On the Almighty Wand*

Rémi Brochenin & Stéphane Demri & Etienne Lozes

LSV, ENS Cachan, CNRS, INRIA Saclay, France

Abstract. We investigate decidability, complexity and expressive power issues for (first-order) separation logic with one record field (herein called SL) and its fragments. SL can specify properties about the memory heap of programs with singly-linked lists. Separation logic with *two* record fields is known to be undecidable by reduction of finite satisfiability for classical predicate logic with one binary relation. Surprisingly, we show that second-order logic is as expressive as SL and as a by-product we get undecidability of SL. This is refined by showing that SL without the separating conjunction is as expressive as SL, whence undecidable too. As a consequence of this deep result, in SL the magic wand can simulate the separating conjunction. By contrast, we establish that SL without the magic wand is decidable with non-elementary complexity by reduction from satisfiability for the first-order theory over finite words. Equivalence between second-order logic and separation logic extends to the case with more than one selector.

1 Introduction

Separation logic. Programming languages with pointer variables have seldom mechanisms to detect errors. An inappropriate management of memory is the source of numerous security holes. Prominent logics for analysing such pointer programs include separation logic [Rey02], pointer assertion logic PAL [JJKS97], TVLA [LAS00], alias logic [BIL04], BI (Bunched Implication) [IO01] and LRP (logic of reachable patterns) [YRS+05] to quote a few examples. Separation logic is an assertion language used in Hoare-like proof systems [Rey02] that are dedicated to verify programs manipulating heaps. Any procedure mechanizing the proof search requires subroutines that check satisfiability of formulae from the assertion language. The main concern of the paper is to analyze the expressive power of the assertion language and its decidability status. Recall that separation logic contains a structural *separation* connective and its adjoint (the separating implication $-\!\!*$, also known as the *magic wand*). Concise and modular proofs can be derived using these connectives, since they can express properties such as non-aliasing and disjoint concurrency. In this perspective, the models of separation logic are pairs made of a store (variable valuation) and a memory heap (partial function with finite domain) that are understood as memory states.

The decidability of the satisfiability problem for separation logic has been intensively studied so far: first-order separation logic with at least two selectors (record fields) is known to be undecidable [CYO01b] by reduction of finite satisfiability for

* Supported by RNTL project AVERILES – R. Brochenin is supported by a DGA/CNRS fellowship.

classical predicate logic with one binary relation [Tra50] (even with no separating connectives). Decidable fragments have been introduced and investigated, see e.g. [BCO04]; such fragments involve some specialized predicates for lists or trees, and some restrictions on first-order quantification. The complexity of various quantifier-free fragments has also been characterized in the past, see e.g. [CYO01b,CYO01a,Rey02,BDL07]. As far as the expressive power is concerned, propositional separation logic can be naturally reduced to propositional calculus or to a fragment to Presburger arithmetic, see [Loz04,CGH05]. On several spatial logics, the adjunct elimination property holds in the absence of first-order quantification, but does not extend to the first-order case, see e.g. [DGG04,Loz05]. To summarize, all the proof techniques used in these works do not adapt to first-order separation logic over the class of models with only one selector, for which both decidability status and the characterization of expressiveness are open.

Our motivations. A long-standing question about separation logic is how it compares with second-order logic. This is a very natural question since separating conjunction and its adjoint are essentially second-order connectives (see also a similar concern on graphs with spatial logics [DGG07]). Moreover, many properties on heaps require second-order logic, for instance to express recursive predicates, or lists and trees properties. In a sense, being able to distinguish the expressive power of these two logical formalisms would justify the use of separation logic. An attempt to make such a comparison can be found in [KR04] but the models are quite different from those considered herein (relational structures with no constraint on the finiteness of the domain of relations). In this paper, our aim is to investigate decidability, complexity and expressive power issues for first-order separation logic with one selector (record field) and its fragments when the models are the standard memory states.

Our contributions. We show that first-order separation logic with one selector (called herein SL) is as expressive as second-order logic over the class of memory states. As a by product, we get that even in presence of a unique selector, 1) first-order separation logic is undecidable (solving an open problem stated in [GM08]), 2) it may express heap properties involving several selectors (passing via second-order logic). This is refined by showing that SL without the separating conjunction is as expressive as SL, whence undecidable too. Our proof also shows that the two formalisms have the same conciseness modulo logspace translations. As a consequence of this deep result, in SL the magic wand can simulate the separating conjunction. So, the magic wand is very powerful, which is interesting because very often this connective, that is not very natural we admit, is excluded from studied fragments of separation logic. Equivalence between second-order logic and separation logic extends to the case with more than one selector by simple adaptation of the one selector case. By contrast, in [KR04], the separating conjunction is sufficient to capture second-order logic but on a different class of models. We also establish that SL without the magic wand is decidable with non-elementary complexity by reduction from satisfiability for the first-order theory over finite words [Sto74] (this result holds already with three variables). Decidability is shown by reduction into weak monadic second-order theory of one unary function that is shown decidable in [Rab69]. It is worth noting that even though the first-order theory of one unary function is known to be not elementary recursive [BGG97], we cannot take advantage of this result since in our models the domain of the unary function is

necessarily finite and finiteness cannot be expressed in most first-order dialects. As a by-product, we obtain that the entailment problem considered in [BCO04] for a fragment of separation logic with one selector is decidable.
Related work. As seen previously, heap properties are formalized in various logical languages [JJKS97,LAS00,Rey02,BIL04,YRS+05] and separation logic is just one prominent of these logics. Verification methods and logics for verifying programs with singly-linked lists can be found for instance in [BCO04,BHMV05,RZ06]. From another perspective, the relationships between logics on graphs with separating features and second-order logic can be found in [DGG07]. Finally, we would like to mention that sabotage modal logics (SML), see e.g. in [LR03], have also the ability to modify the model under evaluation. So far, we are not aware of any work relating separation logic and SML.

Omitted proofs can be found in [BDL08].

2 Preliminaries

In this section, we recall the definition of first-order separation logic with one selector (called herein SL) and second-order logic (SO) over the same class of structures. We introduce a formal notion of expressiveness, provide examples of properties that can be expressed in SL and present a straightforward encoding of SL into a fragment of SO.

2.1 Separation logic and second-order logic

Memory states. Memory states are models for all the logical formalisms we consider herein. They represent the states of the memory for programs manipulating lists. Let Loc be a countably infinite set of *locations* ranged over with $l, l', \ldots$ that represents the set of addresses. A memory state is composed of a pair made of a *store* and a *heap*. Let Var be a countably infinite set of (first-order) *variables* $\mathtt{x}, \mathtt{y}, \mathtt{z}, \ldots$. A *memory state* (also called a *model* in the rest of the document) is a pair (s, h) such that

- s is a variable valuation of the form $s : \mathtt{Var} \to \mathtt{Loc}$ (store),
- h is a partial function $h : \mathtt{Loc} \rightharpoonup \mathtt{Loc}$ with finite domain (heap). We write $\mathtt{dom}(h)$ to denote its domain and $\mathtt{ran}(h)$ to denote its range.

Given a finite set X of variables (for instance occurring in a given formula), we can assume that a model is finite by restricting the domain of the store to X. The variables in Var can be viewed as programming variables, the domain of h as the set of addresses of allocated cells, and $h(l)$ as the value held by the cell at the address l. Two heaps h_1, h_2 are said to be *disjoint*, noted $h_1 \perp h_2$, if their domains are disjoint; when this holds, we write $h_1 * h_2$ to denote the disjoint union $h_1 \uplus h_2$. Given a memory state (s, h) and a location l we write $\widetilde{\sharp} l$ to denote the cardinal of the set $\{l' \in \mathtt{Loc} : h(l') = l\}$ (number of *predecessors* of the location l in (s, h)). A location l' is a *descendant* of l if there is $n \geq 0$ such that $h^n(l) = l'$ ($h^n(l)$ is not always defined).
Formulae in SL *and* SO. Formulae of first-order separation logic with one selector SL are defined by the grammar $\phi := \neg\phi \mid \phi \wedge \phi \mid \exists \mathtt{x}.\phi \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \mathtt{x} = \mathtt{y} \mid \phi * \phi \mid \phi \mathrel{-\!\!*} \phi$. The connective $*$ is called *separating conjunction* whereas the adjoint operator $\mathrel{-\!\!*}$ is

usually called the *magic wand*. We will make use of standard notations for the derived connectives $\forall, \vee, \Rightarrow, \Leftrightarrow$... We also introduce a slight variant of the dual connective for the magic wand, also called the *septraction*: $\phi \mathrel{-\!\!\!\!\!\!*} \psi$ is defined as the formula $\neg((\phi) \mathrel{-\!\!*} (\neg(\psi)))$. We write $\mathtt{FV}(\phi)$ to denote the set of free variables occurring in ϕ and $\mathtt{SL}(*)$ [resp. $\mathtt{SL}(\mathrel{-\!\!*})$] to denote the restriction of $\mathtt{SL}$ without the magic wand [resp. the separating conjunction].

In order to define formulae in $\mathtt{SO}$, we consider a family $\mathtt{VAR} = (\mathtt{VAR}^i)_{i\geq 0}$ of second-order variables, denoted by $\mathtt{P}, \mathtt{Q}, \mathtt{R}, \ldots$ that will be interpreted as finite relations over $\mathtt{Loc}$ ($= \mathtt{Val} = \mathbb{N}$). Each variable in $\mathtt{VAR}^i$ is interpreted as an i-ary relation. An *environment* $\mathcal{E}$ is an interpretation of the second-order variables such that for every $\mathtt{P} \in \mathtt{VAR}^i$, $\mathcal{E}(\mathtt{P})$ is a finite subset of $\mathtt{Loc}^i$. Since we require finiteness of models, the version of second-order logics we shall consider is usually called *weak*.

Formulae of (weak) second-order logic $\mathtt{SO}$ are defined by the grammar $\phi := \neg\phi \mid \phi \wedge \phi \mid \exists \mathtt{x}.\phi \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \mathtt{x} = \mathtt{y} \mid \exists \mathtt{P}.\phi \mid \mathtt{Q}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$, where $\mathtt{P}, \mathtt{Q}$ are second-order variables and $\mathtt{Q} \in \mathtt{VAR}^n$. We write $\mathtt{MSO}$ [resp. $\mathtt{DSO}$] to denote the restriction of $\mathtt{SO}$ to second-order variables in $\mathtt{VAR}^1$ [resp. $\mathtt{VAR}^2$]. As usual, a *sentence* is defined as a formula with no free occurrence of second-order variables.

Satisfaction relations for $\mathtt{SL}$ *and* $\mathtt{SO}$. The logics $\mathtt{SL}$ and $\mathtt{SO}$ share the same class of models, namely the set of memory states. The satisfaction relation for $\mathtt{SO}$ is defined below with argument an environment $\mathcal{E}$ (below $\mathtt{P} \in \mathtt{VAR}^n$).

$$
\begin{array}{lll}
(s,h),\mathcal{E} \models \exists \mathtt{P}.\ \phi & \text{iff} & \text{there is a finite subset } \mathcal{R} \text{ of } \mathtt{Loc}^n, \\
 & & \text{such that } (s,h),\mathcal{E}[\mathtt{P} \mapsto \mathcal{R}] \models \phi \\
(s,h),\mathcal{E} \models \mathtt{P}(\mathtt{x}_1, \cdots, \mathtt{x}_n) & & \\
 & \text{iff} & (s(\mathtt{x}_1), \ldots, s(\mathtt{x}_n)) \in \mathcal{E}(\mathtt{P}) \\
(s,h),\mathcal{E} \models \neg\phi & \text{iff} & \text{not } (s,h),\mathcal{E} \models \phi \\
(s,h),\mathcal{E} \models \phi \wedge \psi & \text{iff} & (s,h),\mathcal{E} \models \phi \text{ and } (s,h),\mathcal{E} \models \psi \\
(s,h),\mathcal{E} \models \exists \mathtt{x}.\ \phi & \text{iff} & \text{there is } l \in \mathtt{Loc} \text{ such that } (s[\mathtt{x} \mapsto l],h),\mathcal{E} \models \phi \\
(s,h),\mathcal{E} \models \mathtt{x} \hookrightarrow \mathtt{y} & \text{iff} & h(s(\mathtt{x})) = s(\mathtt{y}) \\
(s,h),\mathcal{E} \models \mathtt{x} = \mathtt{y} & \text{iff} & s(\mathtt{x}) = s(\mathtt{y})
\end{array}
$$

As usual, when ϕ is a sentence, we write $(s,h) \models \phi$ to denote $(s,h),\mathcal{E} \models \phi$ for any environment $\mathcal{E}$ since $\mathcal{E}$ has no influence on the satisfaction of ϕ. The satisfaction relation for $\mathtt{SL}$ is defined without any environment (or equivalently with no influence of the environment) whereas the clauses that are specific to $\mathtt{SL}$ are the following ones:

$$
\begin{array}{lll}
(s,h) \models \phi_1 * \phi_2 & \text{iff} & \text{there are two heaps } h_1, h_2 \\
 & & \text{such that } h = h_1 * h_2 \text{ and } (s,h_i) \models \phi_i \ (i = 1,2) \\
(s,h) \models \phi_1 \mathrel{-\!\!*} \phi_2 & \text{iff} & \text{for all heaps } h' \bot h, \\
 & & \text{if } (s,h') \models \phi_1 \text{ then } (s, h' * h) \models \phi_2.
\end{array}
$$

So, $(s,h) \models \phi_1 \mathrel{-\!\!\!\!\!\!*} \phi_2$ iff there is $h' \perp h$ such that $(s,h') \models \phi_1$ and $(s, h * h') \models \phi_2$. Validity and satisfiability problems are defined in the usual way. The connective $\mathrel{-\!\!*}$ is the *adjunct* of $*$, meaning that $(\phi * \psi) \Rightarrow \varphi$ is valid iff $\phi \Rightarrow (\psi \mathrel{-\!\!*} \varphi)$ is valid. Observe that $*$ and $\mathrel{-\!\!*}$ are not interdefinable since typically the formula $((\phi * \psi) \Rightarrow \varphi) \Leftrightarrow (\phi \Rightarrow (\psi \mathrel{-\!\!*} \varphi))$ is not valid. This shall be strengthened in the sequel by establishing that $\mathtt{SL}(*)$ is decidable whereas $\mathtt{SL}(\mathrel{-\!\!*})$ is not.

Let F and F′ be two fragments of SL or SO. We say that F′ is at least as expressive as F (written F $\sqsubseteq$ F′) whenever for every sentence $\phi \in$ F, there is $\phi' \in$ F′ such that for every model (s,h), we have $(s,h) \models \phi$ iff $(s,h) \models \phi'$. We write F $\equiv$ F′ if F $\sqsubseteq$ F′ and F′ $\sqsubseteq$ F. A translation from F to F′ is a computable function t : F $\to$ F′ such that for every sentence $\phi \in$ F, for every model (s,h), we have $(s,h) \models \phi$ iff $(s,h) \models t(\phi)$.
Arithmetical constraints. Observe that SL does not contain explicitly arithmetical constraints as in [KR03,MBCC07,BIP08]. However, in Section 4 we show how to compare number of predecessors. Similar developments can be performed to compare lengths of lists but this will come as a corollary of the equivalence between SL and SO.
Another model with data. A more realistic approach to model lists consists in considering two selectors. However, SL behaves as separation logic with two selectors for which one selector is never used. Indeed, we already know that an unrestricted use of the two selectors leads to undecidability. In the paper, we show that even SL satisfiability/validity is already undecidable. It is open how to refer to data values while preserving the decidable results for SL fragments. Possible directions consist either in imposing syntactic restrictions (like the guarded fragment for classical predicate logic) or in forbidding a direct access to data values but allowing predicates of the form "there is a list from x to y with increasing data values" for instance.
More than one selector. It is easy to extend the above definitions to the case with $k \geq 1$ selectors that is also used in the literature. A heap h becomes then a partial function $h : \mathtt{Loc} \rightharpoonup \mathtt{Loc}^k$ with finite domain and atomic formulae of the form $\mathtt{x} \hookrightarrow \mathtt{y}$ are replaced by $\mathtt{x} \hookrightarrow \mathtt{y}_1, \ldots, \mathtt{y}_k$. We write kSL [resp. kSO] to denote the variant of SL [resp. SO] with k selectors. Obviously 1SL [resp. 1SO] corresponds to SL [resp. SO]. We write $k\mathtt{SO}^{k'}$ to denote the restriction of kSO to second-order variables in $\mathtt{VAR}^{k'}$.

2.2 A selection of properties

We present below a series of properties that can be expressed in SL$(*)$.

- The value of x is in the domain of the heap: $\mathtt{alloc}\,(\mathtt{x}) \triangleq \exists \mathtt{y}.\mathtt{x} \hookrightarrow \mathtt{y}$.
- The domain of the heap is restricted to the value of x, and maps it to that of y: $\mathtt{x} \mapsto \mathtt{y} \triangleq \mathtt{x} \hookrightarrow \mathtt{y} \wedge \neg\exists \mathtt{y}.\mathtt{y} \neq \mathtt{x} \wedge \mathtt{alloc}\,(\mathtt{y})$.
- The domain of the heap is empty: $\mathtt{emp} \triangleq \neg\exists \mathtt{x}.\mathtt{alloc}\,(\mathtt{x})$.

Predecessors and special nodes. A *predecessor* of the variable x in the model (s,h) is a location l such that $h(l) = s(\mathtt{x})$. There are formulae in SL$(*)$, namely $\sharp\mathtt{x} \geq n$ and $\sharp\mathtt{x} = n$, such that $\sharp\mathtt{x} \geq n$ [resp. $\sharp\mathtt{x} = n$] holds true exactly in models such that x has more than n predecessors [resp. exactly n predecessors]. For instance, $\sharp\mathtt{x} \geq n$ can defined in the following ways:

$$\overbrace{(\exists \mathtt{y}.\ \mathtt{y} \hookrightarrow \mathtt{x}) * \cdots * (\exists \mathtt{y}.\ \mathtt{y} \hookrightarrow \mathtt{x})}^{n \text{ times}} * \top \quad \text{or} \quad \exists \mathtt{x}_1, \ldots, \mathtt{x}_n.\ \bigwedge_{i \neq j} \mathtt{x}_i \neq \mathtt{x}_j \wedge \bigwedge_{i=1}^{n} \mathtt{x}_i \hookrightarrow \mathtt{x}$$

We define an *extremity* as a location l in a model such that l has at least one predecessor and no predecessor of l has a precedessor. The following formula states that $s(\mathtt{x})$ is an extremity: $\mathtt{extr}(\mathtt{x}) \triangleq (\neg\exists \mathtt{y}.\ \mathtt{y} \hookrightarrow \mathtt{x} \wedge \exists \mathtt{z}.\mathtt{z} \hookrightarrow \mathtt{y}) \wedge \exists \mathtt{y}.\ \mathtt{y} \hookrightarrow \mathtt{x}$.

Reachability and list predicates Reachability in a graph is a standard property that can be expressed in monadic second-order logic. In separation logic, very often a built-in predicate for lists is added, sometimes noted $\mathsf{ls}(\mathtt{x},\mathtt{y})$. Adapting some technique used in the graph logics [DGG07], we show below how this very predicate can be expressed in $\mathtt{SL}(*)$ as well as the reachability predicate $\mathtt{x}{\rightarrow^*}\mathtt{y}$.

A *cyclic list* in a model (s,h) is a non-empty finite sequence $l_1,\ldots,l_n$ $(n \geq 1)$ of locations such that $h(l_n) = l_1$ and for every $i \in \{1,\ldots,n-1\}$, $h(l_i) = l_{i+1}$. A model (s,h) is a *list segment* between $\mathtt{x}$ and $\mathtt{y}$ if there are locations $l_1,\ldots,l_n$ $(n \geq 2)$ such that $s(\mathtt{x}) = l_1$, $s(\mathtt{y}) = l_n$, $l_1 \neq l_n$, $\mathtt{dom}(h) = \{l_1,\ldots,l_{n-1}\}$, and for every $i \in \{1,\ldots,n-1\}$, $h(l_i) = l_{i+1}$. Consider the formula below

$$\begin{array}{rl} \mathtt{x} \xrightarrow{\circlearrowleft}{}^{+} \mathtt{y} & \triangleq \sharp\mathtt{x} = 0 \wedge \mathtt{alloc}\,(\mathtt{x}) \wedge \sharp\mathtt{y} = 1 \wedge \neg\mathtt{alloc}\,(\mathtt{y}) \\ & \quad \wedge \forall \mathtt{z}.\, \mathtt{z} \neq \mathtt{y} \Rightarrow (\sharp\mathtt{z} = 1 \Rightarrow \mathtt{alloc}\,(\mathtt{z})) \wedge \forall \mathtt{z}.\, \sharp\mathtt{z} \leq 1 \end{array}$$

Lemma 1. *Let (s,h) be a model. $(s,h) \models \mathtt{x} \xrightarrow{\circlearrowleft}{}^{+} \mathtt{y}$ iff h is undefined for $s(\mathtt{y})$ and there are unique heaps h_1, h_2 such that $h_1 * h_2 = h$, (s,h_1) is a list segment between $\mathtt{x}$ and $\mathtt{y}$ and (s,h_2) can be decomposed uniquely as a set of cyclic lists.*

We introduce additional formulae: $\mathsf{ls}(\mathtt{x},\mathtt{y}) \triangleq \mathtt{x} \xrightarrow{\circlearrowleft}{}^{+} \mathtt{y} \wedge \neg(\mathtt{x} \xrightarrow{\circlearrowleft}{}^{+} \mathtt{y} * \neg\mathsf{emp})$, $\mathtt{x} \rightarrow^{+} \mathtt{y} \triangleq \top * \mathsf{ls}(\mathtt{x},\mathtt{y})$ and $\mathtt{x}{\rightarrow^*}\mathtt{y} \triangleq \mathtt{x} = \mathtt{y} \vee \mathtt{x} \rightarrow^{+} \mathtt{y}$. They express the properties below.

Lemma 2. *Let (s,h) be a model. (I) $(s,h) \models \mathsf{ls}(\mathtt{x},\mathtt{y})$ iff (s,h) is a list segment between $\mathtt{x}$ and $\mathtt{y}$. (II) $(s,h) \models \mathtt{x}{\rightarrow^*}\mathtt{y}$ [resp. $(s,h) \models \mathtt{x}{\rightarrow^+}\mathtt{y}$] iff $\mathtt{y}$ is a descendant [resp. strict descendant] of $\mathtt{x}$.*

2.3 Preliminary translations

Before showing advanced results in the forthcoming sections, we show below how SL can be encoded into SO by simply mimicking the original semantics and how SO can be encoded in its fragment DSO by representing multiedges by finite sets of edges.

Proposition 3. $\mathtt{SL} \sqsubseteq \mathtt{SO} \equiv \mathtt{DSO}$ *via logspace translations.*

Sections 4 and 5 are devoted to prove that $\mathtt{DSO} \sqsubseteq \mathtt{SL}(\mathrel{-\!\!*})$. We will obtain that $\mathtt{SL}(\mathrel{-\!\!*})$, SL, DSO and SO have the same expressive power (via logspace translations). Consequently, this implies undecidability of the validity problem for any of these logics by the undecidability of classical predicate logic with one binary relation [Tra50]. By contrast, we prove below that $\mathtt{SL}(*)$ is decidable.

3 On the complexity of $\mathtt{SL}(*)$

In this section, we show that $\mathtt{SL}(*)$ satisfiability is decidable but with non-elementary recursive complexity (by reduction from first-order theory over finite words).

Lemma 4. MSO *satisfiability is decidable.*

The proof is based on [Rab69,BGG97]. Using a technique similar to the proof of Lemma 4, we can translate $\mathtt{SL}(*)$ into $\mathtt{MSO}$.

Proposition 5. $\mathtt{SL}(*) \sqsubseteq \mathtt{MSO}$ *via a logspace translation.*

We conjecture that $\mathtt{MSO}$ is strictly more expressive than $\mathtt{SL}(*)$, see the related paper [Mar06].

Corollary 6. $\mathtt{SL}(*)$ *satisfiability is decidable.*

In order to show that satisfiability in $\mathtt{SL}(*)$ is not elementary recursive, we explain below how to encode finite words as memory states. Let $\Sigma = \{a_1, \ldots, a_n\}$ be a finite alphabet. A finite word $w = a_{i_1} \cdot a_{i_2} \cdots a_{i_m}$ is usually represented as the first-order structure $(\{1, \ldots, m\}, <, (\mathtt{P}_a)_{a \in \Sigma})$ where $\mathtt{P}_a$ is the set of positions labelled by the letter a. Similarly, the word w can be represented as a memory state (s_w, h_w) in which

- $\mathtt{x}_{beg} \rightarrow^+ \mathtt{x}_{end}$ for which $\mathtt{x}_{beg}$ and $\mathtt{x}_{end}$ are distinguished variables marking respectively, the beginning and the end of the encoding of w,
- the list segment induced from the satisfaction of $\mathtt{x}_{beg} \rightarrow^+ \mathtt{x}_{end}$ has exactly $m+2$ locations, and any location l of position $j \in \{2, \ldots, m+1\}$ in the list segment (hence excluding $s_w(\mathtt{x}_{beg})$ and $s_w(\mathtt{x}_{end})$) has exactly i_j predecessors. Since $s_w(\mathtt{x}_{beg})$ and $s_w(\mathtt{x}_{end})$ do not encode any position in w, there is no constraint on them.

Similarly, any memory state (s, h) containing a list segment between $\mathtt{x}_{beg}$ and $\mathtt{x}_{end}$ and such that any location on the list segment that is different from $s(\mathtt{x}_{beg})$ and $s(\mathtt{x}_{end})$ has at most $\mid \Sigma \mid$ predecessors corresponds to a unique finite word with the above encoding. In this direction, the memory state may contain other dummy locations but they are irrelevant for the representation of the finite word. Moreover, a memory state can encode only one word since $\mathtt{x}_{beg}$ and $\mathtt{x}_{end}$ are end-markers.

Proposition 7. $\mathtt{SL}(*)$ *is not elementary recursive (even its restriction with 5 variables).*

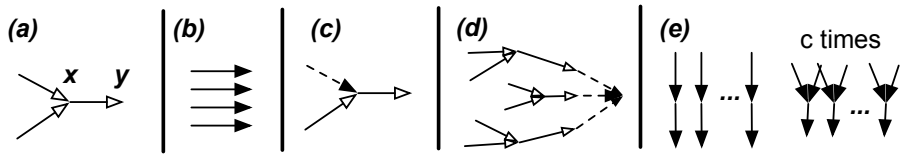
The proof uses the predicate $\mathtt{x} \rightarrow^+ \mathtt{y}$ from Section 2.2. As a corollary, we obtain an alternative decidability proof of the entailment problem for the fragment of $\mathtt{SL}$ considered in [BCO04]. We have established decidability for a fragment of $\mathtt{SL}$ larger than the one considered in [BCO04] (for which the entailment problem is shown in coNP) but of higher complexity.

It is probable that the number of variables can be reduced further while preserving non-elementarity, but it is not very essential at this point, for instance by identifying the limits of the words by patterns (see e.g. Section 4) instead of distinguished variables.

4 Advanced arithmetic constraints

In this section, we show how $\mathtt{SL}(-\!\!*)$ can express properties of the form $\widetilde{\sharp\mathtt{x}} \bowtie \widetilde{\sharp\mathtt{y}} + c$ for any $c \in \mathbb{N}$ and for any relation $\bowtie \in \{=, \geq, \leq\}$ where $\widetilde{\sharp\mathtt{x}}$ denotes the number of predecessors of $s(\mathtt{x})$ in a model. Note that $\widetilde{\sharp\mathtt{x}} \bowtie c$ can be easily expressed in $\mathtt{SL}(-\!\!*)$, even without magic wand. By contrast, expressing a constraint $\widetilde{\sharp\mathtt{x}} \bowtie \widetilde{\sharp\mathtt{y}} + c$ is natural in

Figure 1 Some examples of markers, marked locations, and possible aliasing.



second-order logic. We show below that this can be done also in $\mathtt{SL}(-\!\!*)$. In the sequel, we assume that the current model is denoted by (s, h).

A *marker* in a model (s, h) is a specific pattern in the memory heap that we will intensively use. Formally, a *[resp. strict] marker* in (s, h) is a sequence of distinct locations $l, l_0, \ldots, l_n$ for some $n \geq 0$ such that (a) $h(l_0) = l$ [resp. and $\mathtt{dom}(h) = \{l_0, \ldots, l_n\}$], (b) for $i \in \{1, \ldots, n\}$, $h(l_i) = l_0$ and $\widetilde{\sharp l_i} = 0$ and (c) $\widetilde{\sharp l_0} = n$. We say that this marker is of *degree* n with *endpoint* l. Markers should be thought graphically. Figure 1(a) represents a marker of degree 2 with endpoint $s(\mathtt{y})$.

A model (s, h) is said to be *k-marked* whenever there is no location in $\mathtt{dom}(h)$ that does not belong to a marker of degree k. Moreover it is *strictly* k-marked when no distinct markers share the same endpoint. A model (s, h) is *segmented* whenever $\mathtt{dom}(h) \cap \mathtt{ran}(h) = \emptyset$ and no location has strictly more than one predecessor. Finally, (s, h) is *drown* when no location has one or two predecessors. The lemma below states that many properties involving these notions can be expressed in $\mathtt{SL}(-\!\!*)$.

Lemma 8. *There are formulae* drown, seg, $\mathtt{preM}^2(\mathtt{x})$ *and* $(\mathrm{one} \mid \mathrm{two}[c])$ *($c \geq 0$) in* SL *with no separating connectives such that for every model* (s, h)

(I) $(s, h) \models$ drown *iff* (s, h) *is drown,* **(II)** $(s, h) \models$ seg *iff* (s, h) *is segmented,*
(III) $(s, h) \models \mathtt{preM}^2(\mathtt{x})$ *iff the predecessors of* $s(\mathtt{x})$ *are endpoints of markers of degree 2,*
(IV) $(s, h) \models (\mathrm{one} \mid \mathrm{two}[c])$ *iff there are* h_1, h_2 *such that* $h = h_1 * h_2$, (s, h_1) *is 1-marked and* (s, h_2) *is strictly 2-marked with exactly* c *distinct 2-markers.*

Figure 1(e) contains a model satisfying $(\mathrm{one} \mid \mathrm{two}[c])$ whereas Figure 1(d) presents a location $s(\mathtt{x})$ satisfying $\mathtt{preM}^2(\mathtt{x})$. The above formulae allow to insert in a drown model markers of degree strictly less than 3 and still to safely identify them as markers in the new model. Actually, any location with less than 2 predecessors will necessarily be part of a newly introduced marker. We will assume for now that we are working with a drown model, and later reduce the general case to this one. Observe also if h_1 is segmented and h_2 contains 1-markers, we can obtain 2-markers in $h_1 * h_2$.

We say that two heaps h_1, h_2 are *completely disjoint* if $(\mathtt{dom}(h_1) \cup \mathtt{ran}(h_1)) \cap (\mathtt{dom}(h_2) \cup \mathtt{ran}(h_2)) = \emptyset$. Now assume that h_1 is segmented with $\mid \mathtt{dom}(h_1) \mid = n$, h_2 is drown and, h_1 and h_2 are completely disjoint. Then, there is a 1-marked heap h_1' such that all the predecessors of $s(\mathtt{x})$ are endpoints of 2-markers iff $\widetilde{\sharp \mathtt{x}} \leq n$ (in h_2). If we have the possibility to add to h_1' a strict 2-marked heap with exactly c distinct 2-markers then all the predecessors of $s(\mathtt{x})$ are endpoints of 2-markers iff $\widetilde{\sharp \mathtt{x}} \leq c + n$. This is formalized below.

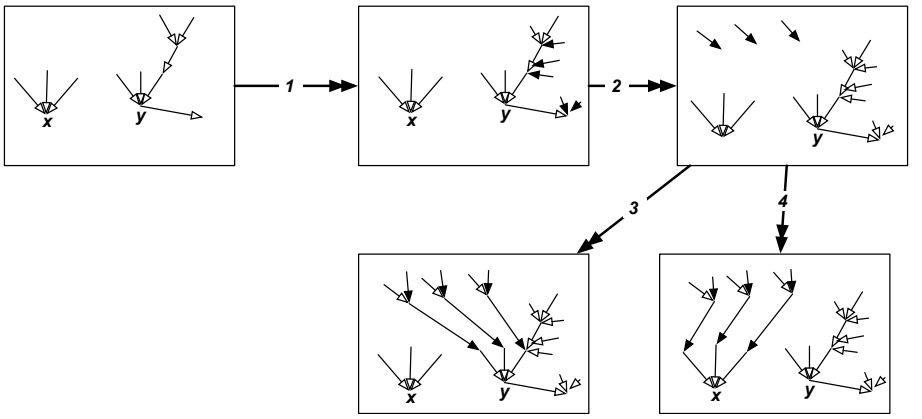
Lemma 9. *Let s be a store and h_1, h_2 be two completely disjoint heaps such that $(s, h_1) \vDash \mathsf{drown} \wedge \sharp\mathtt{x} = i$ and $(s, h_2) \vDash \mathtt{seg} \wedge \sharp\mathtt{x} = 0$. Then, (i) $(s, h_1 * h_2) \models (\mathrm{one} \mid \mathrm{two}[c]) \mathrel{-\!\!*} \mathtt{preM}^2(\mathtt{x})$ iff (ii) $\mid \mathtt{dom}(h_2) \mid \geq (i - c)$.*

So we may test that the number of extra arrows that satisfy seg is less than $\widetilde{\sharp\mathtt{x}}$. Call n the number of these extra arrows, then this scenario says $\widetilde{\sharp\mathtt{x}} - c \leq n$. Now, in order to express properties of the form $\widetilde{\sharp\mathtt{x}} \bowtie \widetilde{\sharp\mathtt{y}} + c$, it is sufficient to express properties of the form $\widetilde{\sharp\mathtt{x}} + c \leq \widetilde{\sharp\mathtt{y}} + c'$ $(c, c' \in \mathbb{N})$ thanks to Boolean connectives. Note moreover that this constraint is equivalent to: for all $n \in \mathbb{N}$, $\widetilde{\sharp\mathtt{y}} - c \leq n$ implies $\widetilde{\sharp\mathtt{x}} - c' \leq n$. This suggests a contest between two players: Spoiler that aims at disproving that the constraint holds, and Duplicator tries to prove it. The contest is the following:

1. Spoiler reduces to the case of a drown model - this will be formalized later.
2. Spoiler picks a segmented heap h' with $\mid \mathtt{dom}(h') \mid = n$.
3. He proves that $\widetilde{\sharp\mathtt{y}} - c \leq n$ using the previous scenario.
4. Then Duplicator must prove $\widetilde{\sharp\mathtt{x}} - c' \leq n$ playing with the same scenario.

Figure 2 summarizes a contest with a successful outcome for Duplicator.

Figure 2 A contest won by Duplicator. $n = 3$, $c = c' = 0$



Let φ_{TD} be the formula that specifies that if a model is not drown then it is only due to the fact that it contains a segmented subheap that is not drown:

$$(\forall \mathtt{x}, \mathtt{y}.\ (\mathtt{x} \hookrightarrow \mathtt{y} \wedge \sharp\mathtt{y} = 1) \Rightarrow (\sharp\mathtt{x} = 0 \wedge \neg \mathtt{alloc}\ (\mathtt{y}))) \wedge (\neg(\exists \mathtt{x}.\ \sharp\mathtt{x} = 2))$$

The formula $\mathsf{contest}(\mathtt{x}, \mathtt{y}, c, c')$ defines a contest (essential to establish Theorem 10).

$$\mathsf{drown} \wedge ((\mathtt{seg} \wedge \sharp\mathtt{x} = 0 \wedge \sharp\mathtt{y} = 0) \mathrel{-\!\!*} \varphi_{TD} \Rightarrow (((\mathrm{one} \mid \mathrm{two}[c]) \mathrel{-\!\!*} \mathtt{preM}^2(\mathtt{y})) \Rightarrow ((\mathrm{one} \mid \mathrm{two}[c]) \mathrel{-\!\!*} \mathtt{preM}^2(\mathtt{x}))).$$

Theorem 10. *For $c, c' \geq 0$, there is a formula ϕ in $\mathtt{SL}(\mathrel{-\!\!*})$ of quadratic size in $c + c'$ such that for every model (s, h), we have $(s, h) \models \phi$ iff $\widetilde{\sharp\mathtt{x}} + c \leq \widetilde{\sharp\mathtt{y}} + c'$.*

The proof contains a simple case analysis depending whether the model can be transformed into a drown one (in order to use $\mathtt{contest}(\mathtt{x},\mathtt{y},c,c')$) without altering the number of precedessors for $s(\mathtt{x})$ and $s(\mathtt{y})$. In Section 5, constraints $\widetilde{\sharp\mathtt{x}}+c\leq\widetilde{\sharp\mathtt{y}}+c'$ with $c,c'\leq 3$ shall be used.

5 $\mathtt{SL}(-\!\!*)$ is equivalent to $\mathtt{SO}$

By Proposition 3, we know that $\mathtt{SL}\sqsubseteq\mathtt{DSO}$ and there is a logspace translation from $\mathtt{SL}$ into $\mathtt{DSO}$ (logspace reductions are closed under compositions). Now, we show the converse. In the sequel, without any loss of generality we require that the sentences in $\mathtt{DSO}$ satisfy the Barendregt convention as far as the second-order variables are concerned. Assuming that a sentence contains the second-order variables $\mathtt{P}_1,\ldots,\mathtt{P}_n$, quantifications over $\mathtt{P}_i$ can only occur in the scope of quantifications over $\mathtt{P}_1,\ldots,\mathtt{P}_{i-1}$ (we call this restriction the *extended Barendregt convention*). Typically, we exclude sentences of the form $\exists\mathtt{P}_2\,\exists\mathtt{P}_1\,\phi$. Observe that any sentence in $\mathtt{DSO}$ can be transformed in logspace into an equivalent sentence verifying this convention. The *quantifier depth* of the occurrence of a subformula ψ in ϕ is therefore the maximal i such that this occurrence is in the scope of $\exists\mathtt{P}_i$ (by convention it is zero if it is not in the scope of any quantification).

Before defining the translation of a $\mathtt{DSO}$ sentence ϕ (with second-order variables $\mathtt{P}_1,\ldots,\mathtt{P}_n$), let us explain how environments can be encoded in $\mathtt{SL}$ which is the key point to simulate second-order quantification. An environment is encoded as part of the memory heap with a specific shape. The interpretation of each variable $\mathtt{P}_i$ is performed as follows. A pair (l,l') is in the interpretation of $\mathtt{P}_i$ iff there are markers with respective endpoint l and l' whose degrees are *consecutive* values strictly between some fixed values $\widetilde{\sharp\mathtt{z}_i^m}$ and $\widetilde{\sharp\mathtt{z}_i^M}$. Here, the distinguished variables $\mathtt{z}_i^m$ and $\mathtt{z}_i^M$ are interpreted as locations that are not in the domain of the original memory heap. In order to avoid confusions between the original memory heap and the part that is dedicated to the encoding of the environment, $\widetilde{\sharp\mathtt{z}_i^m}$ is strictly greater than the degree of any location in the original memory heap. In other words, instead of making the original model drown and using small markers, the model is unchanged and large markers are used.

The translation of the formula ϕ, written $T(\phi)$, is defined with the help of the translation t_j where j records the quantifier depth.

$$T(\phi)\triangleq\exists\mathtt{z}_0^m\,\mathtt{z}_0^M.\,\mathtt{isol}(\mathtt{z}_0^M)\wedge\mathtt{isol}(\mathtt{z}_0^m)\wedge$$

$$[((\forall\mathtt{x}.\,\mathtt{alloc}\,(\mathtt{x})\Rightarrow(\mathtt{x}\hookrightarrow\mathtt{z}_0^M\vee\mathtt{x}\hookrightarrow\mathtt{z}_0^m\vee\mathtt{x}=\mathtt{z}_0^M\vee\mathtt{x}=\mathtt{z}_0^m))\wedge\mathtt{alloc}\,(\mathtt{z}_0^M)\wedge\mathtt{alloc}\,(\mathtt{z}_0^m))\mathrel{\neg\!\!*}$$

$$(\forall\mathtt{x}.\mathtt{x}\neq\mathtt{z}_0^M\wedge\mathtt{x}\neq\mathtt{z}_0^m\Rightarrow(\sharp\mathtt{z}_0^m>2+\sharp\mathtt{x}))\wedge(\sharp\mathtt{z}_0^M=2+\sharp\mathtt{z}_0^m)\wedge\mathtt{extr}(\mathtt{z}_0^m)\wedge\mathtt{extr}(\mathtt{z}_0^M)\wedge t_0(\phi))]$$

The formula $\mathtt{isol}(\mathtt{x})$ is an abbreviation for $\neg\exists\mathtt{y}.\,(\mathtt{x}\hookrightarrow\mathtt{y})\vee(\mathtt{y}\hookrightarrow\mathtt{x})$. Any location with number of predecessors greater than $\widetilde{\sharp\mathtt{z}_0^M}$ is useful to encode a pair of locations for the interpretation of some second-order variable (except the interpretation of distinguished variables of the form either $\mathtt{z}_i^m$ or $\mathtt{z}_i^M$). The translation of the atomic formula $\mathtt{P}_j(\mathtt{x},\mathtt{y})$ in the scope of the quantifications over $\mathtt{P}_1,\ldots,\mathtt{P}_i$ ($j\leq i$) is defined below:

$$t_i(\mathtt{P}_j(\mathtt{x},\mathtt{y}))\triangleq\exists\mathtt{z},\mathtt{z}'\,(\mathtt{z}\hookrightarrow\mathtt{x})\wedge(\mathtt{z}'\hookrightarrow\mathtt{y})\wedge(\sharp\mathtt{z}>\sharp\mathtt{z}_j^m)\wedge(\sharp\mathtt{z}<\sharp\mathtt{z}_j^M)\wedge(\sharp\mathtt{z}'=1+\sharp\mathtt{z})\wedge\mathtt{extr}(\mathtt{z})\wedge\mathtt{extr}(\mathtt{z}')$$

So $(s(\mathtt{x}), s(\mathtt{y}))$ belongs to the interpretation of $\mathtt{P}_j$ when $s(\mathtt{x})$ and $s(\mathtt{y})$ are endpoints of markers with consecutive degrees between $\widetilde{\sharp \mathtt{z}_j^m}$ and $\widetilde{\sharp \mathtt{z}_j^M}$. However, in order to avoid interferences between the encoding of the environment and the original memory heap, we require that the new memory heap satisfies structural properties described below.

Proposition 11. *There is a formula* $\mathtt{envir}(\mathtt{z}, \mathtt{z}')$ *such that* $(s,h) \models \mathtt{envir}$ *iff* $\widetilde{\sharp \mathtt{z}} < \widetilde{\sharp \mathtt{z}'}$, $\widetilde{\sharp \mathtt{z}} \equiv \widetilde{\sharp \mathtt{z}'} + 2$ [3] *and for all* i *in* $[\widetilde{\sharp \mathtt{z}}, \ldots, \widetilde{\sharp \mathtt{z}'}]$,

- *if* $i \equiv \widetilde{\sharp \mathtt{z}} + 1$ [3] *then there is no extremity* l *in* (s,h) *such that* $\widetilde{\sharp l} = i$,
- *if* $i \not\equiv \widetilde{\sharp \mathtt{z}} + 1$ [3] *then there is exactly one location* l *such that* l *is an extremity and* $\widetilde{\sharp l} = i$. *This unique location* l *belongs to* $\mathtt{dom}(h)$.

An i-well-formed model, defined below, can be divided into two disjoint parts such that one part encodes the interpretation of the second-order variables $\mathtt{P}_1, \ldots, \mathtt{P}_i$.

Definition 12. *A memory state* (s,h) *is* i-well-formed *for some* $i \geq 0$ *iff there are heaps* h_1, h_2 *with* $h = h_1 * h_2$ *satisfying the properties below:*

(I) *for every variable* $\mathtt{x}$ *in* $\{\mathtt{z}_1^m, \ldots, \mathtt{z}_i^m\} \cup \{\mathtt{z}_0^M, \ldots, \mathtt{z}_{i-1}^M\}$, $s(\mathtt{x})$ *is an extremity in* (s,h) *and* $\widetilde{\sharp \mathtt{z}_0^m} < \widetilde{\sharp \mathtt{x}} < \widetilde{\sharp \mathtt{z}_i^M}$,

(II) *when* $i \geq j > 0$, $\widetilde{\sharp \mathtt{z}_{j-1}^M} + 1 = \widetilde{\sharp \mathtt{z}_j^m}$,

(III) *there is no location* l *such that* $\widetilde{\sharp l}$ *in* (s,h_1) *is strictly greater than* $\widetilde{\sharp \mathtt{z}_0^m} - 2$ *in* (s,h),

(IV) *if* $l \in \mathtt{dom}(h_2)$ *then there is* $l' \in \{l, h_2(l)\}$ *such that* l' *is an extremity in* h_2,

(V) *any extremity* l *in the model* (s,h_2) *satisfies (1)* $l \notin \mathtt{ran}(h_1)$, *(2)* l *is an extremity in* (s,h), *(3)* $\widetilde{\sharp \mathtt{z}_0^m} \leq \widetilde{\sharp l} \leq \widetilde{\sharp \mathtt{z}_i^M}$ *and (4)* $h(l) \notin \mathtt{dom}(h_2)$.

(VI) $(s,h_2) \models \mathtt{envir}(\mathtt{z}_0^m, \mathtt{z}_i^M)$.

(s,h_1) *is called the* base part *and* (s,h_2) *the* environment part.

Note that in any such decomposition, we have $\mathtt{dom}(h_2) \cap \mathtt{ran}(h_1) = \emptyset$. Moreover, any extremity in h with more than $\widetilde{\sharp \mathtt{z}_0^m}$ predecessors has all predecessors in $\mathtt{dom}(h_2)$. The above decomposition is indeed unique.

Lemma 13. *Whenever* (s,h) *is i-well-formed with base part* h_1 *and environment part* h_2, *there is no* $(h_1', h_2') \neq (h_1, h_2)$ *such that* (s,h) *is i-well-formed with base part* h_1' *and environment part* h_2'.

The *spectrum* of an i-well-formed memory state is defined as the set of numbers of predecessors in the environment part that are greater than 3. Hence such a spectrum has the following shape $\bullet \circ \bullet \bullet \circ \bullet \bullet \cdots \circ \bullet \bullet \circ \bullet \bullet \circ \bullet \bullet \cdots \circ \bullet \bullet \circ \bullet$ corresponding to a finite sequence of successive integers where $\bullet$ indicates the presence of the integer and $\circ$ its absence. The smallest value is $\widetilde{\sharp \mathtt{z}_0^m}$ and the greatest value is $\widetilde{\sharp \mathtt{z}_i^M}$. Two consecutive values in the sequence encode one pair from the interpretation of a second-order variable. Observe that the concatenation of two spectra is still a spectrum. The formula $\mathtt{relation}_{i,X}$ defined below states that part of the memory is 0-well-formed and can serve to encode the interpretation of the variable $\mathtt{P}_i$.

Proposition 14. *Given $i \geq 0$ and X a finite set of variables disjoint from the set of auxiliary variables $\{\mathtt{z}_0^m, \mathtt{z}_0^M, \ldots, \mathtt{z}_i^m, \mathtt{z}_i^M\}$, there is a formula $\mathtt{relation}_{i,X}$ such that for every model (s,h), we have $(s,h) \models \mathtt{relation}_{i,X}$ iff $(s[\mathtt{z}_0^m \mapsto s(\mathtt{z}_i^m), \mathtt{z}_0^M \mapsto s(\mathtt{z}_i^M)], h)$ is 0-well-formed, its base part is empty and for every $\mathtt{x} \in X$, $s(\mathtt{x}) \notin \mathtt{dom}(h)$.*

In order to translate the subformula $\exists \mathtt{P}_i.\ \psi$, we introduce two locations whose numbers of predecessors determine the bounds for the degrees for any marker used to encode a pair for the interpretation of $\mathtt{P}_i$. There is a way to add markers (expressed thanks to the connective $-\!\!*$) that guarantees that the new part of the heap encodes the interpretation of the variable $\mathtt{P}_i$ by using the above formula $\mathtt{relation}_{i,X}$. The translation of $\exists \mathtt{P}_i.\ \psi$ at the $(i-1)$ quantification depth, noted $t_{i-1}(\exists \mathtt{P}_i, \psi)$, is defined by $\exists \mathtt{z}_i^m, \mathtt{z}_i^M.\, \mathtt{isol}(\mathtt{z}_i^m) \wedge \mathtt{isol}(\mathtt{z}_i^M) \wedge (\mathtt{relation}_{i,\mathtt{FV}(\psi)} -\!\!* (\mathtt{envir}(\mathtt{z}_0^m, \mathtt{z}_i^M) \wedge \sharp \mathtt{z}_{i-1}^M + 1 = \sharp \mathtt{z}_i^m \wedge t_i(\psi)))$.

Definition 15. *Let (s,h) be an i-well-formed model with environment part (s,h_2). An environment $\mathcal{E}$* extracted from *(s,h) satisfies for $j \in \{1,\ldots,i\}$, $\mathcal{E}(\mathtt{P}_j) = \mathcal{R}_j$ with $\mathcal{R}_j = \{(h_2(l), h_2(l')) :\ \widetilde{\sharp \mathtt{z}_j^m} < \widetilde{\sharp l},\ \widetilde{\sharp l} + 1 = \widetilde{\sharp l'},\ \widetilde{\sharp l'} < \widetilde{\sharp \mathtt{z}_j^M}$ in $h_2\}$.*

The map t_i is homomorphic for Boolean connectives, and is the identity for atomic formulae of the form either $\mathtt{x} = \mathtt{y}$ or $\mathtt{x} \hookrightarrow \mathtt{y}$. It remains to treat the case for first-order quantification. The main difficulty is to guarantee that first-order variables are not interpreted as locations used in markers encoding second-order quantification. Typically, the number of predecessors of $s(\mathtt{x})$ and $h(s(\mathtt{x}))$ (if its exists) should be less than $\widetilde{\sharp \mathtt{z}_0^M}$ and none of these locations is an extremity. The formula $\mathtt{notonenv}$ is introduced for this purpose: $\mathtt{notonenv}(\mathtt{x}) \triangleq \neg(\exists \mathtt{y}.\ (\mathtt{y} = \mathtt{x} \vee \mathtt{x} \hookrightarrow \mathtt{y}) \wedge (\sharp \mathtt{y} \geq \sharp \mathtt{z}_0^m) \wedge \mathtt{extr}(\mathtt{y}))$. The main reason for introducing $\mathtt{notonenv}(\mathtt{x})$ is to be able to identify locations from the environment part of i-well-formed models, as stated below.

Lemma 16. *Let (s,h) be an i-well-formed with environment part h_2. Then $(s,h) \models \mathtt{notonenv}(\mathtt{x})$ iff $s(\mathtt{x}) \notin \mathtt{dom}(h_2)$.*

The translation $t_i(\exists \mathtt{x}.\ \psi)$ is defined as $\exists \mathtt{x}.\, \mathtt{notonenv}(\mathtt{x}) \wedge t_i(\psi)$. Observe that $T(\phi)$ and ϕ have the same first-order free variables. Correctness of $T(\cdot)$ is based on Proposition 17.

Proposition 17. *Let ϕ be a $\mathtt{DSO}$ formula with second-order variables $\{\mathtt{P}_1, \ldots, \mathtt{P}_n\}$ using the extended Barendregt convention. Let ψ be a subformula of ϕ which occurs under the scope of $\mathtt{P}_1, \ldots, \mathtt{P}_j$ ($0 \leq j \leq n$) with quantified second-order variables in $\{\mathtt{P}_{j+1}, \ldots, \mathtt{P}_n\}$. Let (s,h) be a j-well-formed model with base part h_1 and environment part h_2 such that for each $\mathtt{x} \in \mathtt{FV}(\psi)$, $s(\mathtt{x}) \notin \mathtt{dom}(h_2)$. Let $\mathcal{E}_j$ be an environment extracted from (s,h) ($\{\mathtt{P}_1 \mapsto \mathcal{R}_1, \ldots, \mathtt{P}_j \mapsto \mathcal{R}_j\}$). Then, $(s,h) \models t_j(\psi)$ iff $(s,h_1), \mathcal{E}_j \models \psi$.*

Full proof of Proposition 17 can be found in [BDL08]. We present below simple cases in the analysis. This will entail our main result (Theorem 18).

Proof (sketch with simple cases) Let us start by a preliminary definition. We say that a location l *occurs* in a binary relation $\mathcal{R}$ when there is a location l' such that either $(l,l') \in \mathcal{R}$ or $(l',l) \in \mathcal{R}$. Let ϕ be a $\mathtt{DSO}$ sentence satisfying the extended Barendregt convention. We want to show by induction on ψ that given:

– ψ is a subformula of ϕ which occurs under the scope of $\mathtt{P}_1, \ldots, \mathtt{P}_j$, and with second-order quantifications over elements from $\{\mathtt{P}_{j+1}, \ldots, \mathtt{P}_n\}$,
– (s, h) is j-well-formed with base part h_1 and environment part h_2 such that for every variable $\mathtt{x} \in \mathtt{FV}(\psi)$, we have $s(\mathtt{x}) \notin \mathtt{dom}(h_2)$,
– $\mathcal{E}_j$ is the environment $\{\mathtt{P}_1 \mapsto \mathcal{R}_1, \ldots, \mathtt{P}_j \mapsto \mathcal{R}_j\}$ extracted from h_2,
– no location occurring in $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_j$ belongs to $\mathtt{dom}(h_2)$,

we have $(s, h) \models t_j(\psi)$ iff $(s, h_1), \mathcal{E}_j \models \psi$.

Base case 1: $\psi = \mathtt{P}_k(\mathtt{x}, \mathtt{y})$ with $k \leq j$.

($\rightarrow$) Suppose that $(s, h) \models t_j(\mathtt{P}_k(\mathtt{x}, \mathtt{y}))$. Then, $s(\mathtt{x})$ and $s(\mathtt{y})$ have predecessors in h that are extremities, let us call them respectively $l_\mathtt{x}$ and $l_\mathtt{y}$. In the heap h, we have $\widetilde{\sharp \mathtt{z}_k^m} < \widetilde{\sharp l_\mathtt{x}} = \widetilde{\sharp l_\mathtt{y}} - 1 < \widetilde{\sharp \mathtt{z}_k^M} - 1$. By Definition 12, both $l_\mathtt{x}$ and $l_\mathtt{y}$ have predecessors in $\mathtt{dom}(h_2)$ and all of their predecessors are also in $\mathtt{dom}(h_2)$. Since $\mathtt{z}_k^m$ and $\mathtt{z}_k^M$ have also all of their predecessors in $\mathtt{dom}(h_2)$, we have $\widetilde{\sharp \mathtt{z}_k^m} < \widetilde{\sharp l_\mathtt{x}}$, $\widetilde{\sharp l_\mathtt{x}} + 1 = \widetilde{\sharp l_\mathtt{y}}$ and $\widetilde{\sharp l_\mathtt{y}} < \widetilde{\sharp \mathtt{z}_k^M}$ in h_2. By Definition 15, we get $(h(l_\mathtt{x}), h(l_\mathtt{y})) \in \mathcal{R}_k$, that is $(s(\mathtt{x}), s(\mathtt{y})) \in \mathcal{R}_k$. Consequently, $(s, h_1), \mathcal{E}_j \models \mathtt{P}_k(\mathtt{x}, \mathtt{y})$.

($\leftarrow$) Suppose that $(s, h_1), \mathcal{E}_j \models \mathtt{P}_k(\mathtt{x}, \mathtt{y})$. By definition of $\models$ and $\mathcal{E}_j$, $(s(\mathtt{x}), s(\mathtt{y})) \in \mathcal{R}_k$. So $s(\mathtt{x})$ and $s(\mathtt{y})$ have respectively predecessors $l_\mathtt{x}$ and $l_\mathtt{y}$ in $\mathtt{dom}(h_2)$. In the heap h_2, $l_\mathtt{x}$ and $l_\mathtt{y}$ are extremities and $\widetilde{\sharp \mathtt{z}_k^m} < \widetilde{\sharp l_\mathtt{x}} = \widetilde{\sharp l_\mathtt{y}} - 1 < \widetilde{\sharp \mathtt{z}_k^M} - 1$. By Definition 12, the predecessors of any locations among $s(\mathtt{z}_k^m)$, $l_\mathtt{x}$, $l_\mathtt{y}$ and $s(\mathtt{z}_k^M)$ belong to $\mathtt{dom}(h_2)$. So the above inequalities and equality are also true in h. By Definition 12, the locations $s(\mathtt{z}_k^m)$, $l_\mathtt{x}$, $l_\mathtt{y}$ and $s(\mathtt{z}_k^M)$ are extremities in h. So $(s, h) \models t_j(\mathtt{P}_k(\mathtt{x}, \mathtt{y}))$.

The other base cases are by an easy verification.

Induction step – Case 1: $\psi = \exists \mathtt{x}.\ \psi'$. The statements below are equivalent:

(0) $(s, h) \models t_j(\exists \mathtt{x}\ \psi')$,
(1) there is $l \in \mathtt{Loc}$ such that $(s', h) \models t_j(\psi')$ and $(s', h) \models \mathtt{notonenv}(\mathtt{x})$ with $s' = s[\mathtt{x} \mapsto l]$ (by definition of t_j),
(2) there is $l \in \mathtt{Loc}$ such that $(s', h) \models t_j(\psi')$ and $l \notin \mathtt{dom}(h_2)$ with $s' = s[\mathtt{x} \mapsto l]$ (by Lemma 16),
(3) there is $l \in \mathtt{Loc}$ such that $(s', h_1), \mathcal{E}_j \models \psi'$ and $l \notin \mathtt{dom}(h_2)$ with $s' = s[\mathtt{x} \mapsto l]$ (by induction hypothesis since $\mathtt{FV}(\psi') \subseteq \mathtt{FV}(\exists \mathtt{x}.\ \psi') \cup \{\mathtt{x}\}$),
(4) there is $l \in \mathtt{Loc}$ such that $(s', h_1), \mathcal{E}_j \models \psi'$ with $s' = s[\mathtt{x} \mapsto l]$,
(5) $(s, h_1), \mathcal{E}_j \models \psi$ (by definition of $\models$).

Let us justify below why (4) implies (3). Suppose (4) and $l \in \mathtt{dom}(h_2)$. Since (s, h) is i-well-formed, $l \notin (\mathtt{dom}(h_1) \cup \mathtt{ran}(h_1))$. Since $\mathtt{Loc}$ is an infinite set, there is a location $l' \in (\mathtt{Loc} \setminus (\mathtt{dom}(h_1) \cup \mathtt{ran}(h_1) \cup \mathtt{dom}(h_2))$ such that l' does not occur in $(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_j)$. By equivariance shown in [BDL08], we get $(s[\mathtt{x} \mapsto l'], h_1), \mathcal{E}_j[l \leftarrow l'] \models \psi'$. Suppose *ad absurdum* that l occurs in $\mathcal{R}_k$ for some $1 \leq k \leq j$. So, l has a predecessor that is an extremity in $\mathtt{dom}(h_2)$ and by Definition 12(V(4)), $l \notin \mathtt{dom}(h_2)$, which leads to a contradiction. Hence, $\mathcal{E}_j[l \leftarrow l'] = \mathcal{E}_j$. We have established that $(s[\mathtt{x} \mapsto l'], h_1), \mathcal{E}_j \models \psi'$ and $l' \notin \mathtt{dom}(h_2)$. □

Theorem 18. $\mathtt{SL}(-\!\!*) \equiv \mathtt{SL} \equiv \mathtt{SO} \equiv \mathtt{DSO}$.

$\mathtt{DSO} \sqsubseteq \mathtt{SL}$ stems from Proposition 17. Observe that all the equivalences are obtained with logspace translations. Consequently,

Corollary 19. $\mathtt{SL}(-\!\!*)$ *validity problem is undecidable.*

Undecidability of $\mathtt{SL}(-\!\!*)$ can be obtained more easily by encoding the halting problem for Minsky machines by using the fact that $\sharp\mathtt{x} = \sharp\mathtt{y}$ and $\sharp\mathtt{x} = \sharp\mathtt{y} + 1$ can be expressed in $\mathtt{SL}(-\!\!*)$ (Section 4).

6 Concluding remarks

We have shown that first-order separation logic with one selector is as expressive as second-order logic over the class of memory states, whence undecidable too. Moreover, the restriction to the magic wand preserves the expressive power. This solves two central open problems: the decidability status of SL and the characterization of its expressive power. Additionnally, we have proved that SL without the magic wand is decidable with non-elementary complexity whereas SL restricted to the magic wand is also undecidable. By adapting the above developments, a generalization in presence of $k > 1$ selectors (see Section 2.1) is possible.

Theorem 20. *For every* $k \geq 1$, $k\mathtt{SL} \equiv k\mathtt{SL}(-\!\!*) \equiv k\mathtt{SO}$.

The case $k = 1$ requires a lot of care but a simpler direct proof is possible for $k \neq 1$. Indeed, for $k = 1$ the identification of auxiliary memory cells is performed thanks to structural properties whereas for $k \neq 1$, this can be done by simply checking the presence of distinguished values.

References

[BCO04] J. Berdine, C. Calcagno, and P. O'Hearn. A decidable fragment of separation logic. In *FST&TCS'04*, volume 3328 of *LNCS*, pages 97–109. Springer, 2004.

[BDL07] R. Brochenin, S. Demri, and E. Lozes. Reasoning about sequences of memory states. In *LFCS'07*, volume 4514 of *LNCS*, pages 100–114. Springer, 2007.

[BDL08] R. Brochenin, S. Demri, and E. Lozes. On the almighty wand. Technical report, LSV, ENS de Cachan, 2008.

[BGG97] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

[BHMV05] A. Bouajjani, P. Habermehl, P. Moro, and T. Vojnar. Verifying programs with dynamic 1-selector-linked structured in regular model-checking. In *TACAS'05*, volume 3440 of *LNCS*, pages 13–29. Springer, 2005.

[BIL04] M. Bozga, R. Iosif, and Y. Lakhnech. On logics of aliasing. In *SAS'04*, volume 3148 of *LNCS*, pages 344–360. Springer, 2004.

[BIP08] M. Bozga, R. Iosif, and S. Perarnau. Quantitative separation logic and programs with lists. In *IJCAR'08*, 2008. to appear.

[CGH05] C. Calcagno, Ph. Gardner, and M. Hague. From separation logic to first-order logic. In *FOSSACS'05*, volume 3441 of *LNCS*, pages 395–409. Springer, 2005.

[CYO01a] C. Calcagno, H. Yang, and P. O'Hearn. Computability and complexity results for a spatial assertion language. In *APLAS'01*, pages 289–300, 2001.

[CYO01b] C. Calcagno, H. Yang, and P. O'Hearn. Computability and complexity results for a spatial assertion language for data structures. In *FST&TCS'01*, volume 2245 of *LNCS*, pages 108–119. Springer, 2001.

[DGG04] A. Dawar, Ph. Gardner, and G. Ghelli. Adjunct elimination through games in static ambient logic. In *FST&TCS'04*, volume 3328 of *LNCS*, pages 211–223. Springer, 2004.

[DGG07] A. Dawar, Ph. Gardner, and G. Ghelli. Expressiveness and complexity of graph logic. *I & C*, 205(3):263–310, 2007.

[GM08] D. Galmiche and D. Méry. Tableaux and resource graphs for separation logic, 2008. Submitted.

[IO01] S. Ishtiaq and P. O'Hearn. BI as an assertion language for mutable data structures. In *POPL'01*, pages 14–26, 2001.

[JJKS97] J. Jensen, M. Jorgensen, N. Klarlund, and M. Schwartzbach. Automatic verification of pointer programs using monadic second-order logic. In *PLDI'97*, pages 226–236. ACM, 1997.

[KR03] F. Klaedtke and H. Rueb. Monadic second-order logics with cardinalities. In *ICALP'03*, volume 2719 of *LNCS*, pages 681–696. Springer, 2003.

[KR04] V. Kuncak and M. Rinard. On spatial conjunction as second-order logic. Technical report, MIT CSAIL, October 2004.

[LAS00] T. Lev-Ami and M. Sagiv. TVLA: A system for implementing static analyses. In *SAS'00*, volume 1824 of *LNCS*, pages 280–301. Springer, 2000.

[Loz04] E. Lozes. Separation logic preserves the expressive power of classical logic. In *2nd Workshop on Semantics, Program Analysis, and Computing Environments for Memory Management (SPACE'04)*, 2004.

[Loz05] E. Lozes. Elimination of spatial connectives in static spatial logics. *TCS*, 330(3):475–499, 2005.

[LR03] Ch. Löding and Ph. Rohde. Model checking and satisfiability for sabotage modal logic. In *FST&TCS'03*, volume 2914 of *LNCS*, pages 302–313. Springer, 2003.

[Mar06] J. Marcinkowski. On the expressive power of graph logic. In *CSL'06*, volume 4207 of *LNCS*, pages 486–500. Springer, 2006.

[MBCC07] S. Magill, J. Berdine, E. Clarke, and B. Cook. Arithmetic strengthening for shape analysis. In *SAS'07*, volume 4634 of *LNCS*, pages 419–436. Springer, 2007.

[Rab69] M. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 41:1–35, 1969.

[Rey02] J.C. Reynolds. Separation logic: a logic for shared mutable data structures. In *LICS'02*, pages 55–74. IEEE, 2002.

[RZ06] S. Ranise and C. Zarba. A theory of singly-linked lists and its extensible decision procedure. In *SEFM'06*, pages 206–215. IEEE, 2006.

[Sto74] L. Stockmeyer. *The complexity of decision problems in automata theory and logic*. PhD thesis, Department of Electrical Engineering, MIT, 1974.

[Tra50] B.A. Trakhtenbrot. The impossibility of an algorithm for the decision problem for finite models. *Dokl. Akad. Nauk SSSR*, 70:596–572, 1950. English translation in: AMS Transl. Ser. 2, vol.23 (1063), 1–6.

[YRS+05] G. Yorsh, A. M. Rabinovich, M. Sagiv, A. Meyer, and A. Bouajjani. A logic of reachable patterns in linked data structures. In *FOSSACS'05*, volume 3441 of *LNCS*, pages 94–110. Springer, 2005.