On the Almighty Wand

Rémi Brochenin, Stephane Demri, Etienne Lozes

On the Almighty Wand[1]

Rémi Brochenin[a], Stéphane Demri[a], Etienne Lozes[a,b]

[a]LSV, ENS Cachan, CNRS, INRIA, France
[b]MOVES, RWTH, Aachen, Germany

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, in SL the separating implication (also known as the magic wand) can simulate the separating conjunction. By contrast, we establish that SL without the magic wand is decidable, and we prove a non-elementary complexity by reduction from satisfiability for the first-order theory over finite words. This result is extended with a bounded use of the magic wand that appears in Hoare-style rules. As a generalisation, it is shown that kSL, the separation logic over heaps with $k \geq 1$ record fields, is equivalent to kSO, the second-order logic over heaps with k record fields.

Keywords: separation logic, second-order logic, expressive power, complexity

1. Introduction

Separation logic. Programming languages with pointer variables seldom have mechanisms to detect errors. An inappropriate management of memory is the source of numerous bugs and security holes such as buffer overflow attacks, null pointer dereferences or memory leaks. Prominent logics for analysing such pointer programs include separation logic [35], pointer assertion logic PAL [23], TVLA [28], alias logic [6], BI (Bunched Implication) [22] and LRP (logic of reachable patterns) [39] to quote a few examples. Separation logic (SL) is an assertion language used in Hoare-like proof systems [35] 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 the decidability of its satisfiabily problem. 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.

Magic wand and lists. The complexity of the satisfiability, the model-checking, the validity or the entailment problems for several fragments of separation logic have been intensively studied since the early days of separation logic until quite recently [15, 14, 35, 10, 16]. The magic wand connective makes any of these problems quite difficult to decide (note that these problems are often equivalent in presence of magic wand). The expressive power of $-\!\!*$ is increased by the first-order quantification: SL with magic wand is known to be equivalent to a classical propositional logic [30] if first-order quantifiers are disabled, whereas no adjunct elimination occurs in SL with first-order quantifiers [17, 31]. The same gap exists with respect to decidability: SL without first-order quantifiers is decidable, but it becomes undecidable if first-order quantifiers are taken into account [15]. These known results however crucially relies on the memory model addressing cells with two record fields (undecidability of SL in [15] is by reduction to the first-order theory of a finite *binary* relation). But, despite the predominance of the list-manipulating programs in the case studies of separation logic, the minimality and complexity of SL with magic wand is not known for memory models with only one record field.

Second-order logic. A natural question about separation logic is how it compares with weak second-order logic (SO) and its fragments. This is a very natural question for at least three reasons. Firstly, separating conjunction and its adjoint are essentially second-order connectives (see also a similar concern on graphs with spatial logics [18]), which clearly makes SL be a fragment of SO. Secondly, many properties on heaps require second-order logic, for instance to express recursive predicates, or list and tree properties. Thirdly, SO is known to be a sufficiently expressive assertion language for ensuring the completeness of the Hoare-Floyd logic, and better understanding the relationship between SL and SO could serve to derive the completeness of the proof system of SL.

A first exploration of the relationships between SL and SO can be found in [26], but Kunczak and Rinard considered a separation logic for arbitrary first-order structures, and not the standard, finite, functional heap model of SL. The expressivity of SL with magic wand for lists is thus also an open question.

Our contributions. In this paper, we address simultaneously the decidability, complexity, expressive power, and minimality of first-order separation logic with one selector (record field) with and without magic wand.

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, first-order separation logic is undecidable (solving an open problem stated in [20]). 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 logarithmic-space translations. As a consequence, SL is not a minimal logic, as the magic wand can simulate the separating conjunction, but it does not have the adjunct elimination. Moreover, these results generalize to non-linear recursive data structures: kSL, the separation logic over heaps with $k \geq 1$ record fields, is equivalent to kSO, the second-order logic over heaps with k record fields. Note that we use the loose version of points-to and as far as we can judge, our results are dependent on using the loose points-to. We did not investigate in details which of our results can be adapted to the tight points-to.

The correspondence between first-order separation logic with one selector and weak second-order logic over the class of memory states is particularly interesting from a logical point of view. Indeed, it shows that separating operators are sufficient to express weak second-order quantification; actually we even show that the magic wand suffices. There are well-known examples of standard logics that are shown equivalent to standard other formalisms, such that temporal logics. The celebrated Kamp's theorem states that the the popular linear-time temporal logic LTL is as expressive as first-order logic [24]; here LTL has only the strict until and since operators. This result is refined in [19] where it is shown that unary LTL is as expressive as first-order logic restricted to two individual variables. Similarly, the automata-based approach for formal verification stems from the famous result showing the equivalence between monadic second-order logic and Büchi automata as far as definability of languages of infinite words are concerned [12]. In this paper, our pivot logic is weak second-order logic over the class of memory states.

We also establish that `SL` without the magic wand is decidable, but with a non-elementary complexity (this lower bound is obtained by reduction from satisfiability for the first-order theory over finite words [36], and holds already with three variables). Decidability is shown by reduction to weak monadic second-order theory of one unary (total) function that is shown decidable in [33]. It is worth noting that even though the first-order theory of one unary function is known to be not elementary recursive [4], 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 [3] for a fragment of separation logic with one selector is decidable. We also establish that decidability can be obtained with a restricted use of the magic wand as it occurs in Hoare-like proof systems involving separation logic.

Related work. The closest works to ours are certainly the recent ones on the decidability of separation logic for lists with data [2] and the work on the comparison of the expressive power of monadic second-order logic and the spatial logic for graphs [1]. Although the questions solved in these works do not overlap the results presented herein, these works adopt a point of view quite similar to the one of this paper and give a more complete picture of the topic. More detailed comments are given in related sections of this paper.

The magic wand is rarely considered by the litterature on `SL`, which our result may explain from the complexity point of view. The magic wand is however often behind the scene in recent developments of `SL`. For instance, the bi-abduction problem [21] can be seen as a specialized version of the satisfiability problem for `SL` with magic wand. As a parallel to this work, results stating either the absence of adjunct elimination or the undecidability of satisfiability for logics including a form of magic wand have been independently established for boolean BI [27], propositional `SL` [11], or context logic [13]. The main difference with our work is that the models of these logics include formal propositional variables that can be used to axiomatize the models in any desired way, whereas we are sticking to the heap model.

As seen previously, heap properties are formalized in various logical languages [23, 28, 35, 6, 39] and separation logic is just one prominent example of these logics. However, in this paper we focus on expressive power and decidability issues rather than on verification techniques. Verification methods and logics for verifying programs with singly-linked lists can be found for instance in [3, 5, 34]. From another perspective, the relationships between logics on graphs with separating features and second-order logic can be found in [18]. Finally, we would like to mention that sabotage modal logics (SML) considered in [38, 29] have also the ability to modify the model under evaluation by using new logical connectives. So far, we are not aware of any work relating separation logic and SML.

Plan of the paper. In Section 2, we present the different logical formalisms used in the paper (separation logic `SL` and weak second-order logic `SO`), examples for properties that can be expressed in such languages and a translation from `SL` into `SO`. In Section 3, we show that `SL` restricted to the separating conjunction is decidable with non-elementary complexity. The complexity lower bound is by reduction from the first-order theory over finite words and decidability is obtained by a logarithmic-space reduction into weak monadic second-order theory for one unary function. In Section 4, we extend this decidability result with a restricted use of the magic wand. Section 5 contains many technical contributions about the expressive power of `SL`, in particular we show how to express arithmetical constraints about the memory heap. These results are essential to show in Section 6 that `SO` and `SL` are equivalent in terms of expressivity. This is refined by showing that `SL` restricted to the magic wand (called herein $\mathtt{SL}(-\!\!*)$) is also as expressive as `SO` (and `SL`). In Section 7, we show how the equivalence between separation logic and second-order logic can be extended to memory cells with $k > 1$ record fields. Section 8 contains concluding remarks and open problems for further investigation.

This paper is a completed version of [8].

2. Preliminaries

In this section, we recall the definition of first-order separation logic with one selector (record field), called herein `SL`, and second-order logic over the same class of structures (called herein `SO`). We introduce the concept of being at least as expressive as another fragment, and provide examples of properties that can be expressed in our formalisms. This section ends by presenting a quite straightforward encoding of $\mathtt{SL}$ into fragments of $\mathtt{SO}$ over structures with one unary function.

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 $\mathtt{Loc}$ be a countably infinite set of *locations* ranged over by $l, l', \ldots$ that represents the set of addresses. A memory state is composed of a pair made of a *store* and a *heap*. Let $\mathtt{Var}$ 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 $s : \mathtt{Var} \rightarrow \mathtt{Loc}$ (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 $\mathtt{Var}$ 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. We write $\mathcal{S}$ to denote the set of stores, and $\mathcal{H}$ to denote the set of heaps. A heap h with domain $\{l_1, \ldots, l_n\}$ is sometimes represented by the set of memory cells $\{l_1 \mapsto h(l_1), \ldots, l_n \mapsto h(l_n)\}$. 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 $\sharp l$ 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* [resp. *strict descendant*] of l if there is $n \geq 0$ [resp. $n > 0$] such that $h^n(l) = l'$ ($h^n(l)$ is not always defined). In the rest of the paper, we assume that $\mathtt{Loc} = \mathtt{Val} = \mathbb{N}$, the value *nil* can be encoded by an individual variable in the first-order language.

Formulae in $\mathtt{SL}$ *and* $\mathtt{SO}$. Formulae of first-order separation logic with one selector $\mathtt{SL}$ are defined by the grammar below:

$$\phi := \neg\phi \mid \phi \wedge \phi \mid \exists \mathtt{x}\ \phi \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \mathtt{x} = \mathtt{y} \mid \phi * \phi \mid \phi \mathrel{-\!\!*} \phi$$

The connective $*$ is called *separating conjunction* whereas the adjoint operator $\mathrel{-\!\!*}$, the *separating implication*, is usually called the *magic wand*. We will make use of standard notations for the derived connectives $\forall, \vee, \Rightarrow, \Leftrightarrow$. We write $\mathtt{FV}(\phi)$ to denote the set of free variables occurring in ϕ.

We write $\mathtt{SL}(*)$ [resp. $\mathtt{SL}(\mathrel{-\!\!*})$] to denote the restriction of $\mathtt{SL}$ without the magic wand [resp. without 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 $\mathtt{Loc}$. 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 below:

$$\phi := \neg\phi \mid \phi \wedge \phi \mid \exists \mathtt{x}\ \phi \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \mathtt{x} = \mathtt{y} \mid \exists \mathtt{P}\ \phi \mid \mathtt{Q}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$$

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. Let us mention that the equality $\mathtt{x} = \mathtt{y}$ could be also encoded as by Leibnitz formula $\forall \mathtt{P}.(\mathtt{P}(\mathtt{x}) \Leftrightarrow \mathtt{P}(\mathtt{y}))$.

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 SL is defined without any environment (or equivalently with no influence of the environment). The clauses that are specific to SL are the following ones:

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

Consequently, $\mathrel{-\!\!*}$ is a universal modality whereas $*$ has an existential flavour. 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 SL($*$) is decidable whereas SL($\mathrel{-\!\!*}$) is not.

Septraction. We also introduce a slight variant of the dual connective for the magic wand, also called the *septraction*: $\phi \mathrel{-\!\!\circledast} \psi$ is defined as the formula $\neg((\phi) \mathrel{-\!\!*} (\neg(\psi)))$. It is easy to check that $(s,h) \models \phi_1 \mathrel{-\!\!\circledast} \phi_2$ iff there is $h' \perp h$ such that $(s,h') \models \phi_1$ and $(s, h * h') \models \phi_2$. Septraction is nothing else than an existential version of magic wand. Hence, the septraction operator is quite natural since it states the existence of a disjoint heap satisfying a formula and for which the addition to the original heap satisfies another formula.

Adding the constant `null`. The current version of SL does not contain the constant `null` interpreted by *nil* such that any h is undefined for the value *nil*. Any formula ϕ possibly with the constant `null` can be easily translated into a formula ϕ' of SL such that ϕ is satisfiable iff ϕ' is satisfiable. Indeed, ϕ' can be defined as $\exists\ \mathtt{null}\ (\neg \exists \mathtt{z}\ \mathtt{null} \hookrightarrow \mathtt{z}) \wedge \phi$, where `null` is understood as a distinguished variable. In the sequel, we might use the constant `null` without further notice.

Here are some lemmas that shall be used in the sequel.

Lemma 2.1 *Let (s,h) be a model, $\mathcal{E}$ be an environment, and ψ be a formula in* DSO. *Let l, l' be locations such that*

- $l \notin \mathtt{dom}(h) \cup \mathtt{ran}(h)$.
- $l' \notin \mathtt{dom}(h) \cup \mathtt{ran}(h) \cup \{s(\mathtt{x}) : \mathtt{x} \in \mathtt{FV}(\psi)\}$.

- *l' is not in the finite graph of $\mathcal{E}(\mathtt{P})$ for any second-order variable* $\mathtt{P}$ *occurring in ψ.*

Then $(s[l \leftarrow l'], h), \mathcal{E}[l \leftarrow l'] \models \psi$ iff $(s, h), \mathcal{E} \models \psi$.

In the above statement, $s[l \leftarrow l']$ [resp. $\mathcal{E}[l \leftarrow l']$] denotes the store obtained from s [resp. the environment obtained from $\mathcal{E}$] by replacing every occurrence of l by l' (in the range). Its proof is by simple induction on the structure of ψ.

Lemma 2.2 *For all s, h, $\mathcal{E}$, s', ψ, if $s_{|\mathtt{FV}(\psi)} = s'_{|\mathtt{FV}(\psi)}$, then $(s, h), \mathcal{E} \models \psi$ iff $(s', h), \mathcal{E} \models \psi$.*

The proof of Lemma 2.2 is also by an easy verification.

Let F and F′ be two fragments of SL or SO. We say that F′ is at least as expressive as F (written $\mathrm{F} \sqsubseteq \mathrm{F}'$) whenever for every sentence $\phi \in \mathrm{F}$, there is $\phi' \in \mathrm{F}'$ such that for every model (s, h), we have $(s, h) \models \phi$ iff $(s, h) \models \phi'$. We write $\mathrm{F} \equiv \mathrm{F}'$ if $\mathrm{F} \sqsubseteq \mathrm{F}'$ and $\mathrm{F}' \sqsubseteq \mathrm{F}$. A *translation* from F to F′ is a computable function $t : \mathrm{F} \rightarrow \mathrm{F}'$ such that for every sentence $\phi \in \mathrm{F}$, 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 [25, 32, 7]. However, in Section 5 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 (separation with one selector can only speak about the structure and not about data values). 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”, see e.g. [2].

2.2. A Selection of Properties

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

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

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

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

It is worth noting that the first formula has a unique additional variable $\texttt{y}$ but n occurrences of $*$ whereas the second formula has no separating connectives but n additional variables.

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 $\mathsf{ls}(\texttt{x}, \texttt{y})$. Adapting some technique used in the graph logics [18], we show below how this very predicate can be expressed in $\texttt{SL}(*)$ as well as the reachability predicate $\texttt{x} \rightarrow^* \texttt{y}$.

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 $\texttt{x}$ and $\texttt{y}$ if there are locations $l_1, \ldots, l_n$ $(n \geq 2)$ such that $s(\texttt{x}) = l_1$, $s(\texttt{y}) = l_n$, $l_1 \neq l_n$, $\texttt{dom}(h) = \{l_1, \ldots, l_{n-1}\}$, and for every $i \in \{1, \ldots, n-1\}$, $h(l_i) = l_{i+1}$. Consider the formula below

$$\begin{array}{rcl} \texttt{x} \overset{\circlearrowleft}{\longrightarrow}^{+} \texttt{y} & \triangleq & \sharp\texttt{x} = 0 \wedge \texttt{alloc}\ (\texttt{x}) \\ & & \wedge \sharp\texttt{y} = 1 \wedge \neg\texttt{alloc}\ (\texttt{y}) \\ & & \wedge \forall \texttt{z}\ \texttt{z} \neq \texttt{y} \Rightarrow (\sharp\texttt{z} = 1 \Rightarrow \texttt{alloc}\ (\texttt{z})) \\ & & \wedge \forall \texttt{z}\ \sharp\texttt{z} \leq 1 \end{array}$$

Lemma 2.3 *Let (s, h) be a model. $(s, h) \models \mathtt{x} \overset{\circlearrowleft}{\longrightarrow}^{+} \mathtt{y}$ 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* x *and* y *and (s, h_2) can be decomposed uniquely as a (finite) collection of cyclic lists.*

Proof A location l is *shared* whenever $\sharp l \geq 2$. A location l is *initial* [resp. *final*] whenever $l \in \mathtt{dom}(h) \setminus \mathtt{ran}(h)$ [resp. $l \in \mathtt{ran}(h) \setminus \mathtt{dom}(h)$]. It is easy to show that $(s, h) \models \mathtt{x} \overset{\circlearrowleft}{\longrightarrow}^{+} \mathtt{y}$ if and only if

- $s(\mathtt{x})$ is initial,
- $s(\mathtt{y})$ is final,
- $s(\mathtt{y})$ is the only final location,
- h has no shared location.

It is easy to check that if h is of the form $h_1 * h_2$ having the properties stated in Lemma 2.3, then it satisfies the formula $\mathtt{x} \overset{\circlearrowleft}{\longrightarrow}^{+} \mathtt{y}$, which shows one implication. Let us prove the other implication.

Assume $(s, h) \models \mathtt{x} \overset{\circlearrowleft}{\longrightarrow}^{+} \mathtt{y}$. Since $\mathtt{dom}(h)$ is finite, the set of descendants of $s(\mathtt{x})$ forms either a cyclic list, or a lasso (a list segment followed by a cycle) or a list ended by a final location. Since there are no shared locations, there is no lasso; and since $s(\mathtt{x})$ is initial, it does not belong to a cyclic list. So $s(\mathtt{x})$ has a descendant that is final. It can only be $s(\mathtt{y})$, so h contains a list segment from $s(\mathtt{x})$ to $s(\mathtt{y})$. To end the proof, we must show that the rest of the heap contains cyclic lists only. This is equivalent to say that no location different from $s(\mathtt{x})$ is initial. The proof is *ad absurdum*. Suppose that l is an initial location distinct from $s(\mathtt{x})$. Then by the same reasoning as for $s(\mathtt{x})$, we have $s(\mathtt{y})$ is a descendant of l, so two distinct paths reach $s(\mathtt{y})$, which contradicts the absence of shared locations. $\square$

Now, we can introduce additional formulae (in $\mathtt{SL}(*)$) that are useful in the sequel.

$$\begin{array}{lcl} \mathsf{ls}(\mathtt{x}, \mathtt{y}) & \triangleq & \mathtt{x} \overset{\circlearrowleft}{\longrightarrow}^{+} \mathtt{y} \wedge \neg(\mathtt{x} \overset{\circlearrowleft}{\longrightarrow}^{+} \mathtt{y} * \neg \mathsf{emp}) \\ \mathtt{x} \rightarrow^{+} \mathtt{y} & \triangleq & \top * \mathsf{ls}(\mathtt{x}, \mathtt{y}) \\ \mathtt{x} \rightarrow^{*} \mathtt{y} & \triangleq & \mathtt{x} = \mathtt{y} \vee \mathtt{x} \rightarrow^{+} \mathtt{y} \end{array}$$

These formulae express the properties below.

Lemma 2.4 *Let (s, h) be a model.*

(I) $(s,h) \models \mathsf{ls}(\mathtt{x},\mathtt{y})$ *iff* (s,h) *is a list segment between* x *and* y.

(II) $(s,h) \models \mathtt{x}{\to}^{*}\mathtt{y}$ *[resp.* $(s,h) \models \mathtt{x}{\to}^{+}\mathtt{y}$ *] iff* y *is a descendant [resp. strict descendant] of* 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 internalizing the semantics and how SO can be encoded in its fragment DSO by representing multi-edges by finite sets of edges.

Proposition 2.5 *There is a logarithmic-space translation from* SL *to* SO *(hence* $\mathtt{SL} \sqsubseteq \mathtt{SO}$*).*

Proof For all variables $\mathtt{P},\mathtt{Q},\mathtt{R}$ in $\mathtt{VAR}_2$, let us define the SO formulae below with free occurrences of $\mathtt{P},\mathtt{Q},\mathtt{R}$:

- $init(\mathtt{P}) \ \triangleq\ \forall \mathtt{x},\mathtt{y}\ \mathtt{xPy}{\Leftrightarrow}\mathtt{x}{\hookrightarrow}\mathtt{y}$,
- $heap(\mathtt{P}) \ \triangleq\ \forall \mathtt{x},\mathtt{y},\mathtt{z}\ \mathtt{xPy} \wedge \mathtt{xPz} \Rightarrow \mathtt{y} = \mathtt{z}$ (functionality),
- $\mathtt{P} = \mathtt{Q} * \mathtt{R} \ \ \triangleq\ \forall \mathtt{x},\mathtt{y}\ (\mathtt{xPy}{\Leftrightarrow}(\mathtt{xQy} \vee \mathtt{xRy})) \wedge \neg(\mathtt{xQy} \wedge \mathtt{xRy})$.

Let ϕ be a formula in SL and P be a variable in $\mathtt{VAR}_2$. One can show that for every model (s,h), we have $(s,h) \models \phi$ iff $(s,h) \models \exists \mathtt{P}\ init(\mathtt{P}) \wedge t_{\mathtt{P}}(\phi)$ where $t_{\mathtt{P}}(\cdot)$ is inductively defined as follows ($t_{\mathtt{P}}(\cdot)$ is homomorphic for Boolean connectives and first-order quantification):

$$\begin{aligned}
t_{\mathtt{P}}(\mathtt{x}{\hookrightarrow}\mathtt{y}) &\triangleq\ \mathtt{xPy} \\
t_{\mathtt{P}}(\psi * \varphi) &\triangleq\ \exists \mathtt{Q},\mathtt{Q}'\ \mathtt{P} = \mathtt{Q} * \mathtt{Q}' \wedge t_{\mathtt{Q}}(\psi) \wedge t_{\mathtt{Q}'}(\varphi) \\
t_{\mathtt{P}}(\psi \mathrel{-\!\!*} \varphi) &\triangleq\ \forall \mathtt{Q}((\exists \mathtt{Q}'\ heap(\mathtt{Q}') \wedge \mathtt{Q}' = \mathtt{Q} * \mathtt{P}) \wedge heap(\mathtt{Q}) \wedge t_{\mathtt{Q}}(\psi)) \\
&\quad\ \Rightarrow (\exists \mathtt{Q}'\ heap(\mathtt{Q}') \wedge \mathtt{Q}' = \mathtt{Q} * \mathtt{P} \wedge t_{\mathtt{Q}'}(\varphi))
\end{aligned}$$

In the above clauses, the second-order variables Q and $\mathtt{Q}'$ are fresh. □

Proposition 2.6 *There is a logarithmic-space translation from* SO *to* DSO *(hence* $\mathtt{SO} \sqsubseteq \mathtt{DSO}$*).*

Proof We use the standard graphical representation of a multigraph: a tuple $(l_1,\ldots,l_n)$ is represented by n edges (l_1,l), …, (l_n,l) for some location l. To each variable P in $\mathtt{VAR}_n$, we associate n distinct variables $\mathtt{P}_1,\ldots,\mathtt{P}_n$ in

$\mathtt{VAR}_2$. Let us define the map t, homomorphic for Boolean connectives and first-order quantification, such that t preserves the semantics:

$$t(\exists \mathtt{P}\ \psi) \triangleq \exists \mathtt{P}_1, \ldots, \mathtt{P}_n\ t(\psi)$$

$$t(\mathtt{P}(\mathtt{x}_1, \ldots, \mathtt{x}_n)) \triangleq \exists \mathtt{y}\ \bigwedge_{i=1}^{n} \mathtt{P}_i(\mathtt{x}_i, \mathtt{y}).$$

Correctness of the translation makes an essential use of the fact that in $\mathtt{SO}$, the second-order quantification is over *finite* sets of locations. Indeed, let $\mathcal{R}_1, \ldots, \mathcal{R}_n$ be n finite binary relations and $\mathcal{R}$ be a finite n-ary relation (over $\mathtt{Loc}$). We say that $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ corresponds to $\mathcal{R}$ whenever for all $(l_1, \ldots, l_n) \in \mathtt{Loc}^n$, $(l_1, \ldots, l_n) \in \mathcal{R}$ iff there is $l \in \mathtt{Loc}$ such that for $1 \leq k \leq n$, $(l_k, l) \in \mathcal{R}_k$. We have the following properties:

1. For all finite binary relations $\mathcal{R}_1, \ldots, \mathcal{R}_n$, there is a finite n-ary relation $\mathcal{R}$ such that $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ corresponds to $\mathcal{R}$.
2. Reciprocally, for every finite n-ary relation $\mathcal{R}$, there are n finite binary relations $\mathcal{R}_1, \ldots, \mathcal{R}_n$ such that $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ corresponds to $\mathcal{R}$.

□

Sections 5 and 6 are devoted to prove that $\mathtt{DSO} \sqsubseteq \mathtt{SL}(\mathrel{-\!\!*})$. We will obtain that $\mathtt{SL}(\mathrel{-\!\!*})$, $\mathtt{SL}$, $\mathtt{DSO}$ and $\mathtt{SO}$ 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 [37]. 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 the first-order theory of finite words).

Lemma 3.1 $\mathtt{MSO}$ *satisfiability is decidable.*

Proof The weak monadic second-order theory of unary functions is the theory over structures of the form $(D, f, =)$ where D is a countable domain, f is a unary function, and $=$ is equality. This theory is decidable, see e.g. [4, Corollary 7.2.11]. Since in such a logical language it is possible to express that D is infinite and to simulate that f is a partial function with finite domain (use a monadic predicate symbol to be interpreted as the finite domain of f), one can specify that $(D, f, =)$ augmented with a first-order valuation is isomorphic to a heap. Based on these elementary facts, we define a translation $t_{\mathtt{P}}(.)$, computable in logarithmic space, such that a MSO sentence ϕ is satisfiable iff

$$\overbrace{(\neg\exists \mathtt{P}\ \forall \mathtt{x}\ \mathtt{P(x)})}^{\text{infinity}} \wedge \exists \mathtt{P}\ t_{\mathtt{P}}(\phi)$$

is satisfiable in the weak monadic second-order theory of one unary function, where $t_{\mathtt{P}}(\cdot)$ is defined as follows:

$$\begin{array}{ll} t_{\mathtt{P}}(\mathtt{x} \hookrightarrow \mathtt{y}) & \triangleq \mathtt{P(x)} \wedge f(\mathtt{x}) = \mathtt{y} \\ t_{\mathtt{P}}(\mathtt{x} = \mathtt{y}) & \triangleq \mathtt{x} = \mathtt{y} \\ t_{\mathtt{P}}(\mathtt{Q(x)}) & \triangleq \mathtt{Q(x)} \end{array}$$

$t_{\mathtt{Q}}$ is homomorphic for the Boolean connectives and for quantifications. □

Using a technique similar to the proof of Lemma 3.1, we can translate $\mathtt{SL}(*)$ into $\mathtt{MSO}$.

Proposition 3.2 $\mathtt{SL}(*) \sqsubseteq \mathtt{MSO}$ *via a logspace translation.*

Proof Any formula ϕ in $\mathtt{SL}(*)$ is satisfiable iff

$$\exists \mathtt{P}\ (\forall \mathtt{x}\ \mathtt{P(x)} \Leftrightarrow (\exists \mathtt{y}\ \mathtt{x} \hookrightarrow \mathtt{y})) \wedge t_{\mathtt{P}}(\phi)$$

is satisfiable where $t_{\mathtt{P}}(\cdot)$ is defined as in the proof of Lemma 3.1 with the following clauses:

- $t_{\mathtt{P}}(\mathtt{x} \hookrightarrow \mathtt{y}) \triangleq \mathtt{P(x)} \wedge \mathtt{x} \hookrightarrow \mathtt{y}$,
- $t_{\mathtt{P}}(\mathtt{x} = \mathtt{y}) \triangleq \mathtt{x} = \mathtt{y}$,
- $t_{\mathtt{P}}(\phi * \psi) \triangleq \exists \mathtt{Q}, \mathtt{Q}'\ \mathtt{P} = \mathtt{Q} \uplus \mathtt{Q}' \wedge t_{\mathtt{Q}}(\phi) \wedge t_{\mathtt{Q}'}(\psi)$ where $\mathtt{P} = \mathtt{Q} \uplus \mathtt{Q}'$ is an abbreviation for $\forall \mathtt{x}\ (\mathtt{P(x)} \Leftrightarrow (\mathtt{Q(x)} \vee \mathtt{Q}'\mathtt{(x)})) \wedge \neg(\mathtt{Q(x)} \wedge \mathtt{Q}'\mathtt{(x)})$.

$t_{\mathtt{Q}}$ is homomorphic for the Boolean connectives and for first-order quantification. □

As conjectured in [9], recently it has been shown that $\mathtt{MSO}$ is strictly more expressive than $\mathtt{SL}(*)$ [1].

Corollary 3.3 $\mathtt{SL}(*)$ *satisfiability is decidable.*

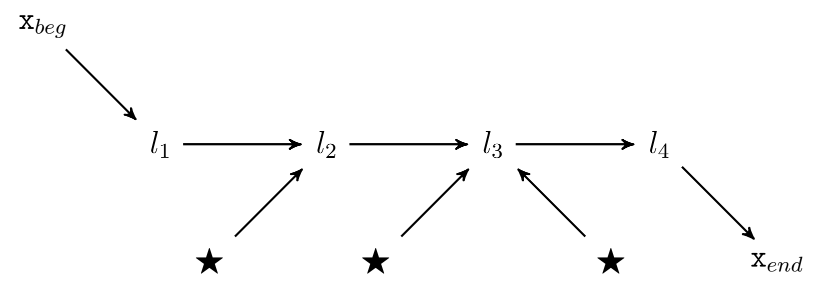
In order to show that satisfiability in $\mathtt{SL}(*)$ is not elementary recursive, we explain below how to encode finite words as memory states. Let $\Sigma = \{a_1, \ldots, a_n\}$ be a finite alphabet. A finite word $w = a_{i_1} \cdot a_{i_2} \cdots a_{i_m}$ is usually represented as the first-order 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

- $\mathtt{x}_{beg} \to^{+} \mathtt{x}_{end}$ holds true and, $\mathtt{x}_{beg}$ and $\mathtt{x}_{end}$ are distinguished variables marking respectively, the beginning and the end of the encoding of w (they do not encode any of its letters),
- the list segment induced from the satisfaction of $\mathtt{x}_{beg} \to^{+} \mathtt{x}_{end}$ 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-1} 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.

In Figure 1, we present a memory state encoding the finite word $a_1 a_2 a_3 a_1$. Throughout the paper, a memory state (s, h) is encoded as a graph representing the heap such that there is an edge from l to l' iff $h(l) = l'$. Locations are represented by letters l (representing themselves), variables $\mathtt{x}$ (representing $s(\mathtt{x})$) or a joker location $\bigstar$ (representing an unspecified location different from all the other locations present in the graph). Although the graph of h is fully specified, we may omit irrelevant variables in the representation of (s, h). In Figure 1, note that each position of the word corresponds to a unique location in the memory state. For instance, the location l_4 has one predecessor encoding the fact that the fourth letter in the word is precisely the first letter a_1. The location l_3 has 3 predecessors encoding that fact that the third letter of the word is precisely the third letter a_3.

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 $\mathrm{card}(\Sigma)$ 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.

Figure 1 Memory state encoding the finite word $a_1a_2a_3a_1$



Proposition 3.4 $\texttt{SL}(*)$ *is not elementary recursive (even its restriction with 5 variables).*

Proof Satisfiability of the first-order theory of finite words [36] is not elementary recursive (this result holds already with three variables). Let us reduce this problem to satisfiability in $\texttt{SL}(*)$. Let ψ_{word} be the formula specifying a word model:

$$(\mathtt{x}_{beg} \rightarrow^{+} \mathtt{x}_{end}) \wedge (\forall \mathtt{x}\ ((\mathtt{x}_{beg} \rightarrow^{+} \mathtt{x}) \wedge (\mathtt{x} \rightarrow^{+} \mathtt{x}_{end})) \Rightarrow \sharp \mathtt{x} \leq \mathrm{card}(\Sigma))$$

It is then easy to show that given a first-order formula ϕ over the signature $(<, (\mathtt{P}_a)_{a\in\Sigma})$, ϕ is satisfiable over finite words iff $\psi_{word} \wedge t(\phi)$ is satisfiable in $\texttt{SL}(*)$ where t is defined as follows:

$$\begin{array}{ll} t(\mathtt{x} < \mathtt{y}) & \triangleq (\mathtt{x} \rightarrow^{+} \mathtt{y}) \\ t(\forall \mathtt{x}\ \psi) & \triangleq \forall \mathtt{x}.\ ((\mathtt{x}_{beg} \rightarrow^{+} \mathtt{x}) \wedge (\mathtt{x} \rightarrow^{+} \mathtt{x}_{end})) \Rightarrow t(\psi) \\ t(\mathtt{P}_{a_i}(\mathtt{x})) & \triangleq \sharp \mathtt{x} = i. \end{array}$$

The translation t is homomorphic for Boolean connectives and remember that $\sharp\mathtt{x} = i$ is a shortcut for a formula in $\texttt{SL}(*)$ of size $\mathcal{O}(i)$ (see Section 2.2). Similarly, $\mathtt{x} \rightarrow^{+} \mathtt{y}$ and $\sharp\mathtt{x} \leq \mathrm{card}(\Sigma)$ belongs to $\texttt{SL}(*)$ (see Section 2.2). One can check that if ϕ contains at most three variables, then $\psi_{word} \wedge t(\phi)$ contains at most five variables.

□

As a corollary of Corollary 3.3, we obtain an alternative decidability proof of the entailment problem for the fragment of SL considered in [3]. We have established decidability for a fragment of SL larger than the one considered in [3] (for which the entailment problem is shown to be 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 unique patterns instead of distinguished variables.

4. A Decidable Fragment of SL with a Restricted Use of $-\!\!*$

In Section 3, we have seen that SL($*$) satisfiability is decidable whereas satisfiability for full SL will be shown to be undecidable. However, SL($*$) is certainly not the largest decidable fragment of SL. In this section, we investigate another decidable extension of SL($*$) thanks to a restricted use of the magic wand; quantification over disjoint heaps is done only for heaps whose domain has cardinality smaller than some fixed n (details will follow). Since the forthcoming extension is closed under negation, this also corresponds to a restricted use of the operator $\overset{\neg}{-\!\!*}$.

4.1. A Restricted Use of $\overset{\neg}{-\!\!*}$

Let us define SL($* + \overset{n}{\overset{\neg}{-\!\!*}}$) as an extension of SL($*$) by adding the binary operators $\overset{n}{\overset{\neg}{-\!\!*}}$ for every $n \in \mathbb{N}$. Unlike the plain operator $\overset{\neg}{-\!\!*}$, a formula with outermost connective $\overset{n}{\overset{\neg}{-\!\!*}}$ states the existence of a disjoint heap for which the cardinality of the domain is bounded by n. More formally, we require that $(s,h) \models \phi_1 \overset{n}{\overset{\neg}{-\!\!*}} \phi_2$ iff there is $h' \perp h$ such that $\mathrm{card}(\mathtt{dom}(h')) \leq n$, $(s,h') \models \phi_1$ and $(s, h * h') \models \phi_2$.

SL($* + \overset{n}{\overset{\neg}{-\!\!*}}$) allows to encode the restricted use of the magic wand in the Hoare-like proof systems as in the backward-reasoning form rule (MUBR) recalled below, see also [35]:

$$\frac{}{\{(\exists \mathtt{z}\ \mathtt{x} \mapsto \mathtt{z}) * ((\mathtt{x} \mapsto \mathtt{y}) -\!\!* \ \phi)\}\ [\mathtt{x}] := \mathtt{y}\ \{\phi\}}$$

It is easy to show that $(\mathtt{x} \mapsto \mathtt{y}) -\!\!* \ \phi$ is equivalent to $\neg((\mathtt{x} \mapsto \mathtt{y}) \overset{1}{\overset{\neg}{-\!\!*}} \neg\phi)$. Typically, whenever the left argument of a formula with outermost connective $-\!\!*$ has only models of bounded size, this trick can be applied again. Let us push a bit further this idea.

4.2. Bounding the Cardinal of Heap Domains

Let $\mathtt{SL}^{-}(*)$ be the fragment of $\mathtt{SL}(*)$ defined by the grammar below and whose formulae are also interpreted over memory states:

$$\phi ::= \perp \mid \mathtt{x} \mapsto \mathtt{y} \mid \mathtt{size} \leq k \mid \mathtt{size} = k \mid \phi * \phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \exists\ \mathtt{x}\ \phi$$

where $k \in \mathbb{N}$. Since $\mathtt{SL}^{-}(*)$ is not closed under negation, it makes sense to consider both $\mathtt{size} \leq k$ and $\mathtt{size} = k$. Anyhow, we shall show that $\mathtt{size} \leq k$ can be expressed differently. The satisfaction relation is defined as for $\mathtt{SL}$ with the obvious following update: $(s, h) \models \mathtt{size} \leq k$ iff $\mathrm{card}(\mathtt{dom}(h)) \leq k$. Observe that $\mathtt{size} = k$ with $k \geq 1$ is also equivalent to the formula below (in $\mathtt{SL}^{-}(*)$):

$$\exists\ \mathtt{x}_1, \ldots, \mathtt{x}_k\ ((\exists\ \mathtt{y}\ \mathtt{x}_1 \mapsto \mathtt{y}) * \cdots * (\exists\ \mathtt{y}\ \mathtt{x}_k \mapsto \mathtt{y}))$$

Let $|\,\phi\,|$ denote the size of a formula ϕ in $\mathtt{SL}^{-}(*)$ with the natural numbers encoded with a unary representation.

Lemma 4.1 *For any $\phi \in \mathtt{SL}^{-}(*)$, if $(s, h) \models \phi$ then* $\mathrm{card}(\mathtt{dom}(h)) \leq |\,\phi\,|$.

The proof is by a straightforward structural induction. Since computing $|\,\phi\,|$ from ϕ can be done in polynomial-time, we obtain the following reduction that becomes especially interesting after showing decidability of $\mathtt{SL}(* + \overset{n}{-\!\!*})$.

Lemma 4.2 *There is a polynomial-time reduction from satisfiability for* $\mathtt{SL}$ *restricted to formulae such that the left argument of any $-\!\!*$-formula belongs to $\mathtt{SL}^{-}(*)$ to satisfiability for $\mathtt{SL}(* + \overset{n}{-\!\!*})$.*

In order to establish the above lemma, it is sufficient to observe that $\phi -\!\!* \psi$ is equivalent to $\neg(\phi \overset{|\phi|}{-\!\!*} \neg\psi)$ whenever $\phi \in \mathtt{SL}^{-}(*)$ and $\psi \in \mathtt{SL}$.

4.3. Symbolic Disjoint Heaps

In order to show decidability for $\mathtt{SL}(* + \overset{n}{-\!\!*})$, we define a reduction into $\mathtt{SL}(*)$. The translation is based on a simple observation: since a formula with outermost connective $\overset{n}{-\!\!*}$ requires the existence of a disjoint heap whose domain size is at most n, this new heap can be encoded by a set of pairs of variables of cardinality n. Hence, a heap of size at most n disjoint from (s, h) can be represented symbolically by a set $\mathtt{C} = \{(\mathtt{y}_1, \mathtt{z}_1), \ldots, (\mathtt{y}_n, \mathtt{z}_n)\}$ such that $\{s(\mathtt{y}_1), \ldots, s(\mathtt{y}_n)\} \cap \mathtt{dom}(h) = \emptyset$ and $s(\mathtt{y}_i) = s(\mathtt{y}_j)$ implies $s(\mathtt{z}_i) = s(\mathtt{z}_j)$, naturally encoding the heap $h(\mathtt{C}) = \{s(\mathtt{y}_i) \mapsto s(\mathtt{z}_i) : s(\mathtt{y}_i) \neq nil,\ 1 \leq i \leq n\}$ assuming that *nil* is a distinguished value represented by some dedicated variable `null`. This is the only place in the paper where *nil* shall be used and as usual we require that *nil* cannot belong to the domain of heaps (see Section 2.1). The set $\mathtt{C} = \{(\mathtt{y}_1, \mathtt{z}_1), \ldots, (\mathtt{y}_n, \mathtt{z}_n)\}$ represents a heap with at most n memory cells, even though $\mathtt{C}$ contains exactly n pairs. However, whenever $s(\mathtt{y}_i) = nil$, the pair $(\mathtt{y}_i, \mathtt{z}_i)$ does not encode any new memory cell. In terms of formulae, $(\mathtt{y}_i, \mathtt{z}_i)$ encodes a memory cell iff $\mathtt{y}_i \neq \mathtt{null}$ holds true. This shall be intensively used in forthcoming formulae.

Let us provide now the formal definitions. A *symbolic disjoint heap* $\mathtt{C}$ for the memory state (s, h) is a finite set of pairs of variables $\{(\mathtt{y}_1, \mathtt{z}_1), \ldots, (\mathtt{y}_n, \mathtt{z}_n)\}$ such that

- $\{s(\mathtt{y}_1), \ldots, s(\mathtt{y}_n)\} \cap \mathtt{dom}(h) = \emptyset$.
- For $1 \leq i, j \leq n$, $s(\mathtt{y}_i) = s(\mathtt{y}_j)$ implies $s(\mathtt{z}_i) = s(\mathtt{z}_j)$.

The heap represented by $\mathtt{C}$, written $h(\mathtt{C})$, is defined by $h(\mathtt{C}) \stackrel{\text{def}}{=} \{s(\mathtt{y}_i) \mapsto s(\mathtt{z}_i) : s(\mathtt{y}_i) \neq nil,\ 1 \leq i \leq n\}$. Observe that $\mathrm{card}(\mathtt{dom}(h(\mathtt{C}))) \leq n$ and $h(\mathtt{C}) \perp h$. $\mathtt{C}$ is said to be of *length* n.

Lemma 4.3 *Given a memory state (s, h) and h' such that $h' \perp h$ and $\mathrm{card}(\mathtt{dom}(h')) \leq n$, there exists a symbolic disjoint heap $\mathtt{C}$ of length n such that $h' = h(\mathtt{C})$ and s' may differ from s at most for the variables occurring in $\mathtt{C}$.*

The proof is by an easy verification by symbolically representing h' with new variables, whence the store s'.

Below we introduce simple formulae useful to separate a symbolic disjoint heap or to extend a symbolic disjoint heap by another symbolic disjoint heap. Given $\mathtt{C} = \{(\mathtt{y}_1, \mathtt{z}_1), \ldots, (\mathtt{y}_n, \mathtt{z}_n)\}$, $\mathtt{C}^0 = \{(\mathtt{y}^0_1, \mathtt{z}^0_1), \ldots, (\mathtt{y}^0_n, \mathtt{z}^0_{n_0})\}$ and $\mathtt{C}^1 = \{(\mathtt{y}^1_1, \mathtt{z}^1_1), \ldots, (\mathtt{y}^1_n, \mathtt{z}^1_{n_1})\}$, we write $\mathtt{C} = \mathtt{C}^0 * \mathtt{C}^1$ to denote the conjunction of the formulae below:

- $h(\mathtt{C})$ is included in $h(\mathtt{C}^0) \cup h(\mathtt{C}^1)$:

$$\bigwedge_{1 \leq i \leq n} \big(\bigvee_{1 \leq i \leq n_0} \mathtt{y}^0_j = \mathtt{y}_i \big) \vee \big(\bigvee_{1 \leq i \leq n_1} \mathtt{y}^1_j = \mathtt{y}_i \big)$$

- $h(\mathtt{C}^0) \cup h(\mathtt{C}^1)$ is included in $h(\mathtt{C})$:

$$\begin{array}{rlcl} & \bigwedge_{1 \leq i \leq n_0} \left(\mathtt{y}_j^0 \neq \mathtt{null} \right. & \Rightarrow & \left. \bigvee_{1 \leq i \leq n} \mathtt{y}_i = \mathtt{y}_j^0 \right) \\ \wedge & \bigwedge_{1 \leq i \leq n_1} \left(\mathtt{y}_j^1 \neq \mathtt{null} \right) & \Rightarrow & \left. \bigvee_{1 \leq i \leq n} \mathtt{y}_i = \mathtt{y}_j^1 \right) \end{array}$$

- $h(\mathtt{C}^0)$ and $h(\mathtt{C}^1)$ encode a function:

$$\bigwedge_{1 \leq j,j' \leq n_0} (\mathtt{y}_j^0 = \mathtt{y}_{j'}^0 \Rightarrow \mathtt{z}_j^0 = \mathtt{z}_{j'}^0) \wedge \bigwedge_{1 \leq j,j' \leq n_1} (\mathtt{y}_j^1 = \mathtt{y}_{j'}^1 \Rightarrow \mathtt{z}_j^1 = \mathtt{z}_{j'}^1)$$

- $h(\mathtt{C}^0)$ and $h(\mathtt{C}^1)$ are disjoint:

$$\bigwedge_{1 \leq i \leq n_0} \bigwedge_{1 \leq i \leq n_1} ((\mathtt{y}_j^0 \neq \mathtt{null}) \vee (\mathtt{y}_{j'}^1 \neq \mathtt{null})) \Rightarrow (\mathtt{y}_j^0 \neq \mathtt{y}_{j'}^1)$$

We provide a few lemmas whose easy proofs are omitted. However, they will be helpful to prove correctness in Section 4.4.

Lemma 4.4 *Let $\mathtt{C}$ be a symbolic disjoint heap of length n for (s,h) and $\mathtt{C}^0 = \{(\mathtt{y}_1^0, \mathtt{z}_1^0), \ldots, (\mathtt{y}_n^0, \mathtt{z}_{n_0}^0)\}$ and $\mathtt{C}^1 = \{(\mathtt{y}_1^1, \mathtt{z}_1^1), \ldots, (\mathtt{y}_n^1, \mathtt{z}_{n_1}^1)\}$ be symbolic disjoint heaps whose variables do not occur in $\mathtt{C}$. Let s' be a store that may differ from s at most for the variables occurring in $\mathtt{C}^0$ and $\mathtt{C}^1$. Assume moreover that $(s',h) \models \mathtt{C} = \mathtt{C}^0 * \mathtt{C}^1$. Then, $h(\mathtt{C}^0)$ and $h(\mathtt{C}^1)$ are symbolic disjoint heaps for (s',h), $h(\mathtt{C}^0) \perp h(\mathtt{C}^1)$ and $h(\mathtt{C}^0) * h(\mathtt{C}^1) = h(\mathtt{C})$.*

Again, the proof is by easy verification and we can also get a converse property.

Lemma 4.5 *Let $\mathtt{C}$ be a symbolic disjoint heap of length n for (s,h). Let $h_0 * h_1 = h(\mathtt{C})$. There exist symbolic disjoint heaps $\mathtt{C}^0$ and $\mathtt{C}^1$ for (s',h) such that variables in $\mathtt{C}$, $\mathtt{C}^0$ and $\mathtt{C}^1$ are mutually disjoint, s' may differ from s at most for the variables occurring in $\mathtt{C}^0$ and $\mathtt{C}^1$, $h_0 = h(\mathtt{C}^0)$, $h_1 = h(\mathtt{C}^1)$ and $(s',h) \models \mathtt{C} = \mathtt{C}^0 * \mathtt{C}^1$.*

Let us now consider the corresponding lemmas to build disjoint heaps.

Lemma 4.6 *Let $\mathtt{C}^0$ be a symbolic disjoint heap for (s,h), $\mathtt{C}$ and $\mathtt{C}^1$ be symbolic disjoint heaps whose variables do not occur in $\mathtt{C}^0$, and such that $(s',h) \models \mathtt{C} = \mathtt{C}^0 * \mathtt{C}^1$, where s' may differ from s at most for the variables occurring in $\mathtt{C}$ and $\mathtt{C}^1$. Then, $\mathtt{C}$ and $\mathtt{C}^1$ are symbolic disjoint heaps for (s',h).*

We can also get a converse property.

Lemma 4.7 *Let* $\mathtt{C}^0$ *be a symbolic disjoint heap for* (s, h) *and,* h' *be disjoint from* $h * h(\mathtt{C}^0)$ *and the cardinal of its domain is less than* n*. There exists a symbolic disjoint heap* $\mathtt{C}^1$ *of length* n *for* (s, h) *such that* $h' = h(\mathtt{C}^1)$*,* $h' * h(\mathtt{C}^0) = h(\mathtt{C}^0 \cup \mathtt{C}^1)$ *and* $(s', h) \models (\mathtt{C}^0 \cup \mathtt{C}^1) = \mathtt{C}^0 * \mathtt{C}^1$ *(*s' *may differ from* s *at most for the variables occurring in* $\mathtt{C}^1$*).*

4.4. The Translation

The recursive translation function is of the form $t(\psi, \mathtt{C}, F)$ where ψ is a subformula to be translated, $\mathtt{C}$ has the format of some symbolic disjoint heap and $F \in \{0, 1\}$ is a flag that specifies whether ψ is evaluated under $h(\mathtt{C})$ $(F = 0)$ or under $h * h(\mathtt{C})$ $(F = 1)$.

Before defining the recursive map t, let us mention that a formula ϕ is translated into $t(\phi, \emptyset, 1)$.

- $t(\mathtt{x} = \mathtt{x}', \mathtt{C}, F) = \mathtt{x} = \mathtt{x}'$.
- $t(\mathtt{x} \hookrightarrow \mathtt{x}', \mathtt{C}, 1) = (\mathtt{x} \hookrightarrow \mathtt{x}') \vee t(\mathtt{x} \hookrightarrow \mathtt{x}', \mathtt{C}, 0)$.
- $t(\mathtt{x} \hookrightarrow \mathtt{x}', \mathtt{C}, 0) = \bigvee_{(\mathtt{y},\mathtt{z}) \in \mathtt{C}} (\mathtt{y} \neq \mathtt{null} \wedge \mathtt{y} = \mathtt{x} \wedge \mathtt{z} = \mathtt{x}')$.
- t is homomorphic for Boolean connectives and first-order quantification (up to renaming quantified variables to avoid capturing variables of $\mathtt{C}$).
- $t(\psi \overset{0}{-\!\!*} \psi', \mathtt{C}, F) = t(\psi, \emptyset, 0) \wedge t(\psi', \mathtt{C}, F)$
- $t(\psi \overset{n}{-\!\!*} \psi', \mathtt{C}, F)$ for $n \geq 1$ is equal to

$$\exists \mathbf{v}\ (\mathtt{C} \cup \mathtt{C}') = \mathtt{C} * \mathtt{C}' \wedge t(\psi, \mathtt{C}', 0) \wedge t(\psi', \mathtt{C} \cup \mathtt{C}', F)\ \wedge \bigwedge_{(\mathtt{y},\mathtt{z}) \in \mathtt{C}'} \neg \mathtt{alloc}\ (\mathtt{y})$$

  where $\mathbf{v}$ is a sequence of n pairs of fresh variables from the symbolic disjoint heap $\mathtt{C}'$.

- $t(\psi * \psi', \mathtt{C}, F)$ with $\mathtt{C}$ of length n is equal to

$$\exists \mathbf{v}\ (t(\psi, \mathtt{C}^0, F)\ *\ t(\psi', \mathtt{C}^1, F))\ \wedge\ \mathtt{C} = \mathtt{C}^0 * \mathtt{C}^1$$

  where $\mathbf{v}$ is a sequence of $2n$ pairs of fresh variables from the symbolic disjoint heaps $\mathtt{C}^0$ and $\mathtt{C}^1$ of length n.

Even though in the worst-case there is an exponential number of ways to divide a heap into two disjoint heaps, our translation remains in polynomial time. The soundness of the translation is guaranteed by the lemma below whose proof is by structural induction and uses the previous lemmas.

Lemma 4.8 *Let* $\mathtt{C}$ *be a symbolic disjoint heap for* (s,h). *For all formulae* ψ *in* $\mathtt{SL}(* + \overset{n}{-\!\!*})$, *we have*

- $(s, h(\mathtt{C})) \models \psi$ *iff* $(s,h) \models t(\psi, \mathtt{C}, 0)$, *and*
- $(s, h * h(\mathtt{C})) \models \psi$ *iff* $(s,h) \models t(\psi, \mathtt{C}, 1)$.

Proof The proof is by structural induction on ψ. The induction hypothesis is of the following form: for every ψ' whose size is strictly smaller than the size of ψ, if $\mathtt{C}'$ be a symbolic disjoint heap for (s'', h''), then we have

$$\begin{array}{lrcll} (1) & (s'', h(\mathtt{C}')) \models \psi' & \text{iff} & (s'', h'') \models t(\psi', \mathtt{C}', 0), & \text{and} \\ (2) & (s'', h'' * h(\mathtt{C}')) \models \psi' & \text{iff} & (s'', h'') \models t(\psi', \mathtt{C}', 1). & \end{array}$$

The base case for atomic formulae is by an easy verification as well as the cases in the induction step for Boolean connectives and first-order quantification. We treat below the case $\psi = \psi_1 * \psi_2$, the case $\psi = \psi_1 \overset{n}{-\!\!*} \psi_2$ can be treated analogously using Lemmas 4.6 and 4.7.

Suppose $(s, h(\mathtt{C})) \models \psi_1 * \psi_2$. There exist h_1 and h_2 such that $h_1 * h_2 = h(\mathtt{C})$, $(s, h_1) \models \psi_1$ and $(s, h_2) \models \psi_2$. By Lemma 4.5, there exist symbolic disjoint heaps $\mathtt{C}_1$ and $\mathtt{C}_2$ (with fresh variables) for (s', h) such that s' may differ from s at most for the variables occurring in $\mathtt{C}_1 \cup \mathtt{C}_2$, $h_1 = h(\mathtt{C}_1)$ and $h_2 = h(\mathtt{C}_2)$. Since each $\mathtt{C}_i$ is a disjoint symbolic heap for (s', h_i), by the induction hypothesis, $(s', h) \models t(\psi_1, \mathtt{C}_1, 0)$ and $(s', h) \models t(\psi_2, \mathtt{C}_2, 0)$. Moreover, $(s', h) \models \mathtt{C} = \mathtt{C}_1 * \mathtt{C}_2$ (observe that satisfaction of $\mathtt{C} = \mathtt{C}_1 * \mathtt{C}_2$ depends only on the store). Hence,

$$(s,h) \models \exists \mathbf{v} \ (t(\psi_1, \mathtt{C}_1, 0) \ * \ t(\psi_2, \mathtt{C}_2, 0)) \ \wedge \ \mathtt{C} = \mathtt{C}_1 * \mathtt{C}_2$$

where $\mathbf{v}$ is the sequence of variables from $\mathtt{C}_1$ and $\mathtt{C}_2$. Consequently, we have $(s,h) \models t(\psi_1 * \psi_2, \mathtt{C}, 0)$.

Similarly, suppose $(s, h * h(\mathtt{C})) \models \psi_1 * \psi_2$. There exist h_1, h_2, h_1' and h_2' such that

$$h_1 * h_2 = h(\mathtt{C}), \qquad h_1' * h_2' = h, \qquad (s, h_1' * h_1) \models \psi_1, \qquad (s, h_2' * h_2) \models \psi_2.$$

By Lemma 4.5, there exist symbolic disjoint heaps $\mathtt{C}_1$ and $\mathtt{C}_2$ (with fresh variables) for (s', h) such that s' may differ from s at most for the variables occurring in $\mathtt{C}_1$, $\mathtt{C}_2$, $h_1 = h(\mathtt{C}_1)$ and $h_2 = h(\mathtt{C}_2)$. Since each $\mathtt{C}_i$ is a symbolic disjoint heap for (s', h_i), by the induction hypothesis,

$$(s', h'_1 * h(\mathtt{C}_1)) \models t(\psi_1, \mathtt{C}_1, 1) \qquad \text{and} \qquad (s', h'_2 * h(\mathtt{C}_2)) \models t(\psi_2, \mathtt{C}_2, 1).$$

Moreover, $(s', h) \models \mathtt{C} = \mathtt{C}_1 * \mathtt{C}_2$. Hence,

$$(s, h) \models \exists \mathbf{v}\ (t(\psi_1, \mathtt{C}_1, 1)\ *\ t(\psi_2, \mathtt{C}_2, 1))\ \wedge\ \mathtt{C} = \mathtt{C}_1 * \mathtt{C}_2$$

So, $(s, h) \models t(\psi_1 * \psi_2, \mathtt{C}, 1)$.

Now suppose $(s, h) \models t(\psi_1 * \psi_2, \mathtt{C}, 0)$, that is

$$(s, h) \models \exists \mathbf{v}\ (t(\psi, \mathtt{C}^0, F)\ *\ t(\psi', \mathtt{C}^1, F))\ \wedge\ \mathtt{C} = \mathtt{C}^0 * \mathtt{C}^1$$

where $\mathbf{v}$ corresponds to the sequence of variables from the fresh symbolic disjoint heaps $\mathtt{C}^0$ and $\mathtt{C}^1$. Hence there exists a store s' that may differ from s at most for the variables occurring in $\mathbf{v}$ such that

$$(s', h) \models (t(\psi_1, \mathtt{C}^0, 0)\ *\ t(\psi_2, \mathtt{C}^1, 0))\ \wedge\ \mathtt{C} = \mathtt{C}^0 * \mathtt{C}^1$$

By the induction hypothesis, $(s', h(\mathtt{C}_1)) \models \psi_1$ and $(s', h(\mathtt{C}_2)) \models \psi_2$. By Lemma 4.4, $h(\mathtt{C}_1) \perp h(\mathtt{C}_2)$ and $h(\mathtt{C}_1) * h(\mathtt{C}_2) = h(\mathtt{C})$. Consequently, we have $(s', h(\mathtt{C})) \models \psi_1 * \psi_2$. Since variables in $\mathbf{v}$ do not occur in $\psi_1 * \psi_2$, we get $(s, h(\mathtt{C})) \models \psi_1 * \psi_2$.

Similarly, $(s, h) \models t(\psi_1 * \psi_2, \mathtt{C}, 1)$ implies $(s, h * h(\mathtt{C})) \models \psi_1 * \psi_2$ by Lemma 4.4. □

This leads to the main result of this section.

Theorem 4.9

(I) *There is a polynomial-time reduction from* $\mathtt{SL}(* + \overset{n}{-\!\!*})$ *satisfiability problem to* $\mathtt{SL}(*)$ *satisfiability problem.*

(II) $\mathtt{SL}(* + \overset{n}{-\!\!*})$ *satisfiability is decidable.*

Proof (II) is a consequence of (I) by using the decidability of $\mathtt{SL}(*)$ satisfiability (see Section 3).

(I) By Lemma 4.8, for every memory state (s, h), we have $(s, h * h(\emptyset)) \models \psi$ iff $(s, h) \models t(\psi, \emptyset, 1)$ where $t(\psi, \emptyset, 1)$ is an $\mathtt{SL}(*)$ formula and $\emptyset$ denotes the empty symbolic disjoint heap. Moreover, we have seen that $t(\psi, \emptyset, 1)$ can be built in polynomial-time assuming that the natural numbers are represented with a unary encoding in ψ. Since $h * h(\emptyset)$ is equal to h, the formulae ψ and $t(\psi, \emptyset, 1)$ hold true at the same memory states. □

We then obtain the following interesting corollary.

Corollary 4.10 *Satisfiability for* $\mathtt{SL}$ *restricted to formulae such that the left argument of any* $-\!\!*$*-formula belongs to* $\mathtt{SL}^{-}(*)$ *is decidable.*

5. Expressing Advanced Arithmetical Constraints in $\mathtt{SL}(-\!\!*)$

In this section, we show how $\mathtt{SL}(-\!\!*)$ can be used to express cardinality constraints on finite sets of locations that are defined by logical predicates. Given a formula $\phi(\mathtt{x})$ with free variable $\mathtt{x}$ and a model (s, h), we write $\sharp\phi$ to denote the number of locations l such that $(s[\mathtt{x} \mapsto l], h) \models \phi(\mathtt{x})$. As a consequence of $\mathtt{SL} \equiv \mathtt{SO}$, for all formulae $\phi(\mathtt{x})$ and $\psi(\mathtt{x})$ from either $\mathtt{SL}$ or $\mathtt{SO}$, there is a formula that can capture a constraint of the form $\sharp\phi < \sharp\psi$. However, in order to show $\mathtt{SL} \equiv \mathtt{SO}$, we shall need to establish this expressiveness result for some specific formulae ϕ and ψ. More precisely, this is done with the predicates "immediate successor of $\mathtt{z}$", $\phi(\mathtt{x}) = \mathtt{x} \hookrightarrow \mathtt{z}$, whence $\sharp\phi$ is the number of predecessors of $\mathtt{z}$. In Section 5.2, numbers of predecessors are compared. The proof of this result is subject to technical complications but its essence is not so intricate, and it is better illustrated by encoding other kinds of cardinality constraints. For this reason, we make a slight detour in our presentation by first sketching the encoding of the cardinality constraints for the predicate $\mathit{reachable}_{\mathtt{z}}(\mathtt{x}) = \mathtt{z} \rightarrow^{*} \mathtt{x}$. This turns out to be a bit simpler to define, and already it provides the key ingredients of our proof. This can be viewed as a warm-up before dealing with the predicate $\mathtt{x} \hookrightarrow \mathtt{z}$.

5.1. Comparing Two List Lengths

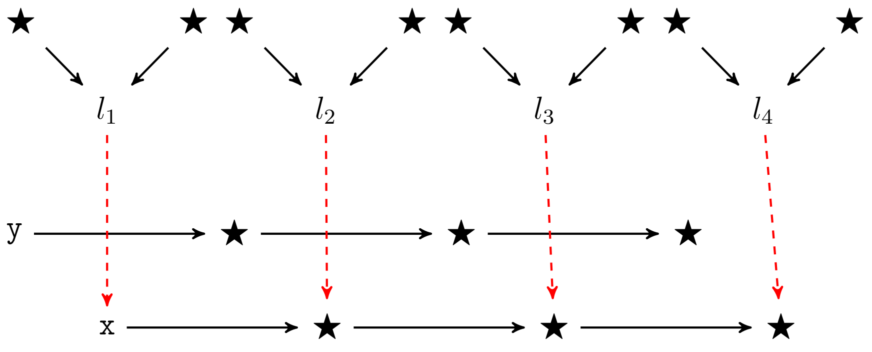
Let us first restrict our attention to models composed of two acyclic lists starting at $\mathtt{x}$ and $\mathtt{y}$ respectively, with no other allocated cells, and with the additional constraint that no location is reachable from $\mathtt{x}$ and $\mathtt{y}$ simultaneously. We aim now at expressing the fact that both lists have the same length n using the magic wand. To do so, we can say that there exist n locations $l_1, \ldots, l_n$ that are not allocated and for which there is a one-one correspondence between these locations and the ones of the list starting at $\mathtt{x}$, and on the other hand there is another one-one correspondence between these same locations and the ones of the list starting at $\mathtt{y}$. The gain for considering non allocated cells is that the one-one correspondence can be materialized by allocating $l_1, \ldots, l_n$ so that each of them points to the cell it is in correspondence with. The trickiest point is then how to materialize the guess of the locations $l_1, \ldots, l_n$ in such a way that it is possible to refer to them later without confusing them with the cells that were initially allocated. To do so, we may observe that in the original model, all locations have at most one predecessor. We can thus identify some extra locations $l_1, \ldots, l_n$ if we impose them to admit exactly two predecessors. With these intuitions in mind, the property that the length of the list starting at `x` is equal to the length of the list starting at `y` can be expressed by a formula of the form below:

$$\phi \quad -\!\!\circledast \big((\psi -\!\!\circledast \varphi(\mathtt{x},\mathtt{y})) \wedge (\psi -\!\!\circledast \varphi(\mathtt{y},\mathtt{x}))\big)$$

where:

- ϕ expresses that all the locations have either 0 or 2 predecessors,
- ψ expresses that all the locations have either 0 or 1 predecessor,
- $\varphi(\mathtt{x},\mathtt{y})$ expresses the situation depicted in Figure 2:
  1. all the locations reachable from `x` have exactly two immediate predecessors, except `x` that has one predecessor only;
  2. among these one or two predecessors, the one that is not reachable from `x` has itself exactly two immediate predecessors which themselves do not have immediate predecessors;
  3. all extra allocated locations are only the ones of the list `y`.

Figure 2 How to compare the length of two lists: situation $\varphi(\mathtt{x},\mathtt{y})$, with sub-model satisfying ϕ in bold, and sub-model satisfying ψ in broken line.



We claim that there exist such formulae ϕ, ψ, and $\varphi(\mathtt{x},\mathtt{y})$ in SL, although we do not plan to provide details herein. We shall do it for constraints about the numbers of predecessors. Before doing so, let us first notice that it is not difficult to adapt this technique to express richer constraints on the length of two lists, as for instance the property that one list is one cell longer than another one, and thus using a reduction to counter machines similar with [7], this entails the undecidability of $\texttt{SL}(-\!\!*)$. However, we were not able to encode $\texttt{SO}$ by using cardinality constraints on list lengths, but rather on comparing numbers of predecessors.

Let us also remark that the above construction relies on the fact that in the considered models, all the locations have at most one predecessor. In the general case, it could be harder to distinguish the locations that are initially allocated in the models, and the ones that correspond to the guessed locations $l_1, \ldots l_n$. This last point justifies why the construction presented at the next section is a bit more technical. Actually, we shall rely on a reduction to models where all the locations have at least three predecessors. However, the key ideas are essentially the same.

5.2. Comparing the Numbers of Predecessors

In this section, we show how $\texttt{SL}(-\!\!*)$ can express properties of the form $\sharp\texttt{x}+c \bowtie \sharp\texttt{y}+c'$ with $c, c' \in \mathbb{N}$ and $\bowtie \in \{=, \geq, \leq\}$ where $\sharp\texttt{x}$ denotes the number of predecessors of $s(\texttt{x})$ in a model. This is a key property in the forthcoming proof establishing that weak second-order logic is equivalent to $\texttt{SL}(-\!\!*)$. Note that $\sharp\texttt{x} \bowtie c$ can be easily expressed in $\texttt{SL}(-\!\!*)$, even without magic wand (indeed c is a fixed value). By contrast, expressing a constraint $\sharp\texttt{x} \bowtie \sharp\texttt{y}+c$ is natural in second-order logic, for instance by introducing an adequate finite binary relation between the predecessors of $\texttt{x}$ and those of $\texttt{y}$. We show below that this can be done also in $\texttt{SL}(-\!\!*)$ but requires much more work.

In a nutshell, expressing constraints of the form $\sharp\texttt{x} + c \bowtie \sharp\texttt{y} + c'$ will be done as follows. First, thanks to Boolean connectives it is sufficient to express properties of the form $\sharp\texttt{x}+c \leq \sharp\texttt{y}+c'$ with $c, c' \in \mathbb{N}$ (strictly speaking, we can assume that $c \times c' = 0$). Moreover, $\sharp\texttt{x} + c \leq \sharp\texttt{y} + c'$ is precisely equivalent to the fact that for all $n \in \mathbb{N}$, $\sharp\texttt{y}-c \leq n$ implies $\sharp\texttt{x}-c' \leq n$ (indeed $N \leq N'$ iff for every $M \geq 0$, we have $N' \leq M$ implies $N \leq M$). Quantification over the set of natural numbers will be simulated by a quantification over disjoint heaps in which n is exactly the cardinal of their domains. Such a quantification is performed thanks to the magic wand and we require that disjoint heaps are segmented and current heap is flooded (to be defined below). A model (s, h) is *segmented* whenever $\texttt{dom}(h) \cap \texttt{ran}(h) = \emptyset$ and no location has strictly more than one predecessor. For instance, the heap h_2 in Figure 4 restricted to cells labelled by 2 is segmented. (s, h) is *flooded* when no location has one or two predecessors. The store s is indeed irrelevant for these concepts. These conditions on heaps are needed in order to guarantee that the heaps obtained from the original heap and the disjoint heaps easily determine which part of the heap has been added. A nice feature is that the fact of being flooded or segmented can be naturally expressed in $\mathtt{SL}(\mathbin{-\!\!*})$ (see Lemma 5.1). Finally, any heap such that $\sharp\mathtt{x}, \sharp\mathtt{y} \geq 3$ can be extended to a flooded heap without modifying the numbers of predecessors for $\mathtt{x}$ and $\mathtt{y}$, respectively. This explains why the term 'flooded' has been chosen. In the case $\sharp\mathtt{x} \leq 2$ or $\sharp\mathtt{y} \leq 2$, we perform a simple case analysis and we obtain Boolean combinations of constraints of the form $\sharp\mathtt{x} \bowtie c''$ or $\sharp\mathtt{y} \bowtie c''$ (that can be easily handled, details will follow).

Lemma 5.1 *There are formulae* $\mathsf{flooded}$ *and* $\mathtt{seg}$ *in* $\mathtt{SL}(\mathbin{-\!\!*})$ *such that for every model* (s, h)*,*

(I) $(s, h) \models \mathsf{flooded}$ *iff* (s, h) *is flooded,*

(II) $(s, h) \models \mathtt{seg}$ *iff* (s, h) *is segmented.*

Proof It is easy to check that the formulae below do the job.

- $\mathsf{flooded} \stackrel{\Delta}{=} \forall \mathtt{x}.\ (\sharp\mathtt{x} = 0 \vee \sharp\mathtt{x} > 2)$.
- $\mathtt{seg} \stackrel{\Delta}{=} \forall \mathtt{x}, \mathtt{y}.\ (\mathtt{x} \hookrightarrow \mathtt{y} \Rightarrow (\sharp\mathtt{y} = 1 \wedge \neg(\exists \mathtt{z}\ \mathtt{z} \hookrightarrow \mathtt{x} \vee \mathtt{y} \hookrightarrow \mathtt{z})))$.

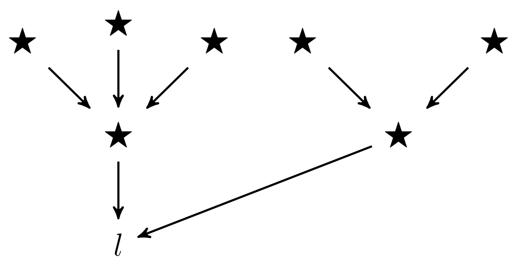
Note that the formulae $\sharp\mathtt{x} = 0$, $\sharp\mathtt{x} > 2$ and $\sharp\mathtt{y} = 1$ are indeed formulae without separating connectives. ☐

Now, we present a few crucial definitions about specific patterns in memory states, namely markers. 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

- $h(l_0) = l$ [resp. and $\mathtt{dom}(h) = \{l_0, \ldots, l_n\}$],
- for every $i \in \{1, \ldots, n\}$, $h(l_i) = l_0$ and $\sharp l_i = 0$,
- $\sharp l_0 = n$.

The marker is said to be of *degree* n with *endpoint* l (n-marker). Markers have simple structure with natural graphical representation. In Figure 3, we present an heap h containing a 2-marker and a 3-marker, both having the same endpoint l. Note that there are disjoint heaps h_1 and h_2 such that $h = h_1 * h_2$, h_1 has a strict 2-marker and h_2 has a strict 3-marker.

Figure 3 An heap with a 2-marker and a 3-marker.



A model (s, h) is said to be *k-marked* whenever there is no location in $\texttt{dom}(h)$ that does not belong to a marker of degree k. Moreover, it is *strictly* k-marked when no distinct markers share the same endpoint (no aliasing).

Markers are essential building blocks to express a constraint of the form $\sharp\texttt{x} - c \leq n$ with $c, n \in \mathbb{N}$. Before presenting the formal treatment, let us explain the principle of the encoding. Assume that h_1 is a flooded heap (i.e, no location has one or two predecessors), and h_2 is a segmented heap such that

1. h_1 and h_2 are disjoint,
2. $\text{card}(\texttt{dom}(h_2)) = n$,
3. $h_1 * h_2$ does not contain locations with two predecessors,
4. if a location l has exactly one predecessor l' in $h_1 * h_2$ then l' has no predecessor and l does not belong to $\texttt{dom}(h_1 * h_2)$.

Hence, $h_1 * h_2$ is almost flooded since the only reason for not being flooded is possibly to contain isolated memory cells from h_2. Figure 4 presents two heaps h_1 and h_2 satisfying the above conditions. Cells of the heap h_2 are labelled by 2. Note also that $h_1 * h_2$ is not flooded because of some isolated cells from h_2 such as $l \mapsto l'$.

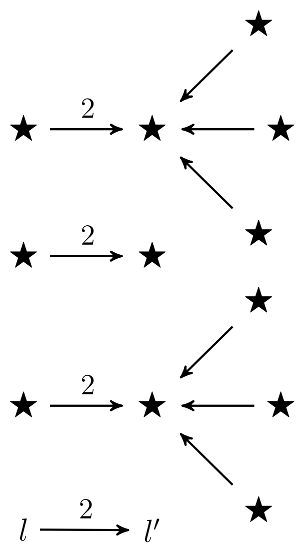
Obviously, $h_1 * h_2$ does not contain any 2-marker and in particular no predecessor of $s(\texttt{x})$ is the endpoint of some 2-marker.

A *c-completion* of $h_1 * h_2$ consists in adding a disjoint heap $h' = h'_1 * h'_2$ such that

1. h'_1 is 1-marked,
2. h'_2 is strictly 2-marked and contains exactly c distinct 2-markers.

Consider the number of 2-markers in the heap $h_1 * h_2 * h'$ resulting from such a completion. First, observe that strictly more than c 2-markers can be

Figure 4 h_1 and h_2 satisfying the conditions 1.-4.

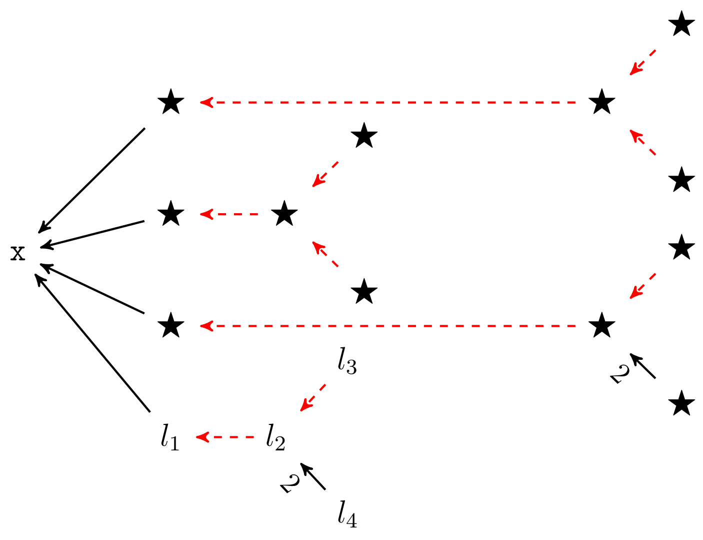


present since an isolated memory cell from h_2 and a 1-marker from h'_1 may produce a 2-marker in $h_1 * h_2 * h'$ (see the locations l_1, l_2, l_3 and l_4 in Figure 5) Second, observe that at least the c 2-markers from h' are still in $h_1 * h_2 * h'$, because the definition of * prevents a 2-marker from combining with a 1-marker to form a 3-marker. Observe also that the insertion of markers of degree strictly less than 3 in the almost flooded heap allows to safely identify them as markers in the new model. Consequently, there are at most $n + c$ predecessors of $s(\mathtt{x})$ that are endpoints of 2-markers in $h_1 * h_2 * h'$. Now, we say that $h_1 * h_2 * h'$ is $\mathtt{x}$-*completed* whenever all the predecessors of $s(\mathtt{x})$ are endpoints of 2-markers.

Figure 5 presents a 2-completion of $h_1 * h_2$ (cells in h_1 are those pointing to $\mathtt{x}$ and cells in h_2 are labelled by 2 whereas the cells of the 2-completion are represented by dashed arrows. The heap restricted to dashed edges satisfies $\mathtt{complete}_2$ – it is composed of two 2-markers and two 1-markers. Moreover, the total resulting heap is $\mathtt{x}$-completed: every predecessor of $\mathtt{x}$ is an endpoint of some 2-marker.

It is easy to observe that $\sharp\mathtt{x} - c \leq n$ iff there is a c-completion h' of $h_1 * h_2$ such that $h_1 * h_2 * h'$ is $\mathtt{x}$-completed (see the exact statement in Lemma 5.3). Lemma 5.2 below states that the heaps obtained by completion can be specified in $\mathtt{SL}(\mathrel{-\!\!*})$.

Figure 5 A 2-completion of $h_1 * h_2$ that leads to a $\mathtt{x}$-completed heap.



Lemma 5.2 *There are formulae* $\mathtt{completed}(\mathtt{x})$ *and* $\mathtt{complete}_c$ *($c \geq 0$) in* $\mathtt{SL}(-\!\!*)$ *such that for every model* (s, h)*,*

(I) $(s,h) \models \mathtt{completed}(\mathtt{x})$ *iff all the predecessors of* $s(\mathtt{x})$ *are endpoints of 2-markers,*

(II) $(s,h) \models \mathtt{complete}_c$ *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.*

Proof The formulae below do the job.

(I) $\mathtt{completed}(\mathtt{x})$ is equal to:

$$\forall \mathtt{y}\ \mathtt{y} \hookrightarrow \mathtt{x} \Rightarrow (\exists \mathtt{z}\ \mathtt{z} \hookrightarrow \mathtt{y} \wedge \sharp\mathtt{z} = 2 \wedge \forall \mathtt{z}'\ \mathtt{z}' \hookrightarrow \mathtt{z} \Rightarrow \sharp\mathtt{z}' = 0))$$

(II) In order to define $\mathtt{complete}_c$ we perform a case analysis and introduce below a few formulae. First, $\psi_0 \triangleq \top$ and let ψ_n be the formula below:

$$\exists \mathtt{x}_1 \cdots \mathtt{x}_n, \mathtt{y}_1 \cdots \mathtt{y}_n$$

$$(\bigwedge_{i \neq j} \mathtt{x}_i \neq \mathtt{x}_j) \wedge (\bigwedge_{i=1}^{n} ((\mathtt{y}_i \hookrightarrow \mathtt{x}_i) \wedge \sharp\mathtt{y}_i = 2 \wedge \forall \mathtt{z}\ \mathtt{z} \hookrightarrow \mathtt{y}_i \Rightarrow \sharp\mathtt{z} = 0))$$

$\psi_c \wedge \neg \psi_{c+1}$ states that the model contains exactly c 2-markers with disjoint endpoints. Let ψ_{cases} be the formula below:

$$\forall \mathtt{x}\ \mathtt{alloc}\ (\mathtt{x}) \Rightarrow (\psi^1_{extr}(\mathtt{x}) \vee \psi^2_{extr}(\mathtt{x}) \vee \psi^1_{end}(\mathtt{x}) \vee \psi^2_{end}(\mathtt{x}))$$

where $\psi^i_{extr}(\mathtt{x})$ [resp. $\psi^i_{end}(\mathtt{x})$] states that $h(s(\mathtt{x}))$ [resp. $h(h(s(\mathtt{x})))$] is the endpoint of some i-marker. By way of example, $\psi^1_{extr}(\mathtt{x})$ is defined as follows:

$$\sharp\mathtt{x} = 1 \wedge (\forall \mathtt{y}\ (\mathtt{y} \hookrightarrow \mathtt{x}) \Rightarrow \sharp\mathtt{y} = 0) \wedge (\exists \mathtt{y}\ \mathtt{x} \hookrightarrow \mathtt{y} \wedge \neg \exists \mathtt{z}\ \mathtt{y} \hookrightarrow \mathtt{z})$$

The formula $\mathtt{complete}_c$ is defined as the conjunction $\psi_c \wedge \neg\psi_{c+1} \wedge \psi_{cases}$.
□

We say that two heaps h_1 and h_2 are *completely disjoint* if $(\mathtt{dom}(h_1) \cup \mathtt{ran}(h_1)) \cap (\mathtt{dom}(h_2) \cup \mathtt{ran}(h_2)) = \emptyset$. Moreover, a pair of heaps (h_1, h_2) is said to be *compatible* whenever

- (s, h_1) is flooded,
- (s, h_2) is segmented,
- h_1 and h_2 are completely disjoint.

Note that h_1 and h_2 from Figure 4 are not compatible since $\mathtt{ran}(h_1) \cap \mathtt{ran}(h_2) \neq \emptyset$.

Lemma 5.3 below presents the formal statement related to the intuitive explanations that were already presented.

Lemma 5.3 *Let s be a store and (h_1, h_2) be a compatible pair of heaps such that* $\mathtt{x}$ *has i predecessors in h_1 for some $i \geq 1$. Then the following two are equivalent:*

(i) $(s, h_1 * h_2) \models \mathtt{complete}_c \mathrel{-\!\!*} \mathtt{completed}(\mathtt{x})$,

(ii) $\mathrm{card}(\mathtt{dom}(h_2)) \geq (i - c)$.

Proof Proof of (i) $\rightarrow$ (ii).
Assume (i). Let h'_1 be an 1-marked heap, h'_2 be a strict 2-marked heap with exactly c 2-markers, and $h = h_1 * h_2 * h'_1 * h'_2$ with $(s, h) \models \mathtt{completed}(\mathtt{x})$. Then, the set of endpoints from 2-markers in h includes $h_1^{-1}(s(\mathtt{x}))$ and its cardinal b satisfies $b \geq i$. Markers of degree 2 witnessing the satisfaction of $\mathtt{completed}(\mathtt{x})$ do not come from h_1 since h_1 is flooded. So, either they come

directly from h_2' or they are markers of degree 1 which have been converted into markers of degree 2 thanks to isolated cells from h_2. Let a be the number of converted markers, then $b \leq a + c$. Since none of h_1, h_2' contributes to the conversion of an 1-marker, the amount of converted markers is bounded by $\mathrm{card}(\texttt{dom}(h_2))$, i.e. $\mathrm{card}(\texttt{dom}(h_2)) \geq a$. Consequently,

$$i - c \leq b - c \leq a \leq \mathrm{card}(\texttt{dom}(h_2)).$$

Proof of (ii) $\rightarrow$ (i).
Assume (ii). In the sequel, we shall introduce locations that are involved in 2-markers; the exponents below in the locations refer to the following intended positions in the schema
$$\begin{tikzcd}[sep=small] A \arrow[dr] & & B \arrow[dl] \\ & C \arrow[d] & \\ & D & \end{tikzcd}$$
for 2-markers (of course "A" and "B" could have been permuted). By letting $N = i-c$, we have $\mathrm{card}(\texttt{dom}(h_2)) \geq N$. The set of locations $h_1^{-1}(s(\texttt{x}))$ (set of predecessors of $s(\texttt{x})$ in h_1) contains $N+c$ elements that can be written $l_1^D, \ldots, l_{N+c}^D$. Since $\mathrm{card}(\texttt{dom}(h_2)) = \mathrm{card}(\texttt{ran}(h_2))$, there exist at least N locations $l_1^C, \ldots, l_N^C$ in $\texttt{ran}(h_2)$. Moreover, since $X = \texttt{dom}(h_1 * h_2) \cup \texttt{ran}(h_1 * h_2)$ is finite, there exist distinct locations $l_1^B, \ldots, l_N^B$ that are not in X. Let h_1' be the heap disjoint from $(h_1 * h_2)$ with the memory cells below:

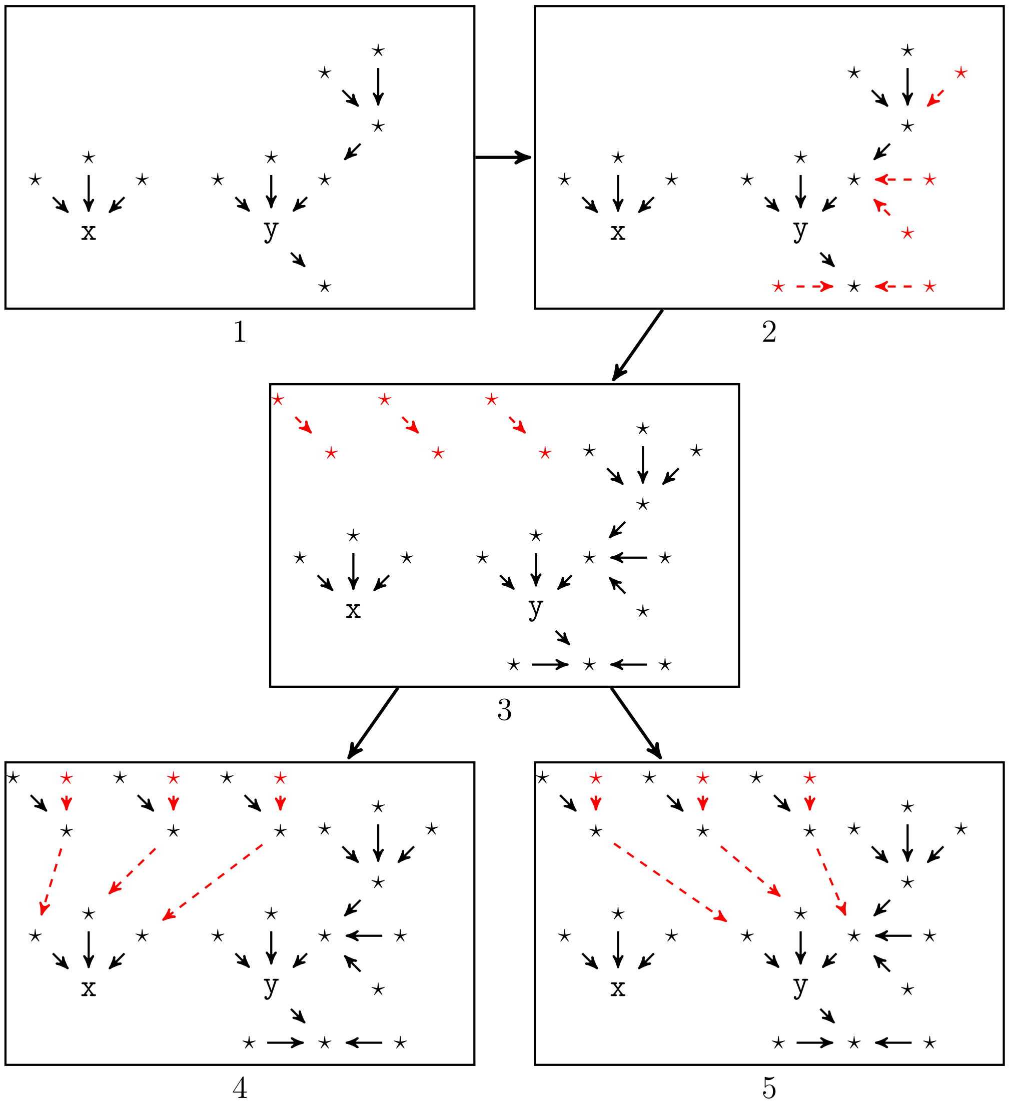
$$h_1' \;=\; \{l_1^B \mapsto l_1^C, l_1^C \mapsto l_1^D, \ldots, l_N^B \mapsto l_N^C, l_N^C \mapsto l_N^D\}$$

Let h_2' be heap disjoint from $(h_1 * h_2 * h_1')$ that contains c 2-markers with endpoints $l_{N+1}^D, \ldots, l_{N+c}^D$ respectively. It is easy to check that $(s, h_1' * h_2') \models \texttt{complete}_c$ and $(s, h_1 * h_2 * h_1' * h_2') \models \texttt{completed}(\texttt{x})$, which is sufficient to guarantee (i). $\Box$

Satisfying that for all $n \in \mathbb{N}$, $\sharp\texttt{y} - c \leq n$ implies $\sharp\texttt{x} - c' \leq n$ suggests a simple contest between two players: Spoiler aims at disproving that the constraint holds, and Duplicator tries to prove it. The whole play of the contest is depicted on Figure 6. The steps of context go as follows:

1. We start with an initial heap h_0 without any hypothesis; if $\sharp\texttt{x} \leq 2$ or $\sharp\texttt{y} \leq 2$, the contest is over (these cases are handled elsewhere), otherwise the contest may start.
2. Spoiler reduces to the case of a flooded model h_1 (whole heap on the second frame of Figure 6) by adding cells (the five new arrows in the second frame) in a controlled way- this will be formalized later.
3. Spoiler picks a segmented heap h_2 (the three new arrows in the third frame) such that $\mathrm{card}(\texttt{dom}(h_2))$ equals n and (h_1, h_2) is compatible.

Figure 6 A contest won by Duplicator. $n = 3$, $c = c' = 0$



4. Spoiler proves that $\sharp\mathtt{y} - c \leq n$ using the previous scenario.
5. Then Duplicator plays and wins if it can prove $\sharp\mathtt{x} - c' \leq n$ (note that Duplicator wins on Figure 6).

Figure 6 summarizes a contest with a successful outcome for Duplicator.

The above contest supposes that it is possible to characterize the heaps $h_1 * h_2$ such that (h_1, h_2) is compatible. A heap h is said to be *almost flooded* whenever there exist h_1 and h_2 such that $h = h_1 * h_2$ and (h_1, h_2) is compatible.

Lemma 5.4 *Let (s, h) be a memory state. h is almost flooded iff $(s, h) \models \widetilde{\mathtt{flooded}}$ with*

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

The proof of Lemma 5.4 is by an easy verification. It remains to define the formula $\mathsf{contest}(\mathtt{x}, \mathtt{y}, c, c')$ that defines a contest and that is essential to establish Theorem 5.5 below.

$$\mathsf{flooded} \wedge ((\mathtt{seg} \wedge \sharp\mathtt{x} = 0 \wedge \sharp\mathtt{y} = 0) \mathrel{-\!\!*} (\widetilde{\mathtt{flooded}}$$
$$\Rightarrow ((\mathtt{complete}_c \mathrel{-\!\!*} \mathtt{completed}(\mathtt{y})) \Rightarrow (\mathtt{complete}_{c'} \mathrel{-\!\!*} \mathtt{completed}(\mathtt{x}))))).$$

Theorem 5.5 *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 $\sharp\mathtt{x}+c \leq \sharp\mathtt{y}+c'$.*

Proof By packing the previous developments, we shall show that

($\heartsuit$) When h is flooded, $(s, h) \models \mathsf{contest}(\mathtt{x}, \mathtt{y}, c, c')$ iff $\sharp\mathtt{x} + c \leq \sharp\mathtt{y} + c'$.

Even though h is not necessarily flooded, when $\sharp\mathtt{x} \geq 3$ and $\sharp\mathtt{y} \geq 3$ it can be safely extended to a flooded heap without modifying the number of predecessors of $\mathtt{x}$ and $\mathtt{y}$. When $\sharp\mathtt{x} \leq 2$ or $\sharp\mathtt{y} \leq 2$ such an extension is not anymore possible. Nevertheless, by a simple case analysis, $\sharp\mathtt{x}+c \leq \sharp\mathtt{y}+c'$ is equivalent to $\bigvee_{i \leq 2}(\sharp\mathtt{x} = i\ \wedge \sharp\mathtt{y} \geq i + c - c') \vee \bigvee_{i \leq 2}(\sharp\mathtt{y} = i\ \wedge \sharp\mathtt{x} \leq i + c' - c)$, which can be easily expressed in $\mathtt{SL}(\mathrel{-\!\!*})$. Let us consider $\phi \stackrel{\text{def}}{=} \phi_{\text{special}} \vee \phi_{\text{main}}$ with $\phi_{\text{main}} \triangleq (\sharp\mathtt{x} = 0 \wedge \sharp\mathtt{y} = 0)\ \mathrel{-\!\!\circledast}\ \mathsf{contest}(\mathtt{x}, \mathtt{y}, c, c')$ and

$$\phi_{\text{special}} \triangleq \bigvee_{i \leq 2}(\sharp\mathtt{x} = i\ \wedge \sharp\mathtt{y} \geq i + c - c') \vee \bigvee_{i \leq 2}(\sharp\mathtt{y} = i\ \wedge \sharp\mathtt{x} \leq i + c' - c)$$

First, it is clear that $\sharp\mathtt{x} + c \leq \sharp\mathtt{y} + c'$ and ($\sharp\mathtt{x} \leq 2$ or $\sharp\mathtt{y} \leq 2$) is equivalent to $(s, h) \models \phi_{\text{special}}$. Now, suppose that $\sharp\mathtt{x} \geq 3$ and $\sharp\mathtt{y} \geq 3$. Assuming that ($\heartsuit$) holds, we have the following equivalences:

(1) $(s,h) \models (\sharp\mathtt{x} = 0 \wedge \sharp\mathtt{y} = 0) \;\mathrel{-\!\!\circledast}\; \mathsf{contest}(\mathtt{x},\mathtt{y},c,c')$.

(2) There is $h' \perp h$ such that $(s,h') \models (\sharp\mathtt{x} = 0 \wedge \sharp\mathtt{y} = 0)$ and $(s,h*h') \models \mathsf{contest}(\mathtt{x},\mathtt{y},c,c')$.

(3) There is $h' \perp h$ such that $(s,h') \models (\sharp\mathtt{x} = 0 \wedge \sharp\mathtt{y} = 0)$ and $(s,h*h') \models \mathsf{flooded}$ and $\sharp\mathtt{y} + c' \geq \sharp\mathtt{x} + c$ (in $h*h'$) by ($\heartsuit$).

(4) $\sharp\mathtt{y} + c' \geq \sharp\mathtt{x} + c$ in h.

Observe that $\sharp\mathtt{x}$ and $\sharp\mathtt{y}$ in h are equal to their values in $h*h'$ since $(s,h') \models (\sharp\mathtt{x} = 0 \wedge \sharp\mathtt{y} = 0)$. Moreover, (4) implies (3) since it is always possible to extend a model into a flooded one while preserving $\sharp\mathtt{x}$ and $\sharp\mathtt{y}$ (when $\sharp\mathtt{x} \geq 3$ and $\sharp\mathtt{y} \geq 3$).

It remains to show that ($\heartsuit$) holds true. The statements below are equivalent (h is assumed to be flooded):

1. $(s,h) \models \mathsf{contest}(\mathtt{x},\mathtt{y},c,c')$.
2. for every segmented disjoint heap h_e such that $(s,h_e) \models \sharp\mathtt{x} = \sharp\mathtt{y} = 0$, if $(s,h*h_e) \models \mathtt{complete}_c \;\mathrel{-\!\!\circledast}\; \mathtt{completed}(\mathtt{y})$ and $h*h_e$ is almost flooded, then $(s,h*h_e) \models \mathtt{complete}_{c'} \;\mathrel{-\!\!\circledast}\; \mathtt{completed}(\mathtt{x})$.
3. for every segmented disjoint heap h_e such that $(s,h_e) \models \sharp\mathtt{x} = \sharp\mathtt{y} = 0$, there exist $h' * h'_e = h * h_e$ such that (h', h'_e) is compatible and the number of predecessors of $\mathtt{x}$ and $\mathtt{y}$ in h are equal to those of $\mathtt{x}$ and $\mathtt{y}$ in h', if $\mathrm{card}(\mathtt{dom}(h')) \geq \sharp\mathtt{y} - c$, then $\mathrm{card}(\mathtt{dom}(h')) \geq \sharp\mathtt{x} - c'$.
4. for every $n \geq 0$, we have $n \geq \sharp\mathtt{y} - c$ in h implies $n \geq \sharp\mathtt{x} - c'$ in h.
5. $\sharp\mathtt{x} + c \leq \sharp\mathtt{y} + c'$.

Lemma 5.4 is used from (1) to (2). Lemma 5.3 is used for the equivalence between (2) and (3). Moreover, one needs to observe that h is flooded, h_e is a disjoint segmented heap, $(s,h_e) \models \sharp\mathtt{x} = \sharp\mathtt{y} = 0$ and $h*h_e$ is almost flooded iff there are $h' * h'_e = h * h_e$ such that (h', h'_e) is compatible and the number of predecessors of $\mathtt{x}$ and $\mathtt{y}$ in h are equal to those of $\mathtt{x}$ and $\mathtt{y}$ in h'. Equivalence between (3) and (4) is due to the fact that for every $n \geq 0$ there is a heap h_e such that $\mathrm{card}(\mathtt{dom}(h_e)) = n$, (h, h_e) is compatible and $(s,h_e) \models \sharp\mathtt{x} = \sharp\mathtt{y} = 0$. $\square$

In Section 6, only constraints of the form $\sharp\mathtt{x} + c \leq \sharp\mathtt{y} + c'$ with $c, c' \leq 3$ are used. In particular, this means that for the forthcoming formulae using advanced arithmetical constraints, $c + c'$ can be viewed as a constant.

6. $\mathtt{SL}(\multimap)$ is Equivalent to $\mathtt{SO}$

By combining Proposition 2.5 and Proposition 2.6, we know that $\mathtt{DSO}$ is at least as expressive as $\mathtt{SL}$ and there is a logarithmic-space translation from $\mathtt{SL}$ into $\mathtt{DSO}$ (logarithmic-space reductions are closed under compositions). Now, we show the converse.

A syntactic convention. 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 $\mathtt{P}_j$ can only occur in the scope of quantifications over $\mathtt{P}_1$, ..., $\mathtt{P}_{j-1}$ (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).

Encoding environments as specific parts of the memory state. Before defining the translation of a $\mathtt{DSO}$ sentence ϕ, let us explain how environments can be encoded in $\mathtt{SL}$. First, let us introduce some terminology. We say that a location l is an *extremity* in a given model if l has at least one predecessor and no predecessor of l has a predecessor. The following formula states that $s(\mathtt{x})$ is an extremity: $\mathtt{extr}(\mathtt{x}) \triangleq (\neg\exists \mathtt{y}.\ (\mathtt{y} \hookrightarrow \mathtt{x} \wedge \exists \mathtt{z}.\mathtt{z} \hookrightarrow \mathtt{y})) \wedge (\exists \mathtt{y}.\ \mathtt{y} \hookrightarrow \mathtt{x})$. In the particular case of a marker, an extremity is the location that points to the endpoint of the marker.

An environment is encoded as a finite set of new markers distinct from the original heap; this heap is called the *environment heap* (and it is written h_E). The main idea is that a pair of locations (l, l') belongs to the interpretation of a dyadic second-order variable if l and l' are the endpoints of two markers of h_E that have respectively degrees d and $d+1$.

Let us illustrate this idea on a simple example. Assume we want to express in $\mathtt{SL}$ the pure $\mathtt{SO}$ sentence "all finite orders have a minimal element", stated by the formula $\forall \mathtt{P}.\phi_{min}(\mathtt{P})$, with $\phi_{min}(\mathtt{P}) \triangleq$

$$\left(\begin{array}{ll} & \forall \mathtt{x}, \mathtt{y}.\mathtt{P}(\mathtt{x}, \mathtt{y}) \Rightarrow \big(\mathtt{P}(\mathtt{x}, \mathtt{x}) \wedge \mathtt{P}(\mathtt{y}, \mathtt{y})\big) \\ \wedge & \forall \mathtt{x}, \mathtt{y}.\big(\mathtt{P}(\mathtt{x}, \mathtt{y}) \wedge \mathtt{P}(\mathtt{y}, \mathtt{x})\big) \Rightarrow \mathtt{x} = \mathtt{y} \\ \wedge & \forall \mathtt{x}, \mathtt{y}, \mathtt{z}.\big(\mathtt{P}(\mathtt{x}, \mathtt{y}) \wedge \mathtt{P}(\mathtt{y}, \mathtt{z})\big) \Rightarrow \mathtt{P}(\mathtt{x}, \mathtt{z}) \end{array}\right) \Rightarrow \exists \mathtt{x}.\forall \mathtt{y}.\mathtt{P}(\mathtt{y}, \mathtt{x}) \Rightarrow \mathtt{x} = \mathtt{y}.$$

We could actually illustrate the idea with any other $\mathtt{SO}$ sentence using one $\mathtt{SO}$ variable only, with this variable quantified in outermost position. Let $\hat{\mathtt{P}}(\mathtt{x},\mathtt{y})$ be the $\mathtt{SL}$ formula

$$\hat{\mathtt{P}}(\mathtt{x},\mathtt{y}) \quad \triangleq \quad \exists \mathtt{x}', \mathtt{y}'.(\mathtt{x}' \hookrightarrow \mathtt{x} \wedge \mathtt{y}' \hookrightarrow \mathtt{y} \wedge \sharp \mathtt{x}' + 1 = \sharp \mathtt{y}').$$

This formula expresses that $\mathtt{x}$ and $\mathtt{y}$ are the endpoints of two markers of consecutive degrees. To any heap h, we can associate the binary relation $\hat{\mathtt{P}}_h$ composed of pairs of such locations. Conversely, any finite binary relation on locations is realized by some $\hat{\mathtt{P}}_h$. As a consequence, the $\mathtt{SO}$ formula $\forall \mathtt{P}.\phi_{min}(\mathtt{P})$ is satisfied by the empty heap if and only if the $\mathtt{SL}$ formula $\top \mathrel{-\!\!*} \phi_{min}(\hat{\mathtt{P}})\big)$ is.

The generalization of this encoding to arbitrary formulas raises several problems. The first problem is to distinguish the environment heap from the the original one (in the example above, this is solved by restricting ourselves to an original empty heap, but this is not possible in general). In the previous section, we solved this issue by first extending the original heap to a flooded heap, and then by using markers of small degrees (one or two) that were clearly distinct from the original heap. The same approach is not possible here, because one may need arbitrarily large degrees. Transforming an original heap into a flooded one in a controlled way is possible for counting the number of predecessors (see Section 5), but it might be much more difficult if the property of interest is not just a property on the number of predecessors, but an arbitrary second-order property. For all these reasons, we adopt a different strategy, and we ensure that the degree of a marker in h_E is strictly *greater* than the maximal number of predecessors of any location from the original heap. Nonetheless, our investigation on counting the number of predecessors is precious (see Section 5), and will be used when expressing that two endpoints l, l' are consecutively marked.

The second problem is, given a pair (l, l') of locations marked by markers of consecutive degrees, to determine the second-order variable $\mathtt{P}_i$ whose interpretation contains (l, l'). In the example above, we only had one second-order variable, but we may not reduce to the case of a unique second-order variable in general. To do so, we impose some more structure on h_E. First, for any natural number n, there is at most one extremity with degree n in h_E. The *spectrum* of h_E is then defined as the finite set of natural numbers n for which there is a marker of degree n in h_E. Second, we require that the spectrum of h_E, depicted as a marking of the sequence of naturals, has the following shape

$$\overbrace{\ldots\circ\circ\circ\circ}^{\text{empty}}\bullet\circ\bullet\bullet\circ\bullet\bullet\cdots\circ\bullet\bullet\circ\bullet\bullet\circ\bullet\bullet\cdots\circ\bullet\bullet\circ\bullet\overbrace{\circ\circ\circ\circ\ldots}^{\text{empty}}$$

(a symbol '$\bullet$' on position n indicates the presence of a marker of degree n, and '$\circ$' its absence). In other words, the spectrum is a finite set of naturals of the form

$$\{n \mid a \leq n \leq b \text{ and } n \neq a+1 \ (\mathsf{mod}\ 3)\}$$

for some $a, b \in \mathbb{N}$ (we later call such a spectrum a *clean* spectrum). This simple and regular structure makes the characterisation of well-formed environment heaps easier at every step of the translation (in particular, every time the environment is extended by a new quantified second-order variable). In order to identify markers that are attached to a given second-order variable,

1. we ensure that the markers of a given second-order variable follow each others in a given interval,
2. these intervals do not overlap for two distinct second-order variables,
3. there is no unused space between these intervals.

This is achieved by introducing, for each $\mathtt{P}_j$, two variables $\mathtt{z}_j^m$ and $\mathtt{z}_j^M$ that are placed on the upper and lower bound of the interval of the interpretation of $\mathtt{P}_j$. For technical reasons, mainly related to bootstrapping, we also consider the two distinguished variables $\mathtt{z}_0^m$ and $\mathtt{z}_0^M$. So, the spectrum of h_E can be graphically depicted as

$$\ldots\circ\circ\circ\circ\ \underbrace{\overset{\sharp\mathtt{z}_0^m}{\bullet}\circ\overset{\sharp\mathtt{z}_0^M}{\bullet}}_{\text{bootstrap}}\ \underbrace{\overset{\sharp\mathtt{z}_1^m}{\bullet}\circ\bullet\bullet\cdots\circ\overset{\sharp\mathtt{z}_1^M}{\bullet}}_{\text{code of }\mathtt{P}_1}\ \underbrace{\overset{\sharp\mathtt{z}_2^m}{\bullet}\circ\bullet\cdots\bullet\circ\overset{\sharp\mathtt{z}_2^M}{\bullet}}_{\text{code of }\mathtt{P}_2}\ \overset{\sharp\mathtt{z}_3^m}{\bullet}\cdots\circ\bullet\bullet\circ\overset{\sharp\mathtt{z}_n^M}{\bullet}\ \circ\circ\circ\circ\ldots$$

6.1. Encoding Environments

First, let us show how to express structural properties about the environment heap. In the proof of Lemma 6.1 below, advanced arithmetical constraints are expressed thanks to Theorem 5.5.

Lemma 6.1 *There is a formula* $\mathtt{PseudoEnv}(\mathtt{z},\mathtt{z}')$ *in* $\mathtt{SL}(\mathrel{-\!\!*})$ *such that the conditions below hold true iff* $(s, h_E) \models \mathtt{PseudoEnv}(\mathtt{z},\mathtt{z}')$*:*

- $\sharp\mathtt{z} < \sharp\mathtt{z}'$, $\sharp\mathtt{z} \equiv \sharp\mathtt{z}' + 2\ (\mathsf{mod}\ 3)$ *and* $\mathtt{z}$ *and* $\mathtt{z}'$ *are extremities.*

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

Proof The formula $\mathtt{PseudoEnv}(\mathtt{z}, \mathtt{z}')$ is the conjunction of the formulae below expressing the following properties:

1. $\sharp \mathtt{z} < \sharp \mathtt{z}'$ and $\mathtt{z}$,$\mathtt{z}'$ are extremities: $\sharp \mathtt{z} < \sharp \mathtt{z}' \wedge \mathtt{extr}(\mathtt{z}) \wedge \mathtt{extr}(\mathtt{z}')$.
2. There is no extremity with number of predecessors equal to either $\sharp \mathtt{z}+1$ or $\sharp \mathtt{z}' - 1$.
$$(\neg \exists \mathtt{x}\ \mathtt{extr}(\mathtt{x}) \wedge \sharp \mathtt{z} + 1 = \sharp \mathtt{x}) \wedge (\neg \exists \mathtt{x}\ \mathtt{extr}(\mathtt{x}) \wedge \sharp \mathtt{z}' = 1 + \sharp \mathtt{x})$$
3. There is an extremity with number of predecessors equal to $\sharp \mathtt{z}+2$ [resp. $\sharp \mathtt{z}' - 2$].
$$\exists \mathtt{x}\ (\mathtt{extr}(\mathtt{x}) \wedge (\sharp \mathtt{z} + 2 = \sharp \mathtt{x})) \wedge \exists \mathtt{x}\ (\mathtt{extr}(\mathtt{x}) \wedge (\sharp \mathtt{z}' = 2 + \sharp \mathtt{x}))$$
4. For every extremity $\mathtt{x}$ with a number of predecessors strictly between $\sharp \mathtt{z}$ and $\sharp \mathtt{z}'$, there is an extremity with a number of predecessors equal to either $\sharp \mathtt{x} + 1$ or $\sharp \mathtt{x} - 1$.
$$\forall \mathtt{x}\ [\mathtt{extr}(\mathtt{x}) \wedge \sharp \mathtt{x} > \sharp \mathtt{z} \wedge \sharp \mathtt{x} < \sharp \mathtt{z}'] \Rightarrow (\exists \mathtt{y}\ \sharp \mathtt{y} = 1 + \sharp \mathtt{x} \vee \exists \mathtt{y}\ \sharp \mathtt{y} + 1 = \sharp \mathtt{x})$$
5. Constraint on two extremities with two consecutive numbers of predecessors:
$$\begin{gathered}\forall \mathtt{x}.\forall \mathtt{y}\ [\mathtt{extr}(\mathtt{x}) \wedge \mathtt{extr}(\mathtt{y}) \wedge (\sharp \mathtt{x} > \sharp \mathtt{z}) \wedge (\sharp \mathtt{x} < \sharp \mathtt{z}') \wedge (\sharp \mathtt{y} > \sharp \mathtt{z}) \wedge \\ (\sharp \mathtt{y} < \sharp \mathtt{z}') \wedge (\sharp \mathtt{y} + 1 = \sharp \mathtt{x})] \Rightarrow \\ [(\neg \exists \mathtt{y}'\ \sharp \mathtt{y}' = 1 + \sharp \mathtt{x}) \wedge (\exists \mathtt{y}'\ \sharp \mathtt{y}' = 2 + \sharp \mathtt{x}) \wedge \\ (\neg \exists \mathtt{y}'\ \sharp \mathtt{y}' + 1 = \sharp \mathtt{y}) \wedge (\exists \mathtt{y}'.\sharp \mathtt{y}' + 2 = \sharp \mathtt{y})]\end{gathered}$$
6. There are no two distinct extremities with an equal number of predecessors.
$$\forall \mathtt{x}[\mathtt{extr}(\mathtt{x}) \wedge \sharp \mathtt{x} \geq \sharp \mathtt{z} \wedge \sharp \mathtt{x} \leq \sharp \mathtt{z}'] \Rightarrow \neg \exists \mathtt{y}\ (\mathtt{extr}(\mathtt{y}) \wedge \sharp \mathtt{x} = \sharp \mathtt{y} \wedge \mathtt{x} \neq \mathtt{y})$$

It is then easy to check that the above conjunction satisfies the statement.

By induction on k ranging from 1 to $(\sharp \mathtt{z}' - \sharp \mathtt{z} - 2)/3$, one can show that there is no extremity l in (s, h_E) such that $\sharp l = \sharp \mathtt{z} + 3k - 2$, and there are extremities l and l' such that $\sharp l = \sharp \mathtt{z} + 3k - 1$ and $\sharp l = \sharp \mathtt{z} + 3k$. This concludes the proof. $\square$

Consequently, if $(s, h_E) \models \mathtt{PseudoEnv}(\mathtt{z}, \mathtt{z}')$, then h_E has a clean spectrum:

$$\overset{\sharp\mathtt{z}}{\bullet} \circ \bullet \bullet \circ \bullet \bullet \cdots \circ \bullet \bullet \circ \bullet \bullet \circ \bullet \bullet \cdots \circ \bullet \bullet \circ \overset{\sharp\mathtt{z}'}{\bullet}$$

In that case, (s, h_E) is called a *pseudo-environment* between $\mathtt{z}$ and $\mathtt{z}'$.

An *environment between* $\mathtt{z}$ *and* $\mathtt{z}'$ is a memory state (s, h_E) such that

(P1) $(s, h_E) \models \mathtt{PseudoEnv}(\mathtt{z}, \mathtt{z}')$.

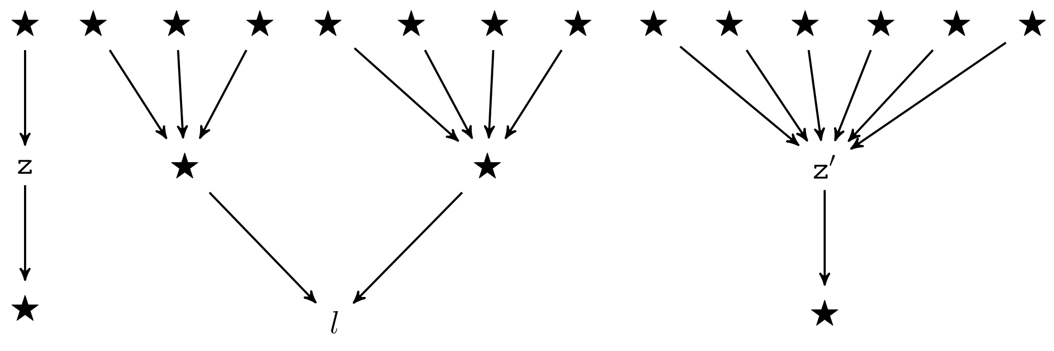
(P2) If $l \in \mathtt{dom}(h_E)$, then either l or $h_E(l)$ is an extremity in h_E.

(P3) For every extremity l in h_E, $l \in \mathtt{dom}(h_E)$ and $h_E(l) \notin \mathtt{dom}(h_E)$.

(P4) For every extremity l in h_E, $\sharp\mathtt{z} \leq \sharp l \leq \sharp\mathtt{z}'$.

Roughly speaking, (s, h_E) is a finite set of *markers* with the above-mentioned spectrum. Figure 7 presents a simple environment with $\sharp\mathtt{z} = 1$ and $\sharp\mathtt{z}' = 6$, which allows to encode a single pair ((l, l) in the present figure). Note that in full generality, the number of pairs that can be encoded by an environment between $\mathtt{z}$ and $\mathtt{z}'$ is equal to $\frac{\sharp\mathtt{z}' - \sharp\mathtt{z} - 2}{3}$.

Figure 7 A simple environment encoding the pair (l, l)



Lemma 6.2 *There exists a formula* $\mathtt{Env}(\mathtt{z}, \mathtt{z}') \in \mathtt{SL}(\multimap)$ *such that for every memory state* (s, h)*, we have* $(s, h) \models \mathtt{Env}(\mathtt{z}, \mathtt{z}')$ *iff* (s, h) *is an environment between* $\mathtt{z}$ *and* $\mathtt{z}'$.

Proof Let us consider the conjunction $\mathtt{Env}(\mathtt{z}, \mathtt{z}')$ of the formulae below.

(F1) $\mathtt{PseudoEnv}(\mathtt{z}, \mathtt{z}')$.

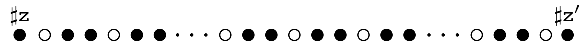
(F2) $\forall \mathtt{x}\ (\mathtt{alloc}\ (\mathtt{x}) \Rightarrow (\mathtt{extr}(\mathtt{x}) \vee \exists \mathtt{y}\ \mathtt{x} \hookrightarrow \mathtt{y} \wedge \mathtt{extr}(\mathtt{y})))$.

(F3) $\forall \mathtt{x}\ \mathtt{extr}(\mathtt{x}) \Rightarrow (\mathtt{alloc}\ (\mathtt{x}) \wedge \exists \mathtt{y}\ \mathtt{x} \hookrightarrow \mathtt{y} \wedge \neg \mathtt{alloc}\ (\mathtt{y}))$.

(F4) $\forall \mathtt{x}\ \mathtt{extr}(\mathtt{x}) \Rightarrow (\sharp \mathtt{z} \leq \sharp \mathtt{x} \wedge \sharp \mathtt{x} \leq \sharp \mathtt{z}')$.

Formula (Fi) captures the condition (Pi). □

Consequently, if $(s, h_E) \models \mathtt{Env}(\mathtt{z}, \mathtt{z}')$, then h_E is equal to a set of markers of the clean spectrum



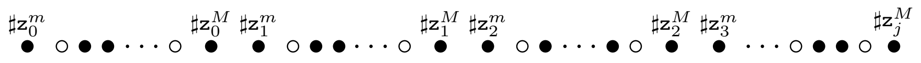
A *j-marked environment* is a memory state (s, h) such that

(PM0) (s, h) is an environment between $\mathtt{z}_0^m$ and $\mathtt{z}_j^M$.

(PM1) for every variable $\mathtt{x}$ in $\{\mathtt{z}_1^m, \ldots, \mathtt{z}_j^m\} \cup \{\mathtt{z}_0^M, \ldots, \mathtt{z}_{j-1}^M\}$, $s(\mathtt{x})$ is an extremity in (s, h) and $\sharp \mathtt{z}_0^m < \sharp \mathtt{x} < \sharp \mathtt{z}_j^M$.

(PM2) For $j \geq i > 0$, $\sharp \mathtt{z}_{i-1}^M + 1 = \sharp \mathtt{z}_i^m$.

Consequently, when (s, h) is a j-marked environment, the spectrum of h_E contains the following values:



Moreover, if (s, h') is another j-marked environment with identical store, then h and h' have the same spectrum.

Definition 6.3 below specifies how a heap can be divided into a base part and an environment part with constraints on the values $\sharp \mathtt{z}_0^m$,$\sharp \mathtt{z}_0^M$, $\ldots$, $\sharp \mathtt{z}_j^m$,$\sharp \mathtt{z}_j^M$. These values are helpful to determine the range of marker degrees that should be considered to encode the interpretation of second-order variables.

Definition 6.3 *A memory state (s, h) is j-*well-formed *for some $j \geq 0$ iff there are heaps h_B, h_E with $h = h_B * h_E$ satisfying the properties below:*

(WF1) *(s, h_E) is a j-marked environment.*

(WF2) *There is no location l such that $\sharp l$ in (s, h_B) is strictly greater than $\sharp \mathtt{z}_0^m - 2$ in (s, h).*

(WF3) $\mathtt{dom}(h_E) \cap \mathtt{ran}(h_B) = \emptyset$.

(s, h_B) is called the base part *and (s, h_E) the* environment part.

Condition (WF3) guarantees that when (s,h) is j-well-formed, for every extremity l in h_E, $\sharp l$ in h_E is equal to $\sharp l$ in h. Consequently, any extremity in h with more than $\sharp \mathtt{z}_0^m$ predecessors has all predecessors in $\mathtt{dom}(h_E)$. Moreover, $(s,h) \models \mathtt{PseudoEnv}(\mathtt{z}_0^m, \mathtt{z}_j^M)$, that is (s,h) is a pseudo-environment between $\mathtt{z}_0^m$ and $\mathtt{z}_j^M$.

We establish below a few lemmas that are helpful in the sequel.

Lemma 6.4 *Let h_E be the environment part of some j-well-formed model. For every location $l \in \mathtt{ran}(h_E)$, either l is an extremity in h_E or there is l' such that $h_E(l') = l$ and l' is an extremity.*

Note that the above property holds true for any environment but we shall use it for j-well-formed models only.

Proof If $l \in \mathtt{ran}(h_E)$, then there is a location l' such that $h_E(l') = l$. By (P2) on (s,h_E), either $h_E(l')$ is an extremity or l' is an extremity. $\square$

Lemma 6.5 below states unicity of decomposition when a memory state is j-well-formed.

Lemma 6.5 (Unicity) *Whenever (s,h) is j-well-formed with base part h_B and environment part h_E, there is no $(h'_B, h'_E) \neq (h_B, h_E)$ such that (s,h) is j-well-formed with base part h'_B and environment part h'_E.*

Proof Let $k_0 = (\sharp \mathtt{z}_j^M - \sharp \mathtt{z}_0^m - 2)/3$ and $S = \{k : k \not\equiv 1 \pmod 3 \text{ and } 0 \leq k \leq 3 \times k_0 + 2\}$ be the spectrum of h_E and h'_E. Indeed, (s,h_E) and (s,h'_E) are both j-marked environments and there are precisely card(S) extremities l in (s,h) such that $\sharp \mathtt{z}_0^m \leq \sharp l \leq \sharp \mathtt{z}_j^M$. For each $k \in S$, we write l_k to denote the *unique* extremity such that $\sharp l_k = \sharp \mathtt{z}_0^m + k$. Notice that each location l_k has no predecessor in h_B by Definition 6.3(WF3), $l_0 = s(\mathtt{z}_0^m)$ and $l_{3k_0+2} = s(\mathtt{z}_j^M)$.

The set $\mathtt{dom}(h_E)$ contains at least the following locations: for every $k \in S$, the location l_k and the $\sharp \mathtt{z}_0^m + k$ predecessors of l_k in h. Let X be the set of the above locations. Assume there is some $l \in (\mathtt{dom}(h_E) \setminus X)$. By (P2), either l or $h_E(l)$ is an extremity in h_E (let us call it l'). Since each predecessor of some location in X is also in X and $l \notin X$, l is not a predecessor of an element in X. Consequently, l' is an extremity that does not belong to $\{l_k : k \in S\}$ (let us call this set Y). Since $\sharp \mathtt{z}_0^m \leq \sharp l' \leq \sharp \mathtt{z}_j^M$, either l' has as many predecessors as an element in Y or $\sharp l' \equiv \sharp \mathtt{z}_0^m + 1 \pmod 3$. This entails that (s,h_E) does not satisfy $\mathtt{PseudoEnv}(\mathtt{z}_0^m, \mathtt{z}_j^M)$ which leads to a contradiction. Consequently, $\mathtt{dom}(h_E) = X$, $h_E = h_{|X}$ (restriction of h to X) and $h_B = h_{|(\mathtt{dom}(h) \setminus X)}$. $\square$

In the sequel, when (s, h) is j-well-formed, by default h_E denotes the environment part and h_B the base part.

We state below a crucial result, basically stating that adding an environment heap to a j-well-formed memory state leads to a $(j+1)$-well-formed memory state. This is central to interpret a new second-order variable (extending the environment part) and this can be performed thanks to $\twoheadrightarrow$ (details will follow).

Lemma 6.6 (Composition) *Let (s, h) be a j-well-formed memory state and (s', h'_E) be a memory state such that*

1. *h'_E is disjoint from h and s' differs from s at most for the variables $\mathtt{z}^m_{j+1}$ and $\mathtt{z}^M_{j+1}$.*
2. *$s'(\mathtt{z}^m_{j+1})$ and $s'(\mathtt{z}^M_{j+1})$ do not belong to $\mathtt{dom}(h) \cup \mathtt{ran}(h)$.*
3. *(s', h'_E) is an environment between $\mathtt{z}^m_{j+1}$ and $\mathtt{z}^M_{j+1}$.*
4. *$(s', h * h'_E) \models \sharp\mathtt{z}^M_j + 1 = \sharp\mathtt{z}^m_{j+1}$.*
5. *$\mathtt{dom}(h'_E) \cap \mathtt{ran}(h) = \emptyset$.*

Then, $(s', h * h'_E)$ is $(j+1)$-well-formed with the base part h_B and the environment part $h_E * h'_E$.

The proof of Lemma 6.6 is tedious and requires some care. We provide the details below.

Proof The proof mainly rests on establishing the property below.

(♠) Any extremity in h_E or in h'_E is an extremity in $h * h'_E$ with exactly the same number of predecessors.

Consequently, this implies that in the model $(s', h * h'_E)$ we have the following relationships:

(✠) $\sharp\mathtt{z}^m_0 < \sharp\mathtt{z}^M_j = \sharp\mathtt{z}^m_{j+1} - 1 < \sharp\mathtt{z}^M_{j+1} - 1$, $\sharp\mathtt{z}^m_0 + 2 \equiv \sharp\mathtt{z}^M_{j+1} \pmod 3$ and $\sharp\mathtt{z}^m_{j+1} \equiv \sharp\mathtt{z}^m_0 \pmod 3$.

Assuming (♠) and (✠), let us check the conditions from Definition 6.3 for ensuring that $(s', h * h'_E)$ is $(j+1)$-well-formed with base part h_B. After doing that, we shall establish that (♠) holds true.

First, we show that $(s', h_E * h'_E)$ is a $(j+1)$-marked environment.

(P1) Let us prove that $(s', h_E * h'_E) \models \mathtt{PseudoEnv}(\mathtt{z}^m_0, \mathtt{z}^M_{j+1})$. Below, the numbers of predecessors are relative to $(s', h_E * h'_E)$. Let $i \in \{\sharp\mathtt{z}^m_0, \ldots, \sharp\mathtt{z}^M_{j+1}\}$.

  - Assume $i \equiv \sharp\mathtt{z}_0^m + 1 \ (\mathsf{mod}\ 3)$. *Ad absurdum*, suppose that there is a location l such that l is an extremity and $\sharp l = i$. Then l is an extremity with i predecessors either in h_E or in h'_E, which leads to a contradiction since (s', h'_E) is an environment between $\mathtt{z}_{j+1}^m$ and $\mathtt{z}_{j+1}^M$ and (s, h_E) is an environment between $\mathtt{z}_0^m$ and $\mathtt{z}_j^M$.
  - Assume $i \not\equiv \sharp\mathtt{z}_0^m + 1 \ (\mathsf{mod}\ 3)$. If $i \in \{\sharp\mathtt{z}_0^m, \ldots, \sharp\mathtt{z}_j^M\}$, then by ($\spadesuit$) there is a unique extremity l_k such that $\sharp l_k = i$. Otherwise ($i \in \{\sharp\mathtt{z}_{j+1}^m, \ldots, \sharp\mathtt{z}_{j+1}^M\}$), by ($\spadesuit$), there is a unique extremity l_k^{new} such that $\sharp l_k^{new} = i$.

(P2) Suppose that $l \in \mathtt{dom}(h_E * h'_E)$. Two cases are distinguished below.

  - $l \in \mathtt{dom}(h_E)$.
    We distinguish again two subcases since h is j-well-formed.
    * In the case l is an extremity in h_E, the location l is an extremity in $h * h'_E$ by ($\spadesuit$). Consequently, l is an extremity in $h_E * h'$.
    * In the case $h(l)$ is an extremity in h_E, the proof is analogous.
  - $l \in \mathtt{dom}(h'_E)$.
    The proof is analogous.

(P3) Let l be an extremity in h_E. Let us show that $h(l) \notin \mathtt{dom}(h_E * h'_E)$. Since (s, h) is j-well-formed, $h(l) \notin \mathtt{dom}(h_E)$. *Ad absurdum*, suppose that $h(l) \in \mathtt{dom}(h'_E)$. Then, either $h(l)$ is an extremity in h'_E or $h(l)$ is a predecessor of an extremity l' in h'_E. In the first case, it leads to a contradiction since the extremities of h'_E are not in $\mathtt{ran}(h_E)$, by hypothesis (5). In the second case, l' is not an extremity in $h * h'_E$ which is in contradiction with ($\spadesuit$). Consequently, $h(l) \notin \mathtt{dom}(h_E * h'_E)$.

Let l be an extremity in h'_E. Since (s', h'_E) is an environment between $\mathtt{z}_{j+1}^m$ and $\mathtt{z}_{j+1}^M$, we know that $h'_E(l) \notin \mathtt{dom}(h'_E)$. It remains to check that $h'_E(l) \notin \mathtt{dom}(h_E)$. *Ad absurdum*, suppose that $h'_E(l) \in \mathtt{dom}(h_E)$. Then there is $l' \in \{h'_E(l), h(h'_E(l))\}$ such that l' is an extremity in h_E. By ($\spadesuit$), l' is an extremity in $h_E * h'_E$. This leads to a contradiction since l has predecessors in $h_E * h'_E$.

(P4) Let l be an extremity in h_E. We have $\sharp\mathtt{z}_0^m \leq \sharp l \leq \sharp\mathtt{z}_j^M < \sharp\mathtt{z}_{j+1}^M$ since (s, h) is j-well-formed and ($\spadesuit$). Let l be an extremity in h'_E. The values $\sharp l$, $\sharp\mathtt{z}_{j+1}^m$ and $\sharp\mathtt{z}_{j+1}^M$ do not change from h'_E to $h * h'_E$. Since (s', h'_E) is an environment between $\mathtt{z}^m_{j+1}$ and $\mathtt{z}^M_{j+1}$, $\sharp\mathtt{z}^m_{j+1} \leq \sharp l \leq \sharp\mathtt{z}^M_{j+1}$. So in $(s', h * h'_E)$, we have $\sharp\mathtt{z}^m_0 < \sharp\mathtt{z}^m_{j+1} \leq \sharp l \leq \sharp\mathtt{z}^M_{j+1}$.

(PM1) By (♠), for each variable $\mathtt{x}$ in $\{\mathtt{z}^m_0, \ldots, \mathtt{z}^m_{j+1}\} \cup \{\mathtt{z}^M_0, \ldots, \mathtt{z}^M_{j+1}\}$, the value $\sharp\mathtt{x}$ remains unchanged from h or h'_E to $h * h'_E$. Considering that h is j-well-formed, h'_E is an environment between $\mathtt{z}^m_{j+1}$ and $\mathtt{z}^M_{j+1}$ and $(s', h * h'_E) \models \sharp\mathtt{z}^M_j + 1 = \sharp\mathtt{z}^m_{j+1}$, we conclude that for every $\mathtt{x} \in \{\mathtt{z}^m_1, \ldots, \mathtt{z}^m_{j+1}\} \cup \{\mathtt{z}^M_0, \ldots, \mathtt{z}^M_j\}$, $s(\mathtt{x})$ is an extremity and $\sharp\mathtt{z}^m_0 < \sharp\mathtt{x} < \sharp\mathtt{z}^M_{j+1}$.

(PM2) Let $0 < i \leq j+1$. If $i \leq j$, then since (s, h) is j-well-formed we obtain $(s, h) \models \sharp\mathtt{z}^M_i + 1 = \sharp\mathtt{z}^m_{i+1}$. By (♠), $(s', h * h'_E) \models \sharp\mathtt{z}^M_i + 1 = \sharp\mathtt{z}^m_{i+1}$ (s' and s agree for these variables). If $i = j+1$, then hypothesis (4) precisely states that $(s', h * h'_E) \models \sharp\mathtt{z}^M_j + 1 = \sharp\mathtt{z}^m_{j+1}$.

It remains to verify the conditions (WF2) and (WF3).

(WF2) Since h and $h * h'_E$ have the same base part and (s, h) is j-well-formed, we get that there is no location l such that $\sharp l$ in (s, h_B) is strictly greater than $\sharp\mathtt{z}^m_0 - 2$ in $(s, h * h_E)$ (equal to $\sharp\mathtt{z}^m_0 - 2$ in (s, h) by (♠)).

(WF3) Since (s, h) is j-well-formed, we have $\mathtt{dom}(h_E) \cap \mathtt{ran}(h_B) = \emptyset$. By hypothesis (5), $\mathtt{dom}(h'_E) \cap \mathtt{ran}(h) = \emptyset$. Consequently, $\mathtt{dom}(h_E * h'_E) \cap \mathtt{ran}(h_B) = \emptyset$.

Now, let us prove that (♠) holds true. First, we prove the case when an extremity is a location of the form $s'(\mathtt{z}^{\blacksquare}_k)$ with $k \in \{0, \ldots, j+1\}$ and $\blacksquare \in \{m, M\}$. By hypothesis (2), $s'(\mathtt{z}^m_{j+1})$ and $s'(\mathtt{z}^M_{j+1})$ do not belong to $\mathtt{ran}(h)$. So the values $\sharp\mathtt{z}^m_{j+1}$ and $\sharp\mathtt{z}^M_{j+1}$ remain unchanged from (s', h'_E) to $(s', h * h'_E)$. Now let $k \in \{0, \ldots, j\}$ and $\blacksquare \in \{m, M\}$. Assume that $\sharp\mathtt{z}^{\blacksquare}_k$ has changed from (s', h) to $(s', h * h'_E)$. Consequently, $s'(\mathtt{z}^{\blacksquare}_k) \in \mathtt{ran}(h'_E)$. By Lemma 6.4, there are two possibilities.

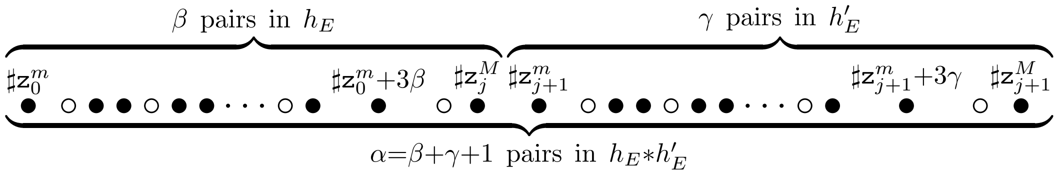
1. $s'(\mathtt{z}^{\blacksquare}_k)$ is an extremity in (s', h'_E).
   As (s', h'_E) is an environment between $\mathtt{z}^m_{j+1}$ and $\mathtt{z}^M_{j+1}$, every extremity belongs to $\mathtt{dom}(h'_E)$, whence $s'(\mathtt{z}^{\blacksquare}_k) \in \mathtt{dom}(h'_E)$. This leads to a contradiction since h and h'_E are disjoint: $s'(\mathtt{z}^{\blacksquare}_k) \in \mathtt{dom}(h_E)$ since (s, h) is j-well-formed.
2. There is a location l such that $h'_E(l) = s'(\mathtt{z}^{\blacksquare}_k)$ (also equal to $s(\mathtt{z}^{\blacksquare}_k)$) and l is an extremity. So $s'(\mathtt{z}^{\blacksquare}_k)$ is not an extremity in $h * h'_E$, which also leads to a contradiction.

Consequently, for all $k \in \{0, \dots, j\}$ and $\blacksquare \in \{m, M\}$, $\sharp\mathsf{z}_k^{\blacksquare}$ is unchanged from h to $h * h'_E$. Based on these preservations and since (s', h'_E) is an environment between z_{j+1}^m and z_{j+1}^M, (s', h) is j-well-formed and $(s', h * h'_E) \models \sharp\mathsf{z}_j^M + 1 = \sharp\mathsf{z}_{j+1}^m$, we can conclude ($\maltese$).

Before treating the proof for the other types of extremities, let us provide a few basic definitions and facts. We define the natural numbers α, β and γ as follows:

$$3\alpha = (\sharp\mathsf{z}_{j+1}^M - \sharp\mathsf{z}_0^m) - 2 \quad 3\beta = (\sharp\mathsf{z}_j^M - \sharp\mathsf{z}_0^m) - 2 \quad 3\gamma = (\sharp\mathsf{z}_{j+1}^M - \sharp\mathsf{z}_{j+1}^m) - 2$$

Notice that $\gamma = \alpha - \beta - 1$. These values are simply related to the spectrum below where the first value is $\sharp\mathsf{z}_0^m$ and the last one is $\sharp\mathsf{z}_{j+1}^M$.


β pairs in h_E
γ pairs in h'_E
$\sharp\mathsf{z}_0^m$
$\sharp\mathsf{z}_0^m{+}3\beta$
$\sharp\mathsf{z}_j^M$
$\sharp\mathsf{z}_{j+1}^m$
$\sharp\mathsf{z}_{j+1}^m{+}3\gamma$
$\sharp\mathsf{z}_{j+1}^M$
$\alpha{=}\beta{+}\gamma{+}1$ pairs in $h_E{*}h'_E$

For $N \geq 1$, let $F_N \triangleq \{k : k \not\equiv 1 \pmod 3 \text{ and } 0 \leq k \leq 3N + 2\}$ and we pose $E_\alpha \triangleq F_\alpha$, $E_\beta \triangleq F_\beta$ and $E_\gamma \triangleq \{n + (3\beta + 3) : n \in F_\gamma\}$. Since (s, h_E) is an environment between z_0^m and z_j^M (remember (s, h) is j-well-formed) and (s', h'_E) is an environment between z_{j+1}^m and z_{j+1}^M, we get that $(s', h * h'_E) \models \mathtt{PseudoEnv}(\mathsf{z}_0^m, \mathsf{z}_{j+1}^M)$. So, for every $k \in E_\alpha$, there is a location l_k^* verifying the properties below in $(s', h * h'_E)$:

- $\sharp l_k^* = \sharp\mathsf{z}_0^m + k$,
- l_k^* is an extremity,
- there is no location l such that $\sharp l = \sharp l_k^*$, $l \neq l_k^*$ and l is an extremity.

Notice that $\sharp l_{3\times\alpha}^* = \sharp\mathsf{z}_{j+1}^M - 2$ in $(s', h * h'_E)$, $l_{3\times\beta+2}^* = s'(\mathsf{z}_j^M)$ and $l_{3\times\beta+3}^* = s'(\mathsf{z}_{j+1}^m)$.

Similarly, as $(s', h) \models \mathtt{PseudoEnv}(\mathsf{z}_0^m, \mathsf{z}_j^M)$, for every $k \in E_\beta$, there is a location l_k verifying the properties below in (s', h):

- $\sharp l_k = \sharp\mathsf{z}_0^m + k$,
- l_k is an extremity,
- there is no location l such that $\sharp l = \sharp l_k$, $l \neq l_k$ and l is an extremity.

Observe that all the extremities in h_E are either of the form l_k, or $s'(\mathtt{z}_0^m)$ or $s'(\mathtt{z}_j^M)$. Moreover, $\sharp l_{3\beta} = \sharp \mathtt{z}_j^M - 2$ in (s', h).

Finally, as $(s', h'_E) \models \mathtt{PseudoEnv}(\mathtt{z}_{j+1}^m, \mathtt{z}_{j+1}^M)$, for every $k \in E_\gamma$, there is a location l_k^{new} verifying the properties below in (s', h'_E):

- $\sharp l_k^{new} = (\sharp \mathtt{z}_{j+1}^m - (3\beta + 3)) + k$,
- l_k^{new} is an extremity,
- there is no location l such that $\sharp l = \sharp l_k^{new}$, $l \neq l_k^{new}$ and l is an extremity.

Observe that all the extremities of h'_E are either of the form l_k^{new}, or $s'(\mathtt{z}_{j+1}^m)$ or $s'(\mathtt{z}_{j+1}^M)$. We can establish additionnal arithmetical properties: $\sharp l_{3\times\gamma}^{new} = \sharp \mathtt{z}_{j+1}^M - 2$ in (s', h'_E) and $\sharp l_k^{new}$ in (s', h'_E) is equal to $\sharp \mathtt{z}_0^m + k$ in $(s', h * h'_E)$.

We are going to prove that for all $k \in E_\beta$, $l_k = l_k^*$, and for all $k \in E_\gamma$, $l_k^{new} = l_k^*$. This will terminate the proof of $(\spadesuit)$ since the only extremities in h_E are $\{l_k : k \in E_\beta\} \cup \{s'(\mathtt{z}_0^m), s'(\mathtt{z}_j^M)\}$ and the only extremities in h'_E are $\{l_k^{new} : k \in E_\gamma\} \cup \{s'(\mathtt{z}_{j+1}^m), s'(\mathtt{z}_{j+1}^M)\}$. The proof is *ad absurdum* and we distinguish two cases (each of them will therefore lead to a contradiction):

(I) There is $k \in E_\beta$ such that $l_k \neq l_k^*$.

(II) There is $k \in E_\gamma$ such that $l_k^{new} \neq l_k^*$.

(I) Let us first establish that $l_k^* \in \mathtt{ran}(h'_E)$ (proof *ad absurdum*). Suppose that $l_k^* \notin \mathtt{ran}(h'_E)$. So, $\sharp l_k^*$ remains unchanged from (s', h) to $(s', h * h'_E)$. As in $(s', h * h'_E)$, we have $\sharp \mathtt{z}_0^m < \sharp l_k^* < \sharp \mathtt{z}_j^M$, and $\mathtt{z}_0^m$ and $\mathtt{z}_j^M$ remain unchanged from (s', h) to $(s', h * h'_E)$, we can infer that $\sharp \mathtt{z}_0^m < \sharp l_k^* < \sharp \mathtt{z}_j^M$ in (s', h). Additionnally, as in $(s', h * h'_E)$, we have $\sharp l_k^* = \sharp \mathtt{z}_0^m + k$, this is also true in (s', h). Finally, as l_k^* is an extremity in $(s', h * h'_E)$, it is also an extremity in (s', h). Consequently, $l_k^* = l_k$, which leads to a contradiction. We have established that $l_k^* \in \mathtt{ran}(h'_E)$. By Lemma 6.4, there are two possibilities:

- l_k^* is an extremity in h'_E.
  Consequently, in h'_E, we have $\sharp l_k^* > \sharp \mathtt{z}_{j+1}^m$. As $\sharp \mathtt{z}_{j+1}^m$ remains unchanged from (s', h'_E) to $(s', h * h'_E)$, in $h * h'_E$ we obtain $\sharp l_k^* > \sharp \mathtt{z}_{j+1}^m = \sharp \mathtt{z}_j^M + 1$, which leads to a contradiction since $\sharp l_k^* = \sharp l_k = \sharp \mathtt{z}_0^m + k$ and $\sharp l_k < \sharp \mathtt{z}_j^M$.
- There is a location l_0 such that l_0 is an extremity in h'_E and $h'_E(l_0) = l_k^*$. So l_k^* is not an extremity in h'_E, and it cannot either be an extremity in $h * h'_E$, which leads to a contradiction.

(II) Let k be the smallest element of E_γ such that $l_k^{new} \neq l_k^*$. In $(s', h*h'_E)$, we know that $\sharp l_k^* > \sharp \mathtt{z}_{j+1}^m > \sharp \mathtt{z}_j^M$. Moreover, as l_k^* is an extremity in $(s', h*h'_E)$, either l_k^* is an extremity in (s', h) too or l_k^* has no predecessor in (s', h). Since no extremity of (s', h) has more than $\sharp \mathtt{z}_j^M$ predecessors (in both h and $h*h'_E$), the location l_k^* cannot have all of its predecessors in $\mathtt{dom}(h)$. Let l_0 be one of the predecessors of l_k^* that belongs to $\mathtt{dom}(h'_E)$, i.e. $h'_E(l_0) = l_k^*$.

Recall that (s', h'_E) is an environment between $\mathtt{z}_{j+1}^m$ and $\mathtt{z}_{j+1}^M$. Since $l_0 \in \mathtt{dom}(h'_E)$, there is $l \in \{l_0, h'_E(l_0)\}$ such that in h'_E:

(a) l is an extremity,

(b) $l \in \mathtt{dom}(h'_E)$,

(c) $\sharp \mathtt{z}_{j+1}^m \leq \sharp l \leq \sharp \mathtt{z}_{j+1}^M$,

(d) no other extremity has exactly $\sharp l$ predecessors.

Indeed, (a) comes from (P2), the conditions (b) and (d) come from $(s', h'_E) \models \mathtt{PseudoEnv}(\mathtt{z}_{j+1}^m, \mathtt{z}_{j+1}^M)$, (c) from satisfaction of (P4).

In the case $l = l_0$, l_k^* is not an extremity in h'_E and hence l_k^* is not an extremity in $h * h'_E$. This leads to a contradiction. Consequently, we have $l = h'_E(l_0) = l_k^*$. Let us conclude the proof.

In h'_E, the location l_k^* is an extremity. As (s', h'_E) is an environment between $\mathtt{z}_{j+1}^m$ and $\mathtt{z}_{j+1}^M$, we have $l_k^* \in \mathtt{dom}(h'_E)$ and $\sharp \mathtt{z}_{j+1}^m \leq \sharp l_k^* \leq \sharp \mathtt{z}_{j+1}^M$ in (s', h'_E). Since $s(\mathtt{z}_{j+1}^m) \neq l_k^*$ and $s(\mathtt{z}_{j+1}^M) \neq l_k^*$, we obtain $\sharp \mathtt{z}_{j+1}^m < \sharp l_k^* < \sharp \mathtt{z}_{j+1}^M$ in h'.

So there is $p \in E_\gamma$ such that $l_k^* = l_p^{new}$. We have that the value $\sharp l_p^{new}$ changes from h'_E to $h * h'_E$, and therefore $l_p^{new} \neq l_p^*$. Since $\sharp l_p^{new}$ can only increase from h'_E to $h * h'_E$, we can conclude that $\sharp l_p^{new}$ in (s', h'_E) is strictly smaller than $\sharp l_p^{new} = \sharp l_k^*$ in $(s', h * h'_E)$. By definition of the locations l_k^* and l_p^{new}, we obtain $p < k$, which leads to a contradiction by minimality of k. $\square$

6.2. The Translation

In this section, we provide the translation from $\mathtt{DSO}$ into $\mathtt{SL}(-\!\!*)$. First, we introduce additional formulae that will be useful in the translation process. It is worth observing that in order to translate first-order quantification, we should guarantee that first-order variables $\mathtt{x}$ are not interpreted as locations from the domain of the environment part. Typically, the number of predecessors of $s(\mathtt{x})$ and $h(s(\mathtt{x}))$ (if it exists) should be less than $\sharp \mathtt{z}_0^M$ and none of these locations is an extremity. The formula $\mathtt{notonenv}(\cdot)$ is introduced for this purpose:

$$\mathtt{notonenv}(\mathtt{x}) \triangleq \neg(\exists \mathtt{y}\ (\mathtt{y} = \mathtt{x} \vee \mathtt{x} \hookrightarrow \mathtt{y}) \wedge (\sharp\mathtt{y} \geq \sharp\mathtt{z}_0^m) \wedge \mathtt{extr}(\mathtt{y})).$$

Lemma 6.7 *Let (s,h) be a j-well-formed model. Then $(s,h) \models \mathtt{notonenv}(\mathtt{x})$ iff $s(\mathtt{x}) \notin \mathtt{dom}(h_E)$.*

Proof As (s,h) is j-well-formed, by Definition 6.3, for any location l, we have $l \in \mathtt{dom}(h_E)$ iff there is a location $l' \in \{l, h(l)\}$ such that in the heap h_E, we have $\sharp l' \geq \sharp\mathtt{z}_0^m$ and l' is an extremity. Moreover, by Definition 6.3, we get in the heap h that $\sharp l' \geq \sharp\mathtt{z}_0^m$ and l' is an extremity. Assume that $s(\mathtt{x}) \in \mathtt{dom}(h_E)$, then thanks to the explanations just above, $(s,h) \not\models \mathtt{notonenv}(\mathtt{x})$. Now, *ad absurdum*, suppose that $s(\mathtt{x}) \notin \mathtt{dom}(h_E)$ and $(s,h) \not\models \mathtt{notonenv}(\mathtt{x})$. Then there is $l \in \{s(\mathtt{x}), h(s(\mathtt{x}))\}$ such that $\sharp l \geq \sharp\mathtt{z}_0^m$ and l is an extremity, by definition of $\mathtt{notonenv}$. Furthermore, by Definition 6.3(WF3), the location l is not an extremity in h_E, all of its precedessors are in h_B. Then by Definition 6.3, $\sharp l \geq \sharp\mathtt{z}_0^m - 2$, which leads to a contradiction. $\square$

The formula $\mathtt{relation}_{j,X}$ defined below is helpful to build environments.

Proposition 6.8 *Let $j \geq 0$ and X be a finite set of variables disjoint from $\{\mathtt{z}_0^m, \mathtt{z}_0^M, \ldots, \mathtt{z}_j^m, \mathtt{z}_j^M\}$. Then, there is a formula $\mathtt{relation}_{j,X}$ such that for every model (s,h), we have $(s,h) \models \mathtt{relation}_{j,X}$ iff (s,h) is an environment between $\mathtt{z}_j^m$ and $\mathtt{z}_j^M$ and for every $\mathtt{x} \in X$, $s(\mathtt{x}) \notin \mathtt{dom}(h)$.*

The formula $\mathtt{relation}_{j,X}$ is simply

$$\mathtt{Env}(\mathtt{z}_j^m, \mathtt{x}_j^M) \wedge \bigwedge_{\mathtt{y} \in X} \neg\mathtt{alloc}\ (\mathtt{y}).$$

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.

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

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

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

The formula $\mathtt{isol}(\mathtt{x})$ is an abbreviation for $\neg\exists\mathtt{y}\ (\mathtt{x} \hookrightarrow \mathtt{y}) \vee (\mathtt{y} \hookrightarrow \mathtt{x})$; this guarantees that $s(\mathtt{x}) \notin \mathtt{dom}(h) \cup \mathtt{ran}(h)$. It remains to define recursively the map $t_j(\cdot)$.

- $t_j(\mathtt{x} = \mathtt{y}) \;\triangleq\; \mathtt{x} = \mathtt{y}$,
- $t_j(\mathtt{x} \hookrightarrow \mathtt{y}) \;\triangleq\; \mathtt{x} \hookrightarrow \mathtt{y}$,
- for $i \leq j$, $t_j(\mathtt{P}_i(\mathtt{x}, \mathtt{y}))$ is defined by

$$\exists \mathtt{z}, \mathtt{z}' \; (\mathtt{z} \hookrightarrow \mathtt{x}) \;\wedge\; (\mathtt{z}' \hookrightarrow \mathtt{y}) \;\wedge\; (\sharp \mathtt{z} > \sharp \mathtt{z}_i^m) \;\wedge\; (\sharp \mathtt{z}' < \sharp \mathtt{z}_i^M) \;\wedge$$

$$(\sharp \mathtt{z}' = 1 + \sharp \mathtt{z}) \;\wedge\; \mathtt{extr}(\mathtt{z}) \;\wedge\; \mathtt{extr}(\mathtt{z}')$$

  So $(s(\mathtt{x}), s(\mathtt{y}))$ belongs to the interpretation of $\mathtt{P}_i$ when $s(\mathtt{x})$ and $s(\mathtt{y})$ are endpoints of markers with consecutive degrees between $\sharp \mathtt{z}_i^m$ and $\sharp \mathtt{z}_i^M$.
- t_j is homomorphic for Boolean connectives.
- $t_j(\exists \mathtt{x} \; \psi) \;\triangleq\; \exists \mathtt{x} \; \mathtt{notonenv}(\mathtt{x}) \wedge t_j(\psi)$.
- $t_j(\exists \mathtt{P}_{j+1}, \psi)$, is defined by

$$\exists \mathtt{z}_{j+1}^m, \mathtt{z}_{j+1}^M \; \mathtt{isol}(\mathtt{z}_{j+1}^m) \wedge \mathtt{isol}(\mathtt{z}_{j+1}^M) \wedge$$

$$(\mathtt{relation}_{j+1, \mathtt{FV}(\psi)} \mathrel{-\!\!\circledast} (\mathtt{PseudoEnv}(\mathtt{z}_0^m, \mathtt{z}_{j+1}^M) \wedge \sharp \mathtt{z}_j^M + 1 = \sharp \mathtt{z}_{j+1}^m \wedge t_{j+1}(\psi)))$$

  In order to translate $\exists \mathtt{P}_{j+1} \; \psi$, 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 $\mathrel{-\!\!\circledast}$) that guarantees that the new part of the heap encodes the interpretation of the variable $\mathtt{P}_{j+1}$ by using the above formula $\mathtt{relation}_{j+1, X}$.

Observe that $T(\phi)$ and ϕ have the same first-order free variables.

6.3. Correctness

Before stating the correctness of the translation $T(\cdot)$, we need to formally define how to extract an environment from a j-well-formed model (but now, that is easy).

Definition 6.9 *Let (s, h) be a j-well-formed model, and let h_E be the associated environment heap. The environment $\mathcal{E}$* extracted from *h is*

$$\mathcal{E}(\mathtt{P}_i) \stackrel{\mathit{def}}{=} \{(h_E(l), h_E(l')) : \; \sharp \mathtt{z}_i^m < \sharp l, \; \sharp l + 1 = \sharp l', \; \sharp l' < \sharp \mathtt{z}_i^M \text{ in } h_E\}$$

for all $i \in \{1, \ldots, j\}$.

Correctness of $T(\cdot)$ is based on Proposition 6.10 below. The proof shall use several results established earlier.

Proposition 6.10 *Let ϕ be a* DSO *formula using the extended Barendregt convention and ψ be a subformula of ϕ at quantifier depth j. Let (s,h) be a j-well-formed model, with base part (s,h_B) and environment part (s,h_E), such that for each $\mathtt{x} \in \mathtt{FV}(\psi)$, $s(\mathtt{x}) \notin \mathtt{dom}(h_E)$. Let $\mathcal{E}_j$ be the environment extracted from h_E. Then, $(s,h) \models t_j(\psi)$ iff $(s,h_B),\mathcal{E}_j \models \psi$.*

Proof 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 $(l,l') \in \mathcal{R}$ or $(l',l) \in \mathcal{R}$. Let ϕ be a DSO sentence satisfying the extended Barendregt convention. We want to show by induction on ψ that given:

- ψ is a subformula of ϕ of quantifier depth j,
- (s,h) is j-well-formed with base part h_B and environment part h_E such that for every variable $\mathtt{x} \in \mathtt{FV}(\psi)$, we have $s(\mathtt{x}) \notin \mathtt{dom}(h_E)$,
- $\mathcal{E}_j$ is the environment $\{\mathtt{P}_1 \mapsto \mathcal{R}_1, \ldots, \mathtt{P}_j \mapsto \mathcal{R}_j\}$ extracted from h_E,
- no location occurring in $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_j$ belongs to $\mathtt{dom}(h_E)$,

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

Base cases.
The base cases $\mathtt{x} = \mathtt{y}$ and $\mathtt{x} \hookrightarrow \mathtt{y}$ are by an easy verification since t_j restricted to them is the identity map. Let us consider the more interesting base case, i.e. when $\psi = \mathtt{P}_k(\mathtt{x},\mathtt{y})$ with $k \leq j$.
$(\rightarrow)$ Suppose that $(s,h) \models t_j(\mathtt{P}_k(\mathtt{x},\mathtt{y}))$. Then, in the heap h, the locations $s(\mathtt{x})$ and $s(\mathtt{y})$ have predecessors in h that are also extremities, let us call them respectively $l_\mathtt{x}$ and $l_\mathtt{y}$. In the heap h, we have $\sharp\mathtt{z}_k^m < \sharp l_\mathtt{x} = \sharp l_\mathtt{y} - 1 < \sharp\mathtt{z}_k^M - 1$. By Definition 6.3, both $l_\mathtt{x}$ and $l_\mathtt{y}$ have predecessors in $\mathtt{dom}(h_E)$ and all of their predecessors are also in $\mathtt{dom}(h_E)$. Since $\mathtt{z}_k^m$ and $\mathtt{z}_k^M$ have also all of their predecessors in $\mathtt{dom}(h_E)$, we have $\sharp\mathtt{z}_k^m < \sharp l_\mathtt{x}$, $\sharp l_\mathtt{x} + 1 = \sharp l_\mathtt{y}$ and $\sharp l_\mathtt{y} < \sharp\mathtt{z}_k^M$ in h_E. By Definition 6.9, 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, $(s,h_B),\mathcal{E}_j \models \mathtt{P}_k(\mathtt{x},\mathtt{y})$.
$(\leftarrow)$ Suppose that $(s,h_B),\mathcal{E}_j \models \mathtt{P}_k(\mathtt{x},\mathtt{y})$. By definition of $\models$ and $\mathcal{E}_j$, we have $(s(\mathtt{x}), s(\mathtt{y})) \in \mathcal{R}_k$. So $s(\mathtt{x})$ and $s(\mathtt{y})$ have respectively predecessors $l_\mathtt{x}$ and $l_\mathtt{y}$ in $\mathtt{dom}(h_E)$. In the heap h_E, $l_\mathtt{x}$ and $l_\mathtt{y}$ are extremities and $\sharp\mathtt{z}_k^m < \sharp l_\mathtt{x} = \sharp l_\mathtt{y} - 1 < \sharp\mathtt{z}_k^M - 1$. By Definition 6.3, the predecessors of any location among $s(\mathtt{z}_k^m)$,

$l_{\mathtt{x}}$, $l_{\mathtt{y}}$ and $s(\mathtt{z}_k^M)$ belong to $\mathtt{dom}(h_E)$. So the above inequalities and equality are also true in h. By Definition 6.3, the locations $s(\mathtt{z}_k^m)$, $l_{\mathtt{x}}$, $l_{\mathtt{y}}$ and $s(\mathtt{z}_k^M)$ are extremities in h. So $(s, h) \models t_j(\mathtt{P}_k(\mathtt{x}, \mathtt{y}))$.

Induction step. Our induction hypothesis is the following: for every subformula ψ' of size strictly less than the size of ψ, for $j \in \{0, \ldots, n\}$ (n is the quantifier depth of ϕ) and for any j-well-formed model (s, h) such that for every variable $\mathtt{x} \in \mathtt{FV}(\psi)$, we have $(s, h) \models t_j(\psi')$ iff $(s, h_B), \mathcal{E}_j \models \psi'$.
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 $l \notin \mathtt{dom}(h_E)$ with $s' = s[\mathtt{x} \mapsto l]$ (by Lemma 6.7),

(3) there is $l \in \mathtt{Loc}$ such that $(s', h_B), \mathcal{E}_j \models \psi'$ and $l \notin \mathtt{dom}(h_E)$ 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 $(s', h_B), \mathcal{E}_j \models \psi'$ with $s' = s[\mathtt{x} \mapsto l]$,

(5) $(s, h_B), \mathcal{E}_j \models \psi$ (by definition of $\models$).

Let us justify below why (4) implies (3). Suppose (4) and $l \in \mathtt{dom}(h_E)$. Since (s, h) is j-well-formed, $l \notin (\mathtt{dom}(h_B) \cup \mathtt{ran}(h_B))$. Since $\mathtt{Loc}$ is an infinite set, there is a location $l' \in (\mathtt{Loc} \setminus (\mathtt{dom}(h_B) \cup \mathtt{ran}(h_B) \cup \mathtt{dom}(h_E))$ such that l' does not occur in $(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_j)$. By Lemma 2.1, $(s[\mathtt{x} \mapsto l'], h_B), \mathcal{E}_j[l \leftarrow l'] \models \psi'$. 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 $\mathtt{dom}(h_E)$ and by (P3), $l \notin \mathtt{dom}(h_E)$, 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 $l' \notin \mathtt{dom}(h_E)$.

Case 2: $\psi = \exists \mathtt{P}_{j+1}\ \psi'$.
($\leftarrow$) Suppose that $(s, h_B), \mathcal{E}_j \models \exists \mathtt{P}_{j+1}\ \psi'$. By definition of the satisfaction relation $\models$, there is $\mathcal{R} \in \mathcal{P}_f(\mathtt{Loc}^2)$ such that $(s, h_B), \mathcal{E}_j[\mathtt{P}_{j+1} \mapsto \mathcal{R}] \models \psi'$. Since we aim at having locations in h_E that do not interfere with the store, we need to be more restrictive about $\mathcal{R}$.

Replacing $\mathcal{R}$ by some $\mathcal{R}'$. We build below a finite binary relation $\mathcal{R}'$ from $\mathcal{R}$ such that no location in $\mathtt{dom}(h_E)$ occurs in $\mathcal{R}'$ and $(s, h_B), \mathcal{E}_j[\mathtt{P}_{j+1} \mapsto \mathcal{R}'] \models \psi'$.

More precisely, $\mathcal{R}'$ will be obtained from $\mathcal{R}$ by replacing its image under a permutation of the set of locations that leaves the locations in s and h_B fixed. The relation $\mathcal{R}'$ is constructed by successively replacing the locations in $\mathtt{dom}(h_E)$ that occur also in $\mathcal{R}$. Suppose that for some $l \in \mathtt{dom}(h_E)$, l occurs also in $\mathcal{R}$. By the induction hypothesis, for every variable $\mathtt{x} \in \mathtt{FV}(\psi')$, $l \neq s(\mathtt{x})$. By Definition 6.3 on (s,h), we have $l \notin (\mathtt{dom}(h_B) \cup \mathtt{ran}(h_B))$. So $l \notin (\mathtt{dom}(h_B) \cup \mathtt{ran}(h_B) \cup \{s(\mathtt{x}) : \mathtt{x} \in \mathtt{FV}(\psi')\})$. As $l \in \mathtt{dom}(h_E)$ and $\mathcal{E}_j$ is extracted from h_E, l does not occur in $(\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_j)$. Moreover, for every location l' that does not occur in $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_j$, we have $\mathcal{E}_j[l \leftarrow l'] = \mathcal{E}_j$.

Since $\{s(\mathtt{x}) : \mathtt{x} \in \mathtt{FV}(\psi')\})$, $\mathtt{dom}(h)$, $\mathtt{ran}(h)$ and $\mathcal{R}_1, \ldots, \mathcal{R}_j$ are finite sets, there is $l' \in \mathtt{Loc}$ such that:

- $l' \notin (\mathtt{dom}(h_B) \cup \mathtt{ran}(h_B) \cup \{s(\mathtt{x}) : \mathtt{x} \in \mathtt{FV}(\psi')\})$ and $l' \notin \mathtt{dom}(h_E)$,
- l' does not occur in $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_j$.

By Lemma 2.1, there is $l' \notin \mathtt{dom}(h_E)$ such that

$$\big(\ s[l \leftarrow l']\ ,\ h_B\ \big)\ \ ,\ \ \mathcal{E}_j[\mathtt{P}_{j+1} \mapsto \mathcal{R}][l \leftarrow l']$$

satisfies ψ'. As $l \notin \{s(\mathtt{x}) : \mathtt{x} \in \mathtt{FV}(\psi)\}$, we also have $s[l \leftarrow l'] = s$. Let $\mathcal{R}''$ be $\mathcal{R}[l \leftarrow l']$. Since $\mathcal{E}_j[l \leftarrow l'] = \mathcal{E}_j$, we obtain $(s, h_B), \mathcal{E}_j[\mathtt{P}_{j+1} \mapsto \mathcal{R}''] \models \psi'$.

If $p \geq 1$ locations in $\mathtt{dom}(h_E)$ occur in $\mathcal{R}$, then $p-1$ locations in $\mathtt{dom}(h_E)$ occur in $\mathcal{R}''$. Applying the above transformation iteratively p times, we can build a relation $\mathcal{R}'$ such that no location in $\mathtt{dom}(h_E)$ occurs in $\mathcal{R}'$ and $(s, h_B), \mathcal{E}_j[\mathtt{P}_{j+1} \mapsto \mathcal{R}'] \models \psi'$.

Hence, $(s, h_B), \mathcal{E}_j \models \exists \mathtt{P}_{j+1}\ \psi'$ if and only if there is a finite binary relation $\mathcal{R} \in \mathcal{P}_f(\mathtt{Loc}^2)$ such that $(s, h_B), \mathcal{E}_j[\mathtt{P}_{j+1} \mapsto \mathcal{R}] \models \psi'$ and no location in $\mathtt{dom}(h_E)$ occurs in $\mathcal{R}$.

Defining (s', h'_E). Let us build s' and h'_E such that

(A) (s', h'_E) is an environment between $\mathtt{z}^m_{j+1}$ and $\mathtt{z}^M_{j+1}$.

(B) $(s', h * h'_E)$ is $(j+1)$-well-formed with the environment part $h_E * h'_E$.

Suppose that $\mathcal{R}$ contains $\alpha \geq 0$ pairs, say $\mathcal{R} = \{(l'_1, l''_1), \ldots, (l'_\alpha, l''_\alpha)\}$. Let us build an environment (s', h'_E) whose spectrum, for $\beta = \sharp \mathtt{z}^M_j + 1$, can be depicted as

$$\overset{\beta}{\bullet}\ \ \circ\ \ \overbrace{(\bullet\ \ \bullet\ \ \circ\ \ \ldots\ \bullet\ \ \bullet\ \ \circ)}^{\alpha \text{ times the pattern } \bullet\bullet\circ}\ \ \overset{\beta+3\alpha+2}{\bullet}$$

Its set of natural numbers S is equal to

$$\{\beta, \beta + 3\alpha + 2\} \quad \cup \quad \{\beta + 3k + 2, \beta + 3k + 3 : \ 0 \leq k \leq \alpha - 1\}.$$

A location l is said to be *fresh* if l is not in the set

$$(\{l'_k, l''_k : 1 \leq k \leq \alpha\} \ \cup \ \mathtt{dom}(h) \ \cup \ \mathtt{ran}(h) \ \cup \ \{s(\mathtt{x}) : \mathtt{x} \in \mathtt{FV}(\psi')\}).$$

By finiteness of the involved objects, let X be the following set of fresh locations (there is no need to provide here precise values):

$$\{l_\gamma : \gamma \in S\} \quad \cup \quad \{l_\gamma^{\gamma'} : \gamma \in S, \ 1 \leq \gamma' \leq \gamma\} \quad \cup \quad \{l''_0, l'_{\alpha+1}\}.$$

The store s' is defined from s by only imposing that $s'(\mathtt{z}^m_{j+1}) = l''_0$ and $s'(\mathtt{z}^M_{j+1}) = l'_{\alpha+1}$. The heap h'_E has domain X and it is defined as follows:

- $h'_E(l_\gamma^{\gamma'}) = l_\gamma$ for $\gamma \in S$ and $1 \leq \gamma' \leq \gamma$,
- $h'_E(l_{\beta+3k+2}) = l'_k$ and $h'_E(l_{\beta+3k+3}) = l''_k$ for $0 \leq k \leq \alpha - 1$,
- $h'_E(l_\beta) = l''_0$ and $h'_E(l_{\beta+3\alpha+2}) = l'_{\alpha+1}$.

By an easy (and long) verification, one can check that (A) and (B) hold true. Moreover, the relations extracted from $h_E * h'_E$ (see Definition 6.9) are precisely $\mathcal{R}_1, \ldots, \mathcal{R}_j, \mathcal{R}$ and for every $\mathtt{x} \in \mathtt{FV}(\psi')$, $s'(\mathtt{x}) \notin \mathtt{dom}(h_E * h'_E)$. By the induction hypothesis, $(s', h_B), \mathcal{E}_j[\mathtt{P}_{j+1} \mapsto \mathcal{R}] \models \psi'$ iff $(s', h * h'_E) \models \psi'$. By Lemma 2.2, $(s', h * h'_E) \models \psi'$.

By (A), $(s', h'_E) \models \mathtt{relation}_{j+1,\mathtt{FV}(\psi')}$. Additionally, by definition of l''_0 and $l'_{\alpha+1}$, we have $(s', h) \models \mathtt{isol}(\mathtt{z}^m_{j+1})$ and $(s', h) \models \mathtt{isol}(\mathtt{z}^M_{j+1})$. Finally, since $(s, h * h'_E)$ is $(j+1)$-well-formed, we have $(s, h * h'_E) \models \mathtt{PseudoEnv}(\mathtt{z}^m_0, \mathtt{z}^M_{j+1})$ (Lemma 6.1) and $(s, h * h'_E) \models \sharp\mathtt{z}^M_j + 1 = \sharp\mathtt{z}^m_{j+1}$. As a conclusion, we have shown $(s, h) \models t_j(\exists \mathtt{P}_{j+1} \ \psi')$.

($\rightarrow$) Suppose that $(s, h) \models t_j(\exists \mathtt{P}_{j+1} \ \psi')$. In other words, there are locations $l, l' \notin (\mathtt{ran}(h) \cup \mathtt{dom}(h))$, and a disjoint heap $h'_E \perp h$ such that the claims below are true

1. h'_E is disjoint from h and s' differs from s at most for the variables $\mathtt{z}^m_{j+1}$ and $\mathtt{z}^M_{j+1}$.
2. $s'(\mathtt{z}^m_{j+1}) = l$ and $s'(\mathtt{z}^M_{j+1}) = l'$ do not belong to $\mathtt{dom}(h)$.
3. (s', h'_E) is an environment between $\mathtt{z}^m_{j+1}$ and $\mathtt{z}^M_{j+1}$.
4. $(s', h * h'_E) \models \sharp\mathtt{z}^M_j + 1 = \sharp\mathtt{z}^m_{j+1}$.

5. $\mathtt{dom}(h'_E) \cap \mathtt{ran}(h) = \emptyset$.

These claims essentially follow from the definition of formula $t_j(\exists \mathtt{P}_{j+1}\ \psi')$, the only difficult part being claim 5. Let us detail this last point: while merging h and h_E, no new marker can be created so any marker in $h * h_E$ is a marker either from h_E or from h, with the same degree. Moreover, $h * h_E$ satisfies $\mathtt{PseudoEnv}(\mathtt{z}_0^m, \mathtt{z}_{j+1}^M)$, so the spectrum of h_E is included in the one of $h * h_E$. Combining these two facts, it follows that all markers of h_E are still markers of the same degree in $h * h_E$, and in particular claim 5 holds.

Now, claims 1-5 are precisely the assumptions from Lemma 6.6 and therefore $(s', h * h'_E)$ is $(j+1)$-well-formed. Observe that (5) is consequence of (3). Since $(s', h * h'_E) \models t_j(\psi')$ and for every $\mathtt{x} \in \mathtt{FV}(\psi')$ $s(\mathtt{x}) \notin \mathtt{dom}(h_E * h'_E)$, we can then apply the induction hypothesis and obtain $(s, h_B), \mathcal{E}_{j+1} \models \psi'$, that is $(s, h_B), \mathcal{E}_j \models \exists \mathtt{P}_{j+1}\ \psi'$ where $\mathcal{E}_{j+1}$ is extracted from $h_E * h'_E$. $\Box$

Here is our main result about the expressive power of SL.

Theorem 6.11 $\mathtt{SL}(\multimap) \equiv \mathtt{SL} \equiv \mathtt{SO} \equiv \mathtt{DSO}$.

Proof The proof follows from the following properties:

- $\mathtt{SL}(\multimap) \sqsubseteq \mathtt{SL}$ and $\mathtt{DSO} \sqsubseteq \mathtt{SO}$ by simply considering syntactic fragments.
- $\mathtt{SL} \sqsubseteq \mathtt{DSO}$ and $\mathtt{SO} \sqsubseteq \mathtt{DSO}$ by Proposition 2.6.
- $\mathtt{DSO} \sqsubseteq \mathtt{SL}(\multimap)$.

It remains to show that $\mathtt{DSO} \sqsubseteq \mathtt{SL}(\multimap)$ by using Proposition 6.10. Let ϕ be a DSO sentence. Without any loss of generality, we can assume that ϕ has no free occurrence of first-order variables of the form $\mathtt{z}_j^{\blacksquare}$ (otherwise, other auxilliary variables are used) and ϕ satisfies the extended Barendregt convention since every DSO sentence can be reduced to an equivalent one in logspace. Let (s, h) be a model. The statements below are equivalent

- $(s, h) \models T(\phi)$,
- There are $h'_E \perp h$, l, l' and $s' = s[\mathtt{z}_0^m \mapsto l, \mathtt{z}_0^M \mapsto l']$ such that
  - l and $l' \notin \mathtt{dom}(h) \cup \mathtt{ran}(h)$,
  - $l, l' \in \mathtt{dom}(h')$ and for every location $l'' \in \mathtt{dom}(h'_E) \setminus \{l, l'\}$, we have $h'_E(l'') \in \{l, l'\}$.
  - In $(s', h * h'_E)$, $\sharp \mathtt{z}_0^M = 2 + \sharp \mathtt{z}_0^m$ and for every $l'' \in \mathtt{dom}(h)$, we have $\sharp \mathtt{z}_0^m \geq 3 + \sharp l''$.

- l and l' are extremities in $(s', h * h'_E)$.
- $(s', h * h'_E) \models t_0(\phi)$.

(by definition of $T(\cdot)$ and $\models$)

- There are $h' \perp h$, l and l' such that
  - $(s', h * h'_E)$ is an environment with $\sharp \mathtt{z}_0^M = 2 + \sharp \mathtt{z}_0^m$.
  - $(s', h * h') \models t_0(\phi)$.

  (by Definition 6.3 and Lemma 6.5)
- There are $h' \perp h$, l and l' such that
  - $(s', h * h'_E)$ is an environment with $\sharp \mathtt{z}_0^M = 2 + \sharp \mathtt{z}_0^m$.
  - $(s', h), \mathcal{E}_0 \models \phi$ for any environment $\mathcal{E}_0$ extracted from h'_E.

  (by Proposition 6.10)
- $(s, h) \models \phi$ since
  - the variables $\mathtt{z}_0^m$ and $\mathtt{z}_0^M$ do not occur in ϕ and ϕ is a sentence.
  - h'_E can always be built since h is essentially a finite structure.

$\square$

Observe that all the equivalences are obtained with logarithmic-space translations. Consequently,

Corollary 6.12 $\mathtt{SL}(-\!\!*)$ *satisfiability problem is undecidable.*

Proof We have seen that for every sentence such that ϕ in $\mathtt{DSO}$, there is an effective way to compute ϕ' in $\mathtt{SL}(-\!\!*)$ such that ϕ and ϕ' hold on exactly the same models. In order to show undecidability of $\mathtt{SL}(-\!\!*)$, it is sufficient to provide a reduction from finitary satisfiability for classical predicate logic restricted to a single binary predicate symbol (see e.g. [37]) to $\mathtt{DSO}$. Let ϕ be a first-order formula built over the binary predicate symbol $\mathtt{R}$. One can easily show that ϕ is satisfiable iff

$$\exists \mathtt{D}\ \exists \mathtt{R}\ (\forall \mathtt{x}\ \mathtt{y}\ \mathtt{R}(\mathtt{x}, \mathtt{y}) \Rightarrow \mathtt{D}(\mathtt{x}, \mathtt{x}) \wedge \mathtt{D}(\mathtt{y}, \mathtt{y})) \wedge t(\phi)$$

is satisfiable. The map t is the identity map for atomic formulae, homomorphic for Boolean connectives and performs a relativization for first-order quantification: $t(\forall \mathtt{x}\ \psi) = \forall \mathtt{x}\ \mathtt{D}(\mathtt{x}, \mathtt{x}) \Rightarrow t(\psi)$. The intention is obviously that $\mathtt{D}(\mathtt{x}, \mathtt{x})$ holds true whenever $\mathtt{x}$ belongs to the finite model. $\square$

Undecidability of $\mathtt{SL}(-\!\!*)$ can be obtained much 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 5). Indeed, computations of length n can be encoded as lists of length $3n$; three successive locations encode a configuration of the machine and for two of those locations, counter values are encoded by the numbers of predecessors. Corollary 6.12 is obtained with the stronger result $\mathtt{SL}(-\!\!*) \equiv \mathtt{DSO}$ since $\mathtt{DSO}$ is undecidable.

7. Extensions with More Than one Selector

In order to express advanced arithmetical constraints (see Section 5) or to encode finite sets of pairs of locations (see Section 6), we have introduced additional parts in the heaps via markers. In order to distinguish these auxiliary markers from the original heap, we have decided to use markers of small degree (as in Section 5) or markers of large degree (as in Section 6). However, in the presence of memory cells with strictly more than one selector it is even easier to identify these auxiliary markers; for example, the memory cells $l \mapsto l'$ introduced in a model to check arithmetical constraints or to encode environments can be replaced by memory cells of the form

$$l \mapsto l', \overbrace{\perp, \ldots, \perp}^{(k-1)\text{ times}}$$

where $\perp$ is a location that is not present in the original model (i.e. not in $\mathtt{ran}(h) \cup \mathtt{dom}(h)$). We write $k\mathtt{SL}$ [resp. $k\mathtt{SO}$] to denote the variant of $\mathtt{SL}$ [resp. $\mathtt{SO}$] with k selectors. In that case, a heap h is defined as a partial function $h : \mathtt{Loc} \rightharpoonup \mathtt{Loc}^k$ with finite domain. The atomic formulae of the form $\mathtt{x} \hookrightarrow \mathtt{y}$ from $\mathtt{SL}$ are replaced by $\mathtt{x} \hookrightarrow \mathtt{y}_1, \ldots, \mathtt{y}_k$. Obviously $1\mathtt{SL}$ [resp. $1\mathtt{SO}$] corresponds to $\mathtt{SL}$ [resp. $\mathtt{SO}$]. We write $k\mathtt{SO}^{k'}$ to denote the restriction of $k\mathtt{SO}$ to second-order variables in $\mathtt{VAR}_{k'}$.

In the rest of this section, we assume that $k > 1$. We dedicate the rest of this section to show Theorem 7.1 below can be proved by adapting what we did for a unique selector. We may overload symbols but no confusion should occur. 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 $k > 1$, this could be done by simply checking the presence of distinguished values.

Theorem 7.1 *For every $k > 1$, $k\mathtt{SL} \equiv k\mathtt{SL}(-\!\!*) \equiv k\mathtt{SO}$.*

We establish Theorem 7.1 by adapting the proof for $k = 1$. However, a simpler proof for $k > 1$ is possible but it would require a different approach that cannot find its place in this paper. First, an obvious adaptation of the proof of Propositions 2.5 and 2.6 allows us to show the statement below.

Lemma 7.2 *$k\mathtt{SL} \sqsubseteq k\mathtt{SO}^{k+1}$ and $k\mathtt{SO}^{k+1} \sqsubseteq k\mathtt{SO}^2$.*

It remains to show that $k\mathtt{SO}^2 \sqsubseteq k\mathtt{SL}(-\!\!*)$. The basic observation is that all the auxiliary memory cells $l \mapsto l'$ introduced in a model to check arithmetical constraints or to encode environments are replaced by memory cells of the form

$$l \mapsto l', \overbrace{\perp, \ldots, \perp}^{(k-1) \text{ times}}$$

where $\perp$ is a location that is not present in the original model. Observe that it is easy to check that a memory cell is auxiliary by simply inspecting the presence of $\perp$. We shall also enforce that in a new memory cell, l' is different from $\perp$ and the $(k-1)$ remaining locations are exactly $\perp$.

Before explaining the adaption, we introduce alternative definitions:

- Given (s, h) and a location l, we write $\sharp l$ to denote the cardinal of $\{l' \in \mathtt{Loc} : h(l') = (l, \ldots)\}$ (number of *1-predecessors* of the location l in (s, h)).
- We write $\mathtt{x} \hookrightarrow \mathtt{y}$ as a shortcut for $\exists \mathtt{y}_2 \cdots \mathtt{y}_k \ \mathtt{x} \hookrightarrow \mathtt{y}, \mathtt{y}_2, \ldots, \mathtt{y}_k$.
- A [resp. *strict*] *marker* in (s, h) is a sequence of distinct locations $l, l_0, \ldots, l_n$ for some $n \geq 0$ (all distinct from $\perp$) such that
  - $h(l_0) = (l, \overbrace{\perp, \ldots, \perp}^{k-1 \text{ times}})$ [resp. and $\mathtt{dom}(h) = \{l_0, \ldots, l_n\}$],
  - for every $i \in \{1, \ldots, n\}$, $h(l_i) = (l_0, \overbrace{\perp, \ldots, \perp}^{k-1 \text{ times}})$ and $\sharp l_i = 0$,
  - $\sharp l_0 = n$.
- We define an *extremity* as a location l in a model such that l has at least one 1-predecessor and no 1-predecessor l' of l appears in some tuple from $\mathtt{ran}(h)$.

Let $\varphi_\perp$ be the formula specifying that auxiliary memory cells are of the above shape:

$$\forall \mathtt{x}, \mathtt{x}_1, \ldots, \mathtt{x}_k.\ \mathtt{x} \hookrightarrow \mathtt{x}_1, \ldots, \mathtt{x}_k \Rightarrow (\mathtt{x} \neq \mathtt{x}_\perp \wedge \mathtt{x}_1 \neq \mathtt{x}_\perp \wedge \bigwedge_{i=1}^{k} \mathtt{x}_i = \mathtt{x}_\perp)$$

Following the developments from Section 5, we can show the following theorem.

Theorem 7.3 *For $c, c' \geq 0$, there is a formula ϕ in $k\mathtt{SL}(\mathrel{-\!\!*})$ of quadratic size in $c + c'$ such that for every model (s, h), we have $(s, h) \models \phi$ iff $\sharp\mathtt{x} + c \leq \sharp\mathtt{y} + c'$.*

Basically, we consider the formula from Section 5 in which we add to the first argument to any subformula with outermost connective either $\overset{\neg}{\mathrel{-\!\!*}}$ or $\mathrel{-\!\!*}$ the conjunct $\varphi_\perp$, exactly when we need to introduce markers. Moreover, in some cases, formulae of the form $\mathtt{x} \hookrightarrow \mathtt{y}$ for the one selector case from Section 5 are replaced by $\mathtt{x} \hookrightarrow \mathtt{y}, \mathtt{x}_\perp, \ldots, \mathtt{x}_\perp$ when markers are involved.

Let us consider the reduction from $k\mathtt{SO}^2$ into $k\mathtt{SL}$. Given a sentence in $k\mathtt{SO}^2$ satisfying the extended Barendregt convention and with n second-order variables, its translation is defined below

$$\exists \mathtt{x}_\perp\ \neg(\exists \mathtt{x}, \mathtt{x}_1, \ldots, \mathtt{x}_k\ (\mathtt{x} \hookrightarrow \mathtt{x}_1, \ldots, \mathtt{x}_k) \wedge (\mathtt{x} = \mathtt{x}_\perp \vee \bigvee_{i=1}^{k} \mathtt{x}_i = \mathtt{x}_\perp)) \wedge T'(\phi)$$

where $T'(\phi)$ is a variant of the map $T(\phi)$ for the one selector case in which the definition of $t_j(\psi)$ is modified as follows.

1. $T'(\phi)$ takes the value below:

$$\begin{gathered}
\exists \mathtt{z}_0^m\ \mathtt{z}_0^M\ \ \mathtt{isol}(\mathtt{z}_0^M) \wedge \mathtt{isol}(\mathtt{z}_0^m) \wedge \\
[((\forall \mathtt{x}.\ \mathtt{alloc}\ (\mathtt{x}) \Rightarrow (\mathtt{x} \hookrightarrow \mathtt{z}_0^M \vee \mathtt{x} \hookrightarrow \mathtt{z}_0^m \vee \mathtt{x} = \mathtt{z}_0^M \vee \mathtt{x} = \mathtt{z}_0^m)) \wedge \\
\mathtt{alloc}\ (\mathtt{z}_0^M) \wedge \mathtt{alloc}\ (\mathtt{z}_0^m)) \wedge \varphi_\perp \overset{\neg}{\mathrel{-\!\!*}} \\
(\forall \mathtt{x}.\mathtt{x} \neq \mathtt{z}_0^M \wedge \mathtt{x} \neq \mathtt{z}_0^m \Rightarrow (\sharp\mathtt{z}_0^m > 2 + \sharp\mathtt{x})) \wedge (\sharp\mathtt{z}_0^M = 2 + \sharp\mathtt{z}_0^m) \wedge \\
\mathtt{extr}(\mathtt{z}_0^m) \wedge \mathtt{extr}(\mathtt{z}_0^M) \wedge t_0(\phi))]
\end{gathered}$$

The formula $\mathtt{isol}(\mathtt{x})$ is an abbreviation for

$$\forall\ \mathtt{y}, \mathtt{y}_1, \ldots, \mathtt{y}_k\ (\mathtt{y} \hookrightarrow \mathtt{y}_1, \ldots, \mathtt{y}_k) \Rightarrow ((\mathtt{y} \neq \mathtt{x}) \wedge \bigwedge_{i=1}^{i=k} (\mathtt{y}_i \neq \mathtt{x})).$$

2. $\texttt{notonenv}(\mathtt{x})$ is defined by

$$\neg(\exists \mathtt{y}\ \mathtt{x} \hookrightarrow \mathtt{y}, \mathtt{x}_\perp, \ldots, \mathtt{x}_\perp \vee \mathtt{y} \hookrightarrow \mathtt{x}, \mathtt{x}_\perp, \ldots, \mathtt{x}_\perp) \wedge \mathtt{x} \neq \mathtt{x}_\perp.$$

3. $t_j(\mathtt{x} = \mathtt{y}) \ \triangleq\ \mathtt{x} = \mathtt{y}$,
4. $t_j(\mathtt{x} \hookrightarrow \mathtt{y}) \ \triangleq\ \mathtt{x} \hookrightarrow \mathtt{y}$,
5. for $i \leq j$, $t_j(\mathtt{P}_i(\mathtt{x}, \mathtt{y}))$ is defined by

$$\exists \mathtt{z}, \mathtt{z}'\ (\mathtt{z} \hookrightarrow \mathtt{x})\ \wedge\ (\mathtt{z}' \hookrightarrow \mathtt{y})\ \wedge\ (\sharp \mathtt{z} > \sharp \mathtt{z}_j^m)\ \wedge\ (\sharp \mathtt{z}' < \sharp \mathtt{z}_j^M) \wedge$$

$$(\sharp \mathtt{z}' = 1 + \sharp \mathtt{z})\ \wedge\ \texttt{extr}(\mathtt{z})\ \wedge\ \texttt{extr}(\mathtt{z}')$$

6. t_j is homomorphic for Boolean connectives.
7. $t_j(\exists \mathtt{x}\ \psi) \ \triangleq\ \exists \mathtt{x}\ \texttt{notonenv}(\mathtt{x}) \wedge t_j(\psi)$.
8. $t_j(\exists \mathtt{P}_{j+1}, \psi)$, is defined by

$$\exists \mathtt{z}_{j+1}^m, \mathtt{z}_{j+1}^M\ \texttt{isol}(\mathtt{z}_{j+1}^m) \wedge \texttt{isol}(\mathtt{z}_{j+1}^M \wedge$$

$$((\texttt{relation}_{j+1,\texttt{FV}(\psi)} \wedge \varphi_\perp) \mathrel{\overline{-\!\!*}}$$

$$(\texttt{PseudoEnv}(\mathtt{z}_0^m, \mathtt{z}_{j+1}^M) \wedge \sharp \mathtt{z}_j^M + 1 = \sharp \mathtt{z}_{j+1}^m \wedge t_{j+1}(\psi)))$$

in which $\texttt{relation}_{j+1,\texttt{FV}(\psi)}$ and $\texttt{PseudoEnv}(\mathtt{z}_0^m, \mathtt{z}_{j+1}^M)$ are slightly updated in order to take into account that the markers are made of memory cells of the form $l \mapsto l', \perp, \ldots, \perp$.

Adapting Definition 6.9 with 1-predecessors, we can state a proposition similar to Proposition 6.10 leading to Theorem 7.1.

8. Concluding Remarks

In the paper, we have mainly studied first-order separation logic with one selector $\texttt{SL}$ for which we have shown the following results:

1. $\texttt{SL}(*)$ is decidable with non-elementary complexity.
2. $\texttt{SL}(* + \overset{n}{\overline{-\!\!*}})$, extending $\texttt{SL}(*)$ with bounded septraction is also decidable.
3. $\texttt{SL}$ is as expressive as weak second-order logic $\texttt{SO}$.
4. $\texttt{SL}$ is as expressive as $\texttt{SL}(-\!\!*)$ as a by-product of our proof technique.
5. $\texttt{SL}(-\!\!*)$ satisfiability is undecidable.

This solves two central open problems: the decidability status of $\texttt{SL}$ and the characterization of its expressive power. Moreover, the above results about expressive power extend naturally to the case with k selectors, for some $k \geq 1$: $k\texttt{SL} \equiv k\texttt{SL}(\mathrel{-\!\!*}) \equiv k\texttt{SO}$.

Acknowledgments. We would like to thank the following colleagues for fruitful discussions related to this work: Anuj Dawar, Arnaud Durand, Didier Galmiche, Dominique Larchey-Wendling, Daniel Méry. Moreover, we wish to thank the anonymous referees for their numerous helpful suggestions and comments on an earlier version of this paper. This has been a great help to improve the quality of the paper.

References

[1] Antonopoulos, T., Dawar, A., 2009. Separating graph logic from MSO. In: FOSSACS'09. Vol. 5504 of Lecture Notes in Computer Science. Springer, pp. 63–77.

[2] Bansal, K., Brochenin, R., Lozes, E., 2009. Beyond shapes: Lists with ordered data. In: FOSSACS'09. Vol. 5504 of Lecture Notes in Computer Science. Springer, pp. 425–439.

[3] Berdine, J., Calcagno, C., O'Hearn, P., 2004. A decidable fragment of separation logic. In: FST&TCS'04. Vol. 3328 of Lecture Notes in Computer Science. Springer, pp. 97–109.

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

[5] Bouajjani, A., Habermehl, P., Moro, P., Vojnar, T., 2005. Verifying programs with dynamic 1-selector-linked structured in regular model-checking. In: TACAS'05. Vol. 3440 of Lecture Notes in Computer Science. Springer, pp. 13–29.

[6] Bozga, M., Iosif, R., Lakhnech, Y., 2004. On logics of aliasing. In: SAS'04. Vol. 3148 of Lecture Notes in Computer Science. Springer, pp. 344–360.

[7] Bozga, M., Iosif, R., Perarnau, S., 2008. Quantitative separation logic and programs with lists. In: IJCAR'08. Vol. 5195 of Lecture Notes in Computer Science. Springer, pp. 34–49.

[8] Brochenin, R., Demri, S., Lozes, E., 2008. On the almighty wand. In: CSL'08. Vol. 5213 of Lecture Notes in Computer Science. Springer, pp. 322–337.

[9] Brochenin, R., Demri, S., Lozes, E., 2008. On the almighty wand. Tech. rep., LSV, ENS de Cachan.

[10] Brochenin, R., Demri, S., Lozes, E., 2009. Reasoning about sequences of memory states. Annals of Pure and Applied Logic 161 (3), 305–323.

[11] Brotherston, J., Kanovich, M. I., 2010. Undecidability of propositional separation logic and its neighbours. In: LICS'10. pp. 130–139.

[12] Büchi, J., 1960. On a decision method in restricted second-order arithmetic. In: Logic, Methodology, and Philosophy of Science. pp. 1–11.

[13] Calcagno, C., Gardner, P., Zarfaty, U., 2007. Context logic as modal logic: completeness and parametric inexpressivity. In: POPL'07. pp. 123–134.

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

[15] Calcagno, C., Yang, H., O'Hearn, P., 2001. Computability and complexity results for a spatial assertion language for data structures. In: FST&TCS'01. Vol. 2245 of Lecture Notes in Computer Science. Springer, pp. 108–119.

[16] Cook, B., Haase, C., Ouaknine, J., Parkinson, M., Worrell, J., 2011. Tractable reasoning in a fragment of separation logic. In: CONCUR'11. Vol. 6901 of Lecture Notes in Computer Science. pp. 235–249.

[17] Dawar, A., Gardner, P., Ghelli, G., 2004. Adjunct elimination through games in static ambient logic. In: FST&TCS'04. Vol. 3328 of Lecture Notes in Computer Science. Springer, pp. 211–223.

[18] Dawar, A., Gardner, P., Ghelli, G., 2007. Expressiveness and complexity of graph logic. Information & Computation 205 (3), 263–310.

[19] Etessami, K., Vardi, M., Wilke, T., 2002. First-order logic with two variables and unary temporal logic. Information & Computation 179 (2), 279–295.

[20] Galmiche, D., Méry, D., 2010. Tableaux and resource graphs for separation logic. Journal of Logic and Computation 20 (1), 189–231.

[21] Gorogiannis, N., Kanovich, M., O'Hearn, P., 2011. The complexity of abduction for separated heap abstractions. In: SAS'11. Vol. 6887 of Lecture Notes in Computer Science. pp. 25–42.

[22] Ishtiaq, S., O'Hearn, P., 2001. BI as an assertion language for mutable data structures. In: POPL'01. pp. 14–26.

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

[24] Kamp, J., 1968. Tense logic and the theory of linear order. Ph.D. thesis, UCLA, USA.

[25] Klaedtke, F., Rueb, H., 2003. Monadic second-order logics with cardinalities. In: ICALP'03. Vol. 2719 of Lecture Notes in Computer Science. Springer, pp. 681–696.

[26] Kuncak, V., Rinard, M., October 2004. On spatial conjunction as second-order logic. Tech. rep., MIT CSAIL.

[27] Larchey-Wendling, D., Galmiche, D., 2010. The Undecidability of Boolean BI through Phase Semantics. In: LICS'10. pp. 140–149.

[28] Lev-Ami, T., Sagiv, M., 2000. TVLA: A system for implementing static analyses. In: SAS'00. Vol. 1824 of Lecture Notes in Computer Science. Springer, pp. 280–301.

[29] Löding, C., Rohde, P., 2003. Model checking and satisfiability for sabotage modal logic. In: FST&TCS'03. Vol. 2914 of Lecture Notes in Computer Science. Springer, pp. 302–313.

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

[31] Lozes, E., 2005. Elimination of spatial connectives in static spatial logics. Theoretical Computer Science 330 (3), 475–499.

[32] Magill, S., Berdine, J., Clarke, E., Cook, B., 2007. Arithmetic strengthening for shape analysis. In: SAS'07. Vol. 4634 of Lecture Notes in Computer Science. Springer, pp. 419–436.

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

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

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

[36] Stockmeyer, L., 1974. The complexity of decision problems in automata theory and logic. Ph.D. thesis, Department of Electrical Engineering, MIT.

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

[38] van Benthem, J., 2005. An essay on sabotage and obstruction. In: Mechanizing Mathematical Reasoning. Essays in Honor of Jorg Siekmann on the Occasion of his 69th Birthday. Springer-Verlag, pp. 268–276.

[39] Yorsh, G., Rabinovich, A. M., Sagiv, M., Meyer, A., Bouajjani, A., 2005. A logic of reachable patterns in linked data structures. In: FOSSACS'05. Vol. 3441 of Lecture Notes in Computer Science. Springer, pp. 94–110.