbol{\chi})$ and in the body of the inductive rule of P, E does not occur in $\boldsymbol{\gamma}$, then

$$\begin{aligned}\mathsf{Abs}(a_i) = &(\neg[Z_1, i] \wedge Z_1 = Z_2 \wedge \boldsymbol{\mu} = \boldsymbol{\nu} \wedge k_i = 0) \vee \\ &([Z_1, i] \wedge \mathsf{Ufld}_1(P(Z_1, \boldsymbol{\mu}; Z_2, \boldsymbol{\nu}; \boldsymbol{\chi}))) \vee \\ &([Z_1, i] \wedge \mathsf{Ufld}_{\geqslant 2}(P(Z_1, \boldsymbol{\mu}; Z_2, \boldsymbol{\nu}; \boldsymbol{\chi}))),\end{aligned}$$

– ϕ_* states the separation constraint of spatial atoms,

$$\phi_* = \bigwedge_{[Z_1,i],[Z_1',j] \in \mathsf{BVars}(\varphi), i \neq j} (Z_1 = Z_1' \wedge [Z_1, i]) \to \neg[Z_1', j].$$

Example 4. Suppose φ is the formula in Example 3. Then

$$\begin{aligned}\mathsf{Abs}(\varphi) = &E_1 = E_4 \wedge x_1 > x_2 + 1 \wedge \\ &((E_1 = E_2 \wedge E_3 = E_4 \wedge x_1 = x_2 \wedge k_1 = 0) \vee \\ &([E_1, 1] \wedge [E_4, 1] \wedge E_1 = E_4 \wedge k_1 = 1 \wedge x_1 = x_2 + k_1) \vee \\ &([E_1, 1] \wedge [E_4, 1] \wedge E_1 \neq E_4 \wedge k_1 \geqslant 2 \wedge x_1 = x_2 + k_1)).\end{aligned}$$

It is easy to see that $\mathsf{Abs}(\varphi)$ is unsatisfiable.

Proposition 3. *For each* $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ *formula* φ*,* φ *is satisfiable iff* $\mathsf{Abs}(\varphi)$ *is satisfiable.*

The satisfiability of $\mathsf{Abs}(\varphi)$ can be discharged by the state-of-the-art SMT solvers, e.g., Z3. It is well known that the satisfiability of the quantifier-free presburger arithmetic formulae can be decided in NP. Hence we have:

Theorem 1. *The satisfiability problem of* $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ *is in* NP*.*

Note that the problem whether the satisfiability problem of $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ is NP-hard is open.

4 Entailment

In this section, we present a complete decision procedure for the entailment problem $\varphi \models \psi$, where φ, ψ are two $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ formulae. We assume, without loss of generality, that $\mathsf{Vars}(\psi) \subseteq \mathsf{Vars}(\varphi)$, both φ and ψ are satisfiable, and $\mathrm{Flds}(\varphi) = \mathrm{Flds}(\psi)$.

On a high level, the decision procedure is similar to that in [11]. Loosely speaking, we construct graph representations $\mathcal{G}_\varphi$ and $\mathcal{G}_\psi$ of φ and ψ respectively and reduce the entailment problem to (a variant of) the graph homomorphism problem from $\mathcal{G}_\psi$ to $\mathcal{G}_\varphi$. However, our decision procedure is considerably more involved due to the additional expressibility of the logic and the non-intuitionistic semantics.

Recall that, in the previous section, from an $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ formula φ one can construct an abstraction $\mathsf{Abs}(\varphi)$. Let $\sim_\varphi$ denote the equivalence relation defined over $\mathsf{LVars}(\varphi)$ as follows: For $X, Y \in \mathsf{LVars}(\varphi)$, $X \sim_\varphi Y$ iff $\mathsf{Abs}(\varphi) \models X = Y$. For $X \in \mathsf{LVars}(\varphi)$, let $[X]_\varphi$ denote the equivalence class of X under $\sim_\varphi$.

4.1 Graph Representations of $\mathsf{SLID_{LC}}[\mathcal{P}]$ Formulae

For a satisfiable $\mathsf{SLID_{LC}}[\mathcal{P}]$ formula φ, we will construct a graph $\mathcal{G}_\varphi$ from φ. Without loss of generality, we assume that φ contains at least one points-to atom or predicate atom.

Assume $\varphi = \Pi \wedge \Delta \wedge \Sigma$ with $\Sigma = a_1 * \ldots * a_n$ $(n \geqslant 1)$, and f_0 denotes the principal location field of φ. (Recall the "uniqueness of predicates" assumption for $\mathsf{SLID_{LC}}[\mathcal{P}]$ formulae in Sect. 2.)

We construct a directed *multigraph* (i.e., a directed graph with parallel arcs) $\mathcal{G}_\varphi = (\mathcal{V}_\varphi, \mathcal{R}_\varphi, \mathcal{L}_\varphi)$:

- $\mathcal{V}_\varphi = \{[E] \mid E \in \mathsf{LVars}(\varphi)\}$, where we use $[E]$ as an abbreviation of $[E]_\varphi$, that is, the equivalence class of $\sim_\varphi$ containing E.
- $\mathcal{R}_\varphi$ is the set of arcs and $\mathcal{L}_\varphi$ is the arc-labeling function, defined as follows:
  - for each pair of location variables (E, F) such that Σ contains a *points-to atom* $a_i = E \mapsto \rho$ and (f_0, F) occurs in ρ for $f_0 \in \mathbb{L}$, there is an arc from $[E]$ to $[F]$ labeled by $f_0[\rho']$, where ρ' is obtained by removing (f_0, F) from ρ — this arc e is said to be *field-labeled* and we write $\mathcal{L}_\varphi(e) = f_0[\rho']$;
  - for each pair of location variables (E, F) such that Σ contains a *predicate atom* $a_i = P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$ and $\mathsf{Abs}(\varphi) \not\models \neg[E, i]$, there is an arc from $[E]$ to $[F]$ labeled by $P(\boldsymbol{\alpha}; \boldsymbol{\beta}; \boldsymbol{\xi})$ — this arc e is said to be *predicate-labeled* and we write $\mathcal{L}_\varphi(e) = P(\boldsymbol{\alpha}; \boldsymbol{\beta}; \boldsymbol{\xi})$.

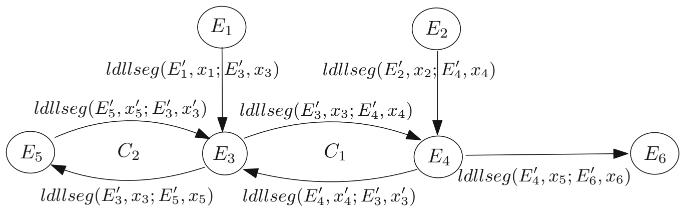
From the construction, each field-labeled or predicate-labeled arc e corresponds to an unique atom a_i in Σ. Let $i(e)$ denote the index i of the atom.

Example 5. Let

$$\begin{aligned} \varphi = {} & \underbrace{ldllseg(E_1, E'_1, x_1; E_3, E'_3, x_3)}_{a_1} * \underbrace{ldllseg(E_2, E'_2, x_2; E_4, E'_4, x_4)}_{a_2} * \\ & \underbrace{ldllseg(E_3, E'_3, x_3; E_4, E'_4, x_4)}_{a_3} * \underbrace{ldllseg(E_4, E'_4, x'_4; E_3, E'_3, x'_3)}_{a_4} * \\ & \underbrace{ldllseg(E_3, E'_3, x_3; E_5, E'_5, x_5)}_{a_5} * \underbrace{ldllseg(E_5, E'_5, x'_5; E_3, E'_3, x'_3)}_{a_6} * \\ & \underbrace{ldllseg(E_4, E'_4, x_5; E_6, E'_6, x_6)}_{a_7}. \end{aligned}$$

The graph $\mathcal{G}_\varphi$ is as illustrated in Fig. 1, where each equivalence class of $\sim_\varphi$ is a singleton and $\mathcal{V}_\varphi = \{[E_1], \ldots, [E_6], [E'_1], \ldots, [E'_6]\}$. Note that there are no arcs between the nodes $[E'_1], \ldots, [E'_6]$.

We use standard graph-theoretic notions, for instance, paths, connected components (CCs) and strongly connected components (SCCs). In particular, a path in $\mathcal{G}_\varphi$ is a (possibly empty) sequence of consecutive arcs in $\mathcal{G}_\varphi$. If there is a path from $[E]$ to $[F]$, then $[F]$ is said to be *reachable* from $[E]$ and $[E]$ is said to be an *ancestor* of $[F]$. For a node $[E]$ and an arc e with source node $[E']$, e is said



E_1
E_2
E_3
E_4
E_5
E_6
C_1
C_2
$ldllseg(E_1', x_1; E_3', x_3)$
$ldllseg(E_2', x_2; E_4', x_4)$
$ldllseg(E_5', x_5'; E_3', x_3')$
$ldllseg(E_3', x_3; E_4', x_4)$
$ldllseg(E_4', x_5; E_6', x_6)$
$ldllseg(E_3', x_3; E_5', x_5)$
$ldllseg(E_4', x_4'; E_3', x_3')$

Fig. 1. The graph $\mathcal{G}_\varphi$

to be *reachable* from $[E]$ if $[E']$ is reachable from $[E]$. A CC or SCC $\mathcal{C}$ of $\mathcal{G}_\varphi$ is said to be *nontrivial* if $\mathcal{C}$ contains at least one arc.

We shall reveal some structural properties of the graph $\mathcal{G}_\varphi$.

Proposition 4. *The graph $\mathcal{G}_\varphi$ satisfies the following properties:*

1. *If there is a field-labeled arc out of $[E]$, then there are no predicate-labeled arcs out of $[E]$.*
2. *For each pair of distinct nodes $[E]$ and $[F]$ in $\mathcal{G}_\varphi$, there is at most one simple path from $[E]$ to $[F]$ in $\mathcal{G}_\varphi$.*

Proposition 5. *Each nontrivial SCC $\mathcal{S}$ satisfies the following constraints.*

– *Each pair of different simple cycles in $\mathcal{S}$ share at most one node — The set of shared nodes is called the set of cut nodes of $\mathcal{S}$, denoted by $\mathsf{Cut}(\mathcal{S})$. Here by "different", we mean that the two sets of arcs in the two cycles are different.*
– *The collection of simple cycles in $\mathcal{S}$ is organised into a tree. More precisely, let $\{C_1, \ldots, C_n\}$ be the set of all the simple cycles in $\mathcal{S}$ and $\mathcal{T}_\mathcal{S} = (\{C_1, \ldots, C_n\}, \mathsf{Cut}(\mathcal{S}), \mathcal{R})$ be the undirected bipartite graph such that for each $i : 1 \leqslant i \leqslant n$, $\{C_i, [E]\} \in \mathcal{R}$ iff $[E] \in \mathsf{Cut}(\mathcal{S}) \cap C_i$. Then $\mathcal{T}_\mathcal{S}$ is a tree.*

Example 6. The graph $\mathcal{G}_\varphi$ in Fig. 1 has just one nontrivial SCC $\mathcal{S}$ comprising the nodes $[E_3], [E_4], [E_5]$. The graph $\mathcal{T}_\mathcal{S} = (\{C_1, C_2\}, \{[E_3]\}, \{\{C_1, [E_3]\}, \{C_2, [E_3]\}\})$ is a tree.

4.2 Entailment Checking by Graph Homomorphisms

As a starting point, we illustrate how a path in $\mathcal{G}_\varphi$ is matchable to an arc in $\mathcal{G}_\psi$, which is the basis of our decision procedure.

Definition 2. *Given an arc e from $[E]_\psi$ to $[F]_\psi$ with label $P'(\boldsymbol{\alpha'}; \boldsymbol{\beta'}; \boldsymbol{\xi'})$ in $\mathcal{G}_\psi$, a (possibly empty) path $\pi = [E_0]_\varphi [E_1]_\varphi \ldots [E_n]_\varphi$ from $[E]_\varphi$ to $[F]_\varphi$ in $\mathcal{G}_\varphi$ is said to be* matchable *to e wrt. $\mathsf{Abs}(\varphi)$ if (1) either π is empty and $\mathsf{Abs}(\varphi) \models E = F \wedge \boldsymbol{\alpha'} = \boldsymbol{\beta'}$, (2) or π is nonempty and there are $\boldsymbol{\alpha'_0}, \boldsymbol{\alpha'_1}, \ldots, \boldsymbol{\alpha'_n}$ such that $\boldsymbol{\alpha'_0} = \boldsymbol{\alpha'}$, $\boldsymbol{\alpha'_n} = \boldsymbol{\beta'}$, and for each $i : 1 \leqslant i \leqslant n$, the arc from $[E_{i-1}]_\varphi$ to $[E_i]_\varphi$ in π is*

– *either a field-labeled arc with the label $f_0[\rho']$ such that* $\mathsf{Abs}(\varphi) \wedge E_{i-1} \mapsto \rho \models P'(E_{i-1}, \boldsymbol{\alpha'_{i-1}}; E_i, \boldsymbol{\alpha'_i}; \boldsymbol{\xi'})$*, where ρ is obtained from ρ' by adding (f_0, E_i);*
– *or a predicate-labeled arc with the label $P(\boldsymbol{\alpha}; \boldsymbol{\beta}; \boldsymbol{\xi})$ such that*

$$\mathsf{Abs}(\varphi) \wedge P(E_{i-1}, \boldsymbol{\alpha}; E_i, \boldsymbol{\beta}; \boldsymbol{\xi}) \models P'(E_{i-1}, \boldsymbol{\alpha'_{i-1}}; E_i, \boldsymbol{\alpha'_i}; \boldsymbol{\xi'}).$$

Note that in the above definition, we abuse the notation slightly, since $\mathsf{Abs}(\varphi)$ may contain Boolean variables $[E', j]$, the integer variables k_j, and disjunctions, thus strictly speaking, $\mathsf{Abs}(\varphi) \wedge E_{i-1} \mapsto \rho$ and $\mathsf{Abs}(\varphi) \wedge P(E_{i-1}, \boldsymbol{\alpha}; E_i, \boldsymbol{\beta}; \boldsymbol{\xi})$ are not $\mathsf{SLID_{LC}}[\mathcal{P}]$ formulae.

Example 7. Let φ be the formula in Example 5 and $\psi = \mathit{dllseg}(E_1, E'_1; E_6, E'_6) * \mathit{dllseg}(E_2, E'_2; E_4, E'_4)$. Then the path $[E_1]_\varphi[E_3]_\varphi[E_4]_\varphi[E_6]_\varphi$ in $\mathcal{G}_\varphi$ is matchable to the arc e from $[E_1]_\psi$ to $[E_6]_\psi$ with the label $\mathit{dllseg}(E'_1; E'_6)$ in $\mathcal{G}_\psi$. More specifically, there are $\boldsymbol{\alpha'_0} = E'_1$, $\boldsymbol{\alpha'_1} = E'_3$, $\boldsymbol{\alpha'_2} = E'_4$, and $\boldsymbol{\alpha'_3} = E'_6$ such that

$$\begin{aligned}
\mathsf{Abs}(\varphi) \wedge \mathit{ldllseg}(E_1, E'_1, x_1; E_3, E'_3, x_3) &\models \mathit{dllseg}(E_1, E'_1; E_3, E'_3),\\
\mathsf{Abs}(\varphi) \wedge \mathit{ldllseg}(E_3, E'_3, x_3; E_4, E'_4, x_4) &\models \mathit{dllseg}(E_3, E'_3; E_4, E'_4),\\
\mathsf{Abs}(\varphi) \wedge \mathit{ldllseg}(E_4, E'_4, x_5; E_6, E'_6, x_6) &\models \mathit{dllseg}(E_4, E'_4; E_6, E'_6).
\end{aligned}$$

Proposition 6. *Suppose φ is an* $\mathsf{SLID_{LC}}[\mathcal{P}]$ *formula, $a = E \mapsto \rho$ or $a = P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$ is a spatial atom in φ, and $P'(E, \boldsymbol{\alpha'}; F, \boldsymbol{\beta'}; \boldsymbol{\xi'})$ is a predicate atom (not necessarily in φ) such that* $\mathsf{Vars}(P'(E, \boldsymbol{\alpha'}; F, \boldsymbol{\beta'}; \boldsymbol{\xi'})) \subseteq \mathsf{Vars}(\varphi)$*. Then (1) the entailment problem* $\mathsf{Abs}(\varphi) \wedge a \models P'(E, \boldsymbol{\alpha'}; F, \boldsymbol{\beta'}; \boldsymbol{\xi'})$ *is in Δ_2^p; and (2) if there exist $\boldsymbol{\alpha'}, \boldsymbol{\alpha''}$ such that* $\mathsf{Abs}(\varphi) \wedge a \models P'(E, \boldsymbol{\alpha'}; F, \boldsymbol{\beta'}; \boldsymbol{\xi'})$ *and* $\mathsf{Abs}(\varphi) \wedge a \models P'(E, \boldsymbol{\alpha''}; F, \boldsymbol{\beta'}; \boldsymbol{\xi'})$*, then* $\mathsf{Abs}(\varphi) \models \boldsymbol{\alpha'} = \boldsymbol{\alpha''}$*. Such an unique $\boldsymbol{\alpha'}$ can be computed effectively from* $\mathsf{Abs}(\varphi)$*, the atom a, $P'(E, -; F, \boldsymbol{\beta'}; \boldsymbol{\xi'})$, and the inductive definition of P and P'.*

The complexity upper bound Δ_2^p in Proposition 6 follows from the fact that, to solve $\mathsf{Abs}(\varphi) \wedge a \models P'(E, \boldsymbol{\alpha'}; F, \boldsymbol{\beta'}; \boldsymbol{\xi'})$, it is necessary to use an oracle to decide the satisfiability of quantifier-free Presburger formulae, which is in NP. The uniqueness of $\boldsymbol{\alpha'}$ in Proposition 6 is guaranteed by the constraints **C2**, **C3**, and **C5** in the inductive definition of predicates.

Proposition 6 shows that Definition 2 is effective, namely,

Proposition 7. *Check whether a path π in $\mathcal{G}_\varphi$ is matchable to a predicate-labeled arc e in $\mathcal{G}_\psi$ can be done in Δ_2^p.*

We are ready to present the decision procedure. We will introduce a concept of allocating plans $\mathcal{AP}$ (cf. Definition 5), which are the pairs $(\mathsf{Abs}_{\mathcal{AP}}[\varphi], \mathcal{G}_{\mathcal{AP}}[\varphi])$, where $\mathsf{Abs}_{\mathcal{AP}}[\varphi]$ is a formula obtained from $\mathsf{Abs}(\varphi)$, and $\mathcal{G}_{\mathcal{AP}}[\varphi]$ is a simplification of $\mathcal{G}_\varphi$. The entailment problem is reduced to checking the existence of a homomorphism from $(\mathsf{Abs}(\psi), \mathcal{G}_\psi)$ to $(\mathsf{Abs}_{\mathcal{AP}}[\varphi], \mathcal{G}_{\mathcal{AP}}[\varphi])$, for each allocating plan $\mathcal{AP}$. For each CC $\mathcal{C}$ of $\mathcal{G}_\varphi$, $\mathsf{Cyc}_\mathcal{C}$ denotes the set of simple cycles in $\mathcal{C}$ and $\mathsf{NScc}_\mathcal{C}$ denotes the set of nontrivial SCCs in $\mathcal{C}$. For $i \in \mathbb{N}$, let $[i] = \{1, \ldots, i\}$.

Definition 3 (Allocating pseudo-plans). *Let $\mathcal{C}_1, \ldots, \mathcal{C}_k$ be an enumeration of the nontrivial CCs of $\mathcal{G}_\varphi$, and for each $i \in [k]$, $\mathsf{Cyc}_{\mathcal{C}_i} = \{C_{i,1}, \ldots, C_{i,l_i}\}$ (where $l_i \geqslant 0$). Then an allocating pseudo-plan Ω for $\mathcal{G}_\varphi$ is a function such that $\Omega(i) \in \{0\} \cup [l_i]$ for each $i \in [k]$.*

Intuitively, $\Omega(i) \in [l_i]$ means that some arc in the simple cycle $C_{i,\Omega(i)}$ is assigned to be an nonempty heap, and accordingly, $\Omega(i) = 0$ means that all arcs in nontrivial SCCs of $\mathcal{C}_i$ are assigned to be empty heaps (cf. Definition 4).

For each arc e with $a_{i(e)} = P(E, \boldsymbol{\alpha}; F, \boldsymbol{\beta}; \boldsymbol{\xi})$, we use ϕ_e to denote $[E, i(e)]$.

Definition 4 ($\Omega[\mathsf{Abs}(\varphi)]$ and feasible allocating pseudo-plans). *Let Ω be an allocating pseudo-plan of $\mathcal{G}_\varphi$. We define $\Omega[\mathsf{Abs}(\varphi)] := \mathsf{Abs}(\varphi) \wedge \bigwedge_{i\in[k]} \zeta_i$, where for each $i \in [k]$, $\zeta_i := \bigvee\limits_{e \in C_{i,\Omega(i)}} \phi_e$ if $\Omega(i) \neq 0$; and $\zeta_i := \bigwedge\limits_{\mathcal{S} \in \mathsf{NScc}_{\mathcal{C}_i}} \bigwedge\limits_{e \in \mathcal{S}} \neg\phi_e$ if $\Omega(i) = 0$. An allocating pseudo-plan Ω is* feasible *if $\Omega[\mathsf{Abs}(\varphi)]$ is satisfiable.*

For an allocating pseudo-plan Ω of $\mathcal{G}_\varphi$, we construct a graph $\Omega[\mathcal{G}_\varphi] = (\mathcal{V}_\Omega, \mathcal{R}_\Omega, \mathcal{L}_\Omega)$ from φ, similarly to $\mathcal{G}_\varphi$, with $\sim_\varphi$ replaced by $\sim_\Omega$ (on $\mathsf{LVars}(\varphi)$) defined as follows: $E \sim_\Omega F$ iff $\Omega[\mathsf{Abs}(\varphi)] \models E = F$.

A directed graph $\mathcal{G}$ is said to be *DAG-like* (DAG: directed acyclic graph) if for each CC $\mathcal{C}$ of $\mathcal{G}$, either $\mathcal{C}$ is a DAG, or $\mathcal{C}$ contains exactly one simple cycle C which is reachable from every node in $\mathcal{C} \setminus C$.

Definition 5 (Allocating plans $\mathcal{AP}$). *Given a formula φ, an allocating plan $\mathcal{AP} = (\mathsf{Abs}_{\mathcal{AP}}[\varphi], \mathcal{G}_{\mathcal{AP}}[\varphi])$ of φ is obtained from $\mathcal{G}_\varphi$ by a sequence of allocating pseudo-plans $\Omega_1, \ldots, \Omega_n$ ($n \geqslant 0$) such that: (1) $\phi_0 = \mathsf{Abs}(\varphi)$, $\mathcal{G}_0 = \mathcal{G}_\varphi$; for each $i : 1 \leqslant i \leqslant n$, (2) Ω_i is a feasible allocating pseudo-plan of $\mathcal{G}_{i-1}$, $\phi_i = \Omega_i[\phi_{i-1}]$, $\mathcal{G}_i = \Omega_i[\mathcal{G}_{i-1}]$; (3) $\mathsf{Abs}_{\mathcal{AP}}[\varphi] = \phi_n$, $\mathcal{G}_{\mathcal{AP}}[\varphi] = \mathcal{G}_n$, and $\mathcal{G}_{\mathcal{AP}}[\varphi]$ is DAG-like.*

For an allocating plan $\mathcal{AP}$ of φ, we use $\Sigma_{\mathcal{AP}}[\varphi]$ to denote the spatial formula corresponding to $\mathcal{G}_{\mathcal{AP}}[\varphi]$. In addition, let $\varphi_{\mathcal{AP}} = \mathsf{Abs}_{\mathcal{AP}}[\varphi] \wedge \Sigma_{\mathcal{AP}}[\varphi]$.

Example 8. Let φ be the formula in Example 5. The graph $\mathcal{G}_\varphi$ contains exactly one nontrivial connected component $\mathcal{C}_1$ (cf. Fig. 1). In addition, suppose Ω_1 and Ω_2 are the allocating pseudo-plans such that $\Omega_1(1) = 1$ and $\Omega_2(1) = 0$. Then $(\Omega_1[\mathsf{Abs}(\varphi)], \Omega_1[\mathcal{G}_\varphi])$ and $(\Omega_2[\mathsf{Abs}(\varphi)], \Omega_2[\mathcal{G}_\varphi])$ are illustrated in Fig. 2. Since both $\Omega_1[\mathcal{G}_\varphi]$ and $\Omega_2[\mathcal{G}_\varphi]$ are DAG-like, we know that $(\Omega_1[\mathsf{Abs}(\varphi)], \Omega_1[\mathcal{G}_\varphi])$ and $(\Omega_2[\mathsf{Abs}(\varphi)], \Omega_2[\mathcal{G}_\varphi])$ are both allocating plans.

Lemma 1. *Let φ, ψ be two $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ formulae such that $\mathsf{Vars}(\psi) \subseteq \mathsf{Vars}(\varphi)$. Then $\varphi \models \psi$ iff the following two conditions hold.*

- $\mathsf{Abs}(\varphi) \models \exists \boldsymbol{Z}.\mathsf{Abs}(\psi)$, *where $\boldsymbol{Z} = \mathsf{Vars}(\mathsf{Abs}(\psi)) \setminus \mathsf{Var}(\psi)$, i.e., the set of additional variables introduced when constructing $\mathsf{Abs}(\psi)$ from ψ.*
- *For each allocating plan $\mathcal{AP}$ of φ, $\varphi_{\mathcal{AP}} \models \psi$.*

By Lemma 1, the entailment problem $\varphi \models \psi$ can be reduced to checking $\varphi_{\mathcal{AP}} \models \psi$ for each allocating plan $\mathcal{AP}$, which we now show that can be further reduced to checking the existence of a (graph) homomorphism from $(\mathsf{Abs}(\psi), \mathcal{G}_\psi)$ to $(\mathsf{Abs}_{\mathcal{AP}}[\varphi], \mathcal{G}_{\mathcal{AP}}[\varphi])$.

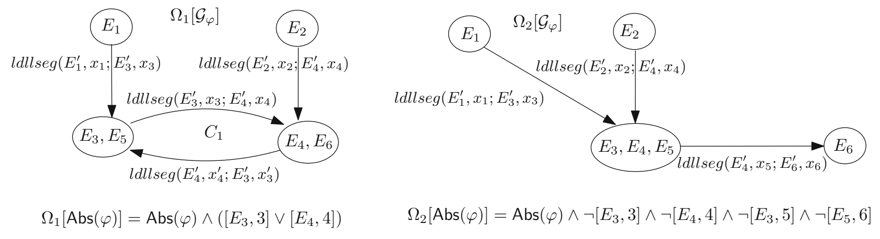

$\Omega_1[\mathcal{G}_\varphi]$
E_1
E_2
$ldllseg(E_1', x_1; E_3', x_3)$
$ldllseg(E_2', x_2; E_4', x_4)$
$ldllseg(E_3', x_3; E_4', x_4)$
E_3, E_5
C_1
E_4, E_6
$ldllseg(E_4', x_4'; E_3', x_3')$
$\Omega_1[\mathsf{Abs}(\varphi)] = \mathsf{Abs}(\varphi) \wedge ([E_3, 3] \vee [E_4, 4])$
$\Omega_2[\mathcal{G}_\varphi]$
E_1
E_2
$ldllseg(E_2', x_2; E_4', x_4)$
$ldllseg(E_1', x_1; E_3', x_3)$
E_3, E_4, E_5
E_6
$ldllseg(E_4', x_5; E_6', x_6)$
$\Omega_2[\mathsf{Abs}(\varphi)] = \mathsf{Abs}(\varphi) \wedge \neg[E_3, 3] \wedge \neg[E_4, 4] \wedge \neg[E_3, 5] \wedge \neg[E_5, 6]$

Fig. 2. $(\Omega_1[\mathsf{Abs}(\varphi)], \Omega_1[\mathcal{G}_\varphi])$ and $(\Omega_2[\mathsf{Abs}(\varphi)], \Omega_2[\mathcal{G}_\varphi])$

Definition 6 (Homomorphisms). *Let $\mathcal{AP}$ be an allocating plan of φ, $\mathcal{G}_{\mathcal{AP}}[\varphi] = (\mathcal{V}_{\mathcal{AP}}, \mathcal{R}_{\mathcal{AP}}, \mathcal{L}_{\mathcal{AP}})$, and $\mathcal{G}_\psi = (\mathcal{V}_\psi, \mathcal{R}_\psi, \mathcal{L}_\psi)$. Then a* homomorphism *from $(\mathsf{Abs}(\psi), \mathcal{G}_\psi)$ to $(\mathsf{Abs}_{\mathcal{AP}}[\varphi], \mathcal{G}_{\mathcal{AP}}[\varphi])$ is a pair of functions (θ, η) where θ is from $\mathcal{V}_\psi$ to $\mathcal{V}_{\mathcal{AP}}$ and η is from $\mathcal{R}_\psi$ to the set of paths in $\mathcal{G}_{\mathcal{AP}}[\varphi]$ satisfying the following constraints.*

- **Variable subsumption:** *For each node $[E] \in \mathcal{V}_\psi$, $[E] \subseteq \theta([E])$.*
- **Field-labeled arcs:** *For each field-labeled arc e from $[E]$ to $[F]$ in $\mathcal{G}_\psi$, $\eta(e)$ is a field-labeled arc from $\theta([E])$ to $\theta([F])$ in $\mathcal{G}_{\mathcal{AP}}[\varphi]$.*
- **Predicate-labeled arcs:** *For each predicate-labeled arc e from $[E]$ to $[F]$ in $\mathcal{G}_\psi$, both $\theta([E])$ and $\theta([F])$ must be in some CC $\mathcal{C}$, and the following conditions are satisfied.*
  - *If $\mathcal{C}$ is a DAG, then*
    - *if $\theta([E]) \neq \theta([F])$, then $\eta(e)$ is the unique simple path from $\theta([E])$ to $\theta([F])$ in $\mathcal{G}_{\mathcal{AP}}[\varphi]$,*
    - *otherwise, $\eta(e)$ is the empty path from $\theta([E])$ to $\theta([F])$.*
  - *Otherwise, let C be the unique simple cycle in $\mathcal{C}$.*
    - *If $\theta([E]) \neq \theta([F])$, moreover, the unique simple path from $\theta([E])$ to $\theta([F])$ in $\mathcal{C}$ is either node-disjoint from C, or contains at least two nodes in C, then $\eta(e)$ is the unique simple path from $\theta([E])$ to $\theta([F])$ in $\mathcal{C}$.*
    - *If $\theta([E]) \neq \theta([F])$, moreover, the unique simple path from $\theta([E])$ to $\theta([F])$ in $\mathcal{C}$ contains exactly one node in C (i.e. $\theta([F])$), then $\eta(e)$ is either the unique simple path from $\theta([E])$ to $\theta([F])$ or the composition of the unique simple path from $\theta([E])$ to $\theta([F])$ and the cycle C.*
    - *If $\theta([E]) = \theta([F])$ and $\theta([F])$ belongs to C, then $\eta(e)$ is either the empty path or the simple cycle C from $\theta([E])$ to $\theta([F])$.*
    - *If $\theta([E]) = \theta([F])$ and $\theta([F])$ does not belong to C, then $\eta(e)$ is the empty path.*
- **Matching of paths to arcs:** *For each arc e in $\mathcal{G}_\psi$, $\eta(e)$ is matchable to e wrt. $\mathsf{Abs}_{\mathcal{AP}}[\varphi]$.*
- **Separation constraint:** *For each pair of distinct arcs e_1, e_2 in $\mathcal{G}_\psi$, $\eta(e_1)$ and $\eta(e_2)$ are arc-disjoint.*
- **Coverage of allarcs in** $\mathcal{G}_{\mathcal{AP}}[\varphi]$**:** *Each arc of $\mathcal{G}_{\mathcal{AP}}[\varphi]$ occurs in $\eta(e)$ for some arc e in $\mathcal{G}_\psi$.*

Lemma 2. *Let φ, ψ be two formulae satisfying the premise of Lemma 1. Then for each allocating plan $\mathcal{AP}$ of $\mathcal{G}_\varphi$, $\varphi_{\mathcal{AP}} \models \psi$ iff there is a homomorphism from $(\mathsf{Abs}(\psi), \mathcal{G}_\psi)$ to $(\mathsf{Abs}_{\mathcal{AP}}[\varphi], \mathcal{G}_{\mathcal{AP}}[\varphi])$.*

Theorem 2. *The entailment problem of $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$ formulae is in Π_3^{P}.*

Complexity Analysis: Deciding whether there exists a homomorphism from $(\mathsf{Abs}(\psi), \mathcal{G}_\psi)$ to $(\mathsf{Abs}_{\mathcal{AP}}[\varphi], \mathcal{G}_{\mathcal{AP}}[\varphi])$ can be done in Σ_2^{P}, by Proposition 7 and guessing a homomorphism (θ, η) in Definition 6. Furthermore, by Lemma 2, $\varphi \not\models \psi$ iff either $\mathsf{Abs}(\varphi) \neq \exists \boldsymbol{Z}.\mathsf{Abs}(\psi)$ (cf. Lemma 1), or there is an allocating plan $\mathcal{AP}$ such that there is no homomorphism from $(\mathsf{Abs}(\psi), \mathcal{G}_\psi)$ to $(\mathsf{Abs}_{\mathcal{AP}}[\varphi], \mathcal{G}_{\mathcal{AP}}[\varphi])$. Hence, deciding $\varphi \not\models \psi$ is in $\mathrm{NP}^{\Pi_2^{\mathrm{P}}} = \Sigma_3^{\mathrm{P}}$. We conclude that the entailment problem is in Π_3^{P}.

5 Conclusion

In this paper, we have defined $\mathsf{SLID}_{\mathsf{LC}}[\mathcal{P}]$, a linearly compositional fragment of separation logic with inductive definitions, where both linear shapes, e.g., singly or doubly linked lists, lists with tail pointers, and data constraints, e.g., sortedness and size constraints, are expressible. We have provided complete decision procedures for both the satisfiability and the entailment problem, with complexity upper-bounds NP and Π_3^{P} respectively. For the satisfiability problem, it turned out that computing the transitive closure of data constraints is critical to the completeness of the decision procedure. For the entailment checking, a novel concept of allocating plans was introduced. Note that we made no efforts to tighten the NP/Π_3^{P} upper-bound or to provide lower-bounds, which might be interesting subjects of further research. More importantly, we believe that the approach introduced in this paper is amenable to implementations and can be extended to handle non-linear shapes (e.g., nested lists, binary search trees) as well as other kinds of data constraints (e.g., set or multiset constraints). These are left as future work.

References

1. Antonopoulos, T., Gorogiannis, N., Haase, C., Kanovich, M., Ouaknine, J.: Foundations for decision problems in separation logic with general inductive predicates. In: Muscholl, A. (ed.) FOSSACS 2014 (ETAPS). LNCS, vol. 8412, pp. 411–425. Springer, Heidelberg (2014)
2. Berdine, J., Calcagno, C., O'Hearn, P.W.: Symbolic execution with separation logic. In: Yi, K. (ed.) APLAS 2005. LNCS, vol. 3780, pp. 52–68. Springer, Heidelberg (2005)
3. Bouajjani, A., Drăgoi, C., Enea, C., Sighireanu, M.: Accurate invariant checking for programs manipulating lists and arrays with infinite data. In: Chakraborty, S., Mukund, M. (eds.) ATVA 2012. LNCS, vol. 7561, pp. 167–182. Springer, Heidelberg (2012)

4. Bozga, M., Iosif, R., Perarnau, S.: Quantitative separation logic and programs with lists. J. Autom. Reasoning **45**(2), 131–156 (2010)
5. Brochenin, R., Demri, S., Lozes, É.: On the almighty wand. Inf. Comput. **211**, 106–137 (2012)
6. Brotherston, J., Distefano, D., Petersen, R.L.: Automated cyclic entailment proofs in separation logic. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS, vol. 6803, pp. 131–146. Springer, Heidelberg (2011)
7. Brotherston, J., Fuhs, C., Perez, J.A.N., Gorogiannis, N.: A decision procedure for satisfiability in separation logic with inductive predicates. In: LICS (2014)
8. Calcagno, C., Distefano, D.: Infer: an automatic program verifier for memory safety of C programs. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 459–465. Springer, Heidelberg (2011)
9. Chin, W.-N., David, C., Nguyen, H.H., Qin, S.: Automated verification of shape, size and bag properties via user-defined predicates in separation logic. Sci. Comput. Program. **77**(9), 1006–1036 (2012)
10. Chu, D., Jaffar, J., Trinh, M.: Automating proofs of data-structure properties in imperative programs. CoRR, abs/1407.6124 (2014)
11. Cook, B., Haase, C., Ouaknine, J., Parkinson, M., Worrell, J.: Tractable reasoning in a fragment of separation logic. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 235–249. Springer, Heidelberg (2011)
12. Demri, S., Deters, M.: Expressive completeness of separation logic with two variables and no separating conjunction. In: CSL-LICS, p. 37 (2014)
13. Enea, C., Lengál, O., Sighireanu, M., Vojnar, T.: Compositional entailment checking for a fragment of separation logic. In: Garrigue, J. (ed.) APLAS 2014. LNCS, vol. 8858, pp. 314–333. Springer, Heidelberg (2014)
14. Enea, C., Sighireanu, M., Wu, Z.: On automated lemma generation for separation logic with inductive definitions. ATVA 2015. LNCS, vol. 9364, pp. 80–96. Springer, Heidelberg (2015). doi:10.1007/978-3-319-24953-7_7
15. Hou, Z., Goré, R., Tiu, A.: Automated theorem proving for assertions in separation logic with all connectives. In: Felty, A.P., Middeldorp, A. (eds.) CADE-25. LNCS(LNAI), vol. 9195, pp. 501–516. Springer, Heidelberg (2015)
16. Iosif, R., Rogalewicz, A., Simacek, J.: The tree width of separation logic with recursive definitions. In: Bonacina, M.P. (ed.) CADE 2013. lncs, vol. 7898, pp. 21–38. Springer, Heidelberg (2013)
17. Iosif, R., Rogalewicz, A., Vojnar, T.: Deciding entailments in inductive separation logic with tree automata. In: Cassez, F., Raskin, J.-F. (eds.) ATVA 2014. LNCS, vol. 8837, pp. 201–218. Springer, Heidelberg (2014)
18. O'Hearn, P.W., Reynolds, J.C., Yang, H.: Local reasoning about programs that alter data structures. In: Fribourg, L. (ed.) CSL 2001 and EACSL 2001. LNCS, vol. 2142, p. 1. Springer, Heidelberg (2001)
19. Pek, E., Qiu, X., Madhusudan, P.: Natural proofs for data structure manipulation in C using separation logic. In: PLDI, pp. 440–451. ACM (2014)
20. Piskac, R., Wies, T., Zufferey, D.: Automating separation logic using SMT. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 773–789. Springer, Heidelberg (2013)
21. Piskac, R., Wies, T., Zufferey, D.: Automating separation logic with trees and data. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 711–728. Springer, Heidelberg (2014)
22. Piskac, R., Wies, T., Zufferey, D.: GRASShopper - complete heap verification with mixed specifications. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014 (ETAPS). LNCS, vol. 8413, pp. 124–139. Springer, Heidelberg (2014)

23. Qiu, X., Garg, P., Stefănescu, A., Madhusudan, P.: Naturalproofs for structure, data, and separation. In: PLDI, pp. 231–242. ACM (2013)
24. Reynolds, J.C.: Separation logic: a logic for shared mutable datastructures. In: LICS, pp. 55–74. ACM (2002)
25. Z3. http://rise4fun.com/z3