Reasoning in the Bernays-Schönfinkel-Ramsey Fragment of Separation Logic

Andrew Reynolds[1], Radu Iosif[2](✉), and Cristina Serban[2]

[1] The University of Iowa, Iowa City, USA
[2] Verimag/CNRS/Université de Grenoble Alpes, Grenoble, France
iosif@imag.fr

Abstract. Separation Logic (SL) is a well-known assertion language used in Hoare-style modular proof systems for programs with dynamically allocated data structures. In this paper we investigate the fragment of first-order SL restricted to the Bernays-Schönfinkel-Ramsey quantifier prefix $\exists^*\forall^*$, where the quantified variables range over the set of memory locations. When this set is uninterpreted (has no associated theory) the fragment is PSPACE-complete, which matches the complexity of the quantifier-free fragment [7]. However, SL becomes undecidable when the quantifier prefix belongs to $\exists^*\forall^*\exists^*$ instead, or when the memory locations are interpreted as integers with linear arithmetic constraints, thus setting a sharp boundary for decidability within SL. We have implemented a decision procedure for the decidable fragment of $\exists^*\forall^*\mathsf{SL}$ as a specialized solver inside a DPLL(T) architecture, within the CVC4 SMT solver. The evaluation of our implementation was carried out using two sets of verification conditions, produced by (i) unfolding inductive predicates, and (ii) a weakest precondition-based verification condition generator. Experimental data shows that automated quantifier instantiation has little overhead, compared to manual model-based instantiation.

1 Introduction

Separation Logic (SL) is a popular logical framework for program verification, used by a large number of methods, ranging from static analysis [6,10,28] to Hoare-style proofs [19] and property-guided abstraction refinement [1]. The salient features that make SL particularly attractive for program verification are the ability of defining (i) recursive data structures using small and natural inductive definitions, (ii) weakest pre- and post-condition calculi that capture the semantics of programs with pointers, and (iii) compositional verification methods, based on the principle of local reasoning (analyzing separately pieces of program working on disjoint heaps).

Consider, for instance, the following inductive definitions, describing an acyclic and a possibly cyclic list segment, respectively:

$$\widehat{\mathsf{ls}}(\mathsf{x},\mathsf{y}) \equiv \mathsf{emp} \wedge \mathsf{x} = \mathsf{y} \ \vee \ \mathsf{x} \neq \mathsf{y} \wedge \exists \mathsf{z} \,.\, \mathsf{x} \mapsto \mathsf{z} * \widehat{\mathsf{ls}}(\mathsf{z},\mathsf{y}) \quad \text{acyclic list segment from } \mathsf{x} \text{ to } \mathsf{y}$$
$$\mathsf{ls}(\mathsf{x},\mathsf{y}) \equiv \mathsf{emp} \wedge \mathsf{x} = \mathsf{y} \ \vee \ \exists \mathsf{u} \,.\, \mathsf{x} \mapsto \mathsf{u} * \mathsf{ls}(\mathsf{u},\mathsf{y}) \quad \text{list segment from } \mathsf{x} \text{ to } \mathsf{y}$$

A. Bouajjani and D. Monniaux (Eds.): VMCAI 2017, LNCS 10145, pp. 462–482, 2017.
DOI: 10.1007/978-3-319-52234-0_25

Intuitively, an acyclic list segment is either empty, in which case the head and the tail coincide ($\mathsf{emp} \wedge \mathsf{x} = \mathsf{y}$), or it contains at least one element which is disjoint from the rest of the list segment. We denote by $\mathsf{x} \mapsto \mathsf{z}$ the fact that x is an allocated memory location, which points to z, and by $\mathsf{x} \mapsto \mathsf{z} * \widehat{\mathsf{ls}}(\mathsf{z}, \mathsf{y})$ the fact that $\mathsf{x} \mapsto \mathsf{z}$ and $\widehat{\mathsf{ls}}(\mathsf{z}, \mathsf{y})$ hold over disjoint parts of the heap. The constraint $\mathsf{x} \neq \mathsf{y}$, in the inductive definition of $\widehat{\mathsf{ls}}$, captures the fact that the tail of the list segment is distinct from every allocated cell in the list segment, which ensures the acyclicity condition. Since this constraint is omitted from the definition of the second (possibly cyclic) list segment $\mathsf{ls}(\mathsf{x}, \mathsf{y})$, its tail y is allowed to point inside the set of allocated cells.

Automated reasoning is the key enabler of push-button program verification. Any procedure that checks the validity of a logical entailment between inductive predicates requires checking the satisfiability of formulae from the base (non-inductive) assertion language, as shown by the example below. Consider a fragment of the inductive proof showing that any acyclic list segment is also a list segment, given below:

$$\dfrac{\dfrac{\widehat{\mathsf{ls}}(\mathsf{z},\mathsf{y}) \vdash \mathsf{ls}(\mathsf{z},\mathsf{y})}{\mathsf{x} \neq \mathsf{y} \wedge \mathsf{x} \mapsto \mathsf{z} * \widehat{\mathsf{ls}}(\mathsf{z},\mathsf{y}) \vdash \exists \mathsf{u} \,.\, \mathsf{x} \mapsto \mathsf{u} * \mathsf{ls}(\mathsf{u},\mathsf{y})}\ \begin{array}{l}\mathsf{x} \neq \mathsf{y} \wedge \mathsf{x} \mapsto \mathsf{z} \models \exists \mathsf{u} \,.\, \mathsf{x} \mapsto \mathsf{u} \\ \text{by instantiation } \mathsf{u} \leftarrow \mathsf{z}\end{array}}{\widehat{\mathsf{ls}}(\mathsf{x},\mathsf{y}) \vdash \mathsf{ls}(\mathsf{x},\mathsf{y})}$$

The first (bottom) inference in the proof corresponds to one of the two cases produced by unfolding both the antecedent and consequent of the entailment (the second case $\mathsf{emp} \wedge \mathsf{x} = \mathsf{y} \vdash \mathsf{emp} \wedge \mathsf{x} = \mathsf{y}$ is trivial and omitted for clarity). The second inference is a simplification of the sequent obtained by unfolding, to a sequent matching the initial one (by renaming z to x), and allows to conclude this branch of the proof by an inductive argument, based on the principle of infinite descent [5].

The simplification applied by the second inference above relies on the validity of the entailment $\mathsf{x} \neq \mathsf{y} \wedge \mathsf{x} \mapsto \mathsf{z} \models \exists \mathsf{u} \,.\, \mathsf{x} \mapsto \mathsf{u}$, which reduces to the (un)satisfiability of the formula $\mathsf{x} \neq \mathsf{y} \wedge \mathsf{x} \mapsto \mathsf{z} \wedge \forall \mathsf{u} \,.\, \neg \mathsf{x} \mapsto \mathsf{u}$. The latter falls into the Bernays-Schönfinkel-Ramsey fragment, defined by the $\exists^*\forall^*$ quantifier prefix, and can be proved unsatisfiable using the instantiation of the universally quantified variable u with the existentially quantified variable z (or a corresponding Skolem constant). In other words, this formula is unsatisfiable because the universal quantified subformula asks that no memory location is pointed to by x, which is contradicted by $\mathsf{x} \mapsto \mathsf{z}$. The instantiation of u that violates the universal condition is $\mathsf{u} \leftarrow \mathsf{z}$, which is carried over in the rest of the proof.

The goal of this paper is mechanizing satisfiability of the Bernays-Schönfinkel-Ramsey fragment of SL, without inductively defined predicates[1]. This fragment is defined by the quantifier prefix of the formulae in prenex normal form. We consider formulae $\exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \,.\, \phi(x_1, \ldots, x_m, y_1, \ldots, y_n)$,

[1] Strictly speaking, the Bernays-Schönfinkel-Ramsey class refers to the $\exists^*\forall^*$ fragment of first-order logic with equality and predicate symbols, but no function symbols [17].

where ϕ is any quantifier-free formula of SL, consisting of pure formulae from given base theory T, and points-to atomic propositions relating terms of T, combined with unrestricted Boolean and separation connectives, and the quantified variables range essentially over the set of memory locations. In a nutshell, the contributions of the paper are two-fold:

1. We draw a sharp boundary between decidability and undecidability, proving essentially that the satisfiability problem for the Bernays-Schönfinkel-Ramsey fragment of SL is PSPACE-complete, if the domain of memory locations is an uninterpreted set, whereas interpreting memory locations as integers with linear arithmetic constraints, leads to undecidability. Moreover, undecidability occurs even for uninterpreted memory locations, if we extend the quantifier prefix to $\exists^*\forall^*\exists^*$.
2. We have implemented an effective decision procedure for quantifier instantiation, based on counterexample-driven learning of conflict lemmas, integrated within the DPLL(T) architecture [12] of the CVC4 SMT solver [2]. Experimental evaluation of our implementation shows that the overhead of the push-button quantifier instantiation is negligible, compared to the time required to solve a quantifier-free instance of the problem, obtained manually, by model inspection.

Related Work. The first theoretical results on the decidability and computational complexity of SL (without inductive definitions) were found by Calcagno, Yang and O'Hearn [7]. They showed that the satisfiability problem for SL is undecidable, in the presence of quantifiers, assuming that each memory location can point to two other locations, i.e. using atomic propositions of the form $\mathsf{x} \mapsto (\mathsf{y}, \mathsf{z})$. Decidability can be recovered by considering the quantifier-free fragment, proved to be PSPACE-complete, by a small model argument [7]. Refinements of these results consider decidable fragments of SL with one record field (atomic points-to propositions $\mathsf{x} \mapsto \mathsf{y}$), and one or two quantified variables. In a nutshell, SL with one record field and separating conjunction only is decidable with non-elementary time complexity, whereas adding the magic wand adjoint leads to undecidability [4]. Decidability, in the presence of the magic wand operator, is recovered by restricting the number of quantifiers to one, in which case the logic becomes PSPACE-complete [9]. This bound is sharp, because allowing two quantified variables leads to undecidability, and decidability with non-elementary time complexity if the magic wand is removed [8].

SMT techniques were applied to deciding the satisfiability of SL in the work of Piskac, Wies and Zufferey [21,22]. They considered quantifier-free fragments of SL with separating conjunction in positive form (not occurring under negation) and without magic wand, and allowed for hardcoded inductive predicates (list and tree segments). In a similar spirit, we previously defined a translation to multi-sorted second-order logic combined with counterexample-driven instantiation for set quantifiers to define a decision procedure for the quantifier-free fragment of SL [25]. In a different vein, a tableau-based semi-decision procedure is given by Méry and Galmiche [11]. Termination of this procedure is guaranteed for the (decidable) quantifier-free fragment of SL, yet no implementation is available for comparison.

A number of automated theorem provers have efficient and complete approaches for the Bernays-Schönfinkel-Ramsey fragment of first-order-logic, also known as effectively propositional logic (EPR) [3,16]. A dedicated approach for EPR in the SMT solver Z3 was developed in [20]. An approach based on finite model finding is implemented in CVC4 [26], which is model-complete for EPR. Our approach is based on counterexample-guided quantifier instantiation, which has been used in the context of SMT solving in previous works [13,23].

2 Preliminaries

We consider formulae in multi-sorted first-order logic. A *signature* Σ consists of a set Σ^{s} of sort symbols and a set Σ^{f} of (sorted) *function symbols* $f^{\sigma_1 \cdots \sigma_n \sigma}$, where $n \geq 0$ and $\sigma_1, \ldots, \sigma_n, \sigma \in \Sigma^{\mathrm{s}}$. If $n = 0$, we call f^σ a *constant symbol.* In this paper, we consider signatures Σ containing the Boolean sort, and write $\top$ and $\bot$ for the Boolean constants *true* and *false*. For this reason, we do not consider predicate symbols as part of a signature, as predicates are viewed as Boolean functions. Additionally, we assume for any finite sequence of sorts $\sigma_1, \ldots, \sigma_n \in \Sigma^{\mathrm{s}}$, the *tuple* sort $\sigma_1 \times \ldots \times \sigma_n$ also belongs to Σ^{s}, and that Σ^{f} includes the i^{th} tuple projection function for each $i = 1, \ldots, n$. For each $k > 0$, let σ^k denote the k-tuple sort $\sigma \times \ldots \times \sigma$.

Let Vars be a countable set of first-order variables, each $x^\sigma \in \mathsf{Vars}$ having an associated sort σ. First-order terms and formulae over the signature Σ (called Σ-terms and Σ-formulae) are defined as usual. For a Σ-formula φ, we denote by $\mathrm{Fvc}(\varphi)$ the set of free variables and constant symbols in φ, and by writing $\varphi(x)$ we mean that $x \in \mathrm{Fvc}(\phi)$. Whenever $\mathrm{Fvc}(\phi) \cap \mathsf{Vars} = \emptyset$, we say that ϕ is a *sentence*, i.e. ϕ has no free variables. A *Σ-interpretation* $\mathcal{I}$ maps:(1) each sort symbol $\sigma \in \Sigma$ to a non-empty set $\sigma^{\mathcal{I}}$, (2) each function symbol $f^{\sigma_1, \ldots, \sigma_n, \sigma} \in \Sigma$ to a total function $f^{\mathcal{I}} : \sigma_1^{\mathcal{I}} \times \ldots \times \sigma_n^{\mathcal{I}} \rightarrow \sigma^{\mathcal{I}}$ where $n > 0$, and to an element of $\sigma^{\mathcal{I}}$ when $n = 0$, and (3) each variable $x^\sigma \in \mathsf{Vars}$ to an element of $\sigma^{\mathcal{I}}$. For an interpretation $\mathcal{I}$ a sort symbol σ and a variable x, we denote by $\mathcal{I}[\sigma \leftarrow S]$ and, respectively $\mathcal{I}[x \leftarrow v]$, the interpretation associating the set S to σ, respectively the value v to x, and which behaves like $\mathcal{I}$ in all other cases[2]. For a Σ-term t, we write $t^{\mathcal{I}}$ to denote the interpretation of t in $\mathcal{I}$, defined inductively, as usual. A satisfiability relation between Σ-interpretations and Σ-formulas, written $\mathcal{I} \models \varphi$, is also defined inductively, as usual. We say that $\mathcal{I}$ is *a model of* φ if $\mathcal{I}$ satisfies φ.

A (multi-sorted first-order) *theory* is a pair $T = (\Sigma, \mathbf{I})$ where Σ is a signature and $\mathbf{I}$ is a non-empty set of Σ-interpretations, the *models* of T. We assume that Σ always contains the equality predicate, which we denote by $\approx$, as well as projection functions for each tuple sort. A Σ-formula φ is *T-satisfiable* if it is satisfied by some interpretation in $\mathbf{I}$. We write E to denote the empty theory (with equality), whose signature consists of a sort U with no additional

[2] By writing $\mathcal{I}[\sigma \leftarrow S]$ we ensure that all variables of sort σ are mapped by $\mathcal{I}$ to elements of S.

function symbols, and LIA to denote the theory of linear integer arithmetic, whose signature consists of the sort Int, the binary predicate symbol $\geq$, function $+$ denoting addition, and the constants $0, 1$ of sort Int, interpreted as usual. In particular, there are no uninterpreted function symbols in LIA. By ELIA we denote the theory obtained by extending the signature of LIA with the sort U of E and the equality over U.

Let $T = (\Sigma, \mathbf{I})$ be a theory and let Loc and Data be two sorts from Σ, with no restriction other than the fact that Loc is always interpreted as a countable set. Also, we consider that Σ has a designated constant symbol $\mathsf{nil}^{\mathsf{Loc}}$. The *Separation Logic* fragment $\mathsf{SL}(T)_{\mathsf{Loc},\mathsf{Data}}$ is the set of formulae generated by the following syntax:

$$\varphi := \phi \mid \mathsf{emp} \mid \mathsf{t} \mapsto \mathsf{u} \mid \varphi_1 * \varphi_2 \mid \varphi_1 \mathrel{-\!\!*} \varphi_2 \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \exists x^\sigma \ . \ \varphi_1(x)$$

where ϕ is a Σ-formula, and t, u are Σ-terms of sorts Loc and Data, respectively. As usual, we write $\forall x^\sigma \ . \ \varphi(x)$ for $\neg\exists x^\sigma \ . \ \neg\varphi(x)$. We omit specifying the sorts of variables and constants when they are clear from the context.

Given an interpretation $\mathcal{I}$, a *heap* is a finite partial mapping $h : \mathsf{Loc}^{\mathcal{I}} \rightharpoonup_{\text{fin}} \mathsf{Data}^{\mathcal{I}}$. For a heap h, we denote by $\text{dom}(h)$ its domain. For two heaps h_1 and h_2, we write $h_1 \# h_2$ for $\text{dom}(h_1) \cap \text{dom}(h_2) = \emptyset$ and $h = h_1 \uplus h_2$ for $h_1 \# h_2$ and $h = h_1 \cup h_2$. We define the *satisfaction relation* $\mathcal{I}, h \models_{\mathsf{SL}} \phi$ inductively, as follows:

$$\begin{array}{lll}
\mathcal{I}, h \models_{\mathsf{SL}} \phi & \iff \mathcal{I} \models \phi \ \text{ if } \phi \text{ is a } \Sigma\text{-formula} \\
\mathcal{I}, h \models_{\mathsf{SL}} \mathsf{emp} & \iff h = \emptyset \\
\mathcal{I}, h \models_{\mathsf{SL}} \mathsf{t} \mapsto \mathsf{u} & \iff h = \{(\mathsf{t}^{\mathcal{I}}, \mathsf{u}^{\mathcal{I}})\} \text{ and } \mathsf{t}^{\mathcal{I}} \not\approx \mathsf{nil}^{\mathcal{I}} \\
\mathcal{I}, h \models_{\mathsf{SL}} \phi_1 * \phi_2 & \iff \text{there exist heaps } h_1, h_2 \text{ s.t. } h = h_1 \uplus h_2 \text{ and } \mathcal{I}, h_i \models_{\mathsf{SL}} \phi_i, i = 1, 2 \\
\mathcal{I}, h \models_{\mathsf{SL}} \phi_1 \mathrel{-\!\!*} \phi_2 & \iff \text{for all heaps } h' \text{ if } h' \# h \text{ and } \mathcal{I}, h' \models_{\mathsf{SL}} \phi_1 \text{ then } \mathcal{I}, h' \uplus h \models_{\mathsf{SL}} \phi_2 \\
\mathcal{I}, h \models_{\mathsf{SL}} \exists x^S . \varphi(x) & \iff \mathcal{I}[x \leftarrow s], h \models_{\mathsf{SL}} \varphi(x), \text{ for some } s \in S^{\mathcal{I}}
\end{array}$$

The satisfaction relation for Σ-formulae, Boolean connectives $\wedge$, $\neg$, and linear arithmetic atoms, are the classical ones from first-order logic. Notice that the range of a quantified variable x^S is the interpretation of its associated sort $S^{\mathcal{I}}$.

A formula φ is said to be *satisfiable* if there exists an interpretation $\mathcal{I}$ and a heap h such that $\mathcal{I}, h \models_{\mathsf{SL}} \varphi$. The (SL, T)-*satisfiability problem* asks, given an SL formula φ, whether there exists an interpretation $\mathcal{I}$ of T and a heap h such that $\mathcal{I}, h \models_{\mathsf{SL}} \varphi$. We write $\varphi \models_{\mathsf{SL}} \psi$ if for every interpretation $\mathcal{I}$ and heap h, if $\mathcal{I}, h \models_{\mathsf{SL}} \varphi$ then $\mathcal{I}, h \models_{\mathsf{SL}} \psi$, and we say that φ *entails* ψ in this case.

The Bernays-Schönfinkel-Ramsey Fragment of SL. In this paper we address the satisfiability problem for the class of sentences $\phi \equiv \exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \ . \ \varphi(x_1, \ldots, x_m, y_1, \ldots, y_n)$, where φ is a quantifier-free formula of $\mathsf{SL}(T)_{\mathsf{Loc},\mathsf{Data}}$. We shall denote this fragment by $\exists^*\forall^*\mathsf{SL}(T)_{\mathsf{Loc},\mathsf{Data}}$. It is easy to see that any sentence ϕ, as above, is satisfiable if and only if the sentence $\forall y_1 \ldots \forall y_n \ . \ \varphi[c_1/x_1, \ldots, c_m/x_m]$ is satisfiable, where $c_1, \ldots, c_m$ are fresh (Skolem) constant symbols. The latter is called the *functional form* of ϕ.

As previously mentioned, SL is used mainly specify properties of a program's heap. If the program under consideration uses pointer arithmetic, as in C or C++, it is useful to consider LIA for the theory of memory addresses. Otherwise, if the program only compares the values of the pointers for equality, as in

Java, one can use E for this purpose. This distinction led us to considering the satisfiability problem for $\exists^*\forall^*\mathsf{SL}(T)_{\mathsf{Loc},\mathsf{Data}}$ in the following cases:

1. Loc is interpreted as the sort U of E and Data as U^k, for some $k \geq 1$. The satisfiability problem for the fragment $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ is PSPACE-complete, and the proof follows a small model property argument.
2. as above, with the further constraint that U is interpreted as an infinite countable set, i.e. of cardinality $\aleph_0$. In this case, we prove a cut-off property stating that all locations not in the domain of the heap and not used in the interpretation of constants, are equivalent from the point of view of an SL formula. This satisfiability problem is reduced to the unconstrained one above, and also found to be PSPACE-complete.
3. both Loc and Data are interpreted as Int, equipped with addition and total order, in which case $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$ is undecidable.
4. Loc is interpreted as the sort U of E, and Data as $U \times \mathsf{Int}$. Then $\exists^*\forall^*\mathsf{SL}(\mathsf{ELIA})_{U,U\times\mathsf{Int}}$ is undecidable.

Additionally, we prove that the fragment $\exists^*\forall^*\exists^*\mathsf{SL}(\mathsf{E})_{U,U^k}$, with two quantifier alternations, is undecidable, if $k \geq 2$. The question whether the fragment $\exists^*\forall^*\mathsf{SL}(\mathsf{ELIA})_{U,\mathsf{Int}}$ is decidable is currently open, and considered for future work.

For space reasons, all missing proofs are given in [24].

3 Decidability and Complexity Results

This section defines the decidable cases of the Bernays-Schönfinkel-Ramsey fragment of SL, with matching undecidable extensions. The decidable fragment $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ relies on a small model property given in Sect. 3.1. Undecidability of $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$ is obtained by a refinement of the undecidability proof for Presburger arithmetic with one monadic predicate [14], in Sect. 3.4.

3.1 Small Model Property

The decidability proof for the quantifier-free fragment of SL [7,30] relies on a small model property. Intuitively, no quantifier-free SL formula can distinguish between heaps in which the number of invisible locations, not in the range of the set of free variables, exceeds a certain threshold, linear in the size of the formula. Then a formula is satisfiable iff it has a heap model of size linear in the size of the input formula.

For reasons of self-containment, we recall a number of definitions and results from [30]. Some of them are slightly modified for our purposes, but these changes have no effect on the validity of the original proofs for the Lemmas 1 and 2 below. In the rest of this section, we consider formulae of $\mathsf{SL}(\mathsf{E})_{U,U^k}$, meaning that (i) $\mathsf{Loc} = U$, and (ii) there exists an integer $k > 0$ such that $\mathsf{Data} = U^k$, where U is the (uninterpreted) sort of E. We fix k for the rest of this section.

Definition 1 *[30, Definition 90]. Given a set of locations S, the equivalence relation $=_S$ between k-tuples of locations is defined as $\langle v_1, \ldots, v_k \rangle =_S \langle v'_1, \ldots, v'_k \rangle$ if and only if*

- *if $v_i \in S$ then $v_i = v'_i$, and*
- *if $v_i \notin S$ then $v'_i \notin S$,*

for all $i = 1, \ldots, k$.

Intuitively, $=_S$ restricts the equality to the elements in S. Observe that $=_S$ is an equivalence relation and that $S \subseteq T$ implies $=_T \subseteq =_S$. For a set S, we write $\|S\|$ for its cardinality, in the following.

Definition 2 *[30, Definition 91]. Given an interpretation $\mathcal{I}$, an integer $n > 0$, a set of variables $X \subseteq \mathsf{Vars}$ and a set of locations $S \subseteq U^{\mathcal{I}}$, for any two heaps $h, h' : U^{\mathcal{I}} \rightharpoonup_{\text{fin}} (U^{\mathcal{I}})^k$, we define $h \sim^{\mathcal{I}}_{n,X,S} h'$ if and only if*

1. *$\mathcal{I}(X) \cap \text{dom}(h) = \mathcal{I}(X) \cap \text{dom}(h')$,*
2. *for all $\ell \in \mathcal{I}(X) \cap \text{dom}(h)$, we have $h(\ell) =_{\mathcal{I}(X) \cup S} h'(\ell)$,*
3. *if $\|\text{dom}(h) \setminus \mathcal{I}(X)\| < n$ then $\|\text{dom}(h) \setminus \mathcal{I}(X)\| = \|\text{dom}(h') \setminus \mathcal{I}(X)\|$,*
4. *if $\|\text{dom}(h) \setminus \mathcal{I}(X)\| \geq n$ then $\|\text{dom}(h') \setminus \mathcal{I}(X)\| \geq n$.*

Observe that, for any $n \leq m$ and $S \subseteq T$ we have $\sim^{\mathcal{I}}_{m,X,T} \subseteq \sim^{\mathcal{I}}_{n,X,S}$. In addition, for any integer $k > 0$, subset $S \subseteq U^{\mathcal{I}}$ and location $\ell \in U^{\mathcal{I}}$, we consider the function $prun^{\ell}_{k,S}(\ell_1, \ldots, \ell_k)$, which replaces each value $\ell_i \notin S$ in its argument list by ℓ.

Lemma 1 *[30, Lemma 94]. Given an interpretation $\mathcal{I}$ and a heap $h : U^{\mathcal{I}} \rightharpoonup_{\text{fin}} (U^{\mathcal{I}})^k$, for each integer $n > 0$, each set of variables $X \subseteq \mathsf{Vars}$, each set of locations $L \subseteq U^{\mathcal{I}}$ such that $L \cap \mathcal{I}(X) = \emptyset$ and $\|L\| = n$, and each location $v \in U^{\mathcal{I}} \setminus (\mathcal{I}(X) \cup \{\mathsf{nil}^{\mathcal{I}}\} \cup L)$, there exists a heap $h' : U^{\mathcal{I}} \rightharpoonup_{\text{fin}} (U^{\mathcal{I}})^k$, with the following properties:*

1. *$h \sim^{\mathcal{I}}_{n,X,L} h'$,*
2. *$\text{dom}(h') \setminus \mathcal{I}(X) \subseteq L$,*
3. *for all $\ell \in \text{dom}(h')$, we have $h'(\ell) = prun^{v}_{k,\mathcal{I}(X) \cup L}(h(\ell))$.*

Next, we define the following measure on quantifier-free SL formulae:

$$\begin{array}{cccc} |\phi * \psi| = |\phi| + |\psi| & |\phi \mathrel{-\!\!*} \psi| = |\psi| & |\phi \wedge \psi| = \max(|\phi|, |\psi|) & |\neg \phi| = |\phi| \\ |\mathsf{t} \mapsto \mathsf{u}| = 1 & |\mathsf{emp}| = 1 & |\phi| = 0 \text{ if } \phi \text{ is a } \Sigma\text{-formula} & \end{array}$$

Intuitively, $|\varphi|$ is the maximum number of invisible locations, that are not in $\mathcal{I}(\text{Fvc}(\varphi))$, and which can be distinguished by the quantifier-free $\mathsf{SL}(\mathsf{E})_{U,U^k}$ formula φ. The crux of the PSPACE-completeness proof for quantifier-free $\mathsf{SL}(\mathsf{E})_{U,U^k}$ is that two heaps equivalent up to $|\varphi|$ invisible locations are also equivalent from the point of view of satisfiability of φ, which provides a small model property for this fragment [7,30].

Lemma 2 *[30, Proposition 95]. Given a quantifier-free $\mathsf{SL}(\mathsf{E})_{U,U^k}$ formula φ, an interpretation $\mathcal{I}$, and two heaps h and h', if $h \sim^{\mathcal{I}}_{|\varphi|,\mathrm{Fvc}(\varphi),\emptyset} h'$ and $\mathcal{I}, h \models_{\mathsf{SL}} \varphi$ then $\mathcal{I}, h' \models_{\mathsf{SL}} \varphi$.*

Our aim is to extend this result to $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$, in the first place. This new small model property is given by the next lemma.

Lemma 3. *Let $\varphi(x_1^U, \ldots, x_n^U)$ be a quantifier-free $\mathsf{SL}(\mathsf{E})_{U,U^k}$-formula, and $\varphi^\forall \equiv \forall x_1^U \ldots \forall x_n^U \, . \, \varphi(x_1^U, \ldots, x_n^U)$ be its universal closure. Then $\varphi^\forall$ has a model if and only if there exists an interpretation $\mathcal{I}$ and a heap $h : U^{\mathcal{I}} \rightharpoonup_{\mathrm{fin}} (U^{\mathcal{I}})^k$ such that $\mathcal{I}, h \models_{\mathsf{SL}} \varphi^\forall$ and:*

1. $\|U^{\mathcal{I}}\| \leq |\varphi| + \|\mathrm{Fvc}(\varphi^\forall)\| + n$,
2. $\mathrm{dom}(h) \subseteq L \cup \mathcal{I}(\mathrm{Fvc}(\varphi^\forall))$,
3. *for all* $\ell \in \mathrm{dom}(h)$, *we have* $h(\ell) \in (\mathcal{I}(\mathrm{Fvc}(\varphi^\forall)) \cup \{\mathsf{nil}^{\mathcal{I}}\} \cup L \cup \{v\})^k$,

where $L \subseteq U^{\mathcal{I}} \backslash \mathcal{I}(\mathrm{Fvc}(\varphi^\forall))$ is a set of locations such that $\|L\| = |\varphi| + n$ and $v \in U^{\mathcal{I}} \backslash (\mathcal{I}(\mathrm{Fvc}(\varphi^\forall)) \cup \{\mathsf{nil}^{\mathcal{I}}\} \cup L)$ is an arbitrary location.

We are ready to prove two decidability results, based on the above small model property, concerning the cases where (i) Loc is interpreted as a countable set with equality, and (ii) Loc is interpreted as an infinite countable set with no other operators than equality.

3.2 Uninterpreted Locations Without Cardinality Constraints

In this section, we consider the satisfiability problem for the fragment $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$, where the location sort U can be interpreted by any (possibly finite) countable set, with no other operations than the equality, and the data sort consists of k-tuples of locations.

Theorem 1. *The satisfiability problem for $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ is* PSPACE-*complete.*

Proof. PSPACE-hardness follows from the fact that satisfiability is PSPACE-complete for quantifier-free $\mathsf{SL}(\mathsf{E})_{U,U^k}$ [7]. To prove membership in PSPACE, consider the formula $\phi \equiv \exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \, . \, \varphi(\mathbf{x}, \mathbf{y})$, where φ is a quantifier-free $\mathsf{SL}(\mathsf{E})_{U,U^k}$ formula. Let $\mathbf{c} = \langle c_1, \ldots, c_m \rangle$ be a tuple of constant symbols, and $\widetilde{\phi} \equiv \forall y_1 \ldots \forall y_n \, . \, \varphi(\mathbf{c}, \mathbf{y})$ be the functional form of ϕ, obtained by replacing x_i with c_i, for all $i = 1, \ldots, m$. By Lemma 3, $\widetilde{\phi}$ has a model if and only if it has a model $\mathcal{I}, h$ such that:

- $\|U^{\mathcal{I}}\| \leq |\varphi| + n + m$,
- $\mathrm{dom}(h) \subseteq L \cup \mathbf{c}^{\mathcal{I}}$,
- $\forall \ell \in \mathrm{dom}(h) \, . \, h(\ell) \in (\mathcal{I}(\mathbf{c}) \cup \{\mathsf{nil}^{\mathcal{I}}\} \cup L \cup \{v\})^k$,

where $L \subseteq U^{\mathcal{I}} \setminus \mathcal{I}(\mathbf{c})$, $\|L\| = |\varphi| + m$ and $v \in U^{\mathcal{I}} \setminus (\mathcal{I}(\mathbf{c}) \cup \{\mathsf{nil}^{\mathcal{I}}\} \cup L)$. We describe below a nondeterministic polynomial space algorithm that decides satisfiability of $\widetilde{\phi}$. First, nondeterministically chose a model $\mathcal{I}, h$ that meets the above requirements. Then we check, for each tuple $\langle u_1, \ldots, u_n \rangle \in (U^{\mathcal{I}})^n$ that $\mathcal{I}[y_1 \leftarrow u_1] \ldots [y_n \leftarrow u_n], h \models_{\mathsf{SL}} \varphi$. In order to enumerate all tuples from $(U^{\mathcal{I}})^n$ we need $n \cdot \lceil \log_2(|\varphi| + n + m) \rceil$ extra bits, and the check for each such tuple can be done in PSPACE, according to [7, Sect. 5]. □

This result is somewhat surprising, because the classical Bernays-Schönfinkel fragment of first-order formulae with predicate symbols (but no function symbols) and quantifier prefix $\exists^*\forall^*$ is known to be NEXPTIME-complete [17, Sect. 7]. The explanation lies in the fact that the interpretation of an arbitrary predicate symbol $P(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ cannot be captured using only points-to atomic propositions, e.g. $\mathsf{x}_1 \mapsto (\mathsf{x}_2, \ldots, \mathsf{x}_n)$, between locations and tuples of locations, due to the interpretation of points-to's as heaps[3] (finite partial functions).

The following lemma sets a first decidability boundary for $\mathsf{SL}(\mathsf{E})_{U,U^k}$, by showing how extending the quantifier prefix to $\exists^*\forall^*\exists^*$ leads to undecidability.

Lemma 4. *The satisfiability problem for $\exists^*\forall^*\exists^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ is undecidable, if $k \geq 2$.*

Observe that the result of Lemma 4 sets a fairly tight boundary between the decidable and undecidable fragments of SL. On the one hand, simplifying the quantifier prefix to $\exists^*\forall^*$ yields a decidable fragment (Theorem 1), whereas $\mathsf{SL}(\mathsf{E})_{U,U}$ $(k = 1)$ without the magic wand $(-\!\!*)$ is decidable with non-elementary time complexity, even when considering an unrestricted quantifier prefix [4].

3.3 Uninterpreted Locations with Cardinality $\aleph_0$

We consider the stronger version of the satisfiability problem for $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$, where U is interpreted as an infinite countable set (of cardinality $\aleph_0$) with no function symbols, other than equality. Instances of this problem occur when, for instance, the location sort is taken to be Int, but no operations are used on integers, except for testing equality.

Observe that this restriction changes the satisfiability status of certain formulae. For instance, $\exists \mathsf{x} \forall \mathsf{y} \,.\, \mathsf{y} \not\approx \mathsf{nil} \Rightarrow (\mathsf{y} \mapsto \mathsf{x} * \top)$ is satisfiable if U is interpreted as a finite set, but becomes unsatisfiable when U is infinite. The reason is that this formula requires every location from $U^{\mathcal{I}}$ apart from nil to be part of the domain of the heap, which is impossible due the fact that only finite heaps are considered by the semantics of SL.

In the following proof, we use the formula $\mathsf{alloc}(\mathsf{x}) \equiv \mathsf{x} \mapsto (\mathsf{x}, \ldots, \mathsf{x}) -\!\!* \bot$, expressing the fact that a location variable x is *allocated*, i.e. its interpretation is part of the heap's domain [4]. Intuitively, we reduce any instance of the $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ satisfiability problem, with U of cardinality $\aleph_0$, to an instance of the same problem without this restriction, by the following cut-off argument:

[3] If $\mathsf{x}_1 \mapsto (\mathsf{x}_2, \ldots, \mathsf{x}_n)$ and $\mathsf{x}_1 \mapsto (\mathsf{x}'_2, \ldots, \mathsf{x}'_n)$ hold, this forces $\mathsf{x}_i = \mathsf{x}'_i$, for all $i = 2, \ldots, n$.

if a free variable is interpreted as a location which is neither part of the heap's domain, nor equal to the interpretation of some constant, then it is not important which particular location is chosen for that interpretation.

Theorem 2. *The satisfiability problem for* $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ *is* PSPACE-*complete if* U *is required to have cardinality* $\aleph_0$.

Proof. PSPACE-hardness follows from the PSPACE-completeness of the satisfiability problem for quantifier-free SL, with uninterpreted locations [7, Sect. 5.2]. Since the reduction from [7, Sect. 5.2] involves no universally quantified variables, the $\aleph_0$ cardinality constraint has no impact on this result.

Let $\exists x_1 \ldots \exists x_m \forall y_1 \ldots \forall y_n \, . \, \varphi(\mathbf{x}, \mathbf{y})$ be a formula, and $\forall y_1 \ldots \forall y_n \, . \, \varphi(\mathbf{c}, \mathbf{y})$ be its functional form, obtained by replacing each x_i with c_i, for $i = 1, \ldots, m$. We consider the following formulae, parameterized by y_i, for $i = 1, \ldots, n$:

$$\begin{aligned}
\psi_0(y_i) &\equiv \mathsf{alloc}(y_i) \\
\psi_1(y_i) &\equiv \textstyle\bigvee_{j=1}^{m} y_i = c_j \\
\psi_2(y_i) &\equiv y_i = d_i \\
\mathsf{external} &\equiv \textstyle\bigwedge_{i=1}^{n} (\neg\mathsf{alloc}(d_i) \wedge \bigwedge_{j=1}^{m} d_i \neq c_j)
\end{aligned}$$

where $\{d_i \mid i = 1, \ldots, n\}$ is a set of fresh constant symbols. We show the following fact:

Fact 1. *There exists an interpretation* $\mathcal{I}$ *and a heap* h *such that* $\|U^{\mathcal{I}}\| = \aleph_0$ *and* $\mathcal{I}, h \models_{\mathsf{SL}} \forall y_1 \ldots \forall y_n \, . \, \varphi(\mathbf{c}, \mathbf{y})$ *iff there exists an interpretation* $\mathcal{I}'$*, not constraining the cardinality of* $U^{\mathcal{I}'}$*, and a heap* h' *such that:*

$$\mathcal{I}', h' \models_{\mathsf{SL}} \mathsf{external} \wedge \forall y_1 \ldots \forall y_n \bigwedge_{\langle t_1, \ldots, t_n \rangle \in \{0,1,2\}^n} \underbrace{\bigwedge_{i=1}^{n} (\psi_{t_i}(y_i) \Rightarrow \varphi(\mathbf{c}, \mathbf{y}))}_{\Psi_{\langle t_1, \ldots, t_n \rangle}}$$

To show membership in PSPACE, consider a nondeterministic algorithm that choses $\mathcal{I}'$ and h' and uses $2n$ extra bits to check that $\mathcal{I}', h' \models_{\mathsf{SL}} \mathsf{extern} \wedge \forall y_1 \ldots \forall y_n \, . \, \Psi_{\langle t_1, \ldots, t_n \rangle}$ separately, for each $\langle t_1, \ldots, t_n \rangle \in \{0, 1, 2\}^n$. By Lemma 3, the sizes of $\mathcal{I}'$ and h' are bounded by a polynomial in the size of $\Psi_{\langle t_1, \ldots, t_n \rangle}$, which is polynomial in the size of φ, and by Theorem 1, each of these checks can be done in polynomial space. □

3.4 Integer Locations with Linear Arithmetic

In the rest of this section we show that the Bernays-Schönfinkel-Ramsey fragment of SL becomes undecidable as soon as we use integers to represent the set of locations and combine SL with linear integer arithmetic (LIA). The proof relies on an undecidability argument for a fragment of Presburger arithmetic with one monadic predicate symbol, interpreted over finite sets. Formally, we denote by $(\exists^*\forall^* \cap \forall^*\exists^*) - \mathsf{LIA}$ the set of formulae consisting of a conjunction between two linear arithmetic formulae, one with quantifier prefix in the language $\exists^*\forall^*$, and another with quantifier prefix $\forall^*\exists^*$.

Theorem 3. *The satisfiability problem is undecidable for the fragment* $(\exists^*\forall^* \cap \forall^*\exists^*) - \mathsf{LIA}$*, with one monadic predicate symbol, interpreted over finite sets of integers.*

Proof. We reduce from the following variant of *Hilbert's 10th Problem*: given a multivariate Diophantine polynomial $R(x_1, \ldots, x_n)$, the problem "does $R(x_1, \ldots, x_n) = 0$ have a solution in $\mathbb{N}^n$?" is undecidable [18].

By introducing sufficiently many free variables, we encode $R(x_1, \ldots, x_n) = 0$ as an equisatisfiable Diophantine system of degree at most two, containing only equations of the form $x = yz$ (resp. $x = y^2$) and linear equations $\sum_{i=1}^{k} a_i x_i = b$, where $a_1, \ldots, a_k, b \in \mathbb{Z}$. Next, we replace each equation of the form $x = yz$, with y and z distinct variables, with the quadratic system $2x + t_y + t_z = t_{y+z} \wedge t_y = y^2 \wedge t_z = z^2 \wedge t_{y+z} = (y+z)^2$, where t_y, t_z and t_{y+z} are fresh (free) variables. In this way, we replace all multiplications between distinct variables by occurrences of the squaring function. Let $\Psi_{R(x_1,\ldots,x_n)=0}$ be the conjunction of the above equations. It is manifest that $R(x_1, \ldots, x_n) = 0$ has a solution in $\mathbb{N}^n$ iff $\Psi_{R(x_1,\ldots,x_n)=0}$ is satisfiable, with all free variables ranging over $\mathbb{N}$.

Now we introduce a monadic predicate symbol P, which is intended to denote a (possibly finite) set of consecutive perfect squares, starting with 0. To capture this definition, we require the following:

$$\begin{aligned} &P(0) \wedge P(1) \wedge \forall x \forall y \forall z \, . \, P(x) \wedge P(y) \wedge P(z) \wedge x < y < z \wedge \\ &(\forall u \, . \, x < u < y \vee y < u < z \Rightarrow \neg P(u)) \Rightarrow z - y = y - x + 2 \end{aligned} \qquad (\mathsf{sqr})$$

Observe that this formula is a weakening of the definition of the infinite set of perfect squares given by Halpern [14], from which the conjunct $\forall x \exists y \, . \, y > x \wedge P(y)$, requiring that P is an infinite set of natural numbers, has been dropped. Moreover, notice that sqr has quantifier prefix $\forall^3\exists$, due to the fact that $\forall u$ occurs implicitly under negation, on the left-hand side of an implication. If P is interpreted as a finite set $P^{\mathcal{I}} = \{p_0, p_1, \ldots, p_N\}$ such that (w.l.o.g.) $p_0 < p_1 < \ldots < p_N$, it is easy to show, by induction on $N > 0$, that $p_i = i^2$, for all $i = 0, 1, \ldots, N$.

The next step is encoding the squaring function using the monadic predicate P. This is done by replacing each atomic proposition $x = y^2$ in $\Psi_{R(x_1,\ldots,x_n)=0}$ by the formula $\theta_{x=y^2} \equiv P(x) \wedge P(x + 2y + 1) \wedge \forall z \, . \, x < z < x + 2y + 1 \Rightarrow \neg P(z)$.

Fact 2. *For each interpretation* $\mathcal{I}$ *mapping* x *and* y *into* $\mathbb{N}$, $\mathcal{I} \models x = y^2$ *iff* $\mathcal{I}$ *can be extended to an interpretation of* P *as a finite set of consecutive perfect squares such that* $\mathcal{I} \models \theta_{x=y^2}$*.*

Let $\Phi_{R(x_1,\ldots,x_n)=0}$ be the conjunction of sqr with the formula obtained by replacing each atomic proposition $x = y^2$ with $\theta_{x=y^2}$ in $\Psi_{R(x_1,\ldots,x_n)=0}$. Observe that each universally quantified variable in $\Phi_{R(x_1,\ldots,x_n)=0}$ occurs either in sqr or in some $\theta_{x=y^2}$, and moreover, each $\theta_{x=y^2}$ belongs to the $\exists^*\forall^*$ fragment of LIA. $\Phi_{R(x_1,\ldots,x_n)=0}$ belongs thus to the $\exists^*\forall^* \cap \forall^*\exists^*$ fragment of LIA, with P being the only monadic predicate symbol. Finally, we prove that $R(x_1, \ldots, x_n) = 0$ has a solution in $\mathbb{N}^n$ iff $\Phi_{R(x_1,\ldots,x_n)=0}$ is satisfiable.

"$\Rightarrow$" Let $\mathcal{I}$ be a valuation mapping $x_1, \ldots, x_n$ into $\mathbb{N}$, such that $\mathcal{I} \models R(x_1, \ldots, x_n) = 0$. Obviously, $\mathcal{I}$ can be extended to a model of $\varPsi_{R(x_1,\ldots,x_n)=0}$ by assigning $t_x^{\mathcal{I}} = (x^{\mathcal{I}})^2$ for all auxiliary variables t_x occurring in $\varPsi_{R(x_1,\ldots,x_n)=0}$. We extend $\mathcal{I}$ to a model of $\varPhi_{R(x_1,\ldots,x_n)=0}$ by assigning $P^{\mathcal{I}} = \{n^2 \mid 0 \leq n \leq \sqrt{m}\}$, where $m = \max\{(x^{\mathcal{I}}+1)^2 \mid x \in \mathrm{Fvc}(\varPsi_{R(x_1,\ldots,x_n)=0})\}$. Clearly $P^{\mathcal{I}}$ meets the requirements of sqr. By Fact 2, we obtain that $\mathcal{I} \models \theta_{x=y^2}$ for each subformula $\theta_{x=y^2}$ of $\varPhi_{R(x_1,\ldots,x_n)=0}$, thus $\mathcal{I} \models \varPhi_{R(x_1,\ldots,x_n)=0}$.

"$\Leftarrow$" If $\mathcal{I} \models \varPsi_{R(x_1,\ldots,x_n)=0}$ then, by sqr, $P^{\mathcal{I}}$ is a set of consecutive perfect squares, and, by Fact 2, $\mathcal{I} \models x = y^2$ for each subformula $\theta_{x=y^2}$ of $\varPhi_{R(x_1,\ldots,x_n)=0}$. Then $\mathcal{I} \models \varPsi_{R(x_1,\ldots,x_n)=0}$ and consequently $\mathcal{I} \models R(x_1, \ldots, x_n) = 0$. $\square$

We consider now the satisfiability problem for the fragment $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$ where both Loc and Data are taken to be the Int sort, equipped with addition and total order. Observe that, in this case, the heap consists of a set of lists, possibly with aliases and circularities. Without losing generality, we consider that Int is interpreted as the set of positive integers[4].

The above theorem cannot be directly used for the undecidability of $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$, by interpreting the (unique) monadic predicate as the (finite) domain of the heap. The problem is with the sqr formula, that defines the interpretation of the monadic predicate as a set of consecutive perfect squares $0, 1, \ldots, n^2$, and whose quantifier prefix lies in the $\forall^*\exists^*$ fragment. We overcome this problem by replacing the sqr formula above with a definition of such sets in $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$. Let us first consider the following properties expressed in SL [4]:

$$\sharp\mathsf{x} \geq 1 \equiv \exists \mathsf{u} \,.\, \mathsf{u} \mapsto \mathsf{x} * \top$$
$$\sharp\mathsf{x} \leq 1 \equiv \forall \mathsf{u} \forall \mathsf{t} \,.\, \neg(\mathsf{u} \mapsto \mathsf{x} * \mathsf{t} \mapsto \mathsf{x} * \top)$$

Intuitively, $\sharp\mathsf{x} \geq 1$ states that x has at least one predecessor in the heap, whereas $\sharp\mathsf{x} \leq 1$ states that x has at most one predecessor. We use $\sharp\mathsf{x} = 0$ and $\sharp\mathsf{x} = 1$ as shorthands for $\neg(\sharp\mathsf{x} \geq 1)$ and $\sharp\mathsf{x} \geq 1 \wedge \sharp\mathsf{x} \leq 1$, respectively. The formula below states that the heap can be decomposed into a list segment starting with x and ending in y, and several disjoint cyclic lists:

$$\begin{aligned}\mathsf{x} \overset{\circlearrowleft}{\longrightarrow}^{+} \mathsf{y} \equiv\ & \sharp\mathsf{x} = 0 \wedge \mathsf{alloc}(\mathsf{x}) \wedge \sharp\mathsf{y} = 1 \wedge \neg\mathsf{alloc}(\mathsf{y}) \wedge \\ & \forall \mathsf{z} \,.\, \mathsf{z} \not\approx \mathsf{y} \Rightarrow (\sharp\mathsf{z} = 1 \Rightarrow \mathsf{alloc}(\mathsf{z})) \wedge \forall \mathsf{z} \,.\, \sharp\mathsf{z} \leq 1\end{aligned}$$

We forbid the existence of circular lists by adding the following arithmetic constraint:

$$\forall \mathsf{u} \forall \mathsf{t} \,.\, \mathsf{u} \mapsto \mathsf{t} * \top \Rightarrow \mathsf{u} < \mathsf{t} \qquad (\mathsf{nocyc})$$

We ask, moreover, that the elements of the list segment starting in x are consecutive perfect squares:

$$\mathsf{consqr}(\mathsf{x}) \equiv \mathsf{x} = 0 \wedge \mathsf{x} \mapsto 1 * \top \wedge \forall \mathsf{z} \forall \mathsf{u} \forall \mathsf{t} \,.\, \mathsf{z} \mapsto \mathsf{u} * \mathsf{u} \mapsto \mathsf{t} * \top \Rightarrow \mathsf{t} - \mathsf{u} = \mathsf{u} - \mathsf{z} + 2 \qquad (\mathsf{consqr})$$

[4] Extending the interpretation of Loc to include negative integers does not make any difference for the undecidability result.

Observe that the formula $\exists \mathsf{x} \exists \mathsf{y} \,.\, \mathsf{x} \overset{\circlearrowleft}{\longrightarrow}^{+} \mathsf{y} \wedge \mathsf{nocyc} \wedge \mathsf{consqr}(\mathsf{x})$ belongs to $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$.

Theorem 4. *The satisfiability problem for* $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$ *is undecidable.*

Proof. We use the same reduction as in the proof of Theorem 3, with two differences:

- we replace sqr by $\exists \mathsf{x} \exists \mathsf{y} \,.\, \mathsf{x} \overset{\circlearrowleft}{\longrightarrow}^{+} \mathsf{y} \wedge \mathsf{nocyc} \wedge \mathsf{consqr}(\mathsf{x})$, and
- define $\theta_{x=y^2} \equiv \mathsf{alloc}(x) \wedge \mathsf{alloc}(x+2y+1) \wedge \forall z \,.\, x < z < x+2y+1 \Rightarrow \neg\mathsf{alloc}(z)$.

□

It is tempting, at this point to ask whether interpreting locations as integers and considering subsets of LIA instead may help recover the decidability. For instance, it has been found that the Bernays-Schönfinkel-Ramsey class is decidable in presence of integers with difference bounds arithmetic [29], and the same type of question can be asked about the fragment of $\exists^*\forall^*\mathsf{SL}(\mathsf{LIA})_{\mathsf{Int},\mathsf{Int}}$, with difference bounds constraints only.

Finally, we consider a variant of the previous undecidability result, in which locations are the (uninterpreted) sort U of E and the data consists of tuples of sort $U \times \mathsf{Int}$. This fragment of SL can be used to reason about lists with integer data. The undecidability of this fragment can be proved along the same lines as Theorem 4.

Theorem 5. *The satisfiability problem for* $\exists^*\forall^*\mathsf{SL}(\mathsf{ELIA})_{U,U\times\mathsf{Int}}$ *is undecidable.*

4 A Procedure for $\exists^*\forall^*$ Separation Logic in an SMT Solver

This section presents a procedure for the satisfiability of $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ inputs[5]. Our procedure builds upon our previous work [25], which gave a decision procedure for quantifier-free $\mathsf{SL}(T)_{\mathsf{Loc},\mathsf{Data}}$ inputs for theories T where the satisfiability problem for quantifier-free T-constraints is decidable. Like existing approaches for quantified formulas in SMT [13,23], our approach is based on incremental quantifier instantiation based on a stream of candidate models returned by a solver for quantifier-free inputs. Our approach for this fragment exploits the small model property given in Lemma 3 to restrict the set of quantifier instantiations it considers to a finite set.

Figure 1 gives a counterexample-guided approach for establishing the satisfiability of input $\exists \mathbf{x} \forall \mathbf{y}\, \varphi(\mathbf{x}, \mathbf{y})$. We first introduce tuples of fresh constants $\mathbf{k}$ and $\mathbf{e}$ of the same type as $\mathbf{x}$ and $\mathbf{y}$ respectively. Our procedure will be based on finding a set of instantiations of $\forall \mathbf{y}\, \varphi(\mathbf{k}, \mathbf{y})$ that are either collectively unsatisfiable or are satisfiable and entail our input. Then, we construct a set L which

[5] The procedure is incorporated into the master branch of the SMT solver CVC4 (https://github.com/CVC4), and can be enabled by command line parameter `--quant-epr`.

```
solve($\exists\mathbf{x}\,\forall\mathbf{y}\,\varphi(\mathbf{x},\mathbf{y})$) where $\mathbf{x} = (x_1,\ldots,x_m)$ and $\mathbf{y} = (y_1,\ldots,y_n)$:

    Let $\mathbf{k} = (k_1,\ldots,k_m)$ and $\mathbf{e} = (e_1,\ldots,e_n)$ be fresh constants of the same type as $\mathbf{x}$ and $\mathbf{y}$.
    Let $L = L' \cup \{k_1,\ldots,k_m\}$ where $L'$ is a set of fresh constants s.t. $\|L'\| = |\varphi(\mathbf{x},\mathbf{y})| + n$.
    Return solve_rec($\exists\mathbf{x}\,\forall\mathbf{y}\,\varphi(\mathbf{x},\mathbf{y}), \emptyset, L$).

solve_rec($\exists\mathbf{x}\,\forall\mathbf{y}\,\varphi(\mathbf{x},\mathbf{y}), \Gamma, L$):

  1. If $\Gamma$ is $(\mathsf{SL},\mathsf{E})$-unsat, return "unsat".
  2. Assume $\exists\mathbf{x}\,\forall\mathbf{y}\,\varphi(\mathbf{x},\mathbf{y})$ is equivalent to $\exists\mathbf{x}\,\forall\mathbf{y}\,\varphi_1(\mathbf{x},\mathbf{y}) \wedge \ldots \wedge \forall\mathbf{y}\,\varphi_p(\mathbf{x},\mathbf{y})$.
     If $\Gamma'_j = \Gamma \cup \{\neg\varphi_j(\mathbf{k},\mathbf{e}) \wedge \bigwedge_{i=1}^{n} \bigvee_{t \in L} e_i \approx t\}$ is $(\mathsf{SL},\mathsf{E})$-unsat for all $j = 1,\ldots,p$, return "sat".
  3. Otherwise, let $\mathcal{I}, h \models_{\mathsf{SL}} \Gamma'_j$ for some $j \in \{1,\ldots,p\}$.
     Let $\mathbf{t} = (t_1,\ldots,t_n)$ be such that $e_i^{\mathcal{I}} = t_i^{\mathcal{I}}$ and $t_i \in L$ for each $i = 1,\ldots,n$.
     Return solve_rec($\exists\mathbf{x}\,\forall\mathbf{y}\,\varphi(\mathbf{x},\mathbf{y}), \Gamma \cup \{\varphi_j(\mathbf{k},\mathbf{t})\}, L$).
```

Fig. 1. A counterexample-guided procedure for $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ formulas $\exists\mathbf{x}\,\forall\mathbf{y}\,\varphi(\mathbf{x},\mathbf{y})$, where U is an uninterpreted sort in the signature of E.

is the union of constants $\mathbf{k}$ and a set L' of fresh constants whose cardinality is equal to $|\varphi(\mathbf{x},\mathbf{y})|$ (see Sect. 3.1) plus the number of universal variables n in our input. Conceptually, L is a finite set of terms from which the instantiations of $\mathbf{y}$ in $\forall\mathbf{y}\,\varphi(\mathbf{k},\mathbf{y})$ can be built.

After constructing L, we call the recursive subprocedure solve_rec on Γ (initially empty) and L. This procedure incrementally adds instances of $\forall\mathbf{y}\,\varphi(\mathbf{k},\mathbf{y})$ to Γ. In step 1, we first check if Γ is (SL,T)-unsatisfiable using the procedure from [25]. If so, our input is (SL,T)-unsatisfiable. Otherwise, in step 2 we consider the *miniscoped* form of our input $\exists\mathbf{x}\,\forall\mathbf{y}\,\varphi_1(\mathbf{x},\mathbf{y}) \wedge \ldots \wedge \forall\mathbf{y}\,\varphi_p(\mathbf{x},\mathbf{y})$, that is, where quantification over $\mathbf{x}$ is distributed over conjunctions. In the following, we may omit quantification on conjunctions φ_j that do not contain variables from $\mathbf{y}$. Given this formula, for each $j = 1,\ldots,p$, we check the (SL,T)-satisfiability of set Γ'_j containing Γ, the negation of $\forall\mathbf{y}\,\varphi_j(\mathbf{k},\mathbf{y})$ where $\mathbf{y}$ is replaced by fresh contants $\mathbf{e}$, and a conjunction of constraints that says each e_i must be equal to at least one term in L for $i = 1,\ldots,n$. If Γ'_j is (SL,T)-unsatisfiable for each $j = 1,\ldots,p$, our input is (SL,T)-satisfiable. Otherwise in step 3, given an interpretation $\mathcal{I}$ and heap h satisfying Γ'_j, we construct a tuple of terms $\mathbf{t} = (t_1,\ldots,t_n)$ used for instantiating $\forall\mathbf{y}\,\varphi_j(\mathbf{k},\mathbf{y})$. For each $i = 1,\ldots,n$, we choose t_i to be a term from L whose interpretation is the same as e_i. The existence of such a t_i is guaranteed by the fact that $\mathcal{I}$ satisfies the constraint from Γ'_j that tells us e_i is equal to at least one such term. This selection ensures that instantiations on each iteration are chosen from a finite set of possibilities and are unique. In practice, the procedure terminates, both for unsatisfiable and satisfiable inputs, before considering all $\mathbf{t}$ from L^n for each $\forall\mathbf{y}\,\varphi_j(\mathbf{x},\mathbf{y})$.

Theorem 6. *Let U be an uninterpreted sort belonging to the signature of* E. *For all* $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ *formulae* ψ *of the form* $\exists \mathbf{x}\, \forall \mathbf{y}\, \varphi(\mathbf{x}, \mathbf{y})$, $\mathsf{solve}(\psi)$*:*

1. *Answers "unsat" only if ψ is* $(\mathsf{SL}, \mathsf{E})$*-unsatisfiable.*
2. *Answers "sat" only if ψ is* $(\mathsf{SL}, \mathsf{E})$*-satisfiable.*
3. *Terminates.*

We discuss a few important details regarding our implementation of the procedure.

Matching Heuristics. When constructing the terms $\mathbf{t}$ for instantiation, it may be the case that $e_i^{\mathcal{I}} = u^{\mathcal{I}}$ for multiple $u \in L$ for some $i \in \{1, \ldots, n\}$. In such cases, the procedure will choose one such u for instantiation. To increase the likelihood of the instantiation being relevant to the satisfiability of our input, we use heuristics for selecting the best possible u among those whose interpretation is equal to e_i in $\mathcal{I}$. In particular, if $e_i^{\mathcal{I}} = u_1^{\mathcal{I}} = u_2^{\mathcal{I}}$, and Γ' contains predicates of the form $e_i \mapsto v$ and $u_1 \mapsto v_1$ for some v, v_1 where $v^{\mathcal{I}} = v_1^{\mathcal{I}}$ but no predicate of the form $u_2 \mapsto v_2$ for some v_2 where $v^{\mathcal{I}} = v_2^{\mathcal{I}}$, then we strictly prefer term u_1 over term u_2 when choosing term t_i for e_i.

Finding Minimal Models. Previous work [26] developed efficient techniques for finding small models for uninterpreted sorts in CVC4. We have found these techniques to be beneficial to the performance of the procedure in Fig. 1. In particular, we use these techniques to find Σ-interpretations $\mathcal{I}$ in $\mathsf{solve\_rec}$ that interpret U as a finite set of minimal size. When combined with the aforementioned matching heuristics, these techniques lead to finding useful instantiations more quickly, since more terms are constrained to be equal to e_i for $i = 1, \ldots, n$ in interpretations $\mathcal{I}$.

Symmetry Breaking. The procedure in Fig. 1 introduces a set of fresh constants L, which in turn introduce the possibility of discovering Σ-interpretations $\mathcal{I}$ that are isomorphic, that is, identical up to renaming of constants in L'. Our procedure adds additional constraints to Γ that do not affect its satisfiability, but reduce the number of isomorphic models. In particular, we consider an ordering $\prec$ on the constants from L', and add constraints that ensure that all models $(\mathcal{I}, h)$ of Γ are such that if $\ell_1^{\mathcal{I}} \notin \mathrm{dom}(h)$, then $\ell_2^{\mathcal{I}} \notin \mathrm{dom}(h)$ for all ℓ_2 such that $\ell_1 \prec \ell_2$.

Example 1. Say we wish to show the validity of the entailment $\mathsf{x} \neq \mathsf{y} \wedge \mathsf{x} \mapsto \mathsf{z} \models_{\mathsf{SL}} \exists \mathsf{u}\, .\, \mathsf{x} \mapsto \mathsf{u}$, from the introductory example (Sect. 1), where $\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{u}$ are of sort U of E. This entailment is valid iff the $\exists^*\forall^*\mathsf{SL}(\mathsf{E})_{U,U^k}$ formula $\exists \mathsf{x} \exists \mathsf{y} \exists \mathsf{z} \forall \mathsf{u}\, .\, \mathsf{x} \not\approx \mathsf{y} \wedge \mathsf{x} \mapsto \mathsf{z} \wedge \neg \mathsf{x} \mapsto \mathsf{u}$ is $(\mathsf{SL}, \mathsf{E})$-unsatisfiable. A run of the procedure in Fig. 1 on this input constructs tuples $\mathbf{k} = (k_x, k_y, k_z)$ and $\mathbf{e} = (e_u)$, and set $L = \{k_x, k_y, k_z, \ell_1, \ell_2\}$, noting that $|\mathsf{x} \not\approx \mathsf{y} \wedge \mathsf{x} \mapsto \mathsf{z} \wedge \neg \mathsf{x} \mapsto \mathsf{u}| = 1$. We then call $\mathsf{solve\_rec}$ where Γ is initially empty. By miniscoping, our input is equivalent to $\exists \mathsf{x} \exists \mathsf{y} \exists \mathsf{z}\, .\, \mathsf{x} \not\approx \mathsf{y} \wedge \mathsf{x} \mapsto \mathsf{z} \wedge \forall \mathsf{u}\, .\, \neg \mathsf{x} \mapsto \mathsf{u}$. On the first two recursive calls to $\mathsf{solve\_rec}$, we may add $k_x \not\approx k_y$ and $k_x \mapsto k_z$ to Γ by trivial instantiation of the first two conjuncts. On the third recursive call, Γ is $(\mathsf{SL}, \mathsf{E})$-satisfiable, and we check the satisfiability of:

$$\Gamma' = \{k_x \neq k_y, k_x \mapsto k_z, k_x \mapsto e_u \wedge (e_u \approx k_x \vee e_u \approx k_y \vee e_u \approx k_z \vee e_u \approx \ell_1 \vee e_u \approx \ell_2)\}$$

Since $k_x \mapsto k_z$ and $k_x \mapsto e_u$ are in Γ', all Σ-interpretations $\mathcal{I}$ and heaps h such that $\mathcal{I}, h \models_{\mathsf{SL}} \Gamma'$ are such that $e_u^{\mathcal{I}} = k_z^{\mathcal{I}}$. Since $k_z \in L$, we may choose to add the instantiation $\neg k_x \mapsto k_z$ to Γ, after which Γ is $(\mathsf{SL}, \mathsf{E})$-unsatisfiable on the next recursive call to solve_rec. Thus, our input is $(\mathsf{SL}, \mathsf{E})$-unsatisfiable and the entailment is valid. ■

A modified version of the procedure in Fig. 1 can be used for $\exists^*\forall^*\mathsf{SL}(T)_{\mathsf{Loc},\mathsf{Data}}$-satisfiability for theories T beyond equality, and where Loc and Data are not restricted to uninterpreted sorts. Notice that in such cases, we cannot restrict Σ-interpretations $\mathcal{I}$ in solve_rec to interpret each e_i as a member of finite set L, and hence we modify solve_rec to omit the constraint restricting variables in $\mathbf{e}$ to be equal to a term from L in the check in Step 2. This modification results in a procedure that is sound both for "unsat" and "sat", but is no longer terminating in general. Nevertheless, it may be used as a heuristic for determining $\exists^*\forall^*\mathsf{SL}(T)_{\mathsf{Loc},\mathsf{Data}}$-(un)satisfiability.

5 Experimental Evaluation

We implemented the solve procedure from Fig. 1 within the CVC4 SMT solver[6] (version 1.5 prerelease). This implementation was tested on two kinds of benchmarks: (i) finite unfoldings of inductive predicates, mostly inspired by benchmarks used in the SL-COMP'14 solver competition [27], and (ii) verification conditions automatically generated by applying the weakest precondition calculus of [15] to the program loops in Fig. 2. All experiments were run on a 2.80 GHz Intel(R) Core(TM) i7 CPU machine with 8 MB of cache[7].

```
while w ≠ nil do
    assert(w.data = $c_0$)
    v := w;
    w := w.next;
    dispose(v);
    do
```

(z)disp

$\mathsf{list}^0(x) \triangleq \mathsf{emp} \wedge x = \mathsf{nil}$

$\mathsf{list}^n(x) \triangleq \exists y . x \mapsto y * \mathsf{list}^{n-1}(y)$

```
while u ≠ nil do
    assert(u.data = $c_0$)
    w := u.next;
    u.next := v;
    v := u;
    u := w;
    do
```

(z)rev

$\mathsf{zlist}^0(x) \triangleq \mathsf{emp} \wedge x = \mathsf{nil}$

$\mathsf{zlist}^n(x) \triangleq \exists y . x \mapsto (c_0, y) * \mathsf{zlist}^{n-1}(y)$

Fig. 2. Program loops

We compared our implementation with the results of applying the CVC4 decision procedure for the quantifier-free fragment of SL [25] to a variant of the

[6] Available at http://cvc4.cs.nyu.edu/web/.

[7] The CVC4 binary and examples used in these experiments are available at http://cs.uiowa.edu/~ajreynol/VMCAI2017-seplog-epr.

benchmarks, obtained by manual quantifier instantiation, as follows. Consider checking the validity of the entailment $\exists \mathbf{x} \,.\, \phi(\mathbf{x}) \models_{\mathsf{SL}} \exists \mathbf{y} \,.\, \psi(\mathbf{y})$, which is equivalent to the unsatisfiability of the formula $\exists \mathbf{x} \forall \mathbf{y} \,.\, \phi(\mathbf{x}) \wedge \neg\psi(\mathbf{y})$. We first check the satisfiability of ϕ. If ϕ is not satisfiable, the entailment holds trivially, so let us assume that ϕ has a model. Second, we check the satisfiability of $\phi \wedge \psi$. Again, if this is unsatisfiable, the entailment cannot hold, because there exists a model of ϕ which is not a model of ψ. Else, if $\phi \wedge \psi$ has a model, we add an equality $x = y$ for each pair of variables $(x, y) \in \mathbf{x} \times \mathbf{y}$ that are mapped to the same term in this model, the result being a conjunction $E(\mathbf{x}, \mathbf{y})$ of equalities. Finally, we check the satisfiability of the formula $\phi \wedge \neg\psi \wedge E$. If this formula is unsatisfiable, the entailment is valid, otherwise, the check is inconclusive. The times in Table 1 correspond to checking satisfiability of $\exists \mathbf{x} \forall \mathbf{y} \,.\, \phi(\mathbf{x}) \wedge \neg\psi(\mathbf{y})$ using the solve procedure (Fig. 1), compared to checking satisfiability of $\phi \wedge \neg\psi \wedge E$, where E is manually generated.

In the first set of experiments (Table 1) we have considered inductive predicates commonly used as verification benchmarks [27]. Here we check the validity of the entailment between lhs and rhs, where both predicates are unfolded $n = 1, 2, 3, 4, 8$ times. The entailment between pos_2^1 and neg_4^1 is skipped because it is not valid (since the negated formula is satisfiable, we cannot generate the manual instantiation).

The second set of experiments considers the verification conditions of the forms $\varphi \Rightarrow \mathsf{wp}(\mathbf{l}, \phi)$ and $\varphi \Rightarrow \mathsf{wp}^n(\mathbf{l}, \phi)$, where $\mathsf{wp}(\mathbf{l}, \phi)$ denotes the weakest precondition of the SL formula ϕ with respect to the sequence of statements $\mathbf{l}$, and $\mathsf{wp}^n(\mathbf{l}, \phi) = \mathsf{wp}(\mathbf{l}, \ldots \mathsf{wp}(\mathbf{l}, \mathsf{wp}(\mathbf{l}, \phi)) \ldots)$ denotes the iterative application of the weakest precondition n times in a row. We consider the loops depicted in Fig. 2, where, for each loop $\mathbf{l}$, we consider the variant $\mathbf{zl}$ as well, which tests that the data values contained within the memory cells are equal to a constant c_0 of sort Loc, by the assertions on line 2. The postconditions are specified by finite unfoldings of the inductive predicates list and zlist.

We observed that, compared to checking the manual instantiation, the fully automated solver was less than 0.5 s slower on 72% of the test cases, and less than 1 s slower on 79% of the test cases. The automated solver experienced 3 timeouts, where the manual instantiation succeeds (for $\widehat{\mathsf{tree}}$ vs. tree with $n = 8$, $\widehat{\mathsf{ts}}$ vs. ts with $n = 3$, and $\mathsf{list}^n(u) * \mathsf{list}^0(v)$ vs. $\mathsf{wp}^n(\mathbf{rev}, u = \mathsf{nil} \wedge \mathsf{list}^n(v))$ with $n = 8$). These timeouts are caused by the first call to the quantifier-free SL decision procedure, which fails to produce a model in less than 300 s (time not accounted for in the manually produced instance of the problem).

6 Conclusions and Future Work

We present theoretical and practical results for the existence of effective decision procedures for the fragment of Separation Logic obtained by restriction of formulae to quantifier prefixes in the set $\exists^*\forall^*$. The theoretical results range from undecidability, when the set of memory locations is taken to be the set of integers and linear arithmetic constraints are allowed, to PSPACE-completeness, when

Table 1. Experimental results

lhs	rhs		$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 8$
Unfoldings of inductive predicates							
$\widehat{\mathsf{ls}}(x,y) \triangleq \mathsf{emp} \wedge x = y \vee \exists z \,.\, x \neq y \wedge x \mapsto z * \widehat{\mathsf{ls}}(z,y)$	$\mathsf{ls}(x,y) \triangleq \mathsf{emp} \wedge x = y \vee \exists z \,.\, x \mapsto z * \mathsf{ls}(z,y)$	solve	<0.01 s	0.02 s	0.03 s	0.05 s	0.21 s
		manual	<0.01 s	<0.01 s	<0.01 s	<0.01 s	<0.01 s
$\widehat{\mathsf{tree}}(x) \triangleq \mathsf{emp} \wedge x = \mathsf{nil} \vee \exists l \exists r \,.\, l \neq r \wedge x \mapsto (l,r) * \mathsf{tree}(l) * \mathsf{tree}(r)$	$\mathsf{tree}(x) \triangleq \mathsf{emp} \wedge x = \mathsf{nil} \vee \exists l \exists r \,.\, x \mapsto (l,r) * \mathsf{tree}(l) * \mathsf{tree}(r)$	solve	<0.01 s	0.04 s	1.43 s	23.42 s	>300 s
		manual	<0.01 s	<0.01 s	<0.01 s	<0.01 s	0.09 s
$\widehat{\mathsf{ts}}(x,a) \triangleq \mathsf{emp} \wedge x = \mathsf{nil} \vee \exists l \exists r \,.\, x \neq y \wedge x \mapsto (l,r) * \widehat{\mathsf{ts}}(l,y) * \mathsf{tree}(r) \vee \exists l \exists r \,.\, x \neq y \wedge x \mapsto (l,r) * \mathsf{tree}(l) * \widehat{\mathsf{ts}}(r,y)$	$\mathsf{ts}(x,a) \triangleq \mathsf{emp} \wedge x = \mathsf{nil} \vee \exists l \exists r \,.\, \wedge x \mapsto (l,r) * \mathsf{ts}(l,y) * \mathsf{tree}(r) \vee \exists l \exists r \,.\, \wedge x \mapsto (l,r) * \mathsf{tree}(l) * \mathsf{ts}(r,y)$	solve	<0.01 s	0.81 s	>300 s	>300 s	>300 s
		manual	<0.01 s	0.03 s	103.89 s	>300 s	>300 s
$\mathsf{pos}_1(x,a) \triangleq x \mapsto a \vee \exists y \exists b \,.\, x \mapsto a * \mathsf{pos}_1(y,b)$	$\mathsf{neg}_1(x,a) \triangleq \neg x \mapsto a \vee \exists y \exists b \,.\, x \mapsto a * \mathsf{neg}_1(y,b)$	solve	0.34 s	0.01 s	0.31 s	0.76 s	21.19 s
		manual	0.04 s	0.05 s	0.08 s	0.12 s	0.53 s
$\mathsf{pos}_1(x,a) \triangleq x \mapsto a \vee \exists y \exists b \,.\, x \mapsto a * \mathsf{pos}_1(y,b)$	$\mathsf{neg}_2(x,a) \triangleq x \mapsto a \vee \exists y \exists b \,.\, \neg x \mapsto a * \mathsf{neg}_2(y,b)$	solve	0.03 s	0.12 s	0.23 s	0.46 s	3.60 s
		manual	0.05 s	0.08 s	0.08 s	0.12 s	0.54 s
$\mathsf{pos}_2(x,a) \triangleq x \mapsto a \vee \exists y \,.\, x \mapsto a * \mathsf{pos}_2(a,y)$	$\mathsf{neg}_3(x,a) \triangleq \neg x \mapsto a \vee \exists y \,.\, x \mapsto a * \mathsf{neg}_3(a,y)$	solve	0.04 s	0.13 s	0.28 s	0.48 s	4.20 s
		manual	0.01 s	0.03 s	0.05 s	0.09 s	0.45 s
$\mathsf{pos}_2(x,a) \triangleq x \mapsto a \vee \exists y \,.\, x \mapsto a * \mathsf{pos}_2(a,y)$	$\mathsf{neg}_4(x,a) \triangleq x \mapsto a \vee \exists y \,.\, \neg x \mapsto a * \mathsf{neg}_4(a,y)$	solve	—	0.08 s	0.15 s	0.26 s	1.33 s
		manual	—	0.03 s	0.06 s	0.09 s	0.46 s
Verification conditions							
$\mathsf{list}^n(w)$	$\mathsf{wp}(\mathbf{disp}, \mathsf{list}^{n-1}(w))$	solve	0.01 s	0.03 s	0.08 s	0.19 s	1.47 s
		manual	<0.01 s	0.01 s	0.02 s	0.05 s	0.26 s
$\mathsf{list}^n(w)$	$\mathsf{wp}^n(\mathbf{disp}, \mathsf{emp} \wedge w = \mathsf{nil})$	solve	0.01 s	0.06 s	0.17 s	0.53 s	7.08 s
		manual	<0.01 s	0.02 s	0.08 s	0.14 s	2.26 s
$\mathsf{zlist}^n(w)$	$\mathsf{wp}(\mathbf{zdisp}, \mathsf{zlist}^{n-1}(w))$	solve	0.04 s	0.05 s	0.09 s	0.19 s	1.25 s
		manual	<0.01 s	0.01 s	0.02 s	0.04 s	0.29 s
$\mathsf{zlist}^n(w)$	$\mathsf{wp}^n(\mathbf{zdisp}, \mathsf{emp} \wedge w = \mathsf{nil})$	solve	0.01 s	0.10 s	0.32 s	0.87 s	11.88 s
		manual	0.01 s	0.02 s	0.07 s	0.15 s	2.20 s
$\mathsf{list}^n(u) * \mathsf{list}^0(v)$	$\mathsf{wp}(\mathbf{rev}, \mathsf{list}^{n-1}(u) * \mathsf{list}^1(v))$	solve	0.38 s	0.06 s	0.11 s	0.16 s	0.56 s
		manual	0.07 s	0.03 s	0.07 s	0.11 s	0.43 s
$\mathsf{list}^n(u) * \mathsf{list}^0(v)$	$\mathsf{wp}^n(\mathbf{rev}, u = \mathsf{nil} \wedge \mathsf{list}^n(v))$	solve	0.38 s	0.07 s	0.30 s	68.68 s	>300 s
		manual	0.08 s	0.06 s	0.11 s	0.23 s	1.79 s
$\mathsf{zlist}^n(u) * \mathsf{zlist}^0(v)$	$\mathsf{wp}(\mathbf{zrev}, \mathsf{zlist}^{n-1}(u) * \mathsf{zlist}^1(v))$	solve	0.22s	0.07 s	0.15 s	0.21 s	0.75 s
		manual	0.04 s	0.02 s	0.04 s	0.06 s	0.31 s
$\mathsf{zlist}^n(u) * \mathsf{zlist}^0(v)$	$\mathsf{wp}^n(\mathbf{zrev}, u = \mathsf{nil} \wedge \mathsf{zlist}^n(v))$	solve	0.23 s	0.09 s	0.17 s	0.30 s	2.06 s
		manual	0.04 s	0.02 s	0.05 s	0.09 s	0.48 s

locations and data in the cells belong to an uninterpreted sort, equipped with equality only. We have implemented a decision procedure for the latter case in the CVC4 SMT solver, using an effective counterexample-driven instantiation of the universal quantifiers. The procedure is shown to be sound, complete and termination is guaranteed when the input belongs to a decidable fragment of SL.

As future work, we aim at refining the decidability chart for $\exists^*\forall^*\mathsf{SL}(T)_{\mathsf{Loc},\mathsf{Data}}$, by considering the case where the locations are interpreted as integers, with weaker arithmetics, such as sets of difference bounds, or octagonal constraints. These results are likely to extend the application range of our tool, to e.g. solvers working on SL with inductive definitions and data constraints. The current implementation should also benefit from improvements of the underlying quantifier-free SL and set theory solvers.

References

1. Albargouthi, A., Berdine, J., Cook, B., Kincaid, Z.: Spatial interpolants. In: Vitek, J. (ed.) ESOP 2015. LNCS, vol. 9032, pp. 634–660. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46669-8_26
2. Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanović, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 171–177. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22110-1_14
3. Baumgartner, P., Fuchs, A., Tinelli, C.: Implementing the model evolution calculus. Int. J. Artif. Intell. Tools **15**(1), 21–52 (2006)
4. Brochenin, R., Demri, S., Lozes, E.: On the almighty wand. Inf. Comput. **211**, 106–137 (2012)
5. Brotherston, J., Simpson, A.: Sequent calculi for induction and infinite descent. J. Logic Comput. **21**(6), 1177–1216 (2011)
6. 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). doi:10.1007/978-3-642-20398-5_33
7. Calcagno, C., Yang, H., O'Hearn, P.W.: Computability and complexity results for a spatial assertion language for data structures. In: Hariharan, R., Vinay, V., Mukund, M. (eds.) FSTTCS 2001. LNCS, vol. 2245, pp. 108–119. Springer, Heidelberg (2001). doi:10.1007/3-540-45294-X_10
8. Demri, S., Deters, M.: Two-variable separation logic and its inner circle. ACM Trans. Comput. Logic **16**(2) (2015). Article no. 15
9. Demri, S., Galmiche, D., Larchey-Wendling, D., Méry, D.: Separation logic with one quantified variable. In: Hirsch, E.A., Kuznetsov, S.O., Pin, J.É., Vereshchagin, N.K. (eds.) CSR 2014. LNCS, vol. 8476, pp. 125–138. Springer, Heidelberg (2014). doi:10.1007/978-3-319-06686-8_10
10. Dudka, K., Peringer, P., Vojnar, T.: Predator: a practical tool for checking manipulation of dynamic data structures using separation logic. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 372–378. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22110-1_29

11. Galmiche, D., Méry, D.: Tableaux and resource graphs for separation logic. J. Logic Comput. **20**(1), 189–231 (2010)
12. Ganzinger, H., Hagen, G., Nieuwenhuis, R., Oliveras, A., Tinelli, C.: DPLL(T): fast decision procedures. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 175–188. Springer, Berlin (2004). doi:10.1007/978-3-540-27813-9_14
13. Ge, Y., Moura, L.: Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 306–320. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02658-4_25
14. Halpern, J.Y.: Presburger arithmetic with unary predicates is π_1^1 complete. J. Symbolic Logic **56**(2), 637–642 (1991)
15. Ishtiaq, S.S., O'Hearn, P.W.: Bi as an assertion language for mutable data structures. ACM SIGPLAN Not. **36**, 14–26 (2001)
16. Korovin, K.: iProver - an instantiation-based theorem prover for first-order logic (system description). In: Proceedings of 4th International Joint Conference on Automated Reasoning, IJCAR 2008, Sydney, Australia, 12–15 August 2008, pp. 292–298 (2008)
17. Lewis, H.R.: Complexity results for classes of quantificational formulas. J. Comput. Syst. Sci. **21**(3), 317–353 (1980)
18. Matiyasevich, Y.: Enumerable sets are diophantine. J. Sovietic Math. **11**, 354–358 (1970)
19. Nguyen, H.H., Chin, W.-N.: Enhancing program verification with lemmas. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 355–369. Springer, Heidelberg (2008). doi:10.1007/978-3-540-70545-1_34
20. Piskac, R., de Moura, L.M., Bjørner, N.: Deciding effectively propositional logic using DPLL and substitution sets. J. Autom. Reasoning **44**(4), 401–424 (2010)
21. 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). doi:10.1007/978-3-642-39799-8_54
22. 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). doi:10.1007/978-3-319-08867-9_47
23. Reynolds, A., Deters, M., Kuncak, V., Tinelli, C., Barrett, C.W.: Counter example-guided quantifier instantiation for synthesis in SMT. In: Proceedings of Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, 18–24 July 2015, Part II, pp. 198–216 (2015)
24. Reynolds, A., Iosif, R., Serban, C.: Reasoning in the Bernays-Schoenfinkel-Ramsey fragment of separation logic. CoRR abs/1610.04707 (2016). http://arxiv.org/abs/1610.04707
25. Reynolds, A., Iosif, R., Serban, C., King, T.: A decision procedure for separation logic in SMT. In: Proceedings of 14th International Symposium on Automated Technology for Verification and Analysis, ATVA 2016, Chiba, Japan, 17–20 October 2016, pp. 244–261 (2016)
26. Reynolds, A., Tinelli, C., Goel, A., Krstić, S.: Finite model finding in SMT. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 640–655. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39799-8_42
27. Sighireanu, M., Cok, D.: Report on SL-COMP 2014. J. Satisfiability Boolean Model. Comput. **1** (2014)

28. Toubhans, A., Chang, B.-Y.E., Rival, X.: An abstract domain combinator for separately conjoining memory abstractions. In: Müller-Olm, M., Seidl, H. (eds.) SAS 2014. LNCS, vol. 8723, pp. 285–301. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10936-7_18
29. Voigt, M., Weidenbach, C.: Bernays-Schönfinkel-Ramsey with simple bounds is nexptime-complete. CoRR abs/1501.07209 (2015)
30. Yang, H.: Local reasoning for stateful programs. Ph.D. thesis, University of Illinois at Urbana-Champaign (2001)