Expressive Completeness of Separation Logic With Two Variables and No Separating Conjunction

STEPHANE DEMRI, LSV, ENS Cachan, CNRS, Université Paris-Saclay, 94235 Cachan, France
MORGAN DETERS, New York University, USA

Separation logic is used as an assertion language for Hoare-style proof systems about programs with pointers, and there is an ongoing quest for understanding its complexity and expressive power. Herein, we show that first-order separation logic with one record field restricted to two variables and the separating implication (no separating conjunction) is as expressive as weak second-order logic, substantially sharpening a previous result. Capturing weak second-order logic with such a restricted form of separation logic requires substantial updates to known proof techniques. We develop these, and as a by-product identify the smallest fragment of separation logic known to be undecidable: first-order separation logic with one record field, two variables, and no separating conjunction. Because we forbid ourselves the use of many syntactic resources, this underscores even further the power of separating implication on concrete heaps.

Categories and Subject Descriptors: F.3.1 [**Specifying and Verifying and Reasoning about Programs**]: Logics of Programs

General Terms: Theory, Verification

Additional Key Words and Phrases: separation logic, expressive completeness, two-variable logics, undecidability

ACM Reference Format:
S. Demri, M. Deters, October 9th, 2015 [paper accepted], September 8th, 2015 [revised submission]. Initial submission: April 3rd, 2015. Expressive Completeness of Separation Logic With Two Variables and No Separating Conjunction *ACM Trans. Comput. Logic* V, N, Article SL2 (January YYYY), 52 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Expressive completeness. The literature is rich with results comparing the expressive power of non-classical logics with most standard logics such as first- or second-order logic. For instance, the celebrated Kamp's Theorem [Kamp 1968; Rabinovich 2014] amounts to stating that linear-time temporal logic (LTL) is equal in expressive power to first-order logic. More generally, we know the expressive completeness of Stavi connectives for general linear time, see e.g. [Gabbay et al. 1994]. This has been refined to the restriction to two variables, leading to the equivalence between unary LTL and FO2, see e.g. [Etessami et al. 1997; Weis 2011] (see also [Sreejith 2013] for related questions). Monadic second-order logic (MSO) is another yardstick logic and, for instance, it is well-known that ω-regular languages are exactly those definable in MSO,

Work partially supported by the EU Seventh Framework Programme (under grant PIOF-GA-2011-301166, DATAVERIF), the Air Force Office of Scientific Research (under award FA9550-09-1-0596), and the National Science Foundation (under grant 0644299). Part of this work was completed when the first author visited the ACSys group at New York University during the outgoing phase of the Marie Curie IOF Project DATAVERIF. This is a completed and extended version of [Demri and Deters 2014].

DOI: http://dx.doi.org/10.1145/0000000.0000000

see e.g. [Straubing 1994]. Similarly, extended temporal logic ETL, defined in [Wolper 1983] and extending LTL, is also known to be equally expressive with MSO. This applies also to linear μ-calculus [Vardi 1988] or to PSL [Lange 2007], to quote a few more examples. On non-linear structures, bisimulation invariant fragment of MSO and modal μ-calculus have been shown equivalent [Janin and Walukiewicz 1996]. In addition, there is a wealth of results relating first-order logic with two variables and non-classical logics, providing a neat characterization of the expressive power of many formalisms since first-order logic and second-order logic are queen logics. For instance, Boolean modal logic with converse and identity is as expressive as first-order logic with two quantified variables (FO2) [Lutz et al. 2001]. For sets of nodes, XPath has also been established equally expressive as FO2, see e.g. an overview in [Marx and de Rijke 2005], see also in [Bojanczyk et al. 2009] a version of FO2 on data trees. Sometimes, a third variable is needed to get expressive completeness. For instance, in [Venema 1991] it is proved that interval logic with connectives Chop, D and T is expressively complete over linear flows of time with respect to first-order logic restricted to three quantified variables. In the realm of interval temporal logics, we also know expressive completeness of metric propositional neighborhood logic with respect to the two-variable fragment of first-order logic for linear orders with successor function, interpreted over natural numbers [Bresolin et al. 2010].

In this paper, we compare separation logic restricted to two variables with (weak) second-order logic over concrete heaps.

Expressive power of separation logic. Separation logic is used as an assertion language for Hoare-style proof systems about programs with pointers [Apt 1981; Reynolds 2002], and there is an ongoing quest for understanding its complexity and expressive power. Alternatively, there are a lot of activities to develop verification methods with decision procedures for fragments of practical use, see e.g. [Cook et al. 2011; Haase et al. 2013]. Many decision procedures have been designed for fragments of separation logics or abstract variants, from analytic methods [Galmiche and Méry 2010; Hou et al. 2014; Hou et al. 2015] to translation to theories handled by SMT solvers [Piskac et al. 2013; Pérez and Rybalchenko 2013; Piskac et al. 2014; Bansal et al. 2015], passing via graph-based algorithms [Haase et al. 2013]. Of course, there are also plenty of methods or heuristics that are even more tailored to verification, see e.g. [Calcagno et al. 2011; Thakur et al. 2014] and there are plenty of other methods to verify heap manipulating programs, see a nice overview in [Chakraborty 2012] for automata-based techniques.

Theoretical issues for separation logic stem from the design of expressive fragments with relatively low complexity (see e.g. [Cook et al. 2011]) to the extension of known decidability results, see e.g. [Bozga et al. 2010; Iosif et al. 2013; Antonopoulos et al. 2014]. Indeed, it is known since [Calcagno et al. 2001] that first-order separation logic with two record fields (herein called 2SL) is undecidable (with a proof that does not require separating connectives and uses Trakhtenbrot's Theorem [Trakhtenbrot 1963]). This is sharpened in [Brochenin et al. 2012] by showing that first-order separation logic with a unique record field (herein called 1SL) is also undecidable, as a consequence of the expressive equivalence between 1SL and weak second-order logic. More recently, 1SL restricted to two variables (1SL2) is shown undecidable too [Demri and Deters 2015] (by reduction from the halting problem for Minsky machines) but without touching the central question of expressive completeness—the purpose of the current paper. From the very beginning, the relationships between separation logic and second-order logic have been quite puzzling (see e.g. an interesting answer with infinite arbitrary structures in [Kuncak and Rinard 2004]). Moreover, comparisons of fragments have been also studied, for instance 1SL($*$) has been established strictly less expressive than MSO in [Antonopoulos and Dawar 2009] (see also the related work [Marcinkowski

2006]). In this paper, we go one step further by showing that two variables suffice to get expressive completeness, which is a real technical *tour de force* while establishing quite a surprising result. As a consequence, we conclude that 1SL2($-\!\!*$) (that is, 1SL2 without separating conjunction) is undecidable, too. This should not be confused with undecidability results from [Brotherston and Kanovich 2014; Larchey-Wendling and Galmiche 2013], which are obtained in an alternative setting with propositional variables and without first-order quantification. It is fair to recall that separating implication (also called the "magic wand") has been less well-studied than separating conjunction in the literature, but its use for program verification is far more recognized nowadays; see e.g. [Lee and Park 2014, Section 1] for a recent, insightful analysis (see also [Hou et al. 2014, Section 8] or [Thakur et al. 2014; Schwerhoff and Summers 2015]).

Our contribution. In this paper, we show that first-order separation logic with one record field, two quantified variables, and no separating conjunction is as expressive as weak second-order logic on heaps; in short, 1SL2($-\!\!*$) $\equiv$ WSOL. Even though conjectured in [Brochenin et al. 2012; Brochenin 2013], it is surprising that two variables suffice, and that further we are able to drop the separating conjunction, thus obtaining expressive completeness and undecidability with only two variables and the magic wand operator. In doing so, we improve previous undecidability results about separation logic [Calcagno et al. 2001; Brochenin et al. 2012; Demri and Deters 2015]. Because we forbid ourselves the use of many syntactic resources, this underlines even further the power of the magic wand. By way of comparison with [Grädel et al. 1999; Immerman et al. 2004], we show undecidability of a two-variable logic with second-order features. Our main undecidability result cannot be derived from [Grädel et al. 1999; Immerman et al. 2004] since in 1SL models, we deal with a single functional binary relation, namely the finite heap. We believe that we have identified the core of separation logic as far as undecidability and expressive completeness are concerned, since, for instance, first-order separation logic with one record field, one quantified variable and an unbounded number of program variables, has recently been shown decidable and it admits a PSPACE-complete satisfiability problem [Demri et al. 2014]. Figure 1 illustrates how the main result of the paper (Theorem 5.13) compares with known results from the literature. Section 2 contains formal definitions of the different logics.

Consequences of 1SL2($-\!\!*$) $\equiv$ WSOL. The first consequence of the equivalence 1SL2($-\!\!*$) $\equiv$ WSOL is certainly that two variables suffice to capture full 1SL and consequently to express any property on heaps that can be stated in weak second-order logic. As a consequence, undecidability holds, and the set of valid formulae in 1SL2($-\!\!*$) is not recursively enumerable. Clearly, this contrasts with many logical formalisms, such as FO2 on first-order structures or FO2 on ω-sequences that are decidable when the number of quantified variables is restricted to two or for which the computational complexity decreases significantly, see e.g. [Grädel et al. 1997; Etessami et al. 1997]. Moreover, such an expressive completeness result can be also used to establish structural completeness of separation logic, as shown in [Lozes 2012]. Hence, our main result has many interesting consequences, apart from the proof technique to show it, which we describe briefly, below.

Proof technique. In our proof, most of the difficulties are concentrated on the use of only two variables: we recycle variables as done for modal logics [Gabbay 1981], but this is insufficient, especially when separating conjunction is also banished. So, as far as the proof for expressive completeness goes, we borrow some of the first principles from [Brochenin et al. 2012], but very quickly we are faced with serious problems when

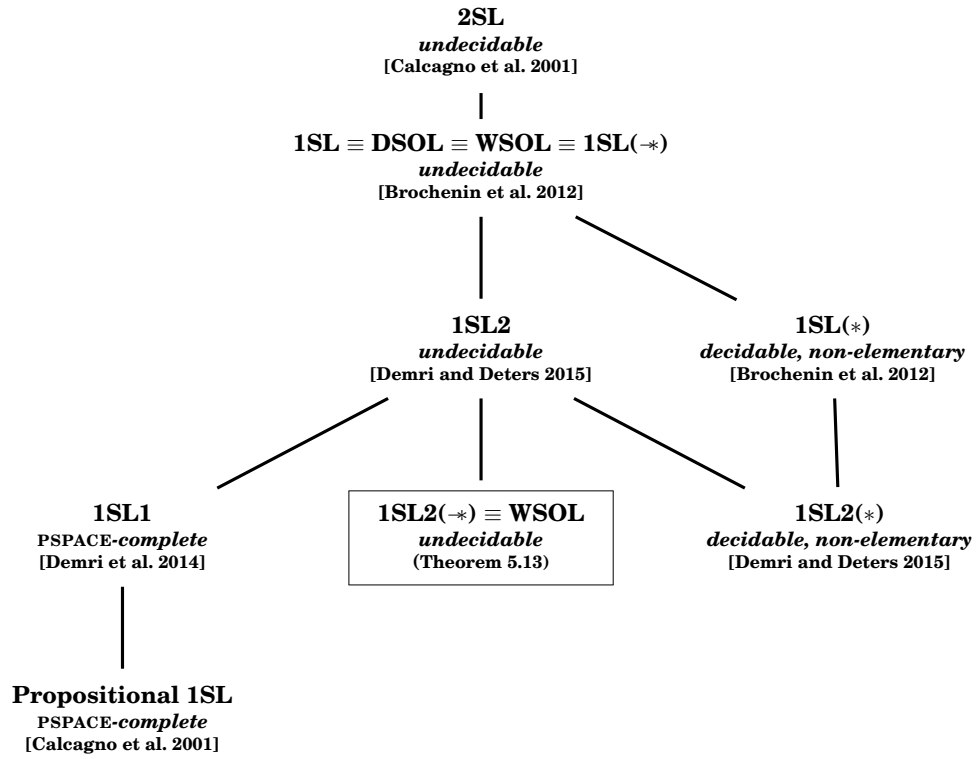


Fig. 1: Contribution of the paper and related work.

we need to identify in some heap at least $k > 0$ heap patterns (a typical example is to specify that at least $k > 0$ locations point to a given location).

Indeed, the standard way to identify such patterns is to use an unbounded number of variables or the separating conjunction. So, in the presence of only two variables and by using only the magic wand operator, instead of chopping the heap in k disjoint subheaps, we add $\mathcal{O}(k)$ new patterns so that the newly combined heap satisfies structural properties that witness the presence of the k patterns in the original heap. This high-level description has to be instantiated as many times as we have to identify different types of patterns, but this new point of view allows us to go far beyond what was known previously (see e.g., the proof of Lemma 3.3). At times, it is not strictly necessary to introduce a radically new method, but instead we can be more thrifty in the way formulae are defined to express desirable properties; of course, this may come with more complex proofs and, above all, more ingenuity to design such formulae. At last, after using the new techniques, after saving syntactic resources on formulae and after using first principles from [Brochenin et al. 2012], we are able to design a lengthy and tedious proof and to conclude that 1SL2($-\!\!*$) is as expressive as weak second-order logic on heaps, and as a by-product it is also the smallest known undecidable fragment of separation logic.

It should be noted that the paper is structured in such a way that we provide more and more complex building blocks to establish our main results. Indeed, after Section 2's preliminary material, Section 3 is dedicated to expressing properties about

reachability and comparing a number of location predecessors against a constant, whereas Section 4 deals with much richer comparisons between numbers of location predecessors. Another contribution of the present paper rests on the fact that we considerably simplify some of the technical insights borrowed from [Brochenin et al. 2012] and therefore the current paper proposes a self-contained proof of the equivalence between 1SL2($\mathit{-\!\!*}$) and weak second-order logic that in many ways is much simpler than what has been done so far, even though our results are stronger. The main result is proven in Section 5. Extensions with program variables or with heaps having $k > 1$ record fields are presented in Section 6. Furthermore, the variant when an infinite domain is allowed, is briefly discussed in Section 6.

2. PRELIMINARIES

2.1. First-order separation logic with one selector (1SL)

A *heap* $\mathfrak{h}$ is a partial function $\mathfrak{h} : \mathbb{N} \rightharpoonup \mathbb{N}$ with finite domain. We write $\mathrm{dom}(\mathfrak{h})$ to denote its *domain* and $\mathrm{ran}(\mathfrak{h})$ to denote its *range*. Two heaps $\mathfrak{h}_1, \mathfrak{h}_2$ are said to be *disjoint*, if their domains are disjoint; when this holds, we write $\mathfrak{h}_1 \uplus \mathfrak{h}_2$ to denote their disjoint union. *Locations* are elements of $\mathbb{N}$ and are denoted by $\mathfrak{l}$, possibly decorated with superscripts or subscripts. We write $\mathfrak{l}_1 \to \mathfrak{l}_2 \to \cdots \to \mathfrak{l}_m$ to mean that for every $i \in [1, m-1]$, $\mathfrak{h}(\mathfrak{l}_i) = \mathfrak{l}_{i+1}$. In that case $\{\mathfrak{l}_1, \ldots, \mathfrak{l}_{m-1}\} \subseteq \mathrm{dom}(\mathfrak{h})$. We write $\widetilde{\sharp}\mathfrak{l}$ to denote the cardinal of the set $\{\mathfrak{l}' \in \mathbb{N} : \mathfrak{h}(\mathfrak{l}') = \mathfrak{l}\}$ made of *predecessors* of $\mathfrak{l}$ (heap $\mathfrak{h}$ is implicit in the expression $\widetilde{\sharp}\mathfrak{l}$) and $\widetilde{\sharp}\mathfrak{l}^{\star}$ to denote the cardinal of $(\{\mathfrak{l}' \in \mathbb{N} : \mathfrak{h}(\mathfrak{l}') = \mathfrak{l}\} \setminus \{\mathfrak{l}\})$. So, $\widetilde{\sharp}\mathfrak{l} = \widetilde{\sharp}\mathfrak{l}^{\star}$ precisely when either $\mathfrak{l} \notin \mathrm{dom}(\mathfrak{h})$ or $\mathfrak{h}(\mathfrak{l}) \neq \mathfrak{l}$ (otherwise $\widetilde{\sharp}\mathfrak{l}^{\star} = \widetilde{\sharp}\mathfrak{l} - 1$ and $\mathfrak{h}(\mathfrak{l}) = \mathfrak{l}$). Most probably, the notation '$\sharp\mathtt{pred}(\mathfrak{l}, \mathfrak{h})$' is more suggestive of its meaning than $\widetilde{\sharp}\mathfrak{l}$ since it contains '$\mathtt{pred}$' by reference to the predecessors of $\mathfrak{l}$ and the heap $\mathfrak{h}$ is made explicit. However, we believe that '$\sharp\mathtt{pred}(\mathfrak{l}, \mathfrak{h})$' is a bit lengthy, especially when it is used in more complex expressions. The same applies for the alternative notation '$\sharp\mathtt{pred}^{\setminus\mathfrak{l}}(\mathfrak{l}, \mathfrak{h})$' instead of $\widetilde{\sharp}\mathfrak{l}^{\star}$.

Usually, in models for separation logic(s), memory states have a heap and also a store, for interpreting program variables (see e.g. [Reynolds 2002]). Herein, there is no need for program variables; we establish expressiveness results without their help. However, an adaption with program variables is presented in Section 6.

Let $\mathtt{FVAR} = \{\mathtt{u}_1, \mathtt{u}_2, \ldots\}$ be a countably infinite set of variables. Formulae of 1SL are defined by the abstract grammar below:

$$\phi \quad ::= \quad \mathtt{u}_i = \mathtt{u}_j \mid \mathtt{u}_i \hookrightarrow \mathtt{u}_j \mid \phi \wedge \phi \mid \neg\phi \mid \phi * \phi \mid \phi \mathit{-\!\!*} \phi \mid \exists\, \mathtt{u}_i\ \phi.$$

The connective $*$ is called the *separating conjunction* and the connective $\mathit{-\!\!*}$ is called the *separating implication* (also known as the *magic wand*). We make use of standard definitions to derive other standard operations ($\forall$, $\vee$, $\Rightarrow$, $\neq$, etc.).

A *valuation* is a map $\mathfrak{f}$ of the form $\mathtt{FVAR} \to \mathbb{N}$. The satisfaction relation $\models$ is parameterized by valuations and is defined as follows (Boolean clauses are omitted):

— $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{u}_i = \mathtt{u}_j$ iff $\mathfrak{f}(\mathtt{u}_i) = \mathfrak{f}(\mathtt{u}_j)$.
— $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{u}_i \hookrightarrow \mathtt{u}_j$ iff $\mathfrak{f}(\mathtt{u}_i) \in \mathrm{dom}(\mathfrak{h})$ and $\mathfrak{h}(\mathfrak{f}(\mathtt{u}_i)) = \mathfrak{f}(\mathtt{u}_j)$.
— $\mathfrak{h} \models_{\mathfrak{f}} \phi_1 * \phi_2$ iff there exist $\mathfrak{h}_1, \mathfrak{h}_2$ such that $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are disjoint, $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$, $\mathfrak{h}_1 \models_{\mathfrak{f}} \phi_1$ and $\mathfrak{h}_2 \models_{\mathfrak{f}} \phi_2$.
— $\mathfrak{h} \models_{\mathfrak{f}} \phi_1 \mathit{-\!\!*} \phi_2$ iff for all $\mathfrak{h}'$, if $\mathfrak{h}$ and $\mathfrak{h}'$ are disjoint, and $\mathfrak{h}' \models_{\mathfrak{f}} \phi_1$ then $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \phi_2$.
— $\mathfrak{h} \models_{\mathfrak{f}} \exists\, \mathtt{u}_i\ \phi$ iff there exists $\mathfrak{l} \in \mathbb{N}$ such that $\mathfrak{h} \models_{\mathfrak{f}[\mathtt{u}_i \mapsto \mathfrak{l}]} \phi$ where $\mathfrak{f}[\mathtt{u}_i \mapsto \mathfrak{l}]$ refers to a map equal to $\mathfrak{f}$ except that $\mathtt{u}_i$ takes the value $\mathfrak{l}$.

We also introduce the so-called *septraction* operator $\mathit{-\!\!\!\!\!\!\rightarrow\!\!\!*}$: $\phi \mathit{-\!\!\!\!\!\!\rightarrow\!\!\!*} \psi$ is defined as the formula $\neg(\phi \mathit{-\!\!*} \neg\psi)$ [Vafeiadis and Parkinson 2007]. So, $\mathfrak{h} \models_{\mathfrak{f}} \phi \mathit{-\!\!\!\!\!\!\rightarrow\!\!\!*} \psi$ iff there exists $\mathfrak{h}'$, disjoint from $\mathfrak{h}$, such that $\mathfrak{h}' \models_{\mathfrak{f}} \phi$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \psi$. The septraction operator states

the existence of a disjoint heap satisfying a formula and for which its addition to the original heap satisfies another formula. Note that the magic wand makes a universal statement about (disjoint) additions to the heap, septraction an existential one.

For every $i \geq 1$, 1SLi denotes the fragment of 1SL restricted to i variables and 1SL$i(\mathbin{-\!\!*})$ denotes its restriction when separating conjunction is disallowed. Let $\mathfrak{L}$ be a logic among 1SL, 1SLi, 1SL$i(\mathbin{-\!\!*})$. The *satisfiability problem* for $\mathfrak{L}$ takes as input a sentence ϕ from $\mathfrak{L}$ and asks whether there is a heap $\mathfrak{h}$ such that $\mathfrak{h} \models \phi$ (regardless of valuation, as ϕ has no free variables).

THEOREM 2.1. *[Brochenin et al. 2012; Demri and Deters 2015] The satisfiability problem for 1SL is undecidable, even if restricted to two individual variables (1SL2).*

2.2. Weak second-order logic (WSOL)

In order to define formulae in WSOL, we consider a family $\mathtt{SVAR} = (\mathtt{SVAR}_i)_{i \geq 1}$ of second-order variables, denoted by $\mathtt{P}, \mathtt{Q}, \mathtt{R}, \ldots$ and interpreted as finite relations over $\mathbb{N}$. Each variable in $\mathtt{SVAR}_i$ is interpreted as an i-ary relation. A second-order valuation $\mathfrak{f}$ is an interpretation of the second-order variables such that for every $\mathtt{P} \in \mathtt{SVAR}_i$, $\mathfrak{f}(\mathtt{P})$ is a finite subset of $\mathbb{N}^i$.

Formulae of WSOL are defined by the grammar below:

$$\phi \quad ::= \quad \mathtt{u}_i = \mathtt{u}_j \mid \mathtt{u}_i \hookrightarrow \mathtt{u}_j \mid \phi \wedge \phi \mid \neg\phi \mid \exists\, \mathtt{u}_i\ \phi \mid \exists\, \mathtt{P}\ \phi \mid \mathtt{P}(\mathtt{u}_1, \ldots, \mathtt{u}_n)$$

where $\mathtt{u}_i, \mathtt{u}_j$, $\mathtt{u}_1$, …, $\mathtt{u}_n$ are first-order variables and the $\mathtt{P}$ are second-order variables with $\mathtt{P} \in \mathtt{SVAR}_n$ for some $n \geq 1$. We write DSOL (dyadic second-order logic) to denote the restriction of WSOL to second-order variables in $\mathtt{SVAR}_2$. Like 1SL, models for WSOL are finite heaps and quantification is performed over all possible locations. The satisfaction relation $\models$ is defined as follows ($\mathfrak{f}$ is a hybrid valuation providing interpretation for both first-order and second-order variables):

— $\mathfrak{h} \models_{\mathfrak{f}} \exists\, \mathtt{P}\ \phi$ iff there exists a *finite* relation $R \subseteq \mathbb{N}^n$ such that $\mathfrak{h} \models_{\mathfrak{f}[\mathtt{P} \mapsto R]} \phi$ where $\mathtt{P} \in \mathtt{SVAR}_n$.
— $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{P}(\mathtt{u}_1, \ldots, \mathtt{u}_n)$ iff $(\mathfrak{f}(\mathtt{u}_1), \ldots, \mathfrak{f}(\mathtt{u}_n)) \in \mathfrak{f}(\mathtt{P})$.

The satisfiability problem for WSOL takes as input a sentence ϕ in WSOL and asks whether there is a heap $\mathfrak{h}$ such that $\mathfrak{h} \models \phi$. By Trakhtenbrot's Theorem (see e.g. [Trakhtenbrot 1963; Börger et al. 1997]), the satisfiability problem for DSOL (and therefore also for WSOL) is undecidable since finite satisfiability for first-order logic with a single binary relation symbol is undecidable. Note that a monadic second-order variable can be simulated by a binary second-order variable from $\mathtt{SVAR}_2$, and this can be used to relativize a formula from DSOL in order to check finite satisfiability.

THEOREM 2.2. *[Brochenin et al. 2012] 1SL, WSOL and DSOL have the same expressive power.*

Consequently, in order to show that 1SL2$(\mathbin{-\!\!*})$ is as expressive as WSOL (our main result), it is sufficient to prove that every sentence from DSOL has an equivalent sentence in 1SL2$(\mathbin{-\!\!*})$. It is worth noting that Theorem 2.2 can be extended to the case with $k > 1$ record fields [Brochenin et al. 2012], which actually requires a simpler proof. A similar adaptation is possible from our main result (see Section 6).

3. EXPRESSING PROPERTIES IN 1SL2$(\mathbin{-\!\!*})$

In the following, let $\mathtt{u}$ and $\overline{\mathtt{u}}$ be the variables $\mathtt{u}_1$ and $\mathtt{u}_2$, in either order. Throughout this article, we build formulae with the quantified variables $\mathtt{u}$ and $\overline{\mathtt{u}}$. Note that any formula $\phi(\mathtt{u})$ with free variable $\mathtt{u}$ can be turned into an equivalent formula with free variable $\overline{\mathtt{u}}$ by permuting the two variables.

One of the first challenges we face in tracking information with 1SL2($-\!\!*$) is that of remembering references to memory cells. With only two variables, expressing simple properties of the heap becomes difficult. If we care to express the property *the domain of the heap is a singleton*, there is no trouble; we can write

$$\exists\, \mathtt{u}\ \left((\exists\, \overline{\mathtt{u}}\ \mathtt{u} \hookrightarrow \overline{\mathtt{u}}) \wedge \forall\, \overline{\mathtt{u}}\ \ (\overline{\mathtt{u}} \neq \mathtt{u} \Rightarrow \neg\, (\exists\, \mathtt{u}\ \overline{\mathtt{u}} \hookrightarrow \mathtt{u}))\right).$$

However, to express that the domain contains exactly two locations, we run into trouble. With additional variables, this would not be a problem, as we would have the ability to refer to these locations at the same time; similarly, with the separating conjunction $*$, there would be no difficulty, as we could express our simpler property twice on two disjoint subheaps, each of which must then contain a single memory cell. But with the severe syntactic restriction of 1SL2($-\!\!*$), a new method is needed.

In this section, we propose a new and natural method to compare a number of predecessors against a constant, and also a way to express a reachability property.

3.1. Warming up with basic properties

Let us begin by defining simple, standard formulae. These are easily seen to be correct.

— $\mathtt{u}$ has a successor in the heap (equivalently, we say it is *allocated*):

$$\mathtt{alloc}(\mathtt{u}) \stackrel{\text{def}}{=} \exists\, \overline{\mathtt{u}}\ \mathtt{u} \hookrightarrow \overline{\mathtt{u}}.$$

— $\mathtt{u}$ is an *isolated location,* that is, it is not in $\mathrm{dom}(\mathfrak{h})$, nor is it in $\mathrm{ran}(\mathfrak{h})$:

$$\mathtt{isoloc}(\mathtt{u}) \stackrel{\text{def}}{=} \neg\mathtt{alloc}(\mathtt{u}) \wedge \neg\exists\, \overline{\mathtt{u}}\ \overline{\mathtt{u}} \hookrightarrow \mathtt{u}.$$

— $\mathrm{dom}(\mathfrak{h})$ has exactly one location:

$$(\mathtt{size} = 1) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \big(\mathtt{alloc}(\mathtt{u}) \wedge \forall\, \overline{\mathtt{u}}\ \ (\mathtt{u} \neq \overline{\mathtt{u}} \Rightarrow \neg\mathtt{alloc}(\overline{\mathtt{u}}))\big).$$

Using the separating conjunction $*$, it is easy to define the formula $(\mathtt{size} = k)$ stating that $\mathrm{dom}(\mathfrak{h})$ has exactly k locations ($k > 1$).

— $\mathtt{u}$ has at least one predecessor:

$$\sharp\mathtt{u} > 0 \stackrel{\text{def}}{=} \exists\, \overline{\mathtt{u}}\ \overline{\mathtt{u}} \hookrightarrow \mathtt{u}.$$

Naturally, since the number of predecessors is nonnegative, we can also write:

$$\sharp\mathtt{u} = 0 \stackrel{\text{def}}{=} \neg(\sharp\mathtt{u} > 0).$$

Let us now proceed with more complicated constructions.

3.2. Counting z-predecessors

In this paper, we will make heavy use of counting predecessors of memory locations. This is critical to many of the technical developments of this work; without an adequate number of variables to refer to locations of interest, we instead "remember" locations by installing a large (and uniquely identifiable) number of predecessors to a location in the heap. At another point in the formula, we can identify this location and operate on it, having not used any of the logic's limited syntactic resources.

As one may imagine, counting and comparing numbers of predecessors is difficult to do in this logic. We first build up a method of counting a certain type of predecessor, then use it to formulate more involved constructions, and finally introduce a way to count the full number of predecessors of a location—that is, to compare its number of predecessors with some given $k \geq 0$. After this is achieved, nearly the entirety of Section 4 extends this to comparing numbers of predecessors between two locations (rather than just a comparison to some given k).

With that in mind, we first introduce the notion of *z-predecessors*, which are predecessors of a location that themselves have no predecessors ('z' is for 'zero'). We can then write:

— $\mathtt{u}$'s predecessors (if it has any) are all z-predecessors:

$$\mathtt{allzpred}(\mathtt{u}) \stackrel{\text{def}}{=} \forall\, \overline{\mathtt{u}}\ \overline{\mathtt{u}} \hookrightarrow \mathtt{u} \Rightarrow \sharp\overline{\mathtt{u}} = 0.$$

— $\mathtt{u}$ has no z-predecessor:

$$\sharp_z\mathtt{u} = 0 \stackrel{\text{def}}{=} \forall\, \overline{\mathtt{u}}\ \overline{\mathtt{u}} \hookrightarrow \mathtt{u} \Rightarrow \sharp\overline{\mathtt{u}} > 0.$$

— $\mathtt{u}$ has at most $k > 0$ z-predecessors:

$$\sharp_z\mathtt{u} \leq k \stackrel{\text{def}}{=} (\mathtt{size} = 1) \mathrel{-\!\!\!\overrightarrow{*}} (\sharp_z\mathtt{u} \leq k-1)$$

where $\sharp_z\mathtt{u} \leq 0$ is defined as $\sharp_z\mathtt{u} = 0$.

These definitions easily allow us to define $\sharp_z\mathtt{u} \bowtie k$ for every $k \in \mathbb{N}$ and $\bowtie \in \{=, <, \leq, >, \geq\}$. The following lemma establishes correctness of the above formulae and all of the derivative forms.

LEMMA 3.1. *Let $\mathfrak{h}$ be a heap, $\mathfrak{f}$ be a valuation, and $k \in \mathbb{N}$. We have $\mathfrak{h} \models_{\mathfrak{f}} \sharp_z\mathtt{u} \leq k$ iff card$(\{\mathfrak{l} \in \mathbb{N} : \widetilde{\sharp}\mathfrak{l} = 0,\ \mathfrak{h}(\mathfrak{l}) = \mathfrak{f}(\mathtt{u})\}) \leq k$.*

PROOF. The proof is by induction on k and the base case with $k = 0$ is by an easy verification. In the induction step, assume $k > 0$ and suppose that the induction hypothesis is the following: $\mathfrak{h} \models_{\mathfrak{f}} \sharp_z\mathtt{u} \leq k'$ iff card$(\{\mathfrak{l} \in \mathbb{N} : \widetilde{\sharp}\mathfrak{l} = 0,\ \mathfrak{h}(\mathfrak{l}) = \mathfrak{f}(\mathtt{u})\}) \leq k'$ for any $k' < k$. In particular this holds for $k' = k - 1$.

($\Leftarrow$) First, suppose that card$(\{\mathfrak{l} \in \mathbb{N} : \widetilde{\sharp}\mathfrak{l} = 0,\ \mathfrak{h}(\mathfrak{l}) = \mathfrak{f}(\mathtt{u})\}) \leq k$.

Case 1: card$(\{\mathfrak{l} \in \mathbb{N} : \widetilde{\sharp}\mathfrak{l} = 0,\ \mathfrak{h}(\mathfrak{l}) = \mathfrak{f}(\mathtt{u})\}) \leq k-1$. Let $\mathfrak{h}'$ be the heap with singleton domain such that $\mathfrak{h}'(\mathfrak{l}) = \mathfrak{l}$ for some $\mathfrak{l} \notin (\mathrm{dom}(\mathfrak{h}) \cup \mathrm{ran}(\mathfrak{h}) \cup \{\mathfrak{f}(\mathtt{u})\})$. Because $\mathfrak{l}$ is quite isolated, we also have that card$(\{\mathfrak{l} \in \mathbb{N} : \widetilde{\sharp}\mathfrak{l} = 0,\ (\mathfrak{h} \uplus \mathfrak{h}')(\mathfrak{l}) = \mathfrak{f}(\mathtt{u})\}) \leq k-1$ and $\mathfrak{h}' \models_{\mathfrak{f}} (\mathtt{size} = 1)$. By (IH), $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \sharp_z\mathtt{u} \leq k-1$. Consequently, by definition of $\models$, we get $\mathfrak{h} \models_{\mathfrak{f}} (\mathtt{size} = 1) \mathrel{-\!\!\!\overrightarrow{*}} \sharp_z\mathtt{u} \leq k-1$, that is $\mathfrak{h} \models_{\mathfrak{f}} \sharp_z\mathtt{u} \leq k$.

Case 2: card$(\{\mathfrak{l} \in \mathbb{N} : \widetilde{\sharp}\mathfrak{l} = 0,\ \mathfrak{h}(\mathfrak{l}) = \mathfrak{f}(\mathtt{u})\}) = k$. Let $\mathfrak{l}_0$ be in $\{\mathfrak{l} \in \mathbb{N} : \widetilde{\sharp}\mathfrak{l} = 0,\ \mathfrak{h}(\mathfrak{l}) = \mathfrak{f}(\mathtt{u})\}$ and $\mathfrak{l}$ be a location not in $(\mathrm{dom}(\mathfrak{h}) \cup \mathrm{ran}(\mathfrak{h}) \cup \{\mathfrak{f}(\mathtt{u})\})$. Let $\mathfrak{h}'$ be the heap with singleton domain such that $\mathfrak{h}'(\mathfrak{l}) = \mathfrak{l}_0$. In $\mathfrak{h} \uplus \mathfrak{h}'$, $\mathfrak{f}(\mathtt{u})$ has one less z-predecessor, so card$(\{\mathfrak{l} \in \mathbb{N} : \widetilde{\sharp}\mathfrak{l} = 0,\ (\mathfrak{h} \uplus \mathfrak{h}')(\mathfrak{l}) = \mathfrak{f}(\mathtt{u})\}) = k-1$ and $\mathfrak{h}' \models_{\mathfrak{f}} (\mathtt{size} = 1)$. By (IH), $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \sharp_z\mathtt{u} \leq k-1$. Consequently, by definition of $\models$, we obtain $\mathfrak{h} \models_{\mathfrak{f}} (\mathtt{size} = 1) \mathrel{-\!\!\!\overrightarrow{*}} \sharp_z\mathtt{u} \leq k-1$, that is $\mathfrak{h} \models_{\mathfrak{f}} \sharp_z\mathtt{u} \leq k$.

($\Rightarrow$) Now suppose that $\mathfrak{h} \models_{\mathfrak{f}} (\mathtt{size} = 1) \mathrel{-\!\!\!\overrightarrow{*}} \sharp_z\mathtt{u} \leq k-1$. There is a heap $\mathfrak{h}'$ disjoint from $\mathfrak{h}$ such that card$(\mathrm{dom}(\mathfrak{h}')) = 1$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \sharp_z\mathtt{u} \leq k-1$. By (IH), $\mathfrak{f}(\mathtt{u})$ has at most $(k-1)$ z-predecessors in $\mathfrak{h} \uplus \mathfrak{h}'$. By removing $\mathfrak{h}'$ from $\mathfrak{h} \uplus \mathfrak{h}'$, one can augment the number of z-predecessors of $\mathfrak{f}(\mathtt{u})$ by at most one (since card$(\mathrm{dom}(\mathfrak{h}')) = 1$). So, $\mathfrak{f}(\mathtt{u})$ has at most k z-predecessors in $\mathfrak{h}$. □

3.3. A matter of knives and forks

To address the problem of referring to many distinct memory locations despite not having sufficient variables to do so, we introduce the notion of *forks*. Forks are simple, recognizable shapes that we add to the heap with the magic wand. The forks can then be found at another point, "deeper" in the formula. Forks are also a critical building block for comparing predecessor counts in Section 4.

A *fork* in $\mathfrak{h}$ is a sequence of distinct locations $\mathfrak{l}, \mathfrak{l}_0, \mathfrak{l}_1, \mathfrak{l}_2$ such that $\mathfrak{h}(\mathfrak{l}_0) = \mathfrak{l}$, $\widetilde{\sharp \mathfrak{l}_0} = 2$, $\mathfrak{h}(\mathfrak{l}_1) = \mathfrak{h}(\mathfrak{l}_2) = \mathfrak{l}_0$ and $\widetilde{\sharp \mathfrak{l}_1} = \widetilde{\sharp \mathfrak{l}_2} = 0$. The *endpoint* of the fork is $\mathfrak{l}$, and its *midpoint* is $\mathfrak{l}_0$. The fork is *isolated* iff $\widetilde{\sharp \mathfrak{l}} = 1$ and $\mathfrak{l} \notin \mathrm{dom}(\mathfrak{h})$. Similarly, a *knife* in $\mathfrak{h}$ is a sequence of distinct locations $\mathfrak{l}, \mathfrak{l}_0, \mathfrak{l}_1$ such that $\mathfrak{h}(\mathfrak{l}_0) = \mathfrak{l}$, $\widetilde{\sharp \mathfrak{l}_0} = 1$, $\mathfrak{h}(\mathfrak{l}_1) = \mathfrak{l}_0$ and $\widetilde{\sharp \mathfrak{l}_1} = 0$. The *endpoint* of the knife is $\mathfrak{l}$, and the *midpoint* of the knife is $\mathfrak{l}_0$. The knife is *isolated* iff $\widetilde{\sharp \mathfrak{l}} = 1$ and $\mathfrak{l} \notin \mathrm{dom}(\mathfrak{h})$.

By way of example, the heap represented in Figure 2 contains three knives, two forks and four endpoints (identified by '$\star$'). Of these, one of the depicted forks and one of the knives are isolated. The locations participating in the "lasso" shape, at the right of the figure, are not part of any knife or fork.

To identify fork and knife endpoints in a heap, we define the following formulae:

$$\begin{aligned} \mathtt{forkendpt}(\mathtt{u}) &\stackrel{\text{def}}{=} \exists\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u} \wedge \sharp_z \overline{\mathtt{u}} = 2 \wedge \mathtt{allzpred}(\overline{\mathtt{u}})) \\ \mathtt{knifeendpt}(\mathtt{u}) &\stackrel{\text{def}}{=} \exists\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u} \wedge \sharp_z \overline{\mathtt{u}} = 1 \wedge \mathtt{allzpred}(\overline{\mathtt{u}}))\,. \end{aligned}$$

Now, let $\mathtt{forky}(\mathtt{u})$ be a formula stating that *all* predecessors of $\mathfrak{f}(\mathtt{u})$, possibly excepting $\mathfrak{f}(\mathtt{u})$ itself, are endpoints of forks:

$$\mathtt{forky}(\mathtt{u}) \stackrel{\text{def}}{=} \forall\, \overline{\mathtt{u}}\ ((\overline{\mathtt{u}} \hookrightarrow \mathtt{u} \wedge \overline{\mathtt{u}} \neq \mathtt{u}) \Rightarrow \mathtt{forkendpt}(\overline{\mathtt{u}}))\,.$$

Three forks, two endpoints, and a forky location are depicted in Figure 3.

Next, let $\mathtt{antiforky}(\mathtt{u})$ be a formula stating that *no* predecessor of $\mathfrak{f}(\mathtt{u})$ is the endpoint of a fork, and let $\mathtt{antiknify}(\mathtt{u})$ be a formula stating that *no* predecessor of $\mathfrak{f}(\mathtt{u})$ is the endpoint of a knife. We define these as

$$\begin{aligned} \mathtt{antiforky}(\mathtt{u}) &\stackrel{\text{def}}{=} \forall\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u} \Rightarrow \neg \mathtt{forkendpt}(\overline{\mathtt{u}})) \\ \mathtt{antiknify}(\mathtt{u}) &\stackrel{\text{def}}{=} \forall\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u} \Rightarrow \neg \mathtt{knifeendpt}(\overline{\mathtt{u}}))\,. \end{aligned}$$

Note the asymmetry between $\mathtt{forky}(\mathtt{u})$ and $\mathtt{antiforky}(\mathtt{u})$: $\mathfrak{f}(\mathtt{u})$ does not have to be the endpoint of a fork for $\mathtt{forky}(\mathtt{u})$ to hold (which would be then impossible to realize for all the predecessors of $\mathfrak{f}(\mathtt{u})$ if $\mathfrak{f}(\mathtt{u})$ were a self-loop, assuming that the number of predecessors of $\mathfrak{f}(\mathtt{u})$ remains constant). It is also easy to enforce that the heap is made of a single fork, which will be useful in later constructions.

LEMMA 3.2. *There exists a formula* $\mathtt{1fork}$ *in 1SL2(*$\mathrel{-\!\!*}$*) such that for all heaps* $\mathfrak{h}$*, we have* $\mathfrak{h} \models \mathtt{1fork}$ *iff* $\mathfrak{h}$ *is made of a single, isolated fork (and nothing else).*

PROOF. In a heap $\mathfrak{h}$, a *trident location*, in short a *tril*, is a location $\mathfrak{l}$ such that

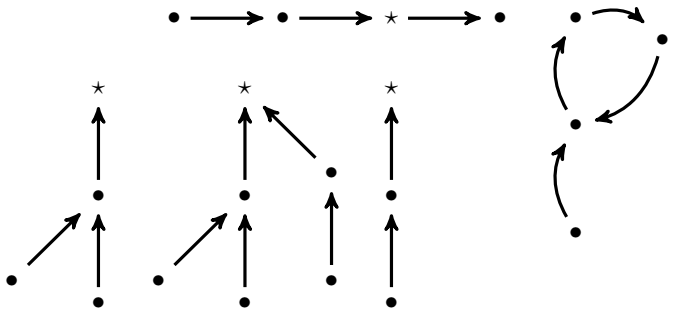


Fig. 2: A heap with three knives, two forks, and four distinct endpoints (marked '$\star$').

(1) $\mathfrak{l} \in \mathrm{dom}(\mathfrak{h})$,
(2) $\mathfrak{l}$ has exactly two predecessors and none of them has a predecessor, and
(3) $\mathfrak{h}(\mathfrak{l}) \notin \mathrm{dom}(\mathfrak{h})$.

So, a tril has exactly two predecessors that are both z-predecessors, and it points to the endpoint of a fork. Therefore it belongs to a fork, but it may not be isolated (the endpoint may have more than one predecessor).

Let $\mathtt{tril}(\mathtt{u})$ be defined as the following 1SL2($-\!\!*$) formula:

$$\mathtt{tril}(\mathtt{u}) \stackrel{\text{def}}{=} \sharp_z \mathtt{u} = 2 \wedge \mathtt{allzpred}(\mathtt{u}) \wedge \exists\, \overline{\mathtt{u}}\ (\mathtt{u} \hookrightarrow \overline{\mathtt{u}} \wedge \neg \mathtt{alloc}(\overline{\mathtt{u}})).$$

It is easy to show that $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{tril}(\mathtt{u})$ iff $\mathfrak{f}(\mathtt{u})$ is a tril. Moreover the following property can be easily established:

—. The heap $\mathfrak{h}$ is made of a single, isolated fork (and nothing else) iff (1) $\mathfrak{h}$ contains a unique tril, and (2) for all locations $\mathfrak{l} \in \mathrm{dom}(\mathfrak{h})$, either $\mathfrak{l}$ is a tril or $\mathfrak{h}(\mathfrak{l})$ is a tril.

Let `1fork` be the formula below, which jointly expresses the properties (1) and (2) above.

$$\mathtt{1fork} \stackrel{\text{def}}{=} \overbrace{\left(\exists\, \mathtt{u}\ \mathtt{tril}(\mathtt{u}) \wedge \forall\, \overline{\mathtt{u}}\ (\mathtt{u} \neq \overline{\mathtt{u}} \Rightarrow \neg \mathtt{tril}(\overline{\mathtt{u}}))\right)}^{\text{property (1)}} \wedge$$

$$\underbrace{\forall\, \mathtt{u}\ \left(\mathtt{alloc}(\mathtt{u}) \Rightarrow (\mathtt{tril}(\mathtt{u}) \vee (\exists\, \overline{\mathtt{u}}\ (\mathtt{u} \hookrightarrow \overline{\mathtt{u}}) \wedge \mathtt{tril}(\overline{\mathtt{u}})))\right)}_{\text{property (2)}}.$$

Clearly, then, $\mathfrak{h} \models \mathtt{1fork}$ iff $\mathfrak{h}$ is made of a single, isolated fork (and nothing else). □

3.4. Counting predecessors

We now have the necessary developments to build formulae that constrain the total number of predecessors (not just z-predecessors) of a location: with $k \in \mathbb{N}$, we define

$$\sharp \mathtt{u} \leq k \stackrel{\text{def}}{=} \begin{cases} \neg \exists\, \overline{\mathtt{u}}\ \overline{\mathtt{u}} \hookrightarrow \mathtt{u} & k = 0 \\ (\mathtt{u} \hookrightarrow \mathtt{u} \wedge \overset{\star}{\sharp} \mathtt{u} \leq k-1) \vee (\neg(\mathtt{u} \hookrightarrow \mathtt{u}) \wedge \overset{\star}{\sharp} \mathtt{u} \leq k) & k > 0 \end{cases}$$

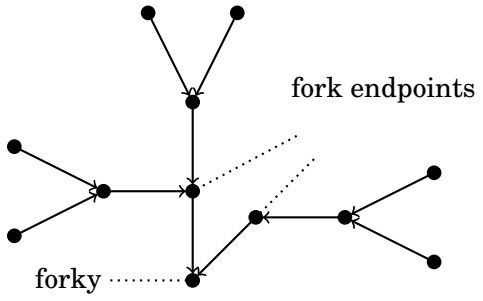


Fig. 3: Forky locations.

where

$$\overset{\star}{\sharp}\mathtt{u} \leq k \stackrel{\text{def}}{=} \begin{cases} \neg\exists\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u} \wedge \overline{\mathtt{u}} \neq \mathtt{u}) & \text{if } k = 0 \\ (\sharp\mathtt{u} = 0) \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!*} (\mathtt{antiforky}(\mathtt{u}) \wedge (\underbrace{\mathtt{1fork} \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!*} \cdots \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!*} \mathtt{1fork}}_{k \text{ times}} \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!*} \mathtt{forky}(\mathtt{u})) & k > 0 \end{cases}$$

In a nutshell, $\mathfrak{f}(\mathtt{u})$ has at most $k > 0$ predecessors if one can make $\mathfrak{f}(\mathtt{u})$ antiforky without changing its predecessor count (which is always possible) and then adding k forks to the heap to make $\mathfrak{f}(\mathtt{u})$ forky (we distinguish the case when $\mathfrak{f}(\mathtt{u})$ is a self-loop).

In Figure 4, we present three heaps; the leftmost heap is the original heap. The heap in the middle is obtained from the leftmost heap by destroying forks pointing to predecessors of the location $\star$ (just add memory cells to destroy the fork shapes). The rightmost heap is obtained from the heap in the middle by adding two forks pointing to predecessors of the location $\star$, except for the predecessor equal to the location $\star$. This amounts to check the satisfaction of $\overset{\star}{\sharp}\mathtt{u} \leq 2$.

LEMMA 3.3. *Let $k \in \mathbb{N}$, $\mathfrak{h}$ be a heap and $\mathfrak{f}$ be a valuation. (I) $\mathfrak{h} \models_{\mathfrak{f}} \overset{\star}{\sharp}\mathtt{u} \leq k$ iff $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} \leq k$. (II) $\mathfrak{h} \models_{\mathfrak{f}} \sharp\mathtt{u} \leq k$ iff $\widetilde{\sharp\mathfrak{f}(\mathtt{u})} \leq k$.*

PROOF. With $k = 0$, the proof is by an easy verification.

With $k \geq 1$, let us start by giving a few definitions that are useful for the proof.

First, we define a family $(\psi_k)_{k\in\mathbb{N}}$ of formulae in 1SL2($\mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!*}$) so that $\psi_0 \stackrel{\text{def}}{=} \mathtt{forky}(\mathtt{u})$ and $\psi_{k+1} \stackrel{\text{def}}{=} \mathtt{1fork} \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!*} \psi_k$. For all formulae in $(\psi_k)_{k\in\mathbb{N}}$, the variable $\mathtt{u}$ is free. Each formula $\sharp\mathtt{u} \leq k$ with $k \geq 1$ is thus equal to

$$\Big((\mathtt{u} \hookrightarrow \mathtt{u}) \wedge \big((\sharp\mathtt{u} = 0) \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!*} (\mathtt{antiforky}(\mathtt{u}) \wedge \psi_{k-1})\big)\Big) \ \vee \\ \Big(\neg(\mathtt{u} \hookrightarrow \mathtt{u}) \wedge \big((\sharp\mathtt{u} = 0) \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!\!\!\!*} (\mathtt{antiforky}(\mathtt{u}) \wedge \psi_k)\big)\Big).$$

Note that in the first disjunct, requiring that $\mathtt{u} \hookrightarrow \mathtt{u}$ holds is not incompatible with the possibility to add a disjoint heap such that $\sharp\mathtt{u} = 0$ holds (thanks to disjointness).

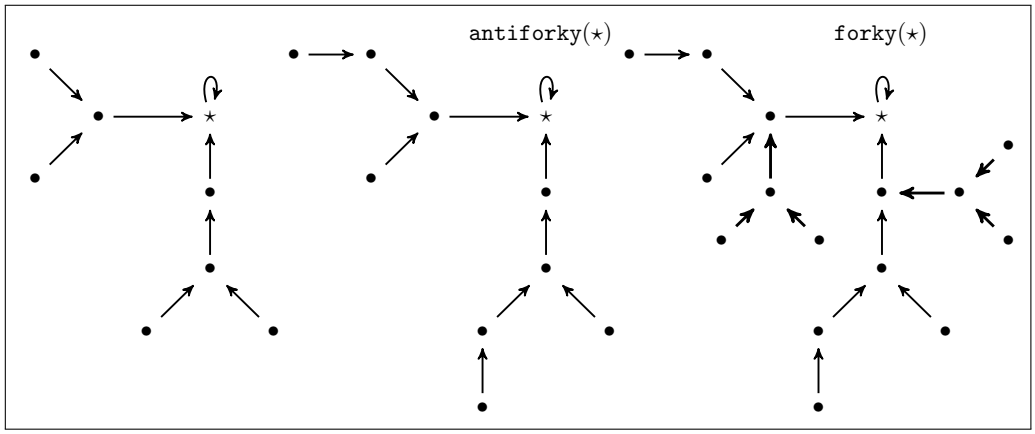


Fig. 4: Removing and adding forks.

Similarly, each formula $\overset{\star}{\sharp}\mathtt{u} \leq k$ with $k \geq 1$ is equal to

$$(\sharp\mathtt{u} = 0) \mathrel{-\!\!\!\!\!\rightarrow\!\!\!\!\!*} (\mathtt{antiforky}(\mathtt{u}) \wedge \psi_k).$$

Given a heap $\mathfrak{h}$ and a location $\mathfrak{l}$, we write $\mathfrak{pne}(\mathfrak{l})$ to denote the number of predecessors of $\mathfrak{l}$ in $\mathfrak{h}$ that are not endpoints of some forks in $\mathfrak{h}$ and that, further, are different from $\mathfrak{l}$. In particular, for all $\mathfrak{l} \in \mathbb{N}$, $\mathfrak{pne}(\mathfrak{l}) \leq \widetilde{\sharp}\mathfrak{l}^{\star}$, and also $\mathfrak{h} \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \mathtt{forky}(\mathtt{u})$ iff $\mathfrak{pne}(\mathfrak{l}) = 0$. We will establish the following property for all $k \geq 0$:

($\mathcal{E}$). For all $\mathfrak{l} \in \mathbb{N}$, for all heaps $\mathfrak{h}$, we have $\mathfrak{h} \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \psi_k$ iff $\mathfrak{pne}(\mathfrak{l}) \leq k$.

For the base case ($k = 0$), we have $\mathfrak{h} \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \psi_0$ iff $\mathfrak{h} \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \mathtt{forky}(\mathtt{u})$ (by definition of ψ_0) iff all the predecessors of $\mathfrak{l}$ (possibly with the exception of $\mathfrak{l}$) are endpoints of some fork (by the property characterizing the formula $\mathtt{forky}(\mathtt{u})$) iff $\mathfrak{pne}(\mathfrak{l}) = 0$ (by definition of $\mathfrak{pne}(\cdot)$) iff $\mathfrak{pne}(\mathfrak{l}) \leq 0$ ($\mathfrak{pne}(\mathfrak{l})$ is always non-negative).

For the induction step, let us assume that $\mathfrak{pne}(\mathfrak{l}) \leq k + 1$.

Case 1: $\mathfrak{pne}(\mathfrak{l}) \leq k$. Let $\mathfrak{h}'$ be some disjoint heap such that $\mathfrak{h}' \models \mathtt{1fork}$ and the endpoint of the unique fork in $\mathfrak{h}'$ is not in $(\mathrm{dom}(\mathfrak{h}) \cup \mathrm{ran}(\mathfrak{h}) \cup \{\mathfrak{l}\})$. It is always possible to construct such an $\mathfrak{h}'$ since $\mathfrak{h}$ is a finite structure. Still, $\mathfrak{pne}(\mathfrak{l}) \leq k$ in $\mathfrak{h} \uplus \mathfrak{h}'$ and therefore by (IH), we have $\mathfrak{h} \uplus \mathfrak{h}' \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \psi_k$. Consequently, there is a disjoint heap $\mathfrak{h}'$ such that $\mathfrak{h}' \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \mathtt{1fork}$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \psi_k$, whence $\mathfrak{h} \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \psi_{k+1}$.

Case 2: $\mathfrak{pne}(\mathfrak{l}) = k + 1$. Let $\mathfrak{l}'$ be some location such that $\mathfrak{h}(\mathfrak{l}') = \mathfrak{l}$ and $\mathfrak{l}' \neq \mathfrak{l}$. Let $\mathfrak{h}'$ be some disjoint heap such that $\mathfrak{h}' \models \mathtt{1fork}$ and the endpoint of the unique fork in $\mathfrak{h}'$ is precisely $\mathfrak{l}'$. Again, it is always possible to construct such an $\mathfrak{h}'$. So, $\mathfrak{pne}(\mathfrak{l}) = k$ in $\mathfrak{h} \uplus \mathfrak{h}'$ and therefore by (IH), we have $\mathfrak{h} \uplus \mathfrak{h}' \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \psi_k$. As for Case 1, we can conclude that $\mathfrak{h} \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \psi_{k+1}$.

Now assume that $\mathfrak{h} \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \psi_{k+1}$, i.e., $\mathfrak{h} \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \mathtt{1fork} \mathrel{-\!\!\!\!\!\rightarrow\!\!\!\!\!*} \psi_k$. There is a heap $\mathfrak{h}'$ disjoint from $\mathfrak{h}$ such that $\mathfrak{h}' \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \mathtt{1fork}$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \psi_k$. By (IH), $\mathfrak{pne}(\mathfrak{l})$ in $\mathfrak{h} \uplus \mathfrak{h}'$ is less than or equal to k (say it equals k'). If the endpoint of the unique fork in $\mathfrak{h}'$ is a predecessor of $\mathfrak{l}$ that is not the endpoint of some fork in $\mathfrak{h}$ and different from $\mathfrak{l}$, then $\mathfrak{pne}(\mathfrak{l})$ in $\mathfrak{h}$ is at most $k' + 1 \leq k + 1$. Otherwise, if the endpoint of the unique fork in $\mathfrak{h}'$ is not a predecessor of $\mathfrak{l}$, then $\mathfrak{pne}(\mathfrak{l})$ in $\mathfrak{h}$ is at most $k' \leq k$. In both cases, $\mathfrak{pne}(\mathfrak{l}) \leq k + 1$ in $\mathfrak{h}$. So, we have established ($\mathcal{E}$).

Now let $\mathfrak{h}$ be some heap. We have (†) $\mathfrak{h} \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} (\sharp\mathtt{u} = 0) \mathrel{-\!\!\!\!\!\rightarrow\!\!\!\!\!*} (\mathtt{antiforky}(\mathtt{u}) \wedge \psi_k))$ iff there exists a heap $\mathfrak{h}'$ disjoint from $\mathfrak{h}$ such that $\widetilde{\sharp}\mathfrak{l}$ in $\mathfrak{h}$ is equal to $\widetilde{\sharp}\mathfrak{l}$ in $\mathfrak{h} \uplus \mathfrak{h}'$, $\mathfrak{h} \uplus \mathfrak{h}' \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \mathtt{antiforky}(\mathtt{u})$ and $\mathfrak{pne}(\mathfrak{l}) \leq k$ in $\mathfrak{h} \uplus \mathfrak{h}'$. $\widetilde{\sharp}\mathfrak{l}$ in $\mathfrak{h}$ is equal to $\widetilde{\sharp}\mathfrak{l}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ (and also $\widetilde{\sharp}\mathfrak{l}^{\star}$ is the same in both heaps), and $\mathfrak{l}$ is antiforky in $\mathfrak{h} \uplus \mathfrak{h}'$, so $\mathfrak{pne}(\mathfrak{l})$ in $\mathfrak{h}$ is less than or equal to $\mathfrak{pne}(\mathfrak{l})$ in $\mathfrak{h} \uplus \mathfrak{h}'$, and $\mathfrak{pne}(\mathfrak{l}) = \widetilde{\sharp}\mathfrak{l}^{\star}$ in $\mathfrak{h} \uplus \mathfrak{h}'$. Now, (†) iff there exists a heap $\mathfrak{h}'$ disjoint from $\mathfrak{h}$ such that $\widetilde{\sharp}\mathfrak{l}$ in $\mathfrak{h}$ is equal to $\widetilde{\sharp}\mathfrak{l}$ in $\mathfrak{h} \uplus \mathfrak{h}'$, $\mathfrak{h} \uplus \mathfrak{h}' \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \mathtt{antiforky}(\mathtt{u})$, and $\widetilde{\sharp}\mathfrak{l}^{\star} \leq k$ in $\mathfrak{h} \uplus \mathfrak{h}'$. Since it is always possible to build a disjoint heap $\mathfrak{h}'$ satisfying those properties, we have that (†) is equivalent to $\widetilde{\sharp}\mathfrak{l}^{\star} \leq k$ in $\mathfrak{h}$. This establishes (I).

By performing a simple case analysis depending whether $\mathfrak{h}(\mathfrak{l}) = \mathfrak{l}$, we can show that for all heaps $\mathfrak{h}$, for all locations $\mathfrak{l}$, for all $k \geq 1$, we have $\mathfrak{h} \models_{[\mathfrak{u}\mapsto\mathfrak{l}]} \sharp\mathtt{u} \leq k$ iff $\widetilde{\sharp}\mathfrak{l} \leq k$. As a conclusion, we have established (II) as well. □

Thus we can express in 1SL2($\mathrel{-\!\!\!\!\!\rightarrow\!\!\!\!\!*}$) the following properties:

— $\mathfrak{f}(\mathtt{u})$ has at least k predecessors: $\sharp\mathtt{u} \geq k \overset{\text{def}}{=} \neg(\sharp\mathtt{u} \leq k - 1)$.
— $\mathfrak{f}(\mathtt{u})$ has exactly k predecessors: $\sharp\mathtt{u} = k \overset{\text{def}}{=} \sharp\mathtt{u} \leq k \wedge \sharp\mathtt{u} \geq k$.

Of course we can also define strict inequalities. Note that these definitions are entirely consistent with the earlier definitions of $\sharp \mathtt{u} > 0$ and $\sharp \mathtt{u} = 0$ given in Section 3.1. Furthermore, all the difficulties to define $\sharp \mathtt{u} \geq k$ vanish if we consider the fragment 1SL2($*$); it is sufficient to consider k identical conjuncts $(\exists\ \overline{\mathtt{u}}\ \overline{\mathtt{u}} \hookrightarrow \mathtt{u})$ built with the separation conjunction.

3.5. Lonely memory cells

The formula below states that $\mathfrak{f}(\mathtt{u})$ is an *isolated cell*: it is allocated but has no predecessors, its successor has no other predecessors besides it, and its successor has itself no successor.

$$\mathtt{isocell}(\mathtt{u}) \stackrel{\text{def}}{=} \sharp \mathtt{u} = 0 \ \wedge\ \big(\exists\ \overline{\mathtt{u}}\ \mathtt{u} \hookrightarrow \overline{\mathtt{u}} \wedge \sharp \overline{\mathtt{u}} = 1 \wedge \neg \mathtt{alloc}(\overline{\mathtt{u}})\big).$$

Now, we can express a useful property of the heap itself. A heap $\mathfrak{h}$ is *segmented* whenever $\mathrm{dom}(\mathfrak{h}) \cap \mathrm{ran}(\mathfrak{h}) = \emptyset$ and no location has strictly more than one predecessor. Otherwise said, all the memory cells in $\mathfrak{h}$ are isolated. This segmentation can be naturally expressed in 1SL2($-\!\!*$):

$$\mathtt{seg} \stackrel{\text{def}}{=} \forall\ \mathtt{u}\ \forall\ \overline{\mathtt{u}}\ \big(\mathtt{u} \hookrightarrow \overline{\mathtt{u}} \Rightarrow (\sharp \overline{\mathtt{u}} = 1 \wedge \sharp \mathtt{u} = 0)\big).$$

3.6. Expressing reachability

In 1SL, reachability can be expressed, and [Demri and Deters 2015] gives a technique for doing so with the two-variable restriction 1SL2, itself a variant of material from [Dawar et al. 2007; Brochenin et al. 2012].

In 1SL2($-\!\!*$) we do not have the luxury of using the separating conjunction, but we can still specify reachability between a pair of locations—say from $\mathfrak{f}(\mathtt{u})$ to $\mathfrak{f}(\overline{\mathtt{u}})$, since our goal is to write a formula parameterized by $\mathtt{u}$ and $\overline{\mathtt{u}}$. However, we need a new technique: we first install a fork at $\mathfrak{f}(\mathtt{u})$, we propagate this fork forward in the heap, and finally we check whether $\mathfrak{f}(\overline{\mathtt{u}})$ is the endpoint of some fork. This is analogous to the way reachability is handled with a monadic second-order predicate, but here, a finite set of locations is identified by propagation of forks. We define $\mathtt{propagate}(\mathtt{u})$ as the formula characterizing the property that $\mathfrak{f}(\mathtt{u})$ is the endpoint of some fork in $\mathfrak{h}$, and that the property of being the endpoint of some fork is propagated along memory cells. The reachability predicate (for reachability from $\mathfrak{f}(\mathtt{u})$ to $\mathfrak{f}(\overline{\mathtt{u}})$) is written $\mathtt{reach}(\mathtt{u}, \overline{\mathtt{u}})$:

$$\begin{array}{lcl} \mathtt{propagate}(\mathtt{u}) & \stackrel{\text{def}}{=} & \mathtt{forkendpt}(\mathtt{u}) \wedge \forall\ \mathtt{u}\ \forall\ \overline{\mathtt{u}}\ \big((\mathtt{u} \hookrightarrow \overline{\mathtt{u}} \wedge \mathtt{forkendpt}(\mathtt{u})) \Rightarrow \mathtt{forkendpt}(\overline{\mathtt{u}})\big) \\ \mathtt{reach}(\mathtt{u}, \overline{\mathtt{u}}) & \stackrel{\text{def}}{=} & \top \mathrel{-\!\!*} \big(\mathtt{propagate}(\mathtt{u}) \Rightarrow \mathtt{forkendpt}(\overline{\mathtt{u}})\big). \end{array}$$

The purpose of the initial "$\top \mathrel{-\!\!*}$" is to enrich the original heap such that the satisfaction of the formula $\mathtt{propagate}(\mathtt{u})$ implies the satisfaction of the formula $\mathtt{forkendpt}(\overline{\mathtt{u}})$. Of course, a disjoint subheap could change the very reachability property we are testing. But some such subheaps do not, and the implicit universal quantification of the magic wand ensures that we are on the safe side: if for *all* combined heaps satisfying $\mathtt{propagate}(\mathtt{u})$, $\mathfrak{f}(\overline{\mathtt{u}})$ is the endpoint of a fork, then we can conclude that $\mathfrak{f}(\mathtt{u})$ reaches $\mathfrak{f}(\overline{\mathtt{u}})$ in the original heap.

LEMMA 3.4. *Given a heap* $\mathfrak{h}$*, a location* $\mathfrak{l}$*, and a valuation* $\mathfrak{f}$*, if* $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}]} \mathtt{propagate}(\mathtt{u})$*, then for all* $k \geq 0$*, if* $\mathfrak{h}^k(\mathfrak{l})$ *exists, then* $\mathfrak{h} \models_{[\overline{\mathtt{u}} \mapsto \mathfrak{h}^k(\mathfrak{l})]} \mathtt{forkendpt}(\overline{\mathtt{u}})$.

PROOF. The proof is by induction on k. To show the base case ($k = 0$), note that for any location $\mathfrak{l}$, $\mathfrak{h}^0(\mathfrak{l}) = \mathfrak{l}$. $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}]} \mathtt{propagate}(\mathtt{u})$ implies $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}]} \mathtt{forkendpt}(\mathtt{u})$, so $\mathfrak{h} \models_{[\overline{\mathtt{u}} \mapsto \mathfrak{h}^k(\mathfrak{l})]} \mathtt{forkendpt}(\overline{\mathtt{u}})$.

Now we assume the statement holds for some $k \geq 0$ and prove the result for $k + 1$. Assuming $\mathfrak{h}^{k+1}(\mathfrak{l})$ exists, we need to show $\mathfrak{h} \models_{[\overline{\mathtt{u}} \mapsto \mathfrak{h}^{k+1}(\mathfrak{l})]} \mathtt{forkendpt}(\overline{\mathtt{u}})$. If $\mathfrak{h}^{k+1}(\mathfrak{l})$ exists

then $\mathfrak{h}^k(\mathfrak{l})$ does also (and is its predecessor). By (IH), we get $\mathfrak{h} \models_{[\overline{\mathfrak{u}} \mapsto \mathfrak{h}^k(\mathfrak{l})]} \texttt{forkendpt}(\overline{\mathfrak{u}})$. Thus $\mathfrak{h} \models_{[\mathfrak{u} \mapsto \mathfrak{h}^k(\mathfrak{l}), \overline{\mathfrak{u}} \mapsto \mathfrak{h}^{k+1}(\mathfrak{l})]} \mathfrak{u} \hookrightarrow \overline{\mathfrak{u}} \wedge \texttt{forkendpt}(\mathfrak{u})$. By the definition of $\texttt{propagate}(\mathfrak{u})$ we have $\mathfrak{h} \models \forall\, \mathfrak{u}\, \forall\, \overline{\mathfrak{u}}\, \big((\mathfrak{u} \hookrightarrow \overline{\mathfrak{u}} \wedge \texttt{forkendpt}(\mathfrak{u})) \Rightarrow \texttt{forkendpt}(\overline{\mathfrak{u}})\big)$, so therefore $\mathfrak{h} \models_{[\overline{\mathfrak{u}} \mapsto \mathfrak{h}^{k+1}(\mathfrak{l})]} \texttt{forkendpt}(\overline{\mathfrak{u}})$. $\square$

LEMMA 3.5. *Given a heap* $\mathfrak{h}$ *and a valuation* $\mathfrak{f}$*, we have* $\mathfrak{h} \models_{\mathfrak{f}} \texttt{reach}(\mathfrak{u}, \overline{\mathfrak{u}})$ *iff* $\mathfrak{h}^k(\mathfrak{f}(\mathfrak{u})) = \mathfrak{f}(\overline{\mathfrak{u}})$ *for some* $k \geq 0$.

PROOF. ($\Rightarrow$) As a direct consequence of $\mathfrak{h} \models_{\mathfrak{f}} \texttt{reach}(\mathfrak{u}, \overline{\mathfrak{u}})$ (that is, expanding the definitions of reach and the magic wand), we know that for all disjoint heaps $\mathfrak{h}'$ such that $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{propagate}(\mathfrak{u})$, the property $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{forkendpt}(\overline{\mathfrak{u}})$ holds. We will next construct such an $\mathfrak{h}'$ where $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{forkendpt}(\overline{\mathfrak{u}})$ and use it to demonstrate that $\mathfrak{f}(\overline{\mathfrak{u}})$ can be reached from $\mathfrak{f}(\mathfrak{u})$ in $\mathfrak{h}$.

First, let $\mathfrak{l}_0^{reach}, \ldots, \mathfrak{l}_k^{reach}$ be the locations reachable from $\mathfrak{f}(\mathfrak{u})$ in $\mathfrak{h}$. Let $\mathit{forks} = \{\mathfrak{l} \in \mathbb{N} : \mathfrak{l}$ is the midpoint of a fork in $\mathfrak{h}\}$ and let $\mathit{free} \stackrel{\text{def}}{=} \mathbb{N} \setminus (\mathrm{dom}(\mathfrak{h}) \cup \mathrm{ran}(\mathfrak{h}) \cup \{\mathfrak{f}(\mathfrak{u}), \mathfrak{f}(\overline{\mathfrak{u}})\})$. Next, let $\mathfrak{l}_i^j \in \mathit{free}$ be some unique location for all $i \in [0, k]$ and $j \in [1, 3]$, and let $\mathfrak{l}_{\mathfrak{l}'}^{fork} \in \mathit{free}$ be some unique location for each $\mathfrak{l}' \in \mathit{forks}$, with these two sets of locations disjoint.

Now, let $\mathfrak{h}'$ be the unique heap whose graph is exactly the set below:

$$\{(\mathfrak{l}_i^1, \mathfrak{l}_i^3), (\mathfrak{l}_i^2, \mathfrak{l}_i^3), (\mathfrak{l}_i^3, \mathfrak{h}^i(\mathfrak{f}(\mathfrak{u}))) : i \in [0, k]\} \cup \{(\mathfrak{l}_{\mathfrak{l}'}^{fork}, \mathfrak{l}') : \mathfrak{l}' \in \mathit{forks}\}.$$

By construction, $\mathfrak{h}'$ is disjoint from $\mathfrak{h}$ (since all memory cells are taken from the set *free*, which does not include any location from $\mathrm{dom}(\mathfrak{h})$). The locations $\mathfrak{l}_0^1, \mathfrak{l}_0^2, \mathfrak{l}_0^3, \mathfrak{f}(\mathfrak{u})$ make up a fork with endpoint $\mathfrak{f}(\mathfrak{u})$, so $\mathfrak{h}' \models_{\mathfrak{f}} \texttt{forkendpt}(\mathfrak{u})$, and note that for all $i > 0$, $\mathfrak{l}_i^1, \mathfrak{l}_i^2, \mathfrak{l}_i^3, \mathfrak{h}^i(\mathfrak{f}(\mathfrak{u}))$ make up a fork with endpoint $\mathfrak{h}^i(\mathfrak{f}(\mathfrak{u}))$. In particular, this means that for all $i \in [1, k]$, $\mathfrak{l}_i^{reach}$ is the endpoint of a fork in $\mathfrak{h} \uplus \mathfrak{h}'$. Thus $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{propagate}(\mathfrak{u})$.

Next, observe that $\mathfrak{l}_0^{reach}, \ldots, \mathfrak{l}_k^{reach}$ are the *only* fork endpoints that exist in $\mathfrak{h} \uplus \mathfrak{h}'$. To see this, note that

(1) the only forks in $\mathfrak{h}'$ are those with endpoints $\mathfrak{l}_0^{reach}, \ldots, \mathfrak{l}_k^{reach}$,
(2) forks in $\mathfrak{h}$ are destroyed such that their endpoints are not endpoints of forks in $\mathfrak{h} \uplus \mathfrak{h}'$ unless they are also endpoints in $\mathfrak{h}'$, and
(3) by construction, $\mathfrak{h}'$ uses locations from the set *free* that excludes $\mathrm{dom}(\mathfrak{h}) \cup \mathrm{ran}(\mathfrak{h})$, so no fork can be "accidentally" constructed by parts from $\mathfrak{h}$ and $\mathfrak{h}'$.

Thus, for every location $\mathfrak{l} \in \mathbb{N}$, $\mathfrak{l}$ is the endpoint of a fork in $\mathfrak{h} \uplus \mathfrak{h}'$ iff $\mathfrak{l} = \mathfrak{l}_i^{reach}$ for some i.

Now, since we assumed $\mathfrak{h} \models_{\mathfrak{f}} \texttt{reach}(\mathfrak{u}, \overline{\mathfrak{u}})$ and we have found a heap $\mathfrak{h}'$ such that $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{propagate}(\mathfrak{u})$, the property $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{forkendpt}(\overline{\mathfrak{u}})$ holds. Since $\mathfrak{f}(\overline{\mathfrak{u}})$ is the endpoint of a fork in $\mathfrak{h} \uplus \mathfrak{h}'$, it must be equal to $\mathfrak{l}_i^{reach}$ (for some i), and consequently must be equal to $\mathfrak{h}^i(\mathfrak{f}(\mathfrak{u}))$ (for some i).

($\Leftarrow$) Assume that $\mathfrak{h}^k(\mathfrak{f}(\mathfrak{u})) = \mathfrak{f}(\overline{\mathfrak{u}})$ for some $k \geq 0$ and we need to show $\mathfrak{h} \models_{\mathfrak{f}} \texttt{reach}(\mathfrak{u}, \overline{\mathfrak{u}})$. To do so, we must show that for all heaps $\mathfrak{h}'$ disjoint from $\mathfrak{h}$, the following property holds true:

($\dagger$) If $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{propagate}(\mathfrak{u})$, then $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{forkendpt}(\overline{\mathfrak{u}})$.

Assuming $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{propagate}(\mathfrak{u})$, $\mathfrak{f}(\mathfrak{u})$ is the endpoint of a fork in $\mathfrak{h} \uplus \mathfrak{h}'$. Now, from Lemma 3.4 we know $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{forkendpt}(\overline{\mathfrak{u}})$, completing the proof. $\square$

It is notable that 1SL2($-\!\!*$) has no need for built-in reachability predicates, in contrast to formalisms from e.g. [Immerman et al. 2004]. In the rest of the paper, we generalize, in a sense, what was done here in an *ad hoc* manner for reachability, so that any second-order property can be represented in 1SL2($-\!\!*$).

4. COMPARING NUMBERS OF PREDECESSORS

The main goal of this section is to define in 1SL2($\mathrel{-\!\!*}$) a formula expressing that $\widetilde{\sharp \mathfrak{f}(\mathfrak{u})} + k \leq \widetilde{\sharp \mathfrak{f}(\overline{\mathfrak{u}})} + k'$, where $k, k' \in \mathbb{N}$, for any heap $\mathfrak{h}$ and valuation $\mathfrak{f}$. Without the restriction on the number of variables, we know that such properties can be expressed in 1SL($\mathrel{-\!\!*}$) [Brochenin et al. 2012]. Note that arithmetical constraints on list lengths can be found in [Bozga et al. 2010] but this is primitive in the logical formalism. By contrast, we show that constraints of the form $\widetilde{\sharp \mathfrak{f}(\mathfrak{u})} + k \leq \widetilde{\sharp \mathfrak{f}(\overline{\mathfrak{u}})} + k'$ can be expressed in 1SL2($\mathrel{-\!\!*}$) itself. Since a property of the form $\widetilde{\sharp \mathfrak{f}(\mathfrak{u})} + k \leq \widetilde{\sharp \mathfrak{f}(\overline{\mathfrak{u}})} + k'$ requires a formula with $\mathcal{O}(k + k')$ variables in 1SL($\mathrel{-\!\!*}$) according to [Brochenin et al. 2012], herein we need to circumvent this issue by proposing an alternative way to express the key properties that are helpful to state that $\widetilde{\sharp \mathfrak{f}(\mathfrak{u})} + k \leq \widetilde{\sharp \mathfrak{f}(\overline{\mathfrak{u}})} + k'$. Below, we still use first principles from [Brochenin et al. 2012] to construct a formula in 1SL2($\mathrel{-\!\!*}$)—mainly, how to build a fork from a knife and an isolated memory cell—but we will need to bypass the serious problem of having only two variables at our disposal, without permitting ourselves any use of the separating conjunction.

4.1. Principles of the construction and principal difficulties with 1SL2($\mathrel{-\!\!*}$)

Let $\mathfrak{h}$ be a heap and $\mathfrak{f}$ be a valuation for which we wish to check whether $\widetilde{\sharp \mathfrak{f}(\mathfrak{u})}^{\star} + k \leq \widetilde{\sharp \mathfrak{f}(\overline{\mathfrak{u}})}^{\star} + k'$ holds (afterward, it will be easy to conclude for $\widetilde{\sharp \mathfrak{f}(\mathfrak{u})} + k \leq \widetilde{\sharp \mathfrak{f}(\overline{\mathfrak{u}})} + k'$, see the proof of Theorem 4.4). Below, we explain which extensions of $\mathfrak{h}$ must be performed to achieve this. There will be a correspondence with formulae in 1SL2($\mathrel{-\!\!*}$) to enforce the construction of these extensions, and this is the subject of the technical developments below. We mainly describe first principles from [Brochenin et al. 2012], but the reader should be warned that in several places, we propose a simplified alternative, apart from the fact that all the formulae need to be part of the restricted fragment 1SL2($\mathrel{-\!\!*}$).

4.1.1. Preparing the heap. The first step consists in preparing the heap by destroying any forks and knives at $\mathfrak{f}(\mathfrak{u})$ and $\mathfrak{f}(\overline{\mathfrak{u}})$, and ensuring there are no isolated memory cells—these properties will be necessary in later steps—while maintaining the number of predecessors at $\mathfrak{f}(\mathfrak{u})$ and $\mathfrak{f}(\overline{\mathfrak{u}})$. To do this, we augment $\mathfrak{h}$ with a heap $\mathfrak{h}_p$ so that the following properties are satisfied:

(a). $\mathfrak{h}_p$ is disjoint from $\mathfrak{h}$;
(b). $\mathfrak{f}(\mathfrak{u})$ has the same number of predecessors in $\mathfrak{h}$ and in $\mathfrak{h} \uplus \mathfrak{h}_p$, say $\widetilde{\sharp \mathfrak{f}(\mathfrak{u})}^{\star} = m \geq 0$;
(c). $\mathfrak{f}(\overline{\mathfrak{u}})$ has the same number of predecessors in $\mathfrak{h}$ and in $\mathfrak{h} \uplus \mathfrak{h}_p$, say $\widetilde{\sharp \mathfrak{f}(\overline{\mathfrak{u}})}^{\star} = \overline{m} \geq 0$;
(d). In $\mathfrak{h} \uplus \mathfrak{h}_p$, $\mathfrak{f}(\mathfrak{u})$ has no predecessor that is an endpoint of a fork or knife;
(e). In $\mathfrak{h} \uplus \mathfrak{h}_p$, $\mathfrak{f}(\overline{\mathfrak{u}})$ has no predecessor that is an endpoint of a fork or knife;
(f). $\mathfrak{h} \uplus \mathfrak{h}_p$ has no isolated memory cell.

Note that this step is always possible, since to destroy the structure of an isolated memory cell, a fork or a knife, it is sufficient to add a memory cell at some position as to no longer have the object—for instance, to destroy a fork that does not involve either $\mathfrak{f}(\mathfrak{u})$ or $\mathfrak{f}(\overline{\mathfrak{u}})$ one can give its midpoint a third predecessor. In the case $\mathfrak{f}(\mathfrak{u})$ (or $\mathfrak{f}(\overline{\mathfrak{u}})$) is the midpoint of a fork, it is sufficient to add a new memory cell $\mathfrak{l}' \to \mathfrak{l}$ where $\mathfrak{l}$ is a predecessor of $\mathfrak{f}(\mathfrak{u})$ (so that the fork is destroyed without modifying the number of predecessors of $\mathfrak{f}(\mathfrak{u})$). We take advantage of the formulae $\mathtt{antiforky}(\cdot)$ and $\mathtt{antiknify}(\cdot)$ to establish properties (d) and (e); the others are straightforward.

4.1.2. Addition of a segmented heap. This step consists in checking whether, for each segmented heap $\mathfrak{h}_s$ satisfying certain properties, the condition ($\mathcal{P}$) defined below holds true. The heap $\mathfrak{h}_s$ must be segmented and also must satisfy the following:

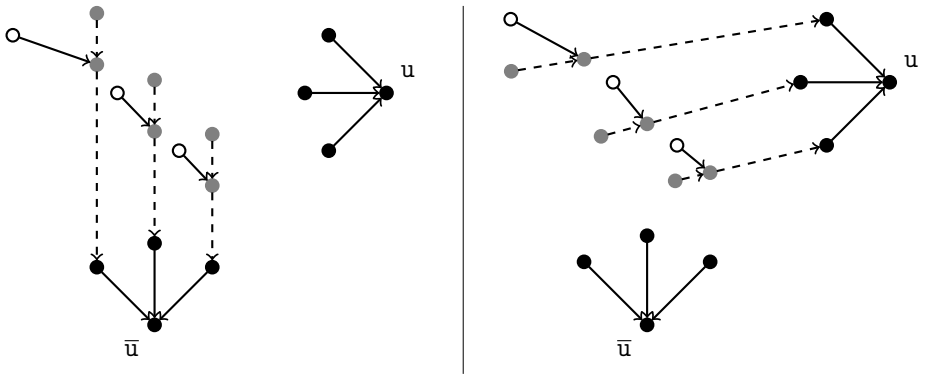


Fig. 5: Using knives and segments to make forks and compare predecessor counts.

(a). $\mathfrak{h}_s$ is disjoint from $\mathfrak{h} \uplus \mathfrak{h}_p$;
(b). $\mathfrak{f}(\mathrm{u})$ and $\mathfrak{f}(\overline{\mathrm{u}})$ have no predecessors in $\mathfrak{h}_s$;
(c). In $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s$, neither $\mathfrak{f}(\mathrm{u})$ nor $\mathfrak{f}(\overline{\mathrm{u}})$ has a predecessor that is an endpoint of a fork or knife (they are *antifork*y and *antiknify*).

Although $\mathfrak{h}_s$ is segmented, we cannot assume that $\mathfrak{h}_s$ is completely disconnected from $\mathfrak{h} \uplus \mathfrak{h}_p$ since $\mathrm{ran}(\mathfrak{h}_s) \cap \mathrm{dom}(\mathfrak{h} \uplus \mathfrak{h}_p)$ may be non-empty but $\mathrm{ran}(\mathfrak{h}_s)$ cannot contain $\mathfrak{f}(\mathrm{u})$ or $\mathfrak{f}(\overline{\mathrm{u}})$. Let n be the number of isolated memory cells in $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s$. The heap $\mathfrak{h}_s$ may have strictly more than n isolated memory cells but those that really matter are the ones that are isolated in $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s$. Note that for each $q \geq 0$, it is possible to build $\mathfrak{h}_s^q$ so that $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s^q$ has exactly q isolated memory cells and $\mathfrak{h}_s^q$ satisfies the above conditions.

In order to construct forks in $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s$ whose endpoints are predecessors of $\mathfrak{f}(\mathrm{u})$ or $\mathfrak{f}(\overline{\mathrm{u}})$, either we can augment the heap with a fork, or we can augment the heap with a knife so that its combination with an isolated memory cell in $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s$ leads to a fork whose endpoint is a predecessor of $\mathfrak{f}(\mathrm{u})$ or $\mathfrak{f}(\overline{\mathrm{u}})$. See Figure 5, which depicts two locations, each with three predecessors. Imagine we are performing the comparison $\widetilde{\sharp \mathfrak{f}(\mathrm{u})}^\star \leq \widetilde{\sharp \mathfrak{f}(\overline{\mathrm{u}})}^\star$ (presently equivalent to $\widetilde{\sharp \mathfrak{f}(\mathrm{u})} \leq \widetilde{\sharp \mathfrak{f}(\overline{\mathrm{u}})}$). First, a segmented heap is added (allocated locations are marked with a white circle in the figure). The left half of the figure shows the addition of a collection of knives through these segments to make $\mathfrak{f}(\overline{\mathrm{u}})$ forky; with the same segments, it must then be possible to add a collection of knives to make $\mathfrak{f}(\mathrm{u})$ forky. Indeed it is; if this is true for *all* such segmented heaps, then it must be that $\widetilde{\sharp \mathfrak{f}(\mathrm{u})}^\star \leq \widetilde{\sharp \mathfrak{f}(\overline{\mathrm{u}})}^\star$. The segments from the segmented heap should really be seen as "potential forks." When k or k' is nonzero, we consider additional forks on one or both sides to compensate for the offset.

This illustrates the principle behind the definition of the following property ($\mathcal{P}$): If there is a heap $\mathfrak{h}_{[k]}$ disjoint from $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s$ made of isolated knives and k isolated forks so that $\mathfrak{f}(\overline{\mathrm{u}})$ is forky in $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s \uplus \mathfrak{h}_{[k]}$, then there is a heap $\mathfrak{h}_{[k']}$ disjoint from $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s$ made of isolated knives and k' isolated forks so that $\mathfrak{f}(\mathrm{u})$ is forky in $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s \uplus \mathfrak{h}_{[k']}$. Note that by the conditions satisfied by $\mathfrak{h}_{[k]}$ or by $\mathfrak{h}_{[k']}$, the number of predecessors of $\mathfrak{f}(\overline{\mathrm{u}})$ [resp. $\mathfrak{f}(\mathrm{u})$] in $\mathfrak{h}_{[k]}$ [resp. $\mathfrak{h}_{[k']}$] is necessarily zero (there is no need to specify explicitly that we do not add predecessors). Indeed, for instance, if in $\mathfrak{h}_{[k]}$ there were an isolated fork or knife having as endpoint $\mathfrak{f}(\overline{\mathrm{u}})$, because $\mathfrak{h}_{[k]}$ is made of isolated knives and forks, one of the predecessors of $\mathfrak{f}(\overline{\mathrm{u}})$ in the newly built heap would not be the endpoint of some fork, and therefore $\mathfrak{f}(\overline{\mathrm{u}})$ would not be forky. That predecessor of $\mathfrak{f}(\overline{\mathrm{u}})$ in the newly built heap would be the midpoint of the isolated fork or knife that has $\mathfrak{f}(\overline{\mathrm{u}})$ as endpoint in $\mathfrak{h}_{[k]}$.

Let us return to arithmetical considerations. The number of forks in $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s \uplus \mathfrak{h}_{[k]}$ whose endpoints are predecessors of $\mathfrak{f}(\overline{\mathrm{u}})$ is bounded by $n+k$ and similarly, the number of forks in $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s \uplus \mathfrak{h}_{[k']}$ whose endpoints are predecessors of $\mathfrak{f}(\mathrm{u})$ is bounded by $n+k'$. Note also that the number of predecessors of $\mathfrak{f}(\mathrm{u})$ is the same in $\mathfrak{h}$ and in $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s \uplus \mathfrak{h}_{[k']}$ and the number of predecessors of $\mathfrak{f}(\overline{\mathrm{u}})$ is the same in $\mathfrak{h}$ and in $\mathfrak{h} \uplus \mathfrak{h}_p \uplus \mathfrak{h}_s \uplus \mathfrak{h}_{[k]}$. So, $n + k \geq \overline{m}$ implies $n + k' \geq m$, i.e. $n \geq \overline{m} - k$ implies $n \geq m - k'$.

4.1.3. Checking the resulting heap. By checking step 2 for all $n \geq 0$, we get that for all $n \geq 0$, we have $n \geq \overline{m} - k$ implies $n \geq m - k'$, which entails that $\overline{m} - k \geq m - k'$, i.e. $m + k \leq \overline{m} + k'$, whenever $m - k' \geq 0$ and $\overline{m} - k \geq 0$. Universal quantification over n is simulated in a formula by using separating implication. When $m < k'$ or $\overline{m} < k$, we make a dedicated case analysis (see the proof of Theorem 4.4).

Below, we present the technical developments.

4.2. Cutlery revisited

Apart from forks, we introduce the notions of *collections* and *large forks* (instrumental in the proof of Lemma 4.1 below). A *large fork* is a sequence of distinct locations $\mathfrak{l}_1, \cdots, \mathfrak{l}_5$ such that $\mathfrak{l}_1$, $\mathfrak{l}_2$, and $\mathfrak{l}_3$ have no predecessors, $\mathfrak{h}(\mathfrak{l}_1) = \mathfrak{h}(\mathfrak{l}_2) = \mathfrak{h}(\mathfrak{l}_3) = \mathfrak{l}_4$, $\widetilde{\sharp \mathfrak{l}_4} = 3$ and $\mathfrak{h}(\mathfrak{l}_4) = \mathfrak{l}_5$. Location $\mathfrak{l}_5$ is called the *endpoint* of the large fork and $\mathfrak{l}_4$ its *midpoint*, and as with forks and knives, the large fork is called *isolated* iff $\widetilde{\sharp \mathfrak{l}_5} = 1$ and $\mathfrak{l}_5 \notin \mathrm{dom}(\mathfrak{h})$. A heap $\mathfrak{h}$ is a *collection of knives and forks* $\stackrel{\text{def}}{\Leftrightarrow}$ there is no location in $\mathrm{dom}(\mathfrak{h})$ that does not belong to an isolated knife or to an isolated fork. Similarly, a heap $\mathfrak{h}$ is a *collection of knives and large forks* $\stackrel{\text{def}}{\Leftrightarrow}$ there is no location in $\mathrm{dom}(\mathfrak{h})$ that does not belong to an isolated knife or isolated large fork.

LEMMA 4.1. *There are formulae* $\mathtt{ksfs}$, $\mathtt{kslfs}$ *and* $\mathtt{ksfs}_{=k}$ *($k \geq 0$) in 1SL2(⇁) such that for every heap* $\mathfrak{h}$,

(1). $\mathfrak{h} \models \mathtt{ksfs}$ *iff* $\mathfrak{h}$ *is a collection of knives and forks,*
(2). $\mathfrak{h} \models \mathtt{kslfs}$ *iff* $\mathfrak{h}$ *is a collection of knives and large forks,*
(3). $\mathfrak{h} \models \mathtt{ksfs}_{=k}$ *iff* $\mathfrak{h}$ *is a collection of knives and forks with exactly* k *forks.*

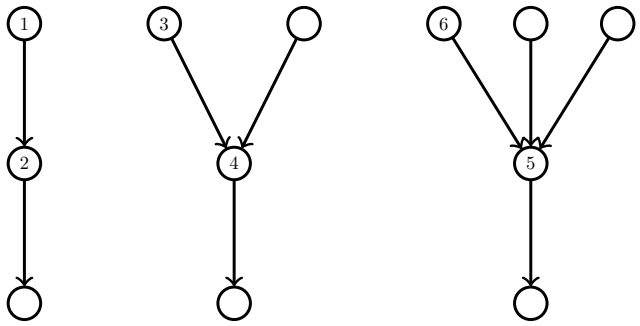
It is also worth noting that $\mathtt{ksfs}_{=k}$ is of linear size in k and it can be built in space $\mathcal{O}(log(k))$ (see the proof below).

PROOF. First, let us introduce auxiliary formulae. For all $\bowtie \in \{\leq, \geq, =\}$ and $i \geq 0$, we define the following formulae (formulae of the form $\sharp \mathrm{u} \bowtie k$ are defined in Lemma 3.3):

$$\sharp \mathrm{u}^0 \bowtie k \stackrel{\text{def}}{=} \sharp \mathrm{u} \bowtie k$$
$$\sharp \mathrm{u}^{+(i+1)} \bowtie k \stackrel{\text{def}}{=} \exists\, \overline{\mathrm{u}}\ \mathrm{u} \hookrightarrow \overline{\mathrm{u}} \wedge \sharp \overline{\mathrm{u}}^{+i} \bowtie k.$$

For instance, $\sharp \mathrm{u}^{+6} \geq 5$ states that there is a (necessarily unique) location at distance 6 from u and its number of predecessors is greater than or equal to 5. Note that (for example) $\neg \sharp \mathrm{u}^{+i} \geq 0$ is *not* equivalent to $\sharp \mathrm{u}^{+i} < 0$ when $i \neq 0$, since the negation applies to the existential quantifier rather than to the comparison. This observation allows a shorthand for expressing a property that an ith successor exists; for example, $\mathfrak{h} \models_{\mathfrak{f}} \sharp \mathrm{u}^{+3} \geq 0$ iff $\{\mathfrak{f}(\mathrm{u}), \mathfrak{h}(\mathfrak{f}(\mathrm{u})), \mathfrak{h}(\mathfrak{h}(\mathfrak{f}(\mathrm{u})))\} \subseteq \mathrm{dom}(\mathfrak{h})$. Symmetrically, $\sharp \mathrm{u}^{-(i+1)} \bowtie k \stackrel{\text{def}}{=} \exists\, \overline{\mathrm{u}}\ \overline{\mathrm{u}} \hookrightarrow \mathrm{u} \wedge \sharp \overline{\mathrm{u}}^{-i} \bowtie k$ but this plays no role in the rest of the paper and this could be useful in some other context, see e.g. [Demri and Deters 2015].

Now, in Figure 6, we introduce a classification of the types of allocated locations in knifes, forks and large forks (we take advantage of obvious symmetries) when they occur in collections made of disjoint structures.



$$
\begin{array}{l}
\varphi_1(\mathtt{u}) \stackrel{\text{def}}{=} (\sharp\mathtt{u} = 0) \wedge (\sharp\mathtt{u}^{+1} = 1) \wedge (\sharp\mathtt{u}^{+2} = 1) \wedge \neg(\sharp\mathtt{u}^{+3} \geq 0) \\
\varphi_2(\mathtt{u}) \stackrel{\text{def}}{=} \exists\, \overline{\mathtt{u}}\ \overline{\mathtt{u}} \hookrightarrow \mathtt{u} \wedge \varphi_1(\overline{\mathtt{u}}) \\
\varphi_3(\mathtt{u}) \stackrel{\text{def}}{=} (\exists\, \overline{\mathtt{u}}\ \mathtt{u} \hookrightarrow \overline{\mathtt{u}} \wedge \mathtt{tril}(\overline{\mathtt{u}})) \wedge \sharp\mathtt{u}^{+2} = 1 \\
\varphi_4(\mathtt{u}) \stackrel{\text{def}}{=} \exists\, \overline{\mathtt{u}}\ \overline{\mathtt{u}} \hookrightarrow \mathtt{u} \wedge \varphi_3(\overline{\mathtt{u}}) \\
\varphi_5(\mathtt{u}) \stackrel{\text{def}}{=} \sharp\mathtt{u} = 3 \wedge \sharp\mathtt{u}^{+1} = 1 \wedge \neg(\sharp\mathtt{u}^{+2} \geq 0) \wedge \mathtt{allzpred}(\mathtt{u}) \\
\varphi_6(\mathtt{u}) \stackrel{\text{def}}{=} \exists\, \overline{\mathtt{u}}\ \mathtt{u} \hookrightarrow \overline{\mathtt{u}} \wedge \varphi_5(\overline{\mathtt{u}})
\end{array}
$$

Fig. 6: Types of allocated locations in a knife, fork, and large fork, and formulae φ_τ for each type τ of location.

In order to define the formulae, we make a case analysis depending on the position of an allocated location on a knife, on a fork or on a large fork. This is a bit tedious but without essential technical difficulty. Figure 6 associates a formula φ_τ for each location of type τ (see the proof of Lemma 3.2 for the definition of $\mathtt{tril}(\mathtt{u})$).

The following properties are then easy to establish (see Figure 6):

($\mathcal{Q}_1$). For all $\mathfrak{h}$ and $\mathfrak{f}$, $\mathfrak{h} \models_\mathfrak{f} \varphi_1(\mathtt{u})$ iff $\mathfrak{f}(\mathtt{u})$ points to the midpoint of an isolated knife.
($\mathcal{Q}_2$). For all $\mathfrak{h}$ and $\mathfrak{f}$, $\mathfrak{h} \models_\mathfrak{f} \varphi_2(\mathtt{u})$ iff $\mathfrak{f}(\mathtt{u})$ is the midpoint of an isolated knife.
($\mathcal{Q}_3$). For all $\mathfrak{h}$ and $\mathfrak{f}$, $\mathfrak{h} \models_\mathfrak{f} \varphi_3(\mathtt{u})$ iff $\mathfrak{f}(\mathtt{u})$ points to the midpoint of an isolated fork.
($\mathcal{Q}_4$). For all $\mathfrak{h}$ and $\mathfrak{f}$, $\mathfrak{h} \models_\mathfrak{f} \varphi_4(\mathtt{u})$ iff $\mathfrak{f}(\mathtt{u})$ is the midpoint of an isolated fork.
($\mathcal{Q}_6$). For all $\mathfrak{h}$ and $\mathfrak{f}$, $\mathfrak{h} \models_\mathfrak{f} \varphi_6(\mathtt{u})$ iff $\mathfrak{f}(\mathtt{u})$ points to the midpoint of an isolated large fork.
($\mathcal{Q}_5$). For all $\mathfrak{h}$ and $\mathfrak{f}$, $\mathfrak{h} \models_\mathfrak{f} \varphi_5(\mathtt{u})$ iff $\mathfrak{f}(\mathtt{u})$ is the midpoint of an isolated large fork.

It is now easy to define $\mathtt{ksfs}$ and $\mathtt{kslfs}$:

$$
\begin{array}{rcl}
\mathtt{ksfs} & \stackrel{\text{def}}{=} & \forall\, \mathtt{u}\ \mathtt{alloc}(\mathtt{u}) \Rightarrow \varphi_1(\mathtt{u}) \vee \varphi_2(\mathtt{u}) \vee \varphi_3(\mathtt{u}) \vee \varphi_4(\mathtt{u}) \\
\mathtt{kslfs} & \stackrel{\text{def}}{=} & \forall\, \mathtt{u}\ \mathtt{alloc}(\mathtt{u}) \Rightarrow \varphi_1(\mathtt{u}) \vee \varphi_2(\mathtt{u}) \vee \varphi_5(\mathtt{u}) \vee \varphi_6(\mathtt{u}).
\end{array}
$$

Next, in order to define $\mathtt{ksfs}_{=k}$, the most natural way would be to use $*$, but in the fragment 1SL2($-\!\!*$), separating conjunction is banished. Similarly, using $\mathcal{O}(k)$ variables would help to identify k forks but again we have only two variables at hand. It is now time to take advantage of large forks in order to identify forks in the original heap.

We define ψ_i as follows:

$$
\begin{array}{rcl}
\psi_0 & \stackrel{\text{def}}{=} & \mathtt{kslfs} \\
\psi_{i+1} & \stackrel{\text{def}}{=} & \exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ \varphi_4(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}}) \wedge \left[(\mathtt{size} = 1 \wedge \overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \mathrel{\vec{-\!\!*}} \psi_i\right].
\end{array}
$$

Let $\mathtt{ksfs}_{=k}$ be defined as $\mathtt{ksfs} \wedge \psi_k$.

(A). Let us show that $\mathfrak{h} \models \texttt{ksfs}$ iff $\mathfrak{h}$ is a collection of knives and forks.
($\Leftarrow$) By way of contradiction, assume $\mathfrak{h}$ is a collection of knives and forks but $\mathfrak{h} \not\models \texttt{ksfs}$. There must exist some location $\mathfrak{l}^{bad}$ such that $\mathfrak{h} \not\models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \texttt{alloc}(\mathtt{u}) \Rightarrow \varphi_1(\mathtt{u}) \vee \varphi_2(\mathtt{u}) \vee \varphi_3(\mathtt{u}) \vee \varphi_4(\mathtt{u})$.
By definition, if $\mathfrak{h}$ is a collection of knives and forks, every location in the domain of the heap belongs to an isolated knife or to an isolated fork. We consider three cases.
— Assume $\mathfrak{l}^{bad} \in \mathrm{dom}(\mathfrak{h})$ belongs to an isolated knife. By definition, an isolated knife consists of three distinct locations, $\mathfrak{l}_0$, $\mathfrak{l}_1$, and $\mathfrak{l}$, such that $\mathfrak{h}(\mathfrak{l}_0) = \mathfrak{l}$, $\widetilde{\sharp\mathfrak{l}_0} = 1$, $\mathfrak{h}(\mathfrak{l}_1) = \mathfrak{l}_0$, $\widetilde{\sharp\mathfrak{l}_1} = 0$, $\widetilde{\sharp\mathfrak{l}} = 1$, and $\mathfrak{l} \notin \mathrm{dom}(\mathfrak{h})$. There are two cases.
(Case $\mathfrak{l}^{bad} = \mathfrak{l}_1$.) $\widetilde{\sharp\mathfrak{l}^{bad}} = 0$, so $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \sharp\mathtt{u} = 0$. $\mathfrak{h}(\mathfrak{l}^{bad}) = \mathfrak{l}_0$ and $\widetilde{\sharp\mathfrak{l}_0} = 1$, so $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \sharp\mathtt{u}^{+1} = 1$. $\mathfrak{h}(\mathfrak{h}(\mathfrak{l}^{bad})) = \mathfrak{l}$ and $\widetilde{\sharp\mathfrak{l}} = 1$, so $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \sharp\mathtt{u}^{+2} = 1$. Finally, $\mathfrak{h}(\mathfrak{h}(\mathfrak{l}^{bad})) \notin \mathrm{dom}(\mathfrak{h})$, so $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \neg(\sharp\mathtt{u}^{+3} \geq 0)$. Thus, $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \varphi_1(\mathtt{u})$, a contradiction.
(Case $\mathfrak{l}^{bad} = \mathfrak{l}_0$.) $\mathfrak{l}^{bad}$ has a single predecessor $\mathfrak{l}_1$ such that $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}_1]} \varphi_1(\mathtt{u})$ by the same reasoning in the previous case. Thus $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \varphi_2(\mathtt{u})$, and we have a contradiction.
— Assume $\mathfrak{l}^{bad} \in \mathrm{dom}(\mathfrak{h})$ belongs to an isolated fork. By definition, an isolated fork consists of four locations $\mathfrak{l}$, $\mathfrak{l}_0$, $\mathfrak{l}_1$, and $\mathfrak{l}_2$, such that $\mathfrak{h}(\mathfrak{l}_0) = \mathfrak{l}$, $\widetilde{\sharp\mathfrak{l}_0} = 2$, $\mathfrak{h}(\mathfrak{l}_1) = \mathfrak{h}(\mathfrak{l}_2) = \mathfrak{l}_0$, $\widetilde{\sharp\mathfrak{l}_1} = \widetilde{\sharp\mathfrak{l}_2} = 0$, $\widetilde{\sharp\mathfrak{l}} = 1$, and $\mathfrak{l} \notin \mathrm{dom}(\mathfrak{h})$.
(Case $\mathfrak{l}^{bad} = \mathfrak{l}_0$.) $\mathfrak{h}(\mathfrak{l}^{bad}) = \mathfrak{l}$ and $\widetilde{\sharp\mathfrak{l}} = 1$, so $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \sharp\mathtt{u}^{+1} = 1$. Further, $\mathfrak{l}^{bad}$ is a tril, so $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \texttt{tril}(\mathtt{u})$, and $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \varphi_4(\mathtt{u})$, and we have a contradiction.
(Case $\mathfrak{l}^{bad} = \mathfrak{l}_1$ or $\mathfrak{l}^{bad} = \mathfrak{l}_2$.) $\mathfrak{h}(\mathfrak{l}^{bad}) = \mathfrak{l}_0$, and $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}_0]} \varphi_4(\mathtt{u})$ by the previous case. Thus $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \varphi_3(\mathtt{u})$, and we have a contradiction.
— Assume $\mathfrak{l}^{bad} \notin \mathrm{dom}(\mathfrak{h})$. Then $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \neg\texttt{alloc}(\mathtt{u})$, and we have a contradiction.
($\Rightarrow$) Now assume $\mathfrak{h} \models \texttt{ksfs}$ but $\mathfrak{h}$ is not a collection of knives and forks.
If $\mathfrak{h}$ is not a collection of knives and forks, then a location $\mathfrak{l}^{bad} \in \mathrm{dom}(\mathfrak{h})$ exists that is not on an isolated knife or isolated fork. Contradiction, since ksfs requires such locations to satisfy one of φ_1, φ_2, φ_3, φ_4, which are on isolated knives and isolated forks.
Thus, $\mathfrak{h}$ is a collection of knives and forks.
(B). Let us show that $\mathfrak{h} \models \texttt{kslfs}$ iff $\mathfrak{h}$ is a collection of knives and large forks (proof very similar to the proof for (A)).
($\Leftarrow$) By way of contradiction, assume $\mathfrak{h}$ is a collection of knives and large forks but $\mathfrak{h} \not\models \texttt{kslfs}$. There must exist some location $\mathfrak{l}^{bad}$ such that $\mathfrak{h} \not\models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \texttt{alloc}(\mathtt{u}) \Rightarrow \varphi_1(\mathtt{u}) \vee \varphi_2(\mathtt{u}) \vee \varphi_5(\mathtt{u}) \vee \varphi_6(\mathtt{u})$.
By definition, if $\mathfrak{h}$ is a collection of knives and large forks, every location in the domain of the heap belongs to an isolated knife or to an isolated large fork. We consider three cases.
— Assume $\mathfrak{l}^{bad} \in \mathrm{dom}(\mathfrak{h})$ belongs to an isolated knife. Identical to the first part of Case (A) proven above.
— Assume $\mathfrak{l}^{bad} \in \mathrm{dom}(\mathfrak{h})$ belongs to an isolated large fork. By definition, an isolated large fork consists of five distinct locations $\mathfrak{l}_1, \cdots, \mathfrak{l}_5$, such that $\mathfrak{l}_1$, $\mathfrak{l}_2$, and $\mathfrak{l}_3$ have no predecessors, $\mathfrak{h}(\mathfrak{l}_1) = \mathfrak{h}(\mathfrak{l}_2) = \mathfrak{h}(\mathfrak{l}_3) = \mathfrak{l}_4$, $\widetilde{\sharp\mathfrak{l}_4} = 3$, $\mathfrak{h}(\mathfrak{l}_4) = \mathfrak{l}_5$, $\widetilde{\sharp\mathfrak{l}_5} = 1$, and $\mathfrak{l}_5 \notin \mathrm{dom}(\mathfrak{h})$.
(Case $\mathfrak{l}^{bad} = \mathfrak{l}_4$.) $\mathfrak{h}(\mathfrak{l}^{bad}) = \mathfrak{l}_5$, $\widetilde{\sharp\mathfrak{l}^{bad}} = 3$, $\widetilde{\sharp\mathfrak{l}_5} = 1$, and $\mathfrak{l}_5 \notin \mathrm{dom}(\mathfrak{h})$, so $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \sharp\mathtt{u} = 3 \wedge \sharp\mathtt{u}^{+1} = 1 \wedge \neg(\sharp\mathtt{u}^{+2} \geq 0) \wedge \texttt{allzpred}(\mathtt{u})$. Thus $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \varphi_5(\mathtt{u})$, and we have a contradiction.

(Case $\mathfrak{l}^{bad} = \mathfrak{l}_1$ or $\mathfrak{l}^{bad} = \mathfrak{l}_2$ or $\mathfrak{l}^{bad} = \mathfrak{l}_3$.) $\mathfrak{h}(\mathfrak{l}^{bad}) = \mathfrak{l}_4$, and $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}_4]} \varphi_5(\mathtt{u})$ by the previous case. Thus $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \varphi_6(\mathtt{u})$, and we have a contradiction.

— Assume $\mathfrak{l}^{bad} \notin \mathrm{dom}(\mathfrak{h})$. Then $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}^{bad}]} \neg\mathtt{alloc}(\mathtt{u})$, and we have a contradiction.

($\Rightarrow$) Now assume $\mathfrak{h} \models \mathtt{kslfs}$ but $\mathfrak{h}$ is not a collection of knives and large forks.

If $\mathfrak{h}$ is not a collection of knives and large forks, then a location $\mathfrak{l}^{bad} \in \mathrm{dom}(\mathfrak{h})$ exists that is not on an isolated knife or isolated large fork. Contradiction, since $\mathtt{kslfs}$ requires such locations to satisfy one of φ_1, φ_2, φ_5, φ_6, which are on isolated knives and isolated large forks.

Thus, $\mathfrak{h}$ is a collection of knives and large forks.

(C). Let us show that $\mathfrak{h} \models \mathtt{ksfs}_{=k}$ iff $\mathfrak{h}$ is a collection of knives and forks with exactly k forks.

We first prove an inductive property of ψ_i.

> We extend the definitions of this section naturally to include a new type of collection. A heap $\mathfrak{h}$ is a *collection of knives, forks and large forks* $\stackrel{\text{def}}{\Leftrightarrow}$ there is no location in $\mathrm{dom}(\mathfrak{h})$ that does not belong to an isolated knife, fork, or large fork.
>
> *Inductive property ($*$):* For all $i \geq 0$, $\mathfrak{h}$ and $\mathfrak{f}$, we have $\mathfrak{h} \models_{\mathfrak{f}} \psi_i$ iff $\mathfrak{h}$ is a collection of knives, forks and large forks with exactly i forks.
>
> Base case $i = 0$ is by an easy verification. Indeed, $\mathfrak{h} \models_{\mathfrak{f}} \psi_0$ iff $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{kslfs}$ (by definition of ψ_0) iff $\mathfrak{h}$ is a collection of knives and large forks (by (B)) iff $\mathfrak{h}$ is a collection of knives, forks and large forks with exactly zero forks.
>
> Now, suppose that the property $(*)$ holds for all $j \leq i$ for some $i \geq 0$. Let us show it for $i + 1$.
>
> ($\Leftarrow$) Assume $\mathfrak{h}$ is a collection of knives, forks and large forks with exactly $i + 1$ forks. Since $\mathfrak{h}$ has at least one fork, let $\mathfrak{l}$ be the midpoint of such a fork and $\mathfrak{l}'$ be an isolated location in $\mathfrak{h}$. So, $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l},\overline{\mathtt{u}}\mapsto\mathfrak{l}']} \varphi_4(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}})$. Let $\mathfrak{h}'$ be the heap such that $\mathrm{card}(\mathrm{dom}(\mathfrak{h}')) = 1$ and $\mathfrak{h}'(\mathfrak{l}') = \mathfrak{l}$. Consequently,
>
> (1) $\mathfrak{h}'$ is disjoint from $\mathfrak{h}$ and $\mathfrak{h}' \models_{[\mathtt{u}\mapsto\mathfrak{l},\overline{\mathtt{u}}\mapsto\mathfrak{l}']} \mathtt{size} = 1 \wedge \overline{\mathtt{u}} \hookrightarrow \mathtt{u}$,
>
> (2) $\mathfrak{h} \uplus \mathfrak{h}'$ is a collection of knives, forks and large forks with exactly i forks. Actually, in $\mathfrak{h} \uplus \mathfrak{h}'$, the location $\mathfrak{l}$ is the midpoint of a large fork and all other locations are untouched by this disjoint union.
>
> By (IH), we have $\mathfrak{h} \uplus \mathfrak{h}' \models_{[\mathtt{u}\mapsto\mathfrak{l},\overline{\mathtt{u}}\mapsto\mathfrak{l}']} \psi_i$. By definition of $\models$, we have $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l},\overline{\mathtt{u}}\mapsto\mathfrak{l}']} (\mathtt{size} = 1 \wedge \overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \mathrel{-\!\!\!\overrightarrow{*}} \psi_i$ and therefore $\mathfrak{h} \models_{\mathfrak{f}} \exists\, \mathtt{u}\, \overline{\mathtt{u}}\, (\varphi_4(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}})) \wedge (\mathtt{size} = 1 \wedge \overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \mathrel{-\!\!\!\overrightarrow{*}} \psi_i$ (for any valuation $\mathfrak{f}$), that is $\mathfrak{h} \models_{\mathfrak{f}} \psi_{i+1}$.
>
> ($\Rightarrow$) Assume that $\mathfrak{h} \models_{\mathfrak{f}} \psi_{i+1}$. There are locations $\mathfrak{l}$ and $\mathfrak{l}'$ such that $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l},\overline{\mathtt{u}}\mapsto\mathfrak{l}']} \varphi_4(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}}) \wedge ((\mathtt{size} = 1 \wedge \overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \mathrel{-\!\!\!\overrightarrow{*}} \psi_i)$. Thus, $\mathfrak{l}$ the midpoint of a fork in $\mathfrak{h}$ and by definition of $\models$, there is a heap $\mathfrak{h}'$ disjoint from $\mathfrak{h}$ such that $\mathfrak{h}' \models_{[\mathtt{u}\mapsto\mathfrak{l},\overline{\mathtt{u}}\mapsto\mathfrak{l}']} \mathtt{size} = 1 \wedge \overline{\mathtt{u}} \hookrightarrow \mathtt{u}$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{[\mathtt{u}\mapsto\mathfrak{l},\overline{\mathtt{u}}\mapsto\mathfrak{l}']} \psi_i$. The heap $\mathfrak{h}'$ is uniquely defined, since $\mathrm{card}(\mathrm{dom}(\mathfrak{h}')) = 1$ and $\mathfrak{h}'(\mathfrak{l}') = \mathfrak{l}$ and by (IH), $\mathfrak{h} \uplus \mathfrak{h}'$ is a collection of knives, forks and large forks with exactly i forks. Since $\mathfrak{h}'$ converts a single fork of $\mathfrak{h}$ into a large fork, $\mathfrak{h}$ must be a collection of knives, forks and large forks with exactly $i + 1$ forks.

Now we can proceed with the proof about $\mathtt{ksfs}_{=k}$.

($\Leftarrow$) Assume $\mathfrak{h}$ is a collection of knives and forks with exactly k forks. By (A), we have $\mathfrak{h} \models \mathtt{ksfs}$ and by $(*)$, we have $\mathfrak{h} \models \psi_k$ (indeed $\mathfrak{h}$ is a collection of knives, forks

and large forks with zero large fork and k forks). Since by definition $\texttt{ksfs}_{=k}$ is equal to $\texttt{ksfs} \wedge \psi_k$, we get $\mathfrak{h} \models \texttt{ksfs}_{=k}$.
($\Rightarrow$) Now suppose that $\mathfrak{h} \models \texttt{ksfs}_{=k}$. Since $\texttt{ksfs}_{=k}$ is equal to $\texttt{ksfs} \wedge \psi_k$, by (A), $\mathfrak{h}$ is a collection of knives and forks, and by the property ($*$), $\mathfrak{h}$ is a collection of knives, forks and large forks with exactly k forks. Consequently, $\mathfrak{h}$ has no large fork and exactly k forks. Hence, $\mathfrak{h}$ is a collection of knives and forks with exactly k forks.

□

4.3. Using collections of forks for comparison

Here is the way we can use the formulae from Lemma 4.1.

LEMMA 4.2. *Let $k \geq 0$, $\mathfrak{h}$ be a heap and $\mathfrak{f}$ be a valuation such that $\mathfrak{h} \models_{\mathfrak{f}} \texttt{antiforky}(\mathtt{u}) \wedge \texttt{antiknify}(\mathtt{u})$, $\mathfrak{h}$ has n isolated memory cells, $m = \widetilde{\sharp \mathfrak{f}(\mathtt{u})}^{\star}$ and $m - k \geq 0$. We have $\mathfrak{h} \models_{\mathfrak{f}} (\texttt{ksfs}_{=k} \mathrel{\overrightarrow{-\!\!*}} \texttt{forky}(\mathtt{u}))$ iff $n \geq m - k$.*

The proof of Lemma 4.2 uses principles similar to what has been done in [Brochenin et al. 2012; Demri and Deters 2015] except that all formulae belong to 1SL2($-\!\!*$) and we propose a simplified version of the lemma and proof (note also our use of $\widetilde{\sharp \mathfrak{f}(\mathtt{u})}^{\star}$).

PROOF. We have $\mathfrak{h} \models_{\mathfrak{f}} (\texttt{ksfs}_{=k} \mathrel{\overrightarrow{-\!\!*}} \texttt{forky}(\mathtt{u}))$ iff (†) there is a heap $\mathfrak{h}'$, disjoint from $\mathfrak{h}$, such that $\mathfrak{h}' \models_{\mathfrak{f}} \texttt{ksfs}_{=k}$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{forky}(\mathtt{u})$. First, let us suppose that (†) holds true with some heap $\mathfrak{h}'$ such that $\mathfrak{h}' \models_{\mathfrak{f}} \texttt{ksfs}_{=k}$.

— The only forks in $\mathfrak{h} \uplus \mathfrak{h}'$ whose endpoints are predecessors of $\mathfrak{f}(\mathtt{u})$ are those from forks in $\mathfrak{h}'$ or those obtained by combining an isolated memory cell from $\mathfrak{h}$ with a knife from $\mathfrak{h}'$. Indeed, by assumption $\mathfrak{h} \models_{\mathfrak{f}} \texttt{antiforky}(\mathtt{u}) \wedge \texttt{antiknify}(\mathtt{u})$.
— The number of forks pointing to a predecessor of $\mathfrak{f}(\mathtt{u})$ in $\mathfrak{h} \uplus \mathfrak{h}'$ is therefore less than or equal to $n + k$.
— The number of predecessors of $\mathfrak{f}(\mathtt{u})$ (other than $\mathfrak{f}(\mathtt{u})$ itself, if it has a self-loop) in $\mathfrak{h} \uplus \mathfrak{h}'$ is greater than or equal to its number of predecessors (other than itself) in $\mathfrak{h}$. However, since $\mathfrak{h}'$ is collection of knives and forks (and therefore all knives and forks are isolated) and $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \texttt{forky}(\mathtt{u})$, the number of predecessors of $\mathfrak{f}(\mathtt{u})$ (excluding $\mathfrak{f}(\mathtt{u})$ itself) in $\mathfrak{h}$ is less than or equal to $n + k$, i.e. $n \geq m - k$.

Now suppose that $n \geq m - k$ and the predecessors of $\mathfrak{f}(\mathtt{u})$ different from $\mathfrak{f}(\mathtt{u})$ are $\mathfrak{p}_1$, …, $\mathfrak{p}_m$. Let $\mathfrak{l}_1^1, \mathfrak{l}_1^2, \ldots, \mathfrak{l}_n^1, \mathfrak{l}_n^2$ be locations such that for every $i \in [1, n]$, we have $\mathfrak{h}(\mathfrak{l}_i^2) = \mathfrak{l}_i^1$ and $\mathfrak{l}_i^2 \to \mathfrak{l}_i^1$ is an isolated memory cell in $\mathfrak{h}$. Let us build $\mathfrak{h}'$ so that it satisfies (†), which is quite easy to realize. Let $\mathfrak{l}_1^{\text{new}}, \ldots, \mathfrak{l}_m^{\text{new}}$ be (new) locations that are not in $\text{dom}(\mathfrak{h}) \cup \text{ran}(\mathfrak{h}) \cup \{\mathfrak{f}(\mathtt{u})\}$. We define $\mathfrak{h}'$ so that it contains exactly $m - k$ knives whose endpoints are predecessors of $\mathfrak{f}(\mathtt{u})$. For every $i \in [1, m - k]$, we define $\mathfrak{h}'(\mathfrak{l}_i^{\text{new}}) \stackrel{\text{def}}{=} \mathfrak{l}_i^1$ and $\mathfrak{h}'(\mathfrak{l}_i^1) \stackrel{\text{def}}{=} \mathfrak{p}_i$ (which is possible because $\mathfrak{l}_i^1 \notin \text{dom}(\mathfrak{h})$). We add k additional forks to $\mathfrak{h}'$ whose endpoints are the remaining k predecessors of $\mathfrak{f}(\mathtt{u})$ different from $\mathfrak{f}(\mathtt{u})$ itself. It is easy to check that $\mathfrak{h}'$ satisfies (†).
Consequently, $\mathfrak{h} \models_{\mathfrak{f}} (\texttt{ksfs}_{=k} \mathrel{\overrightarrow{-\!\!*}} \texttt{forky}(\mathtt{u}))$ iff $n \geq m - k$. □

Let $\texttt{anti}(\mathtt{u}, \overline{\mathtt{u}})$ be $\texttt{antiforky}(\mathtt{u}) \wedge \texttt{antiknify}(\mathtt{u}) \wedge \texttt{antiforky}(\overline{\mathtt{u}}) \wedge \texttt{antiknify}(\overline{\mathtt{u}})$ and let $\texttt{comp}(\mathtt{u}, \overline{\mathtt{u}}, k, k')$ be defined as follows

$$\texttt{comp}(\mathtt{u}, \overline{\mathtt{u}}, k, k') \stackrel{\text{def}}{=} \big[(\texttt{seg} \wedge \sharp \mathtt{u} = 0 \wedge \sharp \overline{\mathtt{u}} = 0) \mathrel{-\!\!*}$$

$$\big(\texttt{anti}(\mathtt{u}, \overline{\mathtt{u}}) \Rightarrow \big(\big[\texttt{ksfs}_{=k} \mathrel{\overrightarrow{-\!\!*}} \texttt{forky}(\overline{\mathtt{u}})\big] \Rightarrow \big[\texttt{ksfs}_{=k'} \mathrel{\overrightarrow{-\!\!*}} \texttt{forky}(\mathtt{u})\big]\big)\big)\big].$$

PROPOSITION 4.3. *Let $k, k' \geq 0$, $\mathfrak{f}$ be a valuation, $\mathfrak{h}$ be a heap such that $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}}) \wedge \neg\exists\ \mathtt{u}\ \mathtt{isocell}(\mathtt{u})$, $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} - k' \geq 0$ and $\widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} - k \geq 0$. We have $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{comp}(\mathtt{u}, \overline{\mathtt{u}}, k, k')$ iff $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} + k \leq \widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} + k'$.*

Note that without any loss of generality we could assume that $k \times k' = 0$ but we provide below a uniform treatment that does not require to distinguish the case $k = 0$ from the case $k' = 0$. Moreover, the assumptions $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} - k' \geq 0$ and $\widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} - k \geq 0$ in Proposition 4.3 are required for the following reasons. In the proof of Proposition 4.3, we use the fact that for all $q, \overline{q} \in \mathbb{N}$, we have $q \leq \overline{q}$ iff for all $n \in \mathbb{N}$, we have $n \geq \overline{q}$ implies $n \geq q$. A similar property holds true for $\mathbb{Z}$, i.e., for all $q, \overline{q} \in \mathbb{Z}$, we have $q \leq \overline{q}$ iff for all $n \in \mathbb{Z}$, we have $n \geq \overline{q}$ implies $n \geq q$. However, with $q, \overline{q} \in \mathbb{Z}$, $q \leq \overline{q}$ is not equivalent to for all $n \in \mathbb{N}$, we have $n \geq \overline{q}$ implies $n \geq q$. That is why we need to assume that q ($\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} - k'$ from the statement of Proposition 4.3) and $\overline{q}$ ($\widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} - k$ in the statement of Proposition 4.3) belong to $\mathbb{N}$, if the quantification for n is over $\mathbb{N}$.

PROOF. Let $\mathfrak{h}$ be such that $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}}) \wedge \neg\exists\ \mathtt{u}\ \mathtt{isocell}(\mathtt{u})$. The statements below are equivalent.

(1) $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{comp}(\mathtt{u}, \overline{\mathtt{u}}, k, k')$.
(2) For every disjoint heap $\mathfrak{h}'$ such that $\mathfrak{h}' \models_{\mathfrak{f}} \mathtt{seg} \wedge \sharp\mathtt{u} = 0 \wedge \sharp\overline{\mathtt{u}} = 0$ and $\mathfrak{h} \uplus \mathfrak{h}' \models \mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}})$, if $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{ksfs}_{=k} \mathrel{-\!\!\!\rightarrow} \mathtt{forky}(\overline{\mathtt{u}})$, then $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{ksfs}_{=k'} \mathrel{-\!\!\!\rightarrow} \mathtt{forky}(\mathtt{u})$. (By definition of $\models_{\mathfrak{f}}$.)
(3) For every $n \geq 0$, let $\mathfrak{h}'$ be a heap disjoint from $\mathfrak{h}$ such that $\mathfrak{h} \uplus \mathfrak{h}'$ has n isolated memory cells, $\mathfrak{h}' \models_{\mathfrak{f}} \mathtt{seg} \wedge \sharp\mathtt{u} = 0 \wedge \sharp\overline{\mathtt{u}} = 0$ and $\mathfrak{h} \uplus \mathfrak{h}' \models \mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}})$, and if $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{ksfs}_{=k} \mathrel{-\!\!\!\rightarrow} \mathtt{forky}(\overline{\mathtt{u}})$, then $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{ksfs}_{=k'} \mathrel{-\!\!\!\rightarrow} \mathtt{forky}(\mathtt{u})$. (By using the fact that it is always possible to add a segmented heap to $\mathfrak{h}$ so that the resulting heap has n isolated memory cells and doesn't change predecessor counts at $\mathfrak{f}(\mathtt{u})$ and $\mathfrak{f}(\overline{\mathtt{u}})$.)
(4) For every $n \geq 0$, we have $n \geq \widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} - k$ in $\mathfrak{h}$ implies $n \geq \widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} - k'$ in $\mathfrak{h}$. (By Lemma 4.2.)
(5) $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} + k \leq \widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} + k'$.

□

Here is the main result in this section using $\mathtt{comp}(\mathtt{u}, \overline{\mathtt{u}}, k, k')$.

THEOREM 4.4. *For $k, k' \geq 0$, there is a formula ϕ in 1SL2($\mathrel{-\!\!\!\rightarrow}$) (of linear size in $k + k'$ and it can be built in space $\mathcal{O}(log(k + k'))$ such that for all $\mathfrak{h}$, $\mathfrak{f}$, we have ($\mathfrak{h} \models_{\mathfrak{f}} \phi$ iff $\widetilde{\sharp\mathfrak{f}(\mathtt{u})} + k \leq \widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})} + k'$).*

We denote such a formula ϕ as $\sharp\mathtt{u} + k \leq \sharp\overline{\mathtt{u}} + k'$ and, as usual, it easily extends to $<, \geq, >, =$. The structure of the proof of Theorem 4.4 is similar to the structure of the proof of [Brochenin et al. 2012, Theorem 5.5], except that now all formulae are in 1SL2($\mathrel{-\!\!\!\rightarrow}$) instead of being defined in the less-constrained 1SL($\mathrel{-\!\!\!\rightarrow}$). Moreover, our case analysis is quite different and we also provide several simplifications, making our new proof even more valuable.

PROOF. Below, we show that for $k, k' \geq 0$, there is a formula $\phi_{k,k'}$ in 1SL2($\mathrel{-\!\!\!\rightarrow}$) (of linear size in $k + k'$ and it can be built in space $\mathcal{O}(log(k + k'))$ such that for every heap $\mathfrak{h}$ and valuation $\mathfrak{f}$, we have $\mathfrak{h} \models_{\mathfrak{f}} \phi_{k,k'}$ iff $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} + k \leq \widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} + k'$. Once we have such a $\phi_{k,k'}$, we can define ϕ as:

$$(\mathtt{u} \hookrightarrow \mathtt{u} \wedge \overline{\mathtt{u}} \hookrightarrow \overline{\mathtt{u}} \wedge \phi_{k,k'}) \ \vee \ (\neg(\mathtt{u} \hookrightarrow \mathtt{u}) \wedge \overline{\mathtt{u}} \hookrightarrow \overline{\mathtt{u}} \wedge \phi_{k,k'+1}) \ \vee$$

$$(\mathtt{u} \hookrightarrow \mathtt{u} \wedge \neg(\overline{\mathtt{u}} \hookrightarrow \overline{\mathtt{u}}) \wedge \phi_{k+1,k'}) \ \vee \ (\neg(\mathtt{u} \hookrightarrow \mathtt{u}) \wedge \neg(\overline{\mathtt{u}} \hookrightarrow \overline{\mathtt{u}}) \wedge \phi_{k,k'}).$$

Observe that the four disjuncts are exclusive.

So it only remains to define the formulae $\phi_{k,k'}$ with $k, k' \geq 0$. By Proposition 4.3, we have the following property for any $\mathfrak{h}, \mathfrak{f}$:

(⋆). When $\mathfrak{h}$ satisfies $\mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}}) \wedge \neg\exists\ \mathtt{u}\ \mathtt{isocell}(\mathtt{u})$, $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} - k' \geq 0$ and $\widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} - k \geq 0$, we have $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{comp}(\mathtt{u}, \overline{\mathtt{u}}, k, k')$ iff $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} + k \leq \widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} + k'$.

Even though the original heap $\mathfrak{h}$ may not satisfy the formula $\mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}}) \wedge \neg\exists\ \mathtt{u}\ \mathtt{isocell}(\mathtt{u})$, it can be safely extended to satisfy this property without modifying the number of predecessors of $\mathfrak{f}(\mathtt{u})$ and $\mathfrak{f}(\overline{\mathtt{u}})$.

Whenever $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} - k' \geq 0$ and $\widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} - k \geq 0$, we have the following equivalences:

(1) $\mathfrak{h} \models_{\mathfrak{f}} (\sharp\mathtt{u} = 0 \wedge \sharp\overline{\mathtt{u}} = 0) \mathrel{-\!\!\!\!\!\rightarrow\!\!\!*} (\mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}}) \wedge (\neg\exists\ \mathtt{u}\ \mathtt{isocell}(\mathtt{u})) \wedge \mathtt{comp}(\mathtt{u}, \overline{\mathtt{u}}, k, k'))$.
(2) There is $\mathfrak{h}'$ disjoint from $\mathfrak{h}$ such that $\mathfrak{h}' \models_{\mathfrak{f}} (\sharp\mathtt{u} = 0 \wedge \sharp\overline{\mathtt{u}} = 0)$, $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}}) \wedge \neg\exists\ \mathtt{u}\ \mathtt{isocell}(\mathtt{u})$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{comp}(\mathtt{u}, \overline{\mathtt{u}}, k, k')$.
(3) There is $\mathfrak{h}'$ disjoint from $\mathfrak{h}$ such that $\mathfrak{h}' \models_{\mathfrak{f}} (\sharp\mathtt{u} = 0 \wedge \sharp\overline{\mathtt{u}} = 0)$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}}) \wedge \neg\exists\ \mathtt{u}\ \mathtt{isocell}(\mathtt{u})$ and $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} + k \leq \widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} + k'$ (in $\mathfrak{h} \uplus \mathfrak{h}'$) by (⋆).
(4) $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} + k \leq \widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} + k'$ in $\mathfrak{h}$.

Observe that $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star}$ and $\widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star}$ in $\mathfrak{h}$ are equal to their values in $\mathfrak{h} \uplus \mathfrak{h}'$ since $\mathfrak{h}' \models_{\mathfrak{f}} (\sharp\mathtt{u} = 0 \wedge \sharp\overline{\mathtt{u}} = 0)$. Moreover, (4) implies (3) since it is always possible to extend a model satisfying $\mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}}) \wedge \neg\exists\ \mathtt{u}\ \mathtt{isocell}(\mathtt{u})$ while preserving $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star}$ and $\widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star}$. By way of example, if $\mathfrak{l}$ is a predecessor of $\mathfrak{f}(\mathtt{u})$, endpoint of the fork $\mathfrak{l}_0$, $\mathfrak{l}_1$, $\mathfrak{l}_2$, $\mathfrak{l}$, then adding a memory cell $\mathfrak{l}_{new} \to \mathfrak{l}_2$ (assuming that $\mathfrak{l}_2 \neq \mathfrak{f}(\overline{\mathtt{u}})$) destroys the fork structure. When $\mathfrak{l}_2 = \mathfrak{f}(\overline{\mathtt{u}})$, we add $\mathfrak{l}_{new} \to \mathfrak{l}_1$ instead. Similarly, if $\mathfrak{f}(\overline{\mathtt{u}}) \to \mathfrak{f}(\mathtt{u})$ and that memory cell is isolated, then it is sufficient to add a new memory cell $\mathfrak{f}(\mathtt{u}) \to \mathfrak{l}_{new}$ with $\mathfrak{l}_{new}$ distinct from $\mathfrak{f}(\mathtt{u})$. The other cases can be treated in a similar fashion.

Without any loss of generality, we can assume that $k \times k' = 0$.

Case $k = 0$ *and* $k' \geq 0$. So $\widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} - k \geq 0$ and we need to make a case analysis depending on the satisfaction of $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} \geq k'$. Note that if $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} < k'$, then obviously $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} \leq \widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} + k'$. So, we write $\phi_{k,k'}$ to denote the formula below:

$$(\overset{\star}{\sharp}\mathtt{u} < k') \vee ((\overset{\star}{\sharp}\mathtt{u} \geq k') \wedge ((\sharp\mathtt{u} = 0 \wedge \sharp\overline{\mathtt{u}} = 0) \mathrel{-\!\!\!\!\!\rightarrow\!\!\!*}$$

$$(\mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}}) \wedge (\neg\exists\ \mathtt{u}\ \mathtt{isocell}(\mathtt{u})) \wedge \mathtt{comp}(\mathtt{u}, \overline{\mathtt{u}}, 0, k')))).$$

Formulae of the form $\overset{\star}{\sharp}\mathtt{u} \geq k'$ (and variants) can be defined thanks to Lemma 3.3.

Case $k' = 0$ *and* $k \geq 0$. So $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} - k' \geq 0$ and we need to make a case analysis depending on the satisfaction of $\widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} \geq k$. Note that if $\widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star} < k$, then $\widetilde{\sharp\mathfrak{f}(\mathtt{u})}^{\star} + k \leq \widetilde{\sharp\mathfrak{f}(\overline{\mathtt{u}})}^{\star}$ cannot hold. So, we write $\phi_{k,k'}$ to denote the formula below:

$$(\overset{\star}{\sharp}\overline{\mathtt{u}} \geq k) \wedge ((\sharp\mathtt{u} = 0 \wedge \sharp\overline{\mathtt{u}} = 0) \mathrel{-\!\!\!\!\!\rightarrow\!\!\!*}$$

$$(\mathtt{anti}(\mathtt{u}, \overline{\mathtt{u}}) \wedge (\neg\exists\ \mathtt{u}\ \mathtt{isocell}(\mathtt{u})) \wedge \mathtt{comp}(\mathtt{u}, \overline{\mathtt{u}}, k, 0))).$$

□

We can now find locations in a heap with a maximal number of predecessors, and we conclude this section with a definition useful in later constructions. Let us introduce the formula $\mathtt{maxdeg}(\mathtt{u})$:

$$\mathtt{maxdeg}(\mathtt{u}) \stackrel{\text{def}}{=} \neg\exists\, \overline{\mathtt{u}}\ \sharp\overline{\mathtt{u}} > \sharp\mathtt{u}.$$

COROLLARY 4.5. *For all* $\mathfrak{h}$ *and* $\mathfrak{f}$*, we have* $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{maxdeg}(\mathtt{u})$ *iff* $\widetilde{\sharp\mathfrak{f}(\mathtt{u})} = max(\{\widetilde{\sharp\mathfrak{l}} : \mathfrak{l} \in \mathbb{N}\})$.

We now have the necessary underlying machinery to tackle the encoding of second-order formulae in our fragment of separation logic.

5. EXPRESSIVE COMPLETENESS FOR 1SL2(−∗)

In order to express sentences in DSOL by sentences in 1SL2(−∗), a hybrid valuation is encoded in the heap by building a disjoint *valuation heap* that takes care of pairs of locations (for interpretation of second-order variables) and that takes care of locations (for interpretation of first-order variables). In principle, this makes sense since every heap has a finite domain and therefore there is always an infinite set of locations that is not in its domain. This leaves enough room to encode a finite amount of information such as the interpretation of second-order variables when they are interpreted by finite sets. We can easily add to the original heap with the magic wand; this permits us to create and update the valuation heap. However, we then must always be able to distinguish between the original heap and the valuation heap.

The main idea to build such a valuation heap rests on the fact that a pair of locations $(\mathfrak{l}, \mathfrak{l}')$ belongs to the interpretation of a second-order variable $\mathtt{P}_i$ whenever $\mathfrak{l}$ and $\mathfrak{l}'$ can be identified in the valuation heap by special patterns involving $\mathfrak{l}$ and $\mathfrak{l}'$ that uniquely characterise the interpretation by $\mathtt{P}_i$. Similarly, a location $\mathfrak{l}$ is the interpretation of a first-order variable whenever $\mathfrak{l}$ can be identified in the valuation heap thanks to some dedicated pattern around $\mathfrak{l}$.

Before explaining further the general principles, let us first provide more information about the above-mentioned patterns. An *entry of degree* $d \geq 2$ is a sequence of distinct locations $\mathfrak{l}_1, \ldots, \mathfrak{l}_d, \mathfrak{l}_{\mathtt{ind}}, \mathfrak{l}$ such that

— $\mathfrak{h}(\mathfrak{l}_1) = \cdots = \mathfrak{h}(\mathfrak{l}_d) = \mathfrak{l}_{\mathtt{ind}}$,
— $\widetilde{\sharp\mathfrak{l}_{\mathtt{ind}}} = d$,
— $\widetilde{\sharp\mathfrak{l}_1} = \cdots = \widetilde{\sharp\mathfrak{l}_d} = 0$, and
— $\mathfrak{h}(\mathfrak{l}_{\mathtt{ind}}) = \mathfrak{l}$.

The location $\mathfrak{l}$ is called the *element*, $\mathfrak{l}_{\mathtt{ind}}$ the *index* and the locations $\mathfrak{l}_1, \ldots, \mathfrak{l}_d$, the *pins*. Entries generalize the notions of forks and large forks from Section 3 and are called *markers* in [Brochenin et al. 2012]. See an entry of degree 4 in the middle of Figure 7. So, the pair of locations $(\mathfrak{l}, \mathfrak{l}')$ is identified as part of the interpretation of $\mathtt{P}_i$ when $\mathfrak{l}$ and $\mathfrak{l}'$ are elements of entries with very large degree. The above-mentioned special patterns are therefore entries, but we require that the degree of the respective entries for $\mathfrak{l}$ and $\mathfrak{l}'$ satisfy some arithmetical constraints, which is possible thanks to Theorem 4.4, and which allows us to relate $\mathfrak{l}$ with $\mathfrak{l}'$.

Then, the principle of the translation consists in building the valuation heap on demand (typically when a quantification appears) and to find special patterns involving entries with large degree whenever an atomic formula needs to be evaluated.

Apart from our essential restriction to 1SL2(−∗) and therefore the need for encoding also first-order valuations, these principles have been introduced in [Brochenin et al. 2012] to translate DSOL formulae into 1SL(−∗) formulae. However, because we are restricted to two first-order variables and because we also require that the separating conjunction is banished, we present below a different way to apply these principles so

that we can show that 1SL2($-\!\!*$) is expressively equivalent to DSOL (and therefore to WSOL).

This high-level description of the formula translation and of the encoding of some hybrid valuation in the heap hides many of the details, which can be found below. However, before explaining how we apply these principles within 1SL2($-\!\!*$), let us emphasize the most obvious and difficult problems to be solved:

(I). we must be able to distinguish the pairs of locations from distinct second-order variables,
(II). we also need to encode first-order valuations, and
(III). the main problem is certainly to access the original heap properly without interference from the valuation heap.

5.1. Left and right parentheses

We introduce variants of entries that are used as delimiters.

A *left j-parenthesis of degree $d \geq 3$ with $j \geq 0$* is a sequence of distinct locations $\mathfrak{l}'_{j+1}$, $\ldots$, $\mathfrak{l}'_1$, $\mathfrak{l}_1$, $\ldots$, $\mathfrak{l}_d$, $\mathfrak{l}_{\mathtt{ind}}$ such that

(u). $\mathfrak{h}(\mathfrak{l}_1) = \cdots = \mathfrak{h}(\mathfrak{l}_d) = \mathfrak{l}_{\mathtt{ind}}$; $\widetilde{\sharp \mathfrak{l}_{\mathtt{ind}}} = d$; $\widetilde{\sharp \mathfrak{l}'_{j+1}} = \widetilde{\sharp \mathfrak{l}_3} = \widetilde{\sharp \mathfrak{l}_4} = \cdots = \widetilde{\sharp \mathfrak{l}_d} = 0$,
(v). $\mathfrak{l}_{\mathtt{ind}} \notin \mathrm{dom}(\mathfrak{h})$; $\mathfrak{l}'_{j+1} \to \mathfrak{l}'_j \to \mathfrak{l}'_{j-1} \to \cdots \to \mathfrak{l}'_1 \to \mathfrak{l}_1$; $\widetilde{\sharp \mathfrak{l}'_j} = \widetilde{\sharp \mathfrak{l}'_{j-1}} = \cdots = \widetilde{\sharp \mathfrak{l}'_1} = \widetilde{\sharp \mathfrak{l}_1} = 1$, and
(w). $\widetilde{\sharp \mathfrak{l}_2} = 0$.

The location $\mathfrak{l}_{\mathtt{ind}}$ is called the *index*. The heap at the left of Figure 7 presents a left j-parenthesis of degree 3.

A *right j-parenthesis of degree $d \geq 3$ with $j \geq 0$* is a sequence of distinct locations $\mathfrak{l}'_{j+1}$, $\ldots$, $\mathfrak{l}'_1$, $\mathfrak{l}''_{j+1}$, $\ldots$, $\mathfrak{l}''_1$, $\mathfrak{l}_1$, $\ldots$, $\mathfrak{l}_d$, $\mathfrak{l}_{\mathtt{ind}}$ such that (u), (v), and

— $\widetilde{\sharp \mathfrak{l}''_{j+1}} = 0$,
— $\widetilde{\sharp \mathfrak{l}''_j} = \widetilde{\sharp \mathfrak{l}''_{j-1}} = \cdots = \widetilde{\sharp \mathfrak{l}''_1} = \widetilde{\sharp \mathfrak{l}_2} = 1$, and
— $\mathfrak{l}''_{j+1} \to \mathfrak{l}''_j \to \mathfrak{l}''_{j-1} \to \cdots \to \mathfrak{l}''_1 \to \mathfrak{l}_2$.

The location $\mathfrak{l}_{\mathtt{ind}}$ is also called the *index*. The heap at the right of Figure 7 presents a right j-parenthesis of degree 5. A j-parenthesis can be understood as an entry, except that the index location is not allocated, and containing one or two paths of length $j+1$, depending on whether it is a left or a right parenthesis.

LEMMA 5.1. *For all $j \geq 0$, there is a formula $\mathtt{lp}_j(\mathtt{u})$ [resp. $\mathtt{rp}_j(\mathtt{u})$] in 1SL2($-\!\!*$) such that for all heaps $\mathfrak{h}$ and valuations $\mathfrak{f}$, we have $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{lp}_j(\mathtt{u})$ [resp. $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{rp}_j(\mathtt{u})$] iff $\mathfrak{f}(\mathtt{u})$ is the index of some left [resp. right] j-parenthesis in $\mathfrak{h}$.*

By the proof below, one can add that $\mathtt{lp}_j(\mathtt{u})$ [resp. $\mathtt{rp}_j(\mathtt{u})$] is of linear size in j and it can be built in space $\mathcal{O}(log(j))$.

PROOF. Let us start by defining formulae for backward paths of length $j+1$:

— $\mathtt{bpath}(1, \mathtt{u}) \stackrel{\text{def}}{=} (\sharp \mathtt{u} = 1) \wedge \forall\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \Rightarrow \sharp \overline{\mathtt{u}} = 0$.
— $\mathtt{bpath}(j+1, \mathtt{u}) \stackrel{\text{def}}{=} (\sharp \mathtt{u} = 1) \wedge \exists\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \wedge \mathtt{bpath}(j, \overline{\mathtt{u}})$.

So, whenever $j \geq 0$, we have $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{bpath}(j+1, \mathtt{u})$ iff there are $\mathfrak{l}_0$, $\ldots$, $\mathfrak{l}_j$ such that $\mathfrak{l}_0 \to \mathfrak{l}_1 \to \cdots \to \mathfrak{l}_j \hookrightarrow \mathfrak{f}(\mathtt{u})$ and $\widetilde{\sharp \mathfrak{l}_0} = 0$, for every $k \in [1, j]$ $\widetilde{\sharp \mathfrak{l}_k} = 1$ and $\widetilde{\sharp \mathfrak{f}(\mathtt{u})} = 1$.

The formula below characterizes the locations such that the predecessors either have no predecessor or have a backward path of length $j+1$ exactly:

$$\varphi_{j+1}(\mathtt{u}) \stackrel{\text{def}}{=} \forall\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \Rightarrow (\sharp\overline{\mathtt{u}} = 0 \vee \mathtt{bpath}(j+1, \overline{\mathtt{u}})).$$

Then, the formulae $\mathtt{lp}_j(\mathtt{u})$ and $\mathtt{rp}_j(\mathtt{u})$ are defined as follows:

— $\mathtt{lp}_j(\mathtt{u}) \stackrel{\text{def}}{=} \neg\mathtt{alloc}(\mathtt{u}) \wedge \varphi_{j+1}(\mathtt{u}) \wedge (\sharp\mathtt{u} \geq 3) \wedge (\exists\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \wedge \mathtt{bpath}(j+1, \overline{\mathtt{u}})) \wedge ((\mathtt{size} = 1) \mathrel{-\!\!\!\overrightarrow{*}} \varphi_{j+2}(\mathtt{u}))$.
— $\mathtt{rp}_j(\mathtt{u}) \stackrel{\text{def}}{=} \neg\mathtt{alloc}(\mathtt{u}) \wedge \varphi_{j+1}(\mathtt{u}) \wedge (\sharp\mathtt{u} \geq 3) \wedge (\exists\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \wedge \mathtt{bpath}(j+1, \overline{\mathtt{u}})) \wedge \neg((\mathtt{size} = 1) \mathrel{-\!\!\!\overrightarrow{*}} \varphi_{j+2}(\mathtt{u})) \wedge ((\mathtt{size} = 1) \mathrel{-\!\!\!\overrightarrow{*}} ((\mathtt{size} = 1) \mathrel{-\!\!\!\overrightarrow{*}} \varphi_{j+2}(\mathtt{u})))$.

The formula $\mathtt{lp}_j(\mathtt{u})$ states that $\mathfrak{f}(\mathtt{u})$ is not allocated, it has at least three predecessors and any predecessor of $\mathfrak{f}(\mathtt{u})$ either has no predecessor or has a backward path of length $j+1$. Moreover, there is at least one predecessor of $\mathfrak{f}(\mathtt{u})$ that has a backward path of length $j+1$ thanks to the satisfaction of the subformula $(\exists\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \wedge \mathtt{bpath}(j+1, \overline{\mathtt{u}}))$. Satisfaction of the subformula $(\mathtt{size} = 1) \mathrel{-\!\!\!\overrightarrow{*}} \varphi_{j+2}(\mathtt{u})$ entails that there is only one such backward path of length $j+1$. A similar analysis can be performed with the formula $\mathtt{rp}_j(\mathtt{u})$ with the exception that it is required to guarantee that there are exactly two predecessors of $\mathfrak{f}(\mathtt{u})$ that have a backward path of length $j+1$. □

In several places, we need to identify the indices from entries as well as their pins. Let $\mathtt{eindex}(\mathtt{u})$ be defined as follows:

$$\mathtt{eindex}(\mathtt{u}) \stackrel{\text{def}}{=} (\sharp_z\mathtt{u} \geq 2) \wedge \mathtt{allzpred}(\mathtt{u}) \wedge \exists\, \overline{\mathtt{u}}\ \mathtt{u} \hookrightarrow \overline{\mathtt{u}}$$

that characterises indices from entries. Let $\mathtt{epin}(\mathtt{u})$ characterise pins from entries: $\mathtt{epin}(\mathtt{u}) \stackrel{\text{def}}{=} \exists\, \overline{\mathtt{u}}\ \mathtt{u} \hookrightarrow \overline{\mathtt{u}} \wedge \mathtt{eindex}(\overline{\mathtt{u}})$. Similarly, we need to characterise the locations from parentheses. We already know how to identify their indices (Lemma 5.1). It remains to identify the other locations via the formula $\mathtt{onpar}_i(\mathtt{u})$ to characterise the locations on some i-parenthesis: roughly speaking, such locations are exactly those that can reach the index of some i-parenthesis in less than $i+2$ steps. Let $\mathtt{onpar}_i(\mathtt{u})$ be the formula

$$\mathtt{onpar}_i(\mathtt{u}) \stackrel{\text{def}}{=} \bigvee_{j=0}^{i+2} \mathtt{dist}_i(j, \mathtt{u})$$

where $\mathtt{dist}_i(j, \mathtt{u})$ is defined as follows:

$$\begin{array}{lcl} \mathtt{dist}_i(0, \mathtt{u}) & \stackrel{\text{def}}{=} & \mathtt{lp}_i(\mathtt{u}) \vee \mathtt{rp}_i(\mathtt{u}) \\ \mathtt{dist}_i(j+1, \mathtt{u}) & \stackrel{\text{def}}{=} & \exists\, \overline{\mathtt{u}}\ (\mathtt{u} \hookrightarrow \overline{\mathtt{u}}) \wedge \mathtt{dist}_i(j, \overline{\mathtt{u}})\ (j \geq 0). \end{array}$$

A rough analysis leads to a construction of $\mathtt{onpar}_i(\mathtt{u})$ in space $\mathcal{O}(log(i))$ and $\mathtt{onpar}_i(\mathtt{u})$ is of quadratic size in i.

LEMMA 5.2. *Let $\mathfrak{h}$ be a heap, $\mathfrak{f}$ be a valuation and $i \geq 0$. Then, $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{onpar}_i(\mathtt{u})$ iff $\mathfrak{f}(\mathtt{u})$ is on some left or right i-parenthesis in $\mathfrak{h}$.*

PROOF. The proof takes advantage of the following properties.

— $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{dist}_i(j, \mathtt{u})$ iff $\mathfrak{f}(\mathtt{u})$ can reach an index location from a left i-parenthesis or from a right i-parenthesis, in j steps for some $j \geq 0$. The proof is obvious, by induction on j.
— If $\mathfrak{f}(\mathtt{u})$ can reach an index location from an i-parenthesis, then $\mathfrak{f}(\mathtt{u})$ is necessarily on an i-parenthesis.
— Every location on an i-parenthesis can reach its index in less than $i+2$ steps.

As a conclusion, $\mathfrak{f}(\mathtt{u})$ is on an i-parenthesis iff it can reach the index of some i-parenthesis in less than $i+2$ steps, which is exactly the way $\mathtt{onpar}_i(\mathtt{u})$ is defined with the help of the generalized disjunction. □

5.2. The role of parentheses

Before explaining the role of parentheses, we introduce the interval of variable indices $[1, K]$ ($K \in \mathbb{N} \setminus \{0\}$) assuming that for each $j \in [1, K]$, either $\mathtt{P}_j$ or $\mathtt{u}_j$ occurs in the DSOL formula to be translated (but not both of them). So, the developments below are relative to a finite set of first-order and second-order variables and this is concretized by the interval $[1, K]$ (always possible since a formula has a finite number of variables).

Let us come back to parentheses and assume that X is a subset of $[0, K]$. In an X-well-formed heap $\mathfrak{h}$ (see Definition 5.8 below), the parentheses play the following role. For each $j \in X$, we have the index location $\mathfrak{lp}_j$ from a distinguished left j-parenthesis and the index location $\mathfrak{rp}_j$ from a distinguished right j-parenthesis. Moreover, let $d^l_j = \widetilde{\sharp \mathfrak{lp}_j}$ and $d^r_j = \widetilde{\sharp \mathfrak{rp}_j}$ (in $\mathfrak{h}$). When $j \in X$ is related to a first-order variable, we require that $d^r_j = d^l_j + 2$ and there is an entry of degree $d^l_j + 1$ such that its element is understood as the interpretation of the variable $\mathtt{u}_j$ (see Figure 7 with $d^r_j = 5$ and $d^l_j = 3$). That explains why the parentheses are viewed as delimiters. Similarly, let $\{(\mathfrak{l}_1, \mathfrak{l}'_1), \ldots, (\mathfrak{l}_\beta, \mathfrak{l}'_\beta)\}$ be a finite set of pairs of locations, understood as the interpretation of a second-order variable $\mathtt{P}_j$ with $j \in X$. In $\mathfrak{h}$, there are 2β entries whose respective degrees are exactly

$$\{d^l_j + 3(i-1) + 1, d^l_j + 3(i-1) + 2 : i \in [1, \beta]\}$$

with $d^r_j = d^l_j + 3\beta + 1$. A pair of entries of respective degrees $d^l_j + 3(i-1) + 1$ and $d^l_j + 3(i-1) + 2$ have exactly as elements $\mathfrak{l}_i$ and $\mathfrak{l}'_i$ respectively, which allows to encode the pair $(\mathfrak{l}_i, \mathfrak{l}'_i)$. All this underlying encoding makes sense only if the left and right parentheses as well as the entries whose degrees are related to their degrees are uniquely determined (see Condition (1) in Definition 5.4, below). For this reason, we introduce a left 0-parenthesis and a right 0-parenthesis with $d^r_0 = d^l_0 + 1$ (0 is not a variable index), the degree d^l_0 is strictly greater than the degree of any location in the original heap, all degrees d^l_j with $j \neq 0$ are strictly greater than d^l_0 and finally, the above-mentioned entries and parentheses are the only ones with their respective degrees. This guarantees that any entry from a pair of entries with successive degrees serving for the interpretation of a second-order variable, cannot serve twice for another pair or for another variable. Below, we provide the technical developments.

We say that a heap $\mathfrak{h}$ is *made of entries and parentheses only* $\stackrel{\text{def}}{\Leftrightarrow}$ every location in $\mathrm{dom}(\mathfrak{h})$ belongs either to a left i-parenthesis for some $i \geq 0$, to a right i-parenthesis for some $i \geq 0$, or to an entry. Given a heap $\mathfrak{h}$ made of entries and parentheses only, we define the set $\mathit{indspect}(\mathfrak{h})$ as follows:

$$\mathit{indspect}(\mathfrak{h}) \stackrel{\text{def}}{=} \{\widetilde{\sharp \mathfrak{l}} : \mathfrak{l} \textit{ is the index of some entry or parenthesis in } \mathfrak{h}\}.$$

The (finite) set $\mathit{indspect}(\mathfrak{h})$ is called the *index spectrum* of $\mathfrak{h}$.

Let $\mathfrak{h}_B$ be a heap such that $\alpha = \mathit{max}(\{\widetilde{\sharp \mathfrak{l}} : \mathfrak{l} \in \mathbb{N}\})$. For instance, if $\mathfrak{h}_B$ has empty domain, then $\alpha = 0$. We have seen in Section 4 that it is possible to characterise the locations that witness this maximal value α thanks to the formula $\mathtt{maxdeg}(\mathtt{u})$. A valuation heap $\mathfrak{h}_V$ for $\mathfrak{h}_B$ is made of entries and parentheses only whose degrees are greater than $\mathit{max}(3, \alpha + 1)$. The heap $\mathfrak{h}_V$ satisfies the following simple conditions (more constraints will follow): $\mathit{min}(\mathit{indspect}(\mathfrak{h}_V))$ is greater than $\mathit{max}(3, \alpha + 1)$ and it is witnessed by the degree of some left 0-parenthesis; each degree in $\mathit{indspect}(\mathfrak{h}_V)$ is witnessed by exactly one entry or parenthesis. The formula $\mathtt{indmin}(\mathtt{u})$ below is satisfied in $\mathfrak{h} = \mathfrak{h}_B \uplus \mathfrak{h}_V$

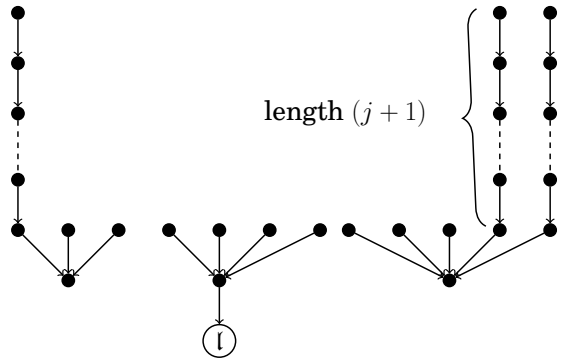


Fig. 7: Encoding $[\mathtt{u}_j \mapsto \mathfrak{l}]$.

by a location $\mathfrak{l}$ witnessing the minimal value in $indspect(\mathfrak{h}_V)$:

$$\mathtt{indmin}(\mathtt{u}) \stackrel{\text{def}}{=} \mathtt{lp}_0(\mathtt{u}) \wedge (\forall\, \overline{\mathtt{u}}\ ((\overline{\mathtt{u}} \neq \mathtt{u}) \wedge \mathtt{lp}_0(\overline{\mathtt{u}})) \Rightarrow \sharp\overline{\mathtt{u}} < \sharp\mathtt{u}).$$

Note that thanks to Section 4, we know that it is possible to compare numbers of predecessors as expressed above. So, $\mathtt{indmin}(\mathtt{u})$ holds when $\mathfrak{f}(\mathtt{u})$ is the unique location that is the index of some left 0-parenthesis with greatest degree.

LEMMA 5.3. *Let $\mathfrak{f}$ be a valuation and $\mathfrak{h}$ be a heap. We have $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{indmin}(\mathtt{u})$ iff $\mathfrak{f}(\mathtt{u})$ is an index of some left 0-parenthesis and there is no other location $\mathfrak{l} \neq \mathfrak{f}(\mathtt{u})$ such that $\widetilde{\sharp\mathfrak{l}} \geq \widetilde{\sharp\mathfrak{f}(\mathtt{u})}$ and $\mathfrak{l}$ is the index of some left 0-parenthesis.*

The proof is by an easy verification by using Lemma 5.1. Once a heap $\mathfrak{h}$ satisfies $\exists\, \mathtt{u}\ \mathtt{indmin}(\mathtt{u})$, the unique location $\mathfrak{l}_0$ such that $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}_0]} \mathtt{indmin}(\mathtt{u})$ (say with $\widetilde{\sharp\mathfrak{l}_0} = d_0$) plays the role of a delimiter between the original heap and the part of the heap that encodes the hybrid valuation. We have seen that an index spectrum is defined for heaps made of entries and parentheses only. This is fine, but below we adapt the definition to heaps $\mathfrak{h}$ satisfying $\exists\, \mathtt{u}\ \mathtt{indmin}(\mathtt{u})$. Let us define the set $spect(\mathfrak{h})$ as follows:

$$spect(\mathfrak{h}) \stackrel{\text{def}}{=} \{\widetilde{\sharp\mathfrak{l}} : \mathfrak{l} \textit{ is an index of some entry or parenthesis in } \mathfrak{h}\} \cap [d_0, +\infty[.$$

The (finite) set $spect(\mathfrak{h})$ is called the *spectrum* of $\mathfrak{h}$. This illustrates how the location $\mathfrak{l}_0$ and the degree $\widetilde{\sharp\mathfrak{l}_0} = d_0$ play the role of separator between the original heap and the valuation heap.

The subheap encoding the valuation is made of parentheses and entries and we shall need to identify the indices of such patterns. The formula $\mathtt{Lindex}(\mathtt{u})$ defined below suffices for this purpose:

$$\mathtt{Lindex}(\mathtt{u}) \stackrel{\text{def}}{=} (\exists\, \overline{\mathtt{u}}\ \mathtt{indmin}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} \leq \sharp\mathtt{u}) \wedge \Big(\Big(\bigvee_{i\in[0,K]} (\mathtt{lp}_i(\mathtt{u}) \vee \mathtt{rp}_i(\mathtt{u}))\Big) \vee \mathtt{eindex}(\mathtt{u})\Big)$$

($\mathtt{u}$ is interpreted as a *large index*). Given $X \subseteq [0, K]$, we shall use also the following formula:

$$\mathtt{Lindex}_X(\mathtt{u}) \stackrel{\text{def}}{=} (\exists\, \overline{\mathtt{u}}\ \mathtt{indmin}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} \leq \sharp\mathtt{u}) \wedge \Big(\Big(\bigvee_{i\in X} (\mathtt{lp}_i(\mathtt{u}) \vee \mathtt{rp}_i(\mathtt{u}))\Big) \vee \mathtt{eindex}(\mathtt{u})\Big).$$

Entries and parentheses with large indices are also called large entries and parentheses, respectively. The elements in $[1, K]$ will be later used as variable indices. Note also that $\mathtt{Lindex}(\mathtt{u})$ is of quadratic size in K and it can be built in space $\mathcal{O}(log(K))$. It is

easy to define a large index that is also the index of a left [resp. right] parenthesis. Let $\mathtt{Llp}_i(\mathtt{u}) \stackrel{\text{def}}{=} \mathtt{Lindex}(\mathtt{u}) \wedge \mathtt{lp}_i(\mathtt{u})$ and $\mathtt{Lrp}_i(\mathtt{u}) \stackrel{\text{def}}{=} \mathtt{Lindex}(\mathtt{u}) \wedge \mathtt{rp}_i(\mathtt{u})$ (see Lemma 5.1). The large index with a maximal degree can be also characterised as follows:

$$\mathtt{maxLindex}(\mathtt{u}) \stackrel{\text{def}}{=} (\forall\, \overline{\mathtt{u}}\ \mathtt{Lindex}(\overline{\mathtt{u}}) \Rightarrow (\sharp\overline{\mathtt{u}} \leq \sharp\mathtt{u})) \wedge \mathtt{Lindex}(\mathtt{u}).$$

Below, we state how the parentheses are organized.

Definition 5.4. Let $X = \{i_0, \ldots, i_s\} \subseteq [0, K]$ with $0 = i_0 < i_1 < \cdots < i_s$. A heap $\mathfrak{h}$ is *X-almost-well-formed* $\stackrel{\text{def}}{\Leftrightarrow}$

(1) For every $j \in [0, s]$, there is a unique location $\mathfrak{l}^l_j$ [resp. $\mathfrak{l}^r_j$] such that $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}^l_j]} \mathtt{Llp}_{i_j}(\mathtt{u})$ [resp. $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}^r_j]} \mathtt{Lrp}_{i_j}(\mathtt{u})$].

(2) For every $j \in [0, s]$, $\widetilde{\sharp\mathfrak{l}^l_j} < \widetilde{\sharp\mathfrak{l}^r_j}$, and $\widetilde{\sharp\mathfrak{l}^r_0} = \widetilde{\sharp\mathfrak{l}^l_0} + 1$.

(3) For every $j \in [1, s]$, we have $\widetilde{\sharp\mathfrak{l}^l_j} = \widetilde{\sharp\mathfrak{l}^r_{j-1}} + 1$.

(4) $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}^r_s]} \mathtt{maxLindex}(\mathtt{u})$.

(5) For every $j \in [1, s]$, if i_j is the index of a first-order variable, then $\widetilde{\sharp\mathfrak{l}^l_j} = \widetilde{\sharp\mathfrak{l}^r_j} - 2$ (see Figure 7).

(6) For every $j \in ([1, K] \setminus X)$, there is no location $\mathfrak{l}$ such that $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}]} \mathtt{Llp}_j(\mathtt{u}) \vee \mathtt{Lrp}_j(\mathtt{u})$.

The definition for X-almost-well-formed heaps mainly specifies the existence of j-parentheses with $j \in X$ and how their respective degrees are related. The degrees are organized as follows and they all belong to the spectrum of $\mathfrak{h}$ (below we let $d^l_j = \widetilde{\sharp\mathfrak{l}^l_j}$ and $d^r_j = \widetilde{\sharp\mathfrak{l}^r_j}$).

$$\begin{array}{ccccccccccccccc}
\models_{[\mathtt{u} \mapsto d^l_0]} \mathtt{indmin}(\mathtt{u}) & & & & & & & & & & & & \models_{[\mathtt{u} \mapsto d^r_s]} \mathtt{maxLindex}(\mathtt{u}) \\
d^l_0 & < & d^r_0 & < & d^l_1 & < & d^r_1 & < & d^l_2 & < & d^r_2 & < \cdots < & d^l_s & < & d^r_s \\
 & & \| & & \| & & & & \| & & & & \| & & \\
 & & d^l_0 + 1 & & d^r_0 + 1 & & & & d^r_1 + 1 & & & & d^r_{s-1} + 1 & &
\end{array}$$

Moreover, when i_j is the index of a first-order variable, we have $d^r_j = d^l_j + 2$.

LEMMA 5.5. *There exists a formula $\mathtt{awfh}_X$ in 1SL2(−∗) of cubic size in K (and it can be built in space $\mathcal{O}(log(K))$) such that $\mathfrak{h} \models \mathtt{awfh}_X$ iff $\mathfrak{h}$ is X-almost-well-formed.*

PROOF. We consider the conjunction of the formulae below, each of which deals with one of the conditions. The cubic size of $\mathtt{awfh}_X$ is essentially due to the fact that a linear amount of formulae (in K) is built and each formula is of quadratic size (here, we use the size properties of already introduced subformulae). Below, arithmetical constraints can be expressed thanks to Theorem 4.4 and its immediate consequences. Condition (1) is taken care by the formula below

$$\bigwedge_{j \in [0,s]} [\exists\, \mathtt{u}\ \mathtt{Llp}_{i_j}(\mathtt{u}) \wedge \neg(\exists\, \overline{\mathtt{u}}\ \mathtt{Llp}_{i_j}(\overline{\mathtt{u}}) \wedge \mathtt{u} \neq \overline{\mathtt{u}})] \wedge \bigwedge_{j \in [0,s]} [\exists\, \mathtt{u}\ \mathtt{Lrp}_{i_j}(\mathtt{u}) \wedge \neg(\exists\, \overline{\mathtt{u}}\ \mathtt{Lrp}_{i_j}(\overline{\mathtt{u}}) \wedge \mathtt{u} \neq \overline{\mathtt{u}})].$$

Condition (2) is expressed as follows:

$$\bigwedge_{j \in [0,s]} [\exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ \mathtt{Llp}_{i_j}(\mathtt{u}) \wedge \mathtt{Lrp}_{i_j}(\overline{\mathtt{u}}) \wedge (\sharp\mathtt{u} < \sharp\overline{\mathtt{u}})] \wedge [\exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ \mathtt{Llp}_0(\mathtt{u}) \wedge \mathtt{Lrp}_0(\overline{\mathtt{u}}) \wedge (\sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 1)].$$

Condition (3) is dealt with

$$\bigwedge_{j\in[1,s]} [\exists\ \mathtt{u}\ \exists\ \overline{\mathtt{u}}\ \mathtt{Llp}_{i_j}(\mathtt{u}) \wedge \mathtt{Lrp}_{i_{j-1}}(\overline{\mathtt{u}}) \wedge (\sharp\mathtt{u} = \sharp\overline{\mathtt{u}} + 1)].$$

Condition (4) is expressed by $\exists\ \mathtt{u}\ \mathtt{Lrp}_{i_s}(\mathtt{u}) \wedge \mathtt{maxLindex}(\mathtt{u})$.
Condition (5) is easily expressed by the formula below:

$$\bigwedge_{j\in[1,s],\ \mathrm{FO}\ i_j} [\exists\ \mathtt{u}\ \exists\ \overline{\mathtt{u}}\ \mathtt{Llp}_{i_j}(\mathtt{u}) \wedge \mathtt{Lrp}_{i_j}(\overline{\mathtt{u}}) \wedge (\sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 2)].$$

Because of the unicity of the left or right parentheses, alternative formulae can be defined by using universal quantifiers instead of existential ones. By way of example, the respective formulae expressing the conditions (3) and (5) with universal quantifiers are the following:

$$\bigwedge_{j\in[1,s]} [\forall\ \mathtt{u}\ \forall\ \overline{\mathtt{u}}\ (\mathtt{Llp}_{i_j}(\mathtt{u}) \wedge \mathtt{Lrp}_{i_{j-1}}(\overline{\mathtt{u}})) \Rightarrow (\sharp\mathtt{u} = \sharp\overline{\mathtt{u}} + 1)].$$

$$\bigwedge_{j\in[1,s],\ \mathrm{FO}\ i_j} [\forall\ \mathtt{u}\ \forall\ \overline{\mathtt{u}}\ (\mathtt{Llp}_{i_j}(\mathtt{u}) \wedge \mathtt{Lrp}_{i_j}(\overline{\mathtt{u}})) \Rightarrow (\sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 2)].$$

Finally, Condition (6) is expressed by:

$$\bigwedge_{j\in[1,K]\setminus X} \neg\exists\ \mathtt{u}\ (\mathtt{Llp}_j(\mathtt{u}) \vee \mathtt{Lrp}_j(\mathtt{u})).$$

□

Let $\mathfrak{h}$ be an X-almost-well-formed heap for some $\{0\} \subseteq X \subseteq [0, K]$ and $i \in X$. We write $\mathtt{vind}_i(\mathtt{u})$ to denote the formula below:

$$\mathtt{Lindex}(\mathtt{u}) \wedge \mathtt{eindex}(\mathtt{u}) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{Llp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} < \sharp\mathtt{u}) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{Lrp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} > \sharp\mathtt{u}).$$

It characterises indices whose degree is strictly between the degree of some large left i-parenthesis and the degree of some large right i-parenthesis. The size of $\mathtt{vind}_i(\mathtt{u})$ is quadratic in K. We write $degrees(i, \mathfrak{h})$ to denote the set:

$$degrees(i, \mathfrak{h}) \stackrel{\text{def}}{=} \{\widetilde{\sharp\mathfrak{l}} \in \mathbb{N} : \mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} \mathtt{vind}_i(\mathtt{u}),\ \mathfrak{l} \in \mathbb{N}\}.$$

LEMMA 5.6. *Let $\mathfrak{h}$ be a heap such that $\mathfrak{h} \models \exists\ \mathtt{u}\ \mathtt{indmin}(\mathtt{u})$ and $i \geq 0$ be such that there are unique locations $\mathfrak{lp}$ and $\mathfrak{rp}$ with $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{lp}]} \mathtt{Llp}_i(\mathtt{u})$ and $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{rp}]} \mathtt{Lrp}_i(\mathtt{u})$. For every $\mathfrak{l} \in \mathbb{N}$, we have $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} \mathtt{vind}_i(\mathtt{u})$ iff $\mathfrak{l}$ is the index of some entry and $\widetilde{\sharp\mathfrak{lp}} < \widetilde{\sharp\mathfrak{l}} < \widetilde{\sharp\mathfrak{rp}}$.*

PROOF. Let $\mathfrak{h}$ be a heap such that:

(1) $\mathfrak{h} \models \exists\ \mathtt{u}\ \mathtt{indmin}(\mathtt{u})$ with $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}_0]} \mathtt{indmin}(\mathtt{u})$ and $\widetilde{\sharp\mathfrak{l}_0} = d_0$.
(2) There is a unique index location $\mathfrak{lp}$ such that $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{lp}]} \mathtt{Llp}_i(\mathtt{u})$ and $\widetilde{\sharp\mathfrak{lp}} \geq d_0$.
(3) There is a unique index location $\mathfrak{rp}$ such that $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{rp}]} \mathtt{Lrp}_i(\mathtt{u})$ and $\widetilde{\sharp\mathfrak{rp}} \geq d_0$.

First, suppose that $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} \mathtt{vind}_i(\mathtt{u})$. By definition of $\mathtt{vind}_i(\mathtt{u})$, we have

$$\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} \mathtt{Lindex}(\mathtt{u}) \wedge \mathtt{eindex}(\mathtt{u}) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{Llp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} < \sharp\mathtt{u}) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{Lrp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} > \sharp\mathtt{u}).$$

Since $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} \mathtt{Lindex}(\mathtt{u}) \wedge \mathtt{eindex}(\mathtt{u})$, $\mathfrak{l}$ is the index of some entry whose degree is greater than d_0. Since $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} (\exists\ \overline{\mathtt{u}}\ \mathtt{Llp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} < \sharp\mathtt{u})$, by (2) we have $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l},\overline{\mathtt{u}}\mapsto\mathfrak{lp}]} \sharp\overline{\mathtt{u}} < \sharp\mathtt{u}$ and therefore $\widetilde{\sharp\mathfrak{lp}} < \widetilde{\sharp\mathfrak{l}}$. Similarly, $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} (\exists\ \overline{\mathtt{u}}\ \mathtt{Lrp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} > \sharp\mathtt{u})$, by (3), we have

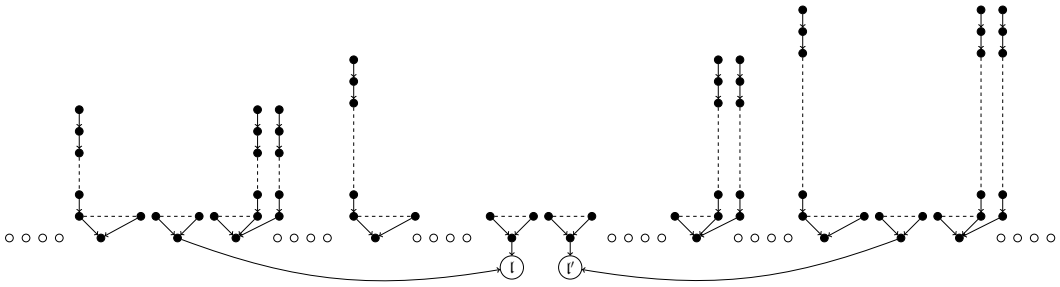


Fig. 8: How the translation of $\mathtt{P}_i(\mathtt{u}_j, \mathtt{u}_k)$ works ($j < i < k$): $(\mathfrak{l}, \mathfrak{l}') \in \mathfrak{V}_{\mathfrak{h}}(\mathtt{P}_i)$.

$\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l},\overline{\mathtt{u}}\mapsto\mathfrak{r}\mathfrak{p}]} \sharp\overline{\mathtt{u}} > \sharp\mathtt{u}$ and therefore $\widetilde{\sharp\mathfrak{l}} < \widetilde{\sharp\mathfrak{r}\mathfrak{p}}$. Consequently, $\mathfrak{l}$ is the index of some entry and $\widetilde{\sharp\mathfrak{l}\mathfrak{p}} < \widetilde{\sharp\mathfrak{l}} < \widetilde{\sharp\mathfrak{r}\mathfrak{p}}$.

Now, suppose that $\mathfrak{l}$ is the index of some entry and $\widetilde{\sharp\mathfrak{l}\mathfrak{p}} < \widetilde{\sharp\mathfrak{l}} < \widetilde{\sharp\mathfrak{r}\mathfrak{p}}$. Since $\mathfrak{l}$ is some index, we have $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} \mathtt{eindex}(\mathtt{u})$. By (2), we get that $d_0 < \widetilde{\sharp\mathfrak{l}}$ and therefore $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} \mathtt{eindex}(\mathtt{u}) \wedge \exists\ \overline{\mathtt{u}}\ \mathtt{indmin}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} \leq \sharp\mathtt{u}$, which implies $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} \mathtt{Lindex}(\mathtt{u})$ by definition of $\mathtt{Lindex}(\mathtt{u})$. By assumption, we have $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l},\overline{\mathtt{u}}\mapsto\mathfrak{l}\mathfrak{p}]} \sharp\overline{\mathtt{u}} < \sharp\mathtt{u}$ and $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l},\overline{\mathtt{u}}\mapsto\mathfrak{r}\mathfrak{p}]} \sharp\overline{\mathtt{u}} > \sharp\mathtt{u}$ and consequently, $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} (\exists\ \overline{\mathtt{u}}\ \mathtt{Llp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} < \sharp\mathtt{u})$ and $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} (\exists\ \overline{\mathtt{u}}\ \mathtt{Lrp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} > \sharp\mathtt{u})$. So, all the conjuncts of $\mathtt{vind}_i(\mathtt{u})$ are satisfied and therefore $\mathfrak{h} \models_{[\mathtt{u}\mapsto\mathfrak{l}]} \mathtt{vind}_i(\mathtt{u})$. $\square$

The formula $\mathtt{elt}_j(\mathtt{u})$ defined below holds true when $\mathtt{u}$ is interpreted as the element of the unique entry attached to the first-order variable $\mathtt{u}_j$.

$$\mathtt{elt}_j(\mathtt{u}) \stackrel{\text{def}}{=} \exists\ \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \wedge \mathtt{vind}_j(\overline{\mathtt{u}}).$$

Let us anticipate a little how the translation from DSOL to 1SL2($*$) works: the translation of $\mathtt{P}_i(\mathtt{u}_j, \mathtt{u}_k)$ can be designed as follows:

$$\exists\ \mathtt{u}\ (\mathtt{elt}_j(\mathtt{u})\ \wedge\ \exists\ \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u}\ \wedge\ \mathtt{vind}_i(\overline{\mathtt{u}}) \wedge$$

$$\exists\ \mathtt{u}\ (\sharp\mathtt{u} = \sharp\overline{\mathtt{u}} + 1\ \wedge\ \mathtt{vind}_i(\mathtt{u})\ \wedge\ \exists\ \overline{\mathtt{u}}\ (\mathtt{u} \hookrightarrow \overline{\mathtt{u}}\ \wedge\ \mathtt{elt}_k(\overline{\mathtt{u}}))))).$$

These definitions take advantage of the fact that there are unique large left and right parentheses for each variable index. Figure 8 illustrates the constraints satisfied by the formula when $j < i < k$. From left to right, the figure represents explicitly a left j-parenthesis, then a right j-parenthesis, then a left i-parenthesis, a right i-parenthesis and a left k-parenthesis, followed finally by a right k-parenthesis. Other entries and parentheses are present in the figure, but they are represented by dots in order to focus on the memory cells relevant to evaluate the formula obtained by the translation of $\mathtt{P}_i(\mathtt{u}_j, \mathtt{u}_k)$. The degrees of parentheses and entries increase from left to right.

5.3. Taking care of valuations

Now that we have a way of identifying that part of the heap that encodes our valuation, we turn our attention to encoding the valuation itself. Below, we introduce a condition for a subheap to be "glued" to an existing valuation. We distinguish three cases.

— A *local* 0*-valuation* is a heap made of a left 0-parenthesis of degree d and a right 0-parenthesis of degree $d+1$ only, for some $d \geq 3$.

— Let $i \in [1, K]$ be the index of some first-order variable. A *local i-valuation* is a heap made of a left i-parenthesis of degree d, an entry of degree $d+1$ and a right i-parenthesis of degree $d+2$ only, for some $d \geq 3$.
— Let $i \in [1, K]$ be the index of some second-order variable. A *local i-valuation* is a heap $\mathfrak{h}$ such that
(1) every location $\mathfrak{l}$ in $\mathrm{dom}(\mathfrak{h})$ belongs either to a left i-parenthesis, to a right i-parenthesis, or to an entry,
(2) $\mathfrak{h}$ contains a unique left [resp. right] i-parenthesis,
(3) the minimal value $min(indspect(\mathfrak{h}))$ is the degree of some left i-parenthesis,
(4) the maximal value $max(indspect(\mathfrak{h}))$ is the degree of some right i-parenthesis,
(5) the index spectrum $indspect(\mathfrak{h})$ is of the form below for some $\alpha \geq 3$, $\beta \geq 0$,

$$\{\alpha\} \cup \{\alpha + 3(i-1) + 1, \alpha + 3(i-1) + 2 : i \in [1, \beta]\} \cup \{\alpha + 3\beta + 1\}$$

(when $\beta = 0$, $indspect(\mathfrak{h})$ is equal to $\{\alpha, \alpha+1\}$),
(6) there are no two distinct index locations with the same degree.

Note that $indspect(\mathfrak{h}) \subseteq [\alpha, \alpha+3\beta+1]$ and the missing values in $[\alpha, \alpha+3\beta+1] \setminus indspect(\mathfrak{h})$ are precisely those in the set $\{\alpha+3(i-1)+3 : i \in [1, \beta]\}$, i.e. in $\{\alpha+3i : i \in [1, \beta]\}$. Since local i-valuations are typically heaps that are added to the current heap to encode the interpretation of a variable, it is essential to be able to characterise them by 1SL2($-\!\!*$) formulae. This is the purpose of the result below.

LEMMA 5.7. *Let $i \in [0, K]$. There is a formula $\mathtt{localval}_i(\mathtt{u})$ in 1SL2($-\!\!*$) (of quadratic size in K and it can be built in space $\mathcal{O}(log(K))$) such that $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{localval}_i(\mathtt{u})$ iff $\mathfrak{h}$ is a local i-valuation and $\mathfrak{f}(\mathtt{u})$ is the index of its left i-parenthesis.*

PROOF. For characterising local 0-valuations, it is sufficient to express the properties below:

(1) any location in the domain is on some left or on some right 0-parenthesis,
(2) there is exactly one left 0-parenthesis whose index is $\mathfrak{f}(\mathtt{u})$,
(3) there is exactly one right 0-parenthesis,
(4) $\widetilde{\sharp\mathfrak{l}} = \widetilde{\sharp\mathfrak{f}(\mathtt{u})} + 1$ where $\mathfrak{l}$ is the index of the unique right 0-parenthesis.

(1)-(4) can be expressed by the formula below:

$$(\forall\, \mathtt{u}\ \mathtt{alloc}(\mathtt{u}) \Rightarrow \mathtt{onpar}_0(\mathtt{u})) \wedge (\mathtt{lp}_0(\mathtt{u}) \wedge \neg(\exists\, \overline{\mathtt{u}}\ \mathtt{lp}_0(\overline{\mathtt{u}}) \wedge \mathtt{u} \neq \overline{\mathtt{u}})) \wedge$$

$$(\exists\, \overline{\mathtt{u}}\ (\mathtt{rp}_0(\overline{\mathtt{u}}) \wedge \neg(\exists\, \mathtt{u}\ \mathtt{rp}_0(\mathtt{u}) \wedge \mathtt{u} \neq \overline{\mathtt{u}})) \wedge (\sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 1)).$$

Formulae of the form $\mathtt{lp}_i(\mathtt{u})$ and $\mathtt{rp}_i(\mathtt{u})$ are provided in the proof of Lemma 5.1 whereas formulae of the form $\mathtt{onpar}_i(\mathtt{u})$ are provided before the statement of Lemma 5.2.

For characterising local i-valuations for some first-order variable $\mathtt{u}_i$, it is sufficient to express the properties below:

(1) any location in the domain is on some left i-parenthesis, or on some right i-parenthesis or on some entry,
(2) there is exactly one left i-parenthesis whose index is $\mathfrak{f}(\mathtt{u})$,
(3) there is exactly one right i-parenthesis,
(4) there is a unique entry, whose degree is d, such that $\widetilde{\sharp\mathfrak{l}} = \widetilde{\sharp\mathfrak{f}(\mathtt{u})} + 2$ and $\widetilde{\sharp\mathfrak{f}(\mathtt{u})} = d - 1$ where $\mathfrak{l}$ is the index of the unique right i-parenthesis.

(1)-(4) can be expressed by the formula below:

$$(\forall\ \mathtt{u}\ \mathtt{alloc}(\mathtt{u}) \Rightarrow \mathtt{onpar}_i(\mathtt{u}) \vee \mathtt{epin}(\mathtt{u}) \vee \mathtt{eindex}(\mathtt{u})) \wedge$$

$$(\mathtt{lp}_i(\mathtt{u}) \wedge \neg(\exists\ \overline{\mathtt{u}}\ \mathtt{lp}_i(\overline{\mathtt{u}}) \wedge \mathtt{u} \neq \overline{\mathtt{u}})) \wedge (\exists\ \overline{\mathtt{u}}\ (\mathtt{rp}_i(\overline{\mathtt{u}}) \wedge \neg(\exists\ \mathtt{u}\ \mathtt{rp}_i(\mathtt{u}) \wedge \mathtt{u} \neq \overline{\mathtt{u}}) \wedge (\sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 2)) \wedge$$

$$(\exists\ \overline{\mathtt{u}}\ \mathtt{eindex}(\overline{\mathtt{u}}) \wedge (\neg\exists \mathtt{u}\ (\mathtt{u} \neq \overline{\mathtt{u}}) \wedge \mathtt{eindex}(\mathtt{u})) \wedge (\sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 1)).$$

For characterising local i-valuations for some second-order variable $\mathtt{P}_i$, it is sufficient to express the properties below:

(1) any location in the domain is on some left i-parenthesis, or on some right i-parenthesis or on some entry,
(2) there is exactly one left i-parenthesis whose index is $\mathfrak{f}(\mathtt{u})$,
(3) there is exactly one right i-parenthesis whose index is the location $\mathfrak{l}$,
(4) any entry has degree in $[\widetilde{\sharp\mathfrak{f}(\mathtt{u})} + 1, \widetilde{\sharp\mathfrak{l}} - 1]$ and its index is the unique one with that degree,
(5) $\widetilde{\sharp\mathfrak{l}} > \widetilde{\sharp\mathfrak{f}(\mathtt{u})}$,
(6) if $\widetilde{\sharp\mathfrak{l}} > \widetilde{\sharp\mathfrak{f}(\mathtt{u})} + 1$, then
  (a) there is an entry with degree $\widetilde{\sharp\mathfrak{f}(\mathtt{u})} + 1$,
  (b) there is an entry with degree $\widetilde{\sharp\mathfrak{f}(\mathtt{u})} + 2$,
  (c) if there are entries with respective degree d and $d+1$, then there is no entry or right i-parenthesis of degree $d+2$,
  (d) if there are entries with respective degree d and $d+1$ and $d+3 < \widetilde{\sharp\mathfrak{l}}$, then there are entries of respective degree $d+3$ and $d+4$.

(1)–(5) can be expressed by the formula below:

$$(\forall\ \mathtt{u}\ \mathtt{alloc}(\mathtt{u}) \Rightarrow \mathtt{onpar}_i(\mathtt{u}) \vee \mathtt{epin}(\mathtt{u}) \vee \mathtt{eindex}(\mathtt{u})) \wedge$$

$$(\mathtt{lp}_i(\mathtt{u}) \wedge \neg(\exists\ \overline{\mathtt{u}}\ \mathtt{lp}_i(\overline{\mathtt{u}}) \wedge \mathtt{u} \neq \overline{\mathtt{u}})) \wedge (\exists\ \overline{\mathtt{u}}\ (\mathtt{rp}_i(\overline{\mathtt{u}}) \wedge \neg(\exists\ \mathtt{u}\ \mathtt{rp}_i(\mathtt{u}) \wedge \mathtt{u} \neq \overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} > \sharp\mathtt{u})) \wedge$$

$$\forall\ \mathtt{u}\ \mathtt{eindex}(\mathtt{u}) \Rightarrow \neg(\exists\ \overline{\mathtt{u}}\ ((\mathtt{eindex}(\overline{\mathtt{u}}) \vee \mathtt{lp}_i(\overline{\mathtt{u}}) \vee \mathtt{rp}_i(\overline{\mathtt{u}}))\ \wedge \sharp\mathtt{u} = \sharp\overline{\mathtt{u}}) \wedge$$

$$(\exists\ \overline{\mathtt{u}}\ \mathtt{rp}_i(\overline{\mathtt{u}}) \wedge (\forall\ \mathtt{u}\ \mathtt{eindex}(\mathtt{u}) \Rightarrow \sharp\mathtt{u} < \sharp\overline{\mathtt{u}} - 1)) \wedge (\forall\ \overline{\mathtt{u}}\ \mathtt{eindex}(\overline{\mathtt{u}}) \Rightarrow \sharp\overline{\mathtt{u}} > \sharp\mathtt{u}).$$

(6a)–(6d) can be expressed by the formula below:

$$(\exists\ \overline{\mathtt{u}}\ \mathtt{rp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} > \sharp\mathtt{u} + 1) \Rightarrow$$

$$(\exists\ \overline{\mathtt{u}}\ \mathtt{eindex}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 1) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{eindex}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 2) \wedge$$

$$\forall\ \mathtt{u}\ (\mathtt{eindex}(\mathtt{u}) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{eindex}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 1)) \Rightarrow \neg(\exists\ \overline{\mathtt{u}}\ (\mathtt{eindex}(\overline{\mathtt{u}}) \vee \mathtt{rp}_i(\overline{\mathtt{u}})) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 2) \wedge$$

$$\forall\ \mathtt{u}\ (\mathtt{eindex}(\mathtt{u}) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{eindex}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 1) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{rp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} > \sharp\mathtt{u} + 3)) \Rightarrow$$

$$((\exists\ \overline{\mathtt{u}}\ \mathtt{eindex}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 3) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{eindex}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 4)).$$

□

The definition for X-almost-well-formed heaps mainly takes care of parentheses. In Definition 5.8 below, constraints on the degrees of large indices are specified.

Definition 5.8. Let $X = \{i_0, \ldots, i_s\} \subseteq [0, K]$ with $0 = i_0 < i_1 < \cdots < i_s$. A heap $\mathfrak{h}$ is *X-well-formed* $\stackrel{\text{def}}{\Leftrightarrow}$ the following conditions hold:

(1) $\mathfrak{h}$ is X-almost-well-formed,
(2) for every $j \in [1,s]$, if i_j is the index of a first-order variable, then $\mathit{degrees}(i_j, \mathfrak{h})$ is a singleton,
(3) for every $j \in [1,s]$, if i_j is the index of a second-order variable, then $\mathit{degrees}(i_j, \mathfrak{h})$ is the set below for some $\alpha_j \geq 3$, $\beta_j \geq 0$:

$$\{\alpha_j + 3(i-1) + 1, \alpha_j + 3(i-1) + 2 : i \in [1, \beta_j]\},$$

(4) for every location $\mathfrak{l}$ such that $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}]} \mathtt{Lindex}(\mathtt{u})$, there is no $\mathfrak{l}' \neq \mathfrak{l}$ such that $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}']} \mathtt{Lindex}(\mathtt{u})$ and $\widetilde{\sharp \mathfrak{l}} = \widetilde{\sharp \mathfrak{l}'}$.
(5) If a location has degree greater than the degree of the unique large left 0-parenthesis, then it is a large index.

When $\mathfrak{h}$ is X-well-formed, we write $\mathfrak{h} = \mathfrak{h}_B \uplus \mathfrak{h}_V$ such that $\mathrm{dom}(\mathfrak{h}_V)$ is made of entries and parentheses of degree $d \geq \widetilde{\sharp \mathfrak{l}_0}$ for some $\mathfrak{l}_0 \in \mathbb{N}$ such that $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}_0]} \mathtt{indmin}(\mathtt{u})$ (i.e., $\mathfrak{l}_0$ is the index of the left 0-parenthesis with the maximal degree). More precisely, $\mathfrak{l} \in \mathrm{dom}(\mathfrak{h}_V)$ iff $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}]} \exists\, \overline{\mathtt{u}}\, (\mathtt{reach}(\mathtt{u}, \overline{\mathtt{u}}) \wedge \mathtt{Lindex}_X(\overline{\mathtt{u}}))$. By Definition 5.8, we have $\mathit{spect}(\mathfrak{h}) = \mathit{indspect}(\mathfrak{h}_V)$ and clearly the decomposition is unique since $\mathfrak{l}_0$ is unique.

Again, well-formed heaps can be characterised by formulae in 1SL2($\multimap$) whose size is cubic in K.

LEMMA 5.9. *Given $\{0\} \subseteq X \subseteq [0,K]$, there is a formula $\mathtt{wfh}_X$ in 1SL2($\multimap$) of cubic size in K (and it can be built in space $\mathcal{O}(log(K))$) such that $\mathfrak{h} \models \mathtt{wfh}_X$ iff $\mathfrak{h}$ is X-well-formed.*

PROOF. We consider the conjunction of the formulae below, each of them deals with one of the five conditions. Condition (1) is obviously taken care by the formula $\mathtt{awfh}_X$ (see the proof of Lemma 5.5). Condition (2) is dealt with the formula below:

$$\bigwedge_{j \in [1,s],\ \mathrm{FO}\ i_j} \exists\, \mathtt{u}\ \mathtt{vind}_{i_j}(\mathtt{u}).$$

Note that since the heap is already X-almost-well-formed, at most one location can satisfy the above existential quantification for each FO variable index i_j. Similarly, Condition (3) is taken care by the formula below (see the proof of Lemma 5.7 and more specifically Condition (6) in that proof with indices from second-order variables):

$$\bigwedge_{j \in [1,s],\ \mathrm{SO}\ i_j} (\exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ \mathtt{Llp}_{i_j}(\mathtt{u}) \wedge \mathtt{Lrp}_{i_j}(\overline{\mathtt{u}}) \wedge (\sharp\overline{\mathtt{u}} > \sharp\mathtt{u} + 1)) \Rightarrow$$

$$\overbrace{[(\exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ \mathtt{Llp}_{i_j}(\mathtt{u}) \wedge \mathtt{vind}_{i_j}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 1)}^{\text{(a)}} \wedge$$

$$\overbrace{(\exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ \mathtt{Llp}_{i_j}(\mathtt{u}) \wedge \mathtt{vind}_{i_j}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 2)}^{\text{(b)}} \wedge$$

$$\overbrace{\forall\, \mathtt{u}\ (\mathtt{vind}_{i_j}(\mathtt{u}) \wedge (\exists\, \overline{\mathtt{u}}\ \mathtt{vind}_{i_j}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 1)) \ \Rightarrow\ \neg(\exists\, \overline{\mathtt{u}}\ (\mathtt{vind}_{i_j}(\overline{\mathtt{u}}) \vee \mathtt{rp}_{i_j}(\overline{\mathtt{u}})) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 2)}^{\text{(c)}} \wedge$$

$$\overbrace{\forall\, \mathtt{u}\ (\mathtt{vind}_{i_j}(\mathtt{u}) \wedge (\exists\, \overline{\mathtt{u}}\ \mathtt{vind}_{i_j}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 1) \wedge (\exists\, \overline{\mathtt{u}}\ \mathtt{Lrp}_{i_j}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} > \sharp\mathtt{u} + 3)) \Rightarrow}^{\text{(d)}}$$

$$((\exists\ \overline{\mathtt{u}}\ \mathtt{vind}_{i_j}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 3) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{vind}_{i_j}(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 4))].$$

The above formula expresses the conditions below, mimicking Condition (6) from the proof of Lemma 5.7:

(a) there is an entry with degree $d^l + 1$ where d^l is the degree of the unique left i_j-parenthesis,
(b) there is an entry with degree $d^l + 2$,
(c) if there are entries with respective degree d and $d + 1$ in $degrees(i_j, \mathfrak{h})$, then there is no entry or right i_j-parenthesis of degree $d + 2$ in $degrees(i_j, \mathfrak{h})$,
(d) if there are entries with respective degree d and $d + 1$ in $degrees(i_j, \mathfrak{h})$ and $d + 3 < d^r$ where d^r is the degree of the unique right i_j-parenthesis, then there are entries of respective degree $d + 3$ and $d + 4$ in $degrees(i_j, \mathfrak{h})$.

Condition (4) is expressed as follows by simply internalizing the condition in 1SL2($*$):

$$\forall\ \mathtt{u}\ \mathtt{Lindex}(\mathtt{u}) \Rightarrow \neg(\exists\ \overline{\mathtt{u}}\ \mathtt{Lindex}(\overline{\mathtt{u}}) \wedge (\mathtt{u} \neq \overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} = \sharp\mathtt{u}).$$

Condition (5) can be expressed as follows:

$$\forall\ \mathtt{u}\ (\exists\ \overline{\mathtt{u}}\ \mathtt{indmin}(\overline{\mathtt{u}}) \wedge \sharp\mathtt{u} \geq \sharp\overline{\mathtt{u}}) \Rightarrow \mathtt{Lindex}_X(\mathtt{u}).$$

□

Let us define formally a valuation from a valuation heap.

Definition 5.10. Let $\mathfrak{h}$ be an X-well-formed heap for some $\{0\} \subseteq X \subseteq [0, K]$.

— For every second-order $i \in X$, we define

$$\mathfrak{V}_{\mathfrak{h}}(\mathtt{P}_i) \stackrel{\text{def}}{=} \{(\mathfrak{h}_V(\mathfrak{l}), \mathfrak{h}_V(\mathfrak{l}')) : \widetilde{\sharp\mathfrak{l}'} = \widetilde{\sharp\mathfrak{l}} + 1,\ \widetilde{\sharp\mathfrak{l}}, \widetilde{\sharp\mathfrak{l}'} \in degrees(i, \mathfrak{h}), \mathfrak{l}, \mathfrak{l}' \text{ are index locations}\}.$$

— For every first-order $i \in X$, $\mathfrak{V}_{\mathfrak{h}}(\mathtt{u}_i) \stackrel{\text{def}}{=} \mathfrak{h}_V(\mathfrak{l})$ where $\mathfrak{l}$ is the unique index location such that $\widetilde{\sharp\mathfrak{l}} \in degrees(i, \mathfrak{h})$.

We say that $\mathfrak{V}_{\mathfrak{h}}$ is the valuation *extracted* from $\mathfrak{h}$.

Below, we present an essential technical result stating how heaps can be composed when a new variable needs to be interpreted. The formulae involved to compose the X-well-formed heap $\mathfrak{h}$ and the local i-valuation heap $\mathfrak{h}'$ are directly used in the translation of quantified formulae (see Section 5.4). Lemma 5.11 is used in the proof of Lemma 5.12.

LEMMA 5.11 (COMPOSITION). *Let $\mathfrak{f}$ be a valuation, $\mathfrak{h}$ be an X-well-formed heap with $\{0\} \subseteq X \subseteq [0, K]$, $i \in [1, K] \setminus X$ with $i > max(X)$, and $\mathfrak{h}'$ be a disjoint heap such that:*

(1) $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}})$,
(2) $\mathfrak{h}' \models_{\mathfrak{f}} \mathtt{localval}_i(\overline{\mathtt{u}})$,
(3) $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{Llp}_i(\overline{\mathtt{u}})$.

Then, $spect(\mathfrak{h} \uplus \mathfrak{h}') = spect(\mathfrak{h}) \uplus indspect(\mathfrak{h}')$.

Roughly speaking, Lemma 5.11 states that given an X-well-formed heap $\mathfrak{h}$, adding a disjoint local i-valuation $\mathfrak{h}'$ with $i \notin X$, leads to an $(X \cup \{i\})$-well-formed heap so that the interpretation of variables with variable indices in X from the extracted valuation, is the same with $\mathfrak{h}$ and with $\mathfrak{h} \uplus \mathfrak{h}'$. The heap $\mathfrak{h} \uplus \mathfrak{h}'$ can be then understood as a *conservative* extension of the heap $\mathfrak{h}$.

The proof of Lemma 5.11 is quite combinatorial and this is the place where we check that the original heap cannot be confused with the valuation heap (and the other way

around). It is important to guarantee, as the proof does, that adding a new part of the valuation does not destroy what has been built so far.

The proof itself is made of an imbrication of case analyses and it takes advantage of our notions of well-formed heaps and local valuations.

Alternative definitions are probably possible (maybe even simpler ones) but their correctness and usefulness should be tested against the satisfaction of Lemma 5.11. We believe that our definitions allow to have relatively clear proofs while avoiding the boredom of repetitive arguments.

PROOF. Let $X = \{i_0, \ldots, i_s\}$ with $0 = i_0 < i_1 < \cdots < i_s$. Since $\mathfrak{h}$ is X-well-formed, for every $j \in X$, there is a location $\mathfrak{lp}_j$ [resp. $\mathfrak{rp}_j$] such that $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{lp}_j]} \mathtt{Llp}_j(\mathtt{u})$ [resp. $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{rp}_j]} \mathtt{Lrp}_j(\mathtt{u})$]. For instance, each $\mathfrak{lp}_j$ is the index of some left j-parenthesis. Similarly, since $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{wfh}_{X \cup \{i\}}$ (so by Lemma 5.9, $\mathfrak{h} \uplus \mathfrak{h}'$ is $(X \cup \{i\})$-well-formed), for every $j \in (X \cup \{i\})$, there is a location $\mathfrak{lp}'_j$ [resp. $\mathfrak{rp}'_j$] such that $\mathfrak{h} \uplus \mathfrak{h}' \models_{[\mathtt{u} \mapsto \mathfrak{lp}'_j]} \mathtt{Llp}_j(\mathtt{u})$ [resp. $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{rp}'_j]} \mathtt{Lrp}_j(\mathtt{u})$].

By Definition 5.8, we have the following inequalities in $\mathfrak{h}$:

$$\widetilde{\sharp\mathfrak{lp}_{i_0}} < \widetilde{\sharp\mathfrak{rp}_{i_0}} < \widetilde{\sharp\mathfrak{lp}_{i_1}} < \widetilde{\sharp\mathfrak{rp}_{i_1}} < \cdots < \widetilde{\sharp\mathfrak{lp}_{i_s}} < \widetilde{\sharp\mathfrak{rp}_{i_s}}. \tag{1}$$

Similarly, we have the following inequalities in $\mathfrak{h} \uplus \mathfrak{h}'$:

$$\widetilde{\sharp\mathfrak{lp}'_{i_0}} < \widetilde{\sharp\mathfrak{rp}'_{i_0}} < \widetilde{\sharp\mathfrak{lp}'_{i_1}} < \widetilde{\sharp\mathfrak{rp}'_{i_1}} < \cdots < \widetilde{\sharp\mathfrak{lp}'_{i_s}} < \widetilde{\sharp\mathfrak{rp}'_{i_s}} < \widetilde{\sharp\mathfrak{lp}'_{i}} < \widetilde{\sharp\mathfrak{rp}'_{i}}. \tag{2}$$

One of the goals of the proof is to show that $(\mathfrak{lp}_{i_0}, \ldots, \mathfrak{rp}_{i_s}) = (\mathfrak{lp}'_{i_0}, \ldots, \mathfrak{rp}'_{i_s})$. Below, we establish the properties (I)–(VIII) and then we show that $\mathit{spect}(\mathfrak{h} \uplus \mathfrak{h}') = \mathit{spect}(\mathfrak{h}) \uplus \mathit{indspect}(\mathfrak{h}')$.

(I) Let us show that $\mathfrak{lp}_0 = \mathfrak{lp}'_0$ and $\widetilde{\sharp\mathfrak{lp}_0}$ in $\mathfrak{h}$ is equal to $\widetilde{\sharp\mathfrak{lp}_0}$ in $\mathfrak{h} \uplus \mathfrak{h}'$. Since $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{indmin}(\mathtt{u})$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{indmin}(\mathtt{u})$, we have $\mathfrak{f}(\mathtt{u}) = \mathfrak{lp}_0 = \mathfrak{lp}'_0$. It remains to prove that $\widetilde{\sharp\mathfrak{lp}_0}$ in $\mathfrak{h}$ is equal to $\widetilde{\sharp\mathfrak{lp}_0}$ in $\mathfrak{h} \uplus \mathfrak{h}'$.

(1) For every $\mathfrak{l} \in \mathrm{dom}(\mathfrak{h})$ such that $\mathfrak{h}(\mathfrak{l}) = \mathfrak{lp}_0$, there is no $\mathfrak{l}' \in \mathrm{dom}(\mathfrak{h}')$ so that $\mathfrak{h}'(\mathfrak{l}') = \mathfrak{l}$ (otherwise $\mathfrak{lp}_0$ is not anymore the index of a 0-parenthesis in $\mathfrak{h} \uplus \mathfrak{h}'$). Consequently, $\widetilde{\sharp\mathfrak{lp}_0}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ is not strictly less than $\widetilde{\sharp\mathfrak{lp}_0}$ in $\mathfrak{h}$.

(2) Let us show now that $\widetilde{\sharp\mathfrak{lp}_0}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ is not strictly more than $\widetilde{\sharp\mathfrak{lp}_0}$ in $\mathfrak{h}$. *Ad absurdum*, suppose that there is a location $\mathfrak{l}$ such that $\mathfrak{h}'(\mathfrak{l}) = \mathfrak{lp}_0$ and $\mathfrak{l}$ has no predecessor in $\mathfrak{h} \uplus \mathfrak{h}'$. So, in $\mathfrak{h}'$, the location $\mathfrak{l}'$ is necessarily the pin of the left i-parenthesis, or the pin of the right i-parenthesis or the pin of some entry. In all these cases, this implies that $\mathfrak{lp}_0$ cannot be anymore the index of a 0-parenthesis in $\mathfrak{h} \uplus \mathfrak{h}'$, which leads to a contradiction.

Consequently, $\widetilde{\sharp\mathfrak{lp}_0}$ in $\mathfrak{h}$ is equal to $\widetilde{\sharp\mathfrak{lp}_0}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ and $\mathfrak{lp}'_0 = \mathfrak{lp}_0$.

(II) Any location in $\mathrm{dom}(\mathfrak{h}')$ belongs either to a left or right i-parenthesis, or to an entry (this is an obvious consequence of the assumption (2) and Lemma 5.7). So, for any $\mathfrak{l} \in \mathrm{dom}(\mathfrak{h}')$, we can associate a unique index location $\mathfrak{l}_{ind}$. We say that a location $\mathfrak{l} \in \mathrm{dom}(\mathfrak{h}')$ associated to the index location $\mathfrak{l}_{ind}$ *wrongly contributes* to a large entry [resp. parenthesis] in $\mathfrak{h} \uplus \mathfrak{h}' \stackrel{\text{def}}{\Leftrightarrow}$ $\mathfrak{l}$ belongs to a large parenthesis [resp. entry] in $\mathfrak{h} \uplus \mathfrak{h}'$ with index location $\mathfrak{l}'_{ind}$ distinct from $\mathfrak{l}_{ind}$. Below we show that no location in $\mathrm{dom}(\mathfrak{h}')$ wrongly contributes to a large entry/parenthesis in $\mathfrak{h} \uplus \mathfrak{h}'$ and that the right situation is actually when $\mathfrak{l}'_{ind} = \mathfrak{l}_{ind}$. Indeed, if $\mathfrak{l}$ is an ancestor of $\mathfrak{l}_{ind}$ in $\mathfrak{h}'$, adding a disjoint heap

preserves that property. So if $\mathfrak{l}$ belongs to an entry, left or right parenthesis in $\mathfrak{h} \uplus \mathfrak{h}'$, the only possible index is $\mathfrak{l}_{ind}$. So, $\mathfrak{l}$ cannot wrongly contribute.

Furthermore, if $\mathfrak{l}$ belongs to $\mathrm{ran}(\mathfrak{h}')$ and $\mathfrak{l} \in \mathrm{dom}(\mathfrak{h}_V)$ with $\mathfrak{h} = \mathfrak{h}_B \uplus \mathfrak{h}_V$ (such a decomposition is possible because $\mathfrak{h}$ is X-well-formed), and the associated index location of $\mathfrak{l}$ in $\mathfrak{h}_V$ is $\mathfrak{l}_{ind}$, then $\mathfrak{l}_{ind}$ is not anymore an index in $\mathfrak{h} \uplus \mathfrak{h}'$. So glueing two components from $\mathfrak{h}'$ and $\mathfrak{h}_V$ respectively, cannot lead to a new component (with possibly a different index).

As a consequence, an allocated location in $\mathfrak{h}'$ that is not an index cannot be transformed into an index in $\mathfrak{h} \uplus \mathfrak{h}'$.

(III) Consequently, any large index $\mathfrak{l}$ in $\mathfrak{h}$ (either from some entry or from some parenthesis) either remains a large index in $\mathfrak{h} \uplus \mathfrak{h}'$ of the same type and degree, or $\mathfrak{l}$ is not anymore part of some entry and parenthesis. So, a large index in $\mathfrak{h}$ cannot be transformed into another type of large index in $\mathfrak{h} \uplus \mathfrak{h}'$. Roughly speaking, the type of an index is determined by the degree and whether the index is from an entry, from a left j-parenthesis or from a right j-parenthesis for some j.

(IV) $\widetilde{\sharp \mathfrak{l}\mathfrak{p}'_i}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ is equal to $\widetilde{\sharp \mathfrak{l}\mathfrak{p}'_i}$ in $\mathfrak{h}'$. This is a consequence of $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{isoloc}(\overline{\mathfrak{u}})$, $\mathfrak{h}' \models_{\mathfrak{f}} \mathtt{localval}_i(\overline{\mathfrak{u}})$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{Llp}_i(\overline{\mathfrak{u}})$.

(V) Let us show that $(\mathfrak{l}\mathfrak{p}_{i_0}, \ldots, \mathfrak{r}\mathfrak{p}_{i_s}) = (\mathfrak{l}\mathfrak{p}'_{i_0}, \ldots, \mathfrak{r}\mathfrak{p}'_{i_s})$, and for every $j \in [0, s]$, $\widetilde{\sharp \mathfrak{l}\mathfrak{p}_{i_j}}$ in $\mathfrak{h}$ is equal to $\widetilde{\sharp \mathfrak{l}\mathfrak{p}_{i_j}}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ and $\widetilde{\sharp \mathfrak{r}\mathfrak{p}_{i_j}}$ in $\mathfrak{h}$ is equal to $\widetilde{\sharp \mathfrak{r}\mathfrak{p}_{i_j}}$ in $\mathfrak{h} \uplus \mathfrak{h}'$. Suppose that there is j such that $\mathfrak{l}\mathfrak{p}_{i_j} \neq \mathfrak{l}\mathfrak{p}'_{i_j}$. By (III), the left i_j-parenthesis with index $\mathfrak{l}\mathfrak{p}'_{i_j}$ cannot be built from (III). By (IV), it cannot be built from the left parenthesis in $\mathfrak{h}'$ neither from the entries in $\mathfrak{h}'$ (because each index is allocated in entries). Similarly, the left i_j-parenthesis with index $\mathfrak{l}\mathfrak{p}'_{i_j}$ cannot be built from the right i-parenthesis from $\mathfrak{h}'$ because it contains already two paths of length $i+1$. Similarly, suppose that there is j such that $\mathfrak{r}\mathfrak{p}_{i_j} \neq \mathfrak{r}\mathfrak{p}'_{i_j}$. By (III), the right i_j-parenthesis with index $\mathfrak{r}\mathfrak{p}'_{i_j}$ cannot be built from (III). By (IV), it cannot be built from the left parenthesis in $\mathfrak{h}'$ neither from the entries in $\mathfrak{h}'$ (because each index is allocated in entries). Similarly, the right i_j-parenthesis with index $\mathfrak{r}\mathfrak{p}'_{i_j}$ cannot be built from the right i-parenthesis from $\mathfrak{h}'$ because of the length of the two paths. Additionally, by (III), for every $j \in [0, s]$, $\widetilde{\sharp \mathfrak{l}\mathfrak{p}_{i_j}}$ in $\mathfrak{h}$ is equal to $\widetilde{\sharp \mathfrak{l}\mathfrak{p}_{i_j}}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ and $\widetilde{\sharp \mathfrak{r}\mathfrak{p}_{i_j}}$ in $\mathfrak{h}$ is equal to $\widetilde{\sharp \mathfrak{r}\mathfrak{p}_{i_j}}$ in $\mathfrak{h} \uplus \mathfrak{h}'$.

(2holes) $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{wfh}_{X \cup \{i\}}$ implies that there are no $d, d+1 \in [min, max]$ where min is equal to $\widetilde{\sharp \mathfrak{l}\mathfrak{p}'_{i_0}}$ in $\mathfrak{h} \uplus \mathfrak{h}'$, max is equal to $\widetilde{\sharp \mathfrak{r}\mathfrak{p}'_i}$ in $\mathfrak{h} \uplus \mathfrak{h}'$, such that neither d nor $d+1$ belongs to $spect(\mathfrak{h} \uplus \mathfrak{h}')$. This is a direct consequence of the conditions in Definition 5.4 and in Definition 5.8. Similarly, there are no $d, d+1 \in [min', max']$ where min' is equal to $\widetilde{\sharp \mathfrak{l}\mathfrak{p}_i}$ in $\mathfrak{h}'$ ($\mathfrak{l}\mathfrak{p}_i$ is the index of the unique left i-parenthesis), max' is equal to $\widetilde{\sharp \mathfrak{r}\mathfrak{p}_i}$ in $\mathfrak{h}'$ ($\mathfrak{r}\mathfrak{p}_i$ is the index of the unique right i-parenthesis), such that neither d nor $d+1$ belongs to $indspect(\mathfrak{h}')$.

(VI) $\widetilde{\sharp \mathfrak{r}\mathfrak{p}'_i}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ is equal to $\widetilde{\sharp \mathfrak{r}\mathfrak{p}'_i}$ in $\mathfrak{h}'$. Let $\mathfrak{r}\mathfrak{p}_i$ be the index of the right i-parenthesis in $\mathfrak{h}'$. Let us first show that $\mathfrak{r}\mathfrak{p}_i = \mathfrak{r}\mathfrak{p}'_i$. We know that $\mathfrak{l}\mathfrak{p}_0 = \mathfrak{l}\mathfrak{p}'_0$, $\mathfrak{l}\mathfrak{p}'_i$ is the index of the left i-parenthesis in $\mathfrak{h}'$ as well as the index of the large left i-parenthesis in $\mathfrak{h} \uplus \mathfrak{h}'$ (by (IV)). Since $\widetilde{\sharp \mathfrak{l}\mathfrak{p}'_i}$ in $\mathfrak{h}'$ is equal to $\widetilde{\sharp \mathfrak{l}\mathfrak{p}'_i}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ (by (IV)), $\widetilde{\sharp \mathfrak{r}\mathfrak{p}'_i}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ is strictly greater than $\widetilde{\sharp \mathfrak{l}\mathfrak{p}'_i}$ (in $\mathfrak{h}'$, or in $\mathfrak{h} \uplus \mathfrak{h}'$ since it is the same value). In particular, $\widetilde{\sharp \mathfrak{l}\mathfrak{p}_0}$ is strictly less than

$\widetilde{\sharp \mathfrak{rp}'_i}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ (since $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{indmin}(\mathfrak{u}) \wedge \mathtt{Llp}_i(\overline{\mathfrak{u}})$). Now let us consider the following facts:

(1) An entry in $\mathfrak{h}'$ cannot be transformed into a right or a left parenthesis in $\mathfrak{h} \uplus \mathfrak{h}'$ (simply because the index of the entry is allocated).
(2) The left i-parenthesis in $\mathfrak{h}'$ is transformed into a left i-parenthesis in $\mathfrak{h} \uplus \mathfrak{h}'$ (IV).

So, the only ways to build a large right i-parenthesis whose degree is greater that $\widetilde{\sharp \mathfrak{lp}'_i}$ is either by considering the right i-parenthesis from $\mathfrak{h}'$ with index $\mathfrak{rp}_i$ or by building another large right i-parenthesis with a different index either from the entries of $\mathfrak{h}'$ or from the right i-parenthesis in $\mathfrak{h}'$, which is not possible by (II). Consequently, $\mathfrak{rp}_i = \mathfrak{rp}'_i$. So far, we can observe that $\widetilde{\sharp \mathfrak{rp}'_i}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ is greater than $\widetilde{\sharp \mathfrak{rp}'_i}$ in $\mathfrak{h}'$. Let us show that we have indeed equality between these two degrees. Note that $indspect(\mathfrak{h}')$ can be defined as some set below where $\alpha = \widetilde{\sharp \mathfrak{lp}'_i}$ (in $\mathfrak{h}'$) and $\beta \geq 0$:

$$\{\alpha\} \cup \{\alpha + 3(j-1) + 1, \alpha + 3(j-1) + 2 : j \in [1, \beta]\} \cup \{\alpha + 3\beta + 1\}.$$

Indeed, $\mathfrak{h}'$ is a local i-valuation. Since $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{isoloc}(\overline{\mathfrak{u}})$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{Llp}_i(\overline{\mathfrak{u}})$, we also have that $\widetilde{\sharp \mathfrak{lp}'_i}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ is equal to α (this is actually (IV)).

So, $indspect(\mathfrak{h}')$ contains the following values:

$$\alpha, \alpha+1, \alpha+2, \alpha+4, \alpha+5, \ldots, \alpha+3(\beta-1)+1, \alpha+3(\beta-1)+2, \alpha+3\beta+1.$$

The values in $spect(\mathfrak{h} \uplus \mathfrak{h}') \cap [\alpha, +\infty[$ can only be obtained from the degree of index locations from $\mathfrak{h}'$ and therefore are of the form

$$\alpha + \alpha_0, \alpha + 1 + \alpha_1^1, \alpha + 2 + \alpha_2^1, \alpha + 4 + \alpha_1^2, \alpha + 5 + \alpha_2^2, \ldots$$

$$\ldots, \alpha + 3(\beta-1) + 1 + \alpha_1^\beta, \alpha + 3(\beta-1) + 2 + \alpha_2^\beta, \alpha + 3\beta + 1 + \alpha_{\beta+1}$$

with $\alpha_0, \alpha_{\beta+1}, \alpha_1^1, \alpha_2^1, \ldots, \alpha_1^\beta, \alpha_2^\beta \geq 0$. These latter values should be understood as the potential differences of degrees.

Since $\mathfrak{h} \models_{\mathfrak{f}} \mathtt{isoloc}(\overline{\mathfrak{u}})$ and $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{Llp}_i(\overline{\mathfrak{u}})$, we have $\alpha_0 = 0$ (this is actually (IV)). Since $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{wfh}_{X \cup \{i\}}$, there is $\beta' \geq 0$ such that $spect(\mathfrak{h} \uplus \mathfrak{h}') \cap [\alpha, +\infty[$ is equal to

$$\{\alpha\} \cup \{\alpha + 3(j-1) + 1, \alpha + 3(j-1) + 2 : j \in [1, \beta']\} \cup \{\alpha + 3\beta' + 1\}. \tag{3}$$

So necessarily $\beta' \geq \beta$.

Moreover, if $\beta = 0$, then $\alpha_{\beta+1} = 0$, otherwise $spect(\mathfrak{h} \uplus \mathfrak{h}') \cap [\alpha, +\infty[$ cannot be of the form of Equation (3). Assuming that $\beta \geq 1$, we can show by induction on j that $\alpha_1^j = 0$ and then $\alpha_2^j = 0$ with $j \in [1, \beta]$. For instance, $\alpha_1^1 \neq 0$ leads to a contradiction, since $\alpha + 1$ would not belong to $spect(\mathfrak{h} \uplus \mathfrak{h}') \cap [\alpha, +\infty[$ and therefore could not be of the form of Equation (3) by (2holes). Then, $\alpha_2^1 \neq 0$ leads to a contradiction, since $\alpha + 2$ would not belong to $spect(\mathfrak{h} \uplus \mathfrak{h}') \cap [\alpha, +\infty[$ (while $\alpha + 1$ does) and therefore could not be of the form of Equation (3). The same reasoning can be performed for all $j \in [2, \beta]$.

(VII) For every $\mathfrak{l}$ such that $\mathfrak{h} \models_{[\mathfrak{u} \mapsto \mathfrak{l}]} \mathtt{Lindex}(\mathfrak{u})$, we have $\mathfrak{h} \uplus \mathfrak{h}' \models_{[\mathfrak{u} \mapsto \mathfrak{l}]} \mathtt{Lindex}(\mathfrak{u})$, and $\widetilde{\sharp \mathfrak{l}}$ in $\mathfrak{h}$ is equal to $\widetilde{\sharp \mathfrak{l}}$ in $\mathfrak{h} \uplus \mathfrak{h}'$. Let $\mathfrak{l}$ be a large index in $\mathfrak{h}$. By (III), either $\mathfrak{l}$ is a large index in $\mathfrak{h} \uplus \mathfrak{h}'$ or $\mathfrak{l}$ is not anymore an index in $\mathfrak{h} \uplus \mathfrak{h}'$. In the latter case, $\mathfrak{l}$ cannot be the index of some j-parenthesis with $j \in X$ by (V). This latter case with $\mathfrak{l}$ being a large entry index is nevertheless ruled out because $\mathfrak{h} \uplus \mathfrak{h}' \models \mathtt{wfh}_{X \cup \{i\}}$ and there will be then a missing value in the spectrum $spect(\mathfrak{h} \uplus \mathfrak{h}')$ by (2holes).

As a consequence, an allocated location in $\mathfrak{h}$ that is not a large index, cannot be transformed into an index in $\mathfrak{h} \uplus \mathfrak{h}'$, otherwise we would have two large indices with

the same degree which is not possible since $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{wfh}_{X \cup \{i\}}$.

(VIII) For every $\mathfrak{l}$ such that $\mathfrak{h}' \models_{[\mathfrak{u} \mapsto \mathfrak{l}]} \mathtt{eindex}(\mathfrak{u})$, we have $\mathfrak{h} \uplus \mathfrak{h}' \models_{[\mathfrak{u} \mapsto \mathfrak{l}]} \mathtt{Lindex}(\mathfrak{u})$, and $\widetilde{\sharp \mathfrak{l}}$ in $\mathfrak{h}'$ is equal to $\widetilde{\sharp \mathfrak{l}}$ in $\mathfrak{h} \uplus \mathfrak{h}'$. Let $\mathfrak{l}$ be an index in $\mathfrak{h}'$. By (VI), if $\mathfrak{l}$ is the index of the left [resp. right] i-parenthesis, then $\mathfrak{l}$ is also the index of the left [resp. right] i-parenthesis in $\mathfrak{h} \uplus \mathfrak{h}'$ with the same amount of predecessors. If $\mathfrak{h}' \models_{[\mathfrak{u} \mapsto \mathfrak{l}]} \mathtt{eindex}(\mathfrak{u})$ and $\mathfrak{h} \uplus \mathfrak{h}' \not\models_{[\mathfrak{u} \mapsto \mathfrak{l}]} \mathtt{Lindex}(\mathfrak{u})$, there will be then a missing value in the spectrum $spect(\mathfrak{h} \uplus \mathfrak{h}')$, which would lead to a contradiction by invoking (2holes).

We are then able to conclude the statement of the lemma by showing the propositions below:

(†). $spect(\mathfrak{h}) \subseteq spect(\mathfrak{h} \uplus \mathfrak{h}')$.
(† †). $indspect(\mathfrak{h}') \subseteq spect(\mathfrak{h} \uplus \mathfrak{h}')$.
(† † †). $spect(\mathfrak{h} \uplus \mathfrak{h}') \subseteq spect(\mathfrak{h}) \uplus indspect(\mathfrak{h}')$.

Proof of (†). Let $n \in spect(\mathfrak{h})$. So, there is $\mathfrak{l} \in \mathbb{N}$ such that $\widetilde{\sharp \mathfrak{l}} = n$ and $\mathfrak{l}$ is an index of some entry or parenthesis in $\mathfrak{h}$ such that $n \geq d_0$ with $\widetilde{\sharp \mathfrak{lp}_0} = d_0$. Hence, $\mathfrak{l}$ is a large index and by (VII), $\mathfrak{l}$ is also a large index in $\mathfrak{h} \uplus \mathfrak{h}'$ with $\widetilde{\sharp \mathfrak{l}}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ greater or equal to d_0. This is true only if $n \geq \widetilde{\sharp \mathfrak{lp}'_{i_0}}$. So $n \in spect(\mathfrak{h} \uplus \mathfrak{h}')$.

Proof of († †). Let $n \in indspect(\mathfrak{h}')$. In the case there is some index $\mathfrak{l}$ of some entry such that $\widetilde{\sharp \mathfrak{l}} = n$, by (VIII), $\mathfrak{l}$ is a large index in $\mathfrak{h} \uplus \mathfrak{h}'$ and therefore $n \in spect(\mathfrak{h} \uplus \mathfrak{h}')$. In the case $n = \widetilde{\sharp \mathfrak{lp}'_i}$, the value $\widetilde{\sharp \mathfrak{lp}'_i}$ in $\mathfrak{h} \uplus \mathfrak{h}'$ is also equal to n by (IV), and by the assumption (3) of the lemma, $\mathfrak{lp}'_i$ is a large left parenthesis in $\mathfrak{h} \uplus \mathfrak{h}'$, whence $n \in spect(\mathfrak{h} \uplus \mathfrak{h}')$.

The case $n = \widetilde{\sharp \mathfrak{rp}'_i}$ admits a similar treatment by using (VI).

Proof of († † †). Let $n \in spect(\mathfrak{h} \uplus \mathfrak{h}')$. In the case $n \in [\widetilde{\sharp \mathfrak{lp}'_{i_0}}, \widetilde{\sharp \mathfrak{rp}'_{i_s}}]$, by (V), we get $n \in [\widetilde{\sharp \mathfrak{lp}_{i_0}}, \widetilde{\sharp \mathfrak{rp}_{i_s}}]$. Suppose that $n \notin spect(\mathfrak{h})$ and

$$n \in [\widetilde{\sharp \mathfrak{lp}_{i_0}}, \widetilde{\sharp \mathfrak{rp}_{i_s}}] \setminus \{\widetilde{\sharp \mathfrak{lp}_{i_0}}, \widetilde{\sharp \mathfrak{rp}_{i_0}}, \ldots, \widetilde{\sharp \mathfrak{lp}_{i_s}}, \widetilde{\sharp \mathfrak{rp}_{i_s}}\}.$$

Considering the structure of the spectrum of $\mathfrak{h}$, $n-1$ and $n+1$ belong to $spect(\mathfrak{h})$. By (VII), $n-1$ and $n+1$ belong to $spect(\mathfrak{h} \uplus \mathfrak{h}')$, which leads to a contradiction since there cannot be such three consecutive values ($n-1$, n and $n+1$) in $spect(\mathfrak{h} \uplus \mathfrak{h}')$.

Since $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{f}} \mathtt{wfh}_{X \cup \{i\}}$, we can conclude that $\widetilde{\sharp \mathfrak{lp}_i}'$ (in $\mathfrak{h} \uplus \mathfrak{h}'$) is equal to $\widetilde{\sharp \mathfrak{lp}'_{i_s}} + 1$ (also equal to $\widetilde{\sharp \mathfrak{lp}'_{i_s}} + 1$ in $\mathfrak{h}$). In the remaining case $n \in [\widetilde{\sharp \mathfrak{lp}'_i}, \widetilde{\sharp \mathfrak{rp}'_i}]$, suppose that $n \notin indspect(\mathfrak{h}')$ and $n \in (\widetilde{\sharp \mathfrak{lp}'_i}, \widetilde{\sharp \mathfrak{rp}'_i})$. Considering the structure of the index spectrum of $\mathfrak{h}'$, $n-1$ and $n+1$ belong to $indspect(\mathfrak{h}')$. By (VIII), $n-1$ and $n+1$ belong to $spect(\mathfrak{h} \uplus \mathfrak{h}')$, which leads to a contradiction since there cannot be such three consecutive values ($n-1$, n and $n+1$) in $(\widetilde{\sharp \mathfrak{lp}'_i}, \widetilde{\sharp \mathfrak{rp}'_i})$. □

5.4. A reduction from DSOL into 1SL2(⇀)

Below, we define a translation from a sentence ϕ in DSOL into a sentence in 1SL2(⇀) that uses only logarithmic space. Without any loss of generality, we assume that

(1) two occurrences of quantified variables in ϕ have distinct variable indices (e.g., $\mathtt{P}_4$ and $\mathtt{u}_4$ cannot both occur in ϕ and "$\forall\, \mathtt{u}_4$" cannot occur more than once) and
(2) if $\exists\, \mathtt{u}_i\ \psi_1$ is a subformula of $\exists\, \mathtt{u}_j\ \psi_2$, then $i > j$ and this holds for any combination of first-order/second-order variables.

At the outset, we may rename variables so that these simple conditions are satisfied. We assume that the variable indices for (first-order or second-order) variables are among $[1, K]$. Obviously, K is less than the size of the formula ϕ to be translated.

The translation of the formula ϕ, written $\mathtt{T}(\phi)$, first applies a top-level translation $t_{top}(\cdot)$ which takes care of initializing the valuation heap (mainly to introduce the left 0-parenthesis and the right 0-parenthesis); then, a recursive map $t(\cdot)$ is applied. So, $\mathtt{T}(\phi) \stackrel{\text{def}}{=} t_{top}(\phi)$ where $t_{top}(\phi)$ is defined as follows:

$$t_{top}(\phi) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \mathtt{isoloc}(\mathtt{u}) \wedge (\mathtt{localval}_0(\mathtt{u}) \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!*} (\mathtt{wfh}_{\{0\}} \wedge$$

$$\mathtt{indmin}(\mathtt{u}) \wedge (\forall\, \overline{\mathtt{u}}\ ((\mathtt{u} \neq \overline{\mathtt{u}}) \wedge \neg \mathtt{Lrp}_0(\overline{\mathtt{u}})) \Rightarrow (\sharp\overline{\mathtt{u}} < \sharp\mathtt{u})) \wedge t(\{0\}, \phi))).$$

The first step of the translation consists in adding 0-parentheses so that the heap that evaluates $t(\{0\}, \phi)$ is $\{0\}$-well-formed. The translation map $t(\cdot)$ has two arguments: the formula to be transformed and the set of variable indices for variables that have been quantified so far. The map $t(\cdot)$ is inductively defined as follows ($X \subseteq [0, K]$, ψ subformula of ϕ):

— $t(X, \cdot)$ is homomorphic for Boolean connectives,

— $t(X, \mathtt{u}_i = \mathtt{u}_j) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{elt}_j(\mathtt{u})$,

— $t(X, \mathtt{u}_i \hookrightarrow \mathtt{u}_j) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ (\mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{elt}_j(\overline{\mathtt{u}}) \wedge \mathtt{u} \hookrightarrow \overline{\mathtt{u}})$,

— $t(X, \mathtt{P}_i(\mathtt{u}_j, \mathtt{u}_k)) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ (\mathtt{elt}_j(\mathtt{u}) \wedge \exists\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u} \wedge \mathtt{vind}_i(\overline{\mathtt{u}}) \wedge \exists\, \mathtt{u}\ (\sharp\mathtt{u} = \sharp\overline{\mathtt{u}} + 1 \wedge \mathtt{vind}_i(\mathtt{u}) \wedge \exists\, \overline{\mathtt{u}}\ (\mathtt{u} \hookrightarrow \overline{\mathtt{u}} \wedge \mathtt{elt}_k(\overline{\mathtt{u}})))))$.

— Translating the quantifiers themselves is a bit trickier, as we need to introduce the new entries to the valuation heap by applying the magic wand. For the quantifier $\exists\, \mathtt{u}_i$, we choose two locations $\mathfrak{l}$ and $\mathfrak{l}'$ such that $\mathfrak{l}$ is the index of the left 0-parenthesis and $\mathfrak{l}'$ is an isolated location in the original heap.
We construct a new heap that is a local i-valuation while enforcing that the index of the left 0-parenthesis is preserved and $\mathfrak{l}'$ becomes the index of the unique large left i-parenthesis (see Lemma 5.11). $t(X, \exists\, \mathtt{u}_i\ \psi) \stackrel{\text{def}}{=}$

$$\exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ ((\mathtt{indmin}(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}})) \wedge (\mathtt{localval}_i(\overline{\mathtt{u}}) \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!*}$$

$$(\mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{Llp}_i(\overline{\mathtt{u}}) \wedge t(X \cup \{i\}, \psi)))).$$

— The translation with second-order variables is analogous (the formula $\mathtt{localval}_i(\overline{\mathtt{u}})$ below is actually defined differently, see the proof of Lemma 5.7):

$$t(X, \exists\, \mathtt{P}_i\ \psi) \stackrel{\text{def}}{=} \exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ ((\mathtt{indmin}(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}})) \wedge$$

$$(\mathtt{localval}_i(\overline{\mathtt{u}}) \mathrel{-\!\!\!\!\!\!\rightarrow\!\!\!*} (\mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{Llp}_i(\overline{\mathtt{u}}) \wedge t(X \cup \{i\}, \psi)))).$$

Every subformula $t(X, \psi)$ has no free variable from $\mathtt{free}(\psi) \subseteq X$ where $\mathtt{free}(\psi)$ denotes the set of variable indices in ψ from either first-order or second-order *free* variables. As noted by one anonymous referee, a standard trick is to convert first-order variables into second-order ones so that the proof has only to deal with one type of variable. Herein, we do not quite eliminate first-order variables but we provide a uniform treatment for first-order quantifications and second-order quantifications, which essentially amounts to dealing with a single type of encoding.

Below, we state the correctness lemma that allows us to get Theorem 5.13 (the proof is by structural induction).

LEMMA 5.12 (CORRECTNESS). *Let ϕ be a DSOL sentence of the above form, ψ be one of its subformulae and $(\mathtt{free}(\psi) \cup \{0\}) \subseteq X \subseteq [0, K]$. Let $\mathfrak{h} = \mathfrak{h}_B \uplus \mathfrak{h}_V$ be a X-well-formed heap and $\mathfrak{V}_\mathfrak{h}$ be the valuation extracted from $\mathfrak{h}$. Then, $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \psi$ iff $\mathfrak{h} \models t(X, \psi)$.*

PROOF. The proof is by structural induction.
Base case 1: ψ is equal to $\mathtt{u}_i = \mathtt{u}_j$.
Since $\mathfrak{h}$ is X-well-formed and $i, j \in X$, $\mathfrak{V}_\mathfrak{h}(\mathtt{u}_i)$ is equal to $\mathfrak{h}_V(\mathfrak{l})$ where $\mathfrak{l}$ is the unique index location such that $\widetilde{\sharp\mathfrak{l}} \in \mathit{degrees}(i, \mathfrak{h})$. Similarly, $\mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$ is equal to $\mathfrak{h}_V(\mathfrak{l}')$ where $\mathfrak{l}'$ is the unique index location such that $\widetilde{\sharp\mathfrak{l}'} \in \mathit{degrees}(j, \mathfrak{h})$. Uniqueness is a consequence of Definition 5.8(2).

Let us recall that $\mathtt{elt}_i(\mathtt{u}) = \exists\ \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \wedge \mathtt{vind}_i(\overline{\mathtt{u}})$ with $\mathtt{vind}_i(\mathtt{u}) = \mathtt{Lindex}(\mathtt{u}) \wedge \mathtt{eindex}(\mathtt{u}) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{Llp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} < \sharp\mathtt{u}) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{Lrp}_i(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} > \sharp\mathtt{u})$. Similarly, $\mathtt{elt}_j(\mathtt{u}) = \exists\ \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \wedge \mathtt{vind}_j(\overline{\mathtt{u}})$.

First, let us suppose that $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \mathtt{u}_i = \mathtt{u}_j$. This means that $\mathfrak{V}_\mathfrak{h}(\mathtt{u}_i)$ and $\mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$ are equal, say to $\mathfrak{l}'$ and therefore there is a unique index location $\mathfrak{l}_i$ such that $\mathfrak{h}_V(\mathfrak{l}_i) = \mathfrak{l}'$ and $\widetilde{\sharp\mathfrak{l}_i} \in \mathit{degrees}(i, \mathfrak{h})$. Moreover, there is a unique index location $\mathfrak{l}_j$ such that $\mathfrak{h}_V(\mathfrak{l}_j) = \mathfrak{l}'$ and $\widetilde{\sharp\mathfrak{l}_j} \in \mathit{degrees}(j, \mathfrak{h})$. Since $\widetilde{\sharp\mathfrak{l}_i}$ and $\widetilde{\sharp\mathfrak{l}_j}$ belong to the index spectrum of $\mathfrak{h}_V$, the locations $\mathfrak{l}_i$ and $\mathfrak{l}_j$, have the same number of predecessors in $\mathfrak{h}$ and in $\mathfrak{h}_V$ and they are also indices in $\mathfrak{h}$ (see Lemma 5.11). Consequently, $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}']} \mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{elt}_j(\mathtt{u})$ (see also Lemma 5.6), whence $\mathfrak{h} \models \exists\ \mathtt{u}\ \mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{elt}_j(\mathtt{u})$.

Now suppose that $\mathfrak{h} \models \exists\ \mathtt{u}\ \mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{elt}_j(\mathtt{u})$. There is a location $\mathfrak{l}'$ such that $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}']} \mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{elt}_j(\mathtt{u})$. Therefore there are index locations $\mathfrak{l}_i$ and $\mathfrak{l}_j$ such that $\widetilde{\sharp\mathfrak{l}_i} \in \mathit{degrees}(i, \mathfrak{h})$, $\widetilde{\sharp\mathfrak{l}_j} \in \mathit{degrees}(j, \mathfrak{h})$, $\mathfrak{h}(\mathfrak{l}_i) = \mathfrak{l}'$ and $\mathfrak{h}(\mathfrak{l}_j) = \mathfrak{l}'$. By Lemma 5.11, $\mathfrak{h}_V(\mathfrak{l}_i) = \mathfrak{l}'$, $\widetilde{\sharp\mathfrak{l}_i} \in \mathit{indspect}(\mathfrak{h}_V)$, $\mathfrak{h}_V(\mathfrak{l}_j) = \mathfrak{l}'$ and $\widetilde{\sharp\mathfrak{l}_j} \in \mathit{indspect}(\mathfrak{h}_V)$. By definition of $\mathfrak{V}_\mathfrak{h}$ (see Definition 5.10), this implies that $\mathfrak{V}_\mathfrak{h}(\mathtt{u}_i) = \mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$ and therefore $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \mathtt{u}_i = \mathtt{u}_j$.

Base case 2: ψ is equal to $\mathtt{u}_i \hookrightarrow \mathtt{u}_j$.
Again, since $\mathfrak{h}$ is X-well-formed and $i, j \in X$, $\mathfrak{V}_\mathfrak{h}(\mathtt{u}_i)$ is equal to $\mathfrak{h}_V(\mathfrak{l})$ where $\mathfrak{l}$ is the unique location such that $\widetilde{\sharp\mathfrak{l}} \in \mathit{degrees}(i, \mathfrak{h})$. Similarly, $\mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$ is equal to $\mathfrak{h}_V(\mathfrak{l})$ where $\mathfrak{l}$ is the unique location such that $\widetilde{\sharp\mathfrak{l}} \in \mathit{degrees}(j, \mathfrak{h})$.

First, let us suppose that $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \mathtt{u}_i \hookrightarrow \mathtt{u}_j$. This means that $\mathfrak{h}_B(\mathfrak{V}_\mathfrak{h}(\mathtt{u}_i)) = \mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$, and there is a unique location $\mathfrak{l}_i$ such that $\mathfrak{h}_V(\mathfrak{l}_i) = \mathfrak{V}_\mathfrak{h}(\mathtt{u}_i)$ and $\widetilde{\sharp\mathfrak{l}_i} \in \mathit{degrees}(i, \mathfrak{h})$. There is also a unique location $\mathfrak{l}_j$ such that $\mathfrak{h}_V(\mathfrak{l}_j) = \mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$ and $\widetilde{\sharp\mathfrak{l}_j} \in \mathit{degrees}(j, \mathfrak{h})$. Since $\widetilde{\sharp\mathfrak{l}_i}$ and $\widetilde{\sharp\mathfrak{l}_j}$ belong to the index spectrum of $\mathfrak{h}_V$, the locations $\mathfrak{l}_i$ and $\mathfrak{l}_j$, have the same number of predecessors in $\mathfrak{h}$ and in $\mathfrak{h}_V$ and they are also indices in $\mathfrak{h}$ (see Lemma 5.11). There are index locations $\mathfrak{l}', \mathfrak{l}''$ such that $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}', \overline{\mathtt{u}} \mapsto \mathfrak{l}'']} \mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{elt}_j(\overline{\mathtt{u}}) \wedge \mathtt{u} \hookrightarrow \overline{\mathtt{u}}$. Note that $\mathfrak{h}_B(\mathfrak{V}_\mathfrak{h}(\mathtt{u}_i)) = \mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$ implies $\mathfrak{h}(\mathfrak{V}_\mathfrak{h}(\mathtt{u}_i)) = \mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$ since $\mathfrak{h}_B$ is a subheap of $\mathfrak{h}$. Consequently, $\mathfrak{h} \models \exists\ \mathtt{u}\ \exists\ \overline{\mathtt{u}}\ \mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{elt}_j(\overline{\mathtt{u}}) \wedge \mathtt{u} \hookrightarrow \overline{\mathtt{u}}$.

Now suppose that $\mathfrak{h} \models \exists\ \mathtt{u}\ \exists\ \overline{\mathtt{u}}\ \mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{elt}_j(\overline{\mathtt{u}}) \wedge \mathtt{u} \hookrightarrow \overline{\mathtt{u}}$. Consequently, there are locations $\mathfrak{l}'$ and $\mathfrak{l}''$ such that $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}', \overline{\mathtt{u}} \mapsto \mathfrak{l}'']} \mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{elt}_j(\overline{\mathtt{u}}) \wedge \mathtt{u} \hookrightarrow \overline{\mathtt{u}}$. Therefore there are locations $\mathfrak{l}_i$ and $\mathfrak{l}_j$ such that $\widetilde{\sharp\mathfrak{l}_i} \in \mathit{degrees}(i, \mathfrak{h})$, $\widetilde{\sharp\mathfrak{l}_j} \in \mathit{degrees}(j, \mathfrak{h})$, $\mathfrak{h}(\mathfrak{l}_i) = \mathfrak{l}'$, $\mathfrak{h}(\mathfrak{l}_j) = \mathfrak{l}''$ and of course $\mathfrak{h}(\mathfrak{l}') = \mathfrak{l}''$. By Lemma 5.11 and since $\mathit{indspect}(\mathfrak{h}_V) = \mathit{spect}(\mathfrak{h})$, $\mathfrak{h}_V(\mathfrak{l}_i) = \mathfrak{l}'$, $\widetilde{\sharp\mathfrak{l}_i} \in \mathit{indspect}(\mathfrak{h}_V)$, $\mathfrak{h}_V(\mathfrak{l}_j) = \mathfrak{l}''$ and $\widetilde{\sharp\mathfrak{l}_j} \in \mathit{indspect}(\mathfrak{h}_V)$. By construction of $\mathfrak{V}_\mathfrak{h}$ (see Definition 5.10), this implies that $\mathfrak{h}(\mathfrak{V}_\mathfrak{h}(\mathtt{u}_i)) = \mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$ and therefore $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \mathtt{u}_i \hookrightarrow \mathtt{u}_j$. Note that $\mathfrak{V}_\mathfrak{h}(\mathtt{u}_i) \notin \mathrm{dom}(\mathfrak{h}_V)$ since $\mathfrak{h}$ is X-well-formed.

Base case 3: ψ is equal to $\mathtt{P}_i(\mathtt{u}_j, \mathtt{u}_k)$.
Suppose that $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \mathtt{P}_i(\mathtt{u}_j, \mathtt{u}_k)$. By definition of the satisfaction relation $\models$, this is equivalent to $(\mathfrak{V}_\mathfrak{h}(\mathtt{u}_j), \mathfrak{V}_\mathfrak{h}(\mathtt{u}_k)) \in \mathfrak{V}_\mathfrak{h}(\mathtt{P}_i)$. By definition of $\mathfrak{V}_\mathfrak{h}$, $\mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$ is equal to $\mathfrak{h}_V(\mathfrak{l}_j)$ where $\mathfrak{l}_j$ is the unique index location such that $\widetilde{\sharp\mathfrak{l}_j} \in \mathit{degrees}(j, \mathfrak{h})$. Similarly, $\mathfrak{V}_\mathfrak{h}(\mathtt{u}_k)$ is equal to $\mathfrak{h}_V(\mathfrak{l}_k)$ where $\mathfrak{l}_k$ is the unique index location such that $\widetilde{\sharp\mathfrak{l}_k} \in \mathit{degrees}(k, \mathfrak{h})$. So,

$\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}_j]} \mathtt{vind}_j(\mathtt{u})$ and $\mathfrak{h} \models_{[\overline{\mathtt{u}} \mapsto \mathfrak{l}_k]} \mathtt{vind}_k(\overline{\mathtt{u}})$. Moreover, by definition of $\mathfrak{V}_\mathfrak{h}$, there are index locations $\mathfrak{l}_i$ and $\mathfrak{l}'_i$ such that

(1) $\widetilde{\sharp \mathfrak{l}'_i} = \widetilde{\sharp \mathfrak{l}_i} + 1$,
(2) $\widetilde{\sharp \mathfrak{l}_i}, \widetilde{\sharp \mathfrak{l}'_i} \in \mathit{degrees}(i, \mathfrak{h})$,
(3) $\mathfrak{h}(\mathfrak{l}_i) = \mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$ and $\mathfrak{h}(\mathfrak{l}'_i) = \mathfrak{V}_\mathfrak{h}(\mathtt{u}_k)$,
(4) $\mathfrak{h} \models_{[\overline{\mathtt{u}} \mapsto \mathfrak{l}_i]} \mathtt{vind}_i(\overline{\mathtt{u}})$ and $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}'_i]} \mathtt{vind}_i(\mathtt{u})$.

Finally, the formulae $\mathtt{elt}_j(\mathtt{u})$ and $\mathtt{elt}_k(\overline{\mathtt{u}})$ are defined so that $\mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)]} \mathtt{elt}_j(\mathtt{u})$ and $\mathfrak{h} \models_{[\overline{\mathtt{u}} \mapsto \mathfrak{V}_\mathfrak{h}(\mathtt{u}_k)]} \mathtt{elt}_k(\overline{\mathtt{u}})$. So, we have

— $\mathfrak{l}_i \to \mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$ (i.e. $\mathfrak{h}(\mathfrak{l}_i) = \mathfrak{V}_\mathfrak{h}(\mathtt{u}_j)$),
— $\widetilde{\sharp \mathfrak{l}'_i} = \widetilde{\sharp \mathfrak{l}_i} + 1$,
— $\mathfrak{l}'_i \to \mathfrak{V}_\mathfrak{h}(\mathtt{u}_k)$ (i.e. $\mathfrak{h}(\mathfrak{l}'_i) = \mathfrak{V}_\mathfrak{h}(\mathtt{u}_k)$).

This guarantees the satisfaction of

$$\mathfrak{h} \models \exists\, \mathtt{u}\, \Big(\mathtt{elt}_j(\mathtt{u}) \wedge \exists\, \overline{\mathtt{u}}\, \big(\overline{\mathtt{u}} \hookrightarrow \mathtt{u}\ \wedge\ \mathtt{vind}_i(\overline{\mathtt{u}}) \wedge$$

$$\exists\, \mathtt{u}\ \big(\sharp \mathtt{u} = \sharp \overline{\mathtt{u}} + 1 \wedge \mathtt{vind}_i(\mathtt{u}) \wedge \exists\, \overline{\mathtt{u}}\ \big(\mathtt{u} \hookrightarrow \overline{\mathtt{u}} \wedge \mathtt{elt}_k(\overline{\mathtt{u}})\big)\big)\big)\Big).$$

The proof in the other direction is by an easy verification and similar since all of the above implications are indeed equivalences.

Induction step. The induction hypothesis is the following: for every subformula ψ' of size strictly less than the size of ψ, for every $\mathtt{free}(\psi') \subseteq X' \subseteq [0, K]$, we have $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \psi'$ iff $\mathfrak{h} \models t(X', \psi')$. The cases when the outermost connective is Boolean are by an easy verification.

Case 1: ψ is equal to $\exists\, \mathtt{u}_i\ \psi'$. Suppose that $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \exists\, \mathtt{u}_i\ \psi'$. By definition of the satisfaction relation $\models$, there is $\mathfrak{l} \in \mathbb{N}$ such that $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}[\mathtt{u}_i \mapsto \mathfrak{l}]} \psi'$. In case $\mathfrak{l}$ belongs to the set Y defined below,

$$Y = \mathrm{dom}(\mathfrak{h}_V) \cup \{\mathfrak{l} \in \mathbb{N} : \mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}]} \bigvee_{j \in X} (\mathtt{Llp}_j(\mathtt{u}) \vee \mathtt{Lrp}_j(\mathtt{u}))\}$$

(and therefore $\mathfrak{l}$ is an isolated location in $\mathfrak{h}_B$), we pick another location $\mathfrak{l}'$ that does not belong to Y and that is also isolated in $\mathfrak{h}_B$. It is then easy to show that $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}[\mathtt{u}_i \mapsto \mathfrak{l}]} \psi'$ iff $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}[\mathtt{u}_i \mapsto \mathfrak{l}']} \psi'$. So, without any loss of generality, below we assume that $\mathfrak{l}$ does not belong to Y.

Let us build $\mathfrak{h}^i_V$ and an assignment $\mathfrak{f}$ such that:

(1) $\mathfrak{h}^i_V \models_\mathfrak{f} \mathtt{localval}_i(\overline{\mathtt{u}})$,
(2) $\mathfrak{h} \models_\mathfrak{f} \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}})$,
(3) $\mathfrak{h} \uplus \mathfrak{h}^i_V \models_\mathfrak{f} \mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{Llp}_i(\overline{\mathtt{u}})$.

Assume that $max(X) = j$ and m is the degree of the right j-parenthesis with greatest degree. It is easy to define a local i-valuation $\mathfrak{h}^i_V$ disjoint from $\mathfrak{h}$ such that the degree of the left i-parenthesis is $m+1$, the degree of the right i-parenthesis is $m+3$, the degree of the unique entry is $m+2$, its element is precisely $\mathfrak{l}$ and all the locations in its domain are isolated in $\mathfrak{h}$ (always possible since $\mathrm{dom}(\mathfrak{h}) \cup \mathrm{ran}(\mathfrak{h})$ is finite).

It is not difficult to check that $\mathfrak{h}^i_V$ and $\mathfrak{f}$ satisfy the above conditions. Since $\mathfrak{h} \uplus \mathfrak{h}^i_V$ is $(X \cup \{i\})$-well-formed by construction, by Lemma 5.11, we have $\mathfrak{V}_\mathfrak{h}[\mathtt{u}_i \mapsto \mathfrak{l}]$ equal to $\mathfrak{V}_{\mathfrak{h} \uplus \mathfrak{h}^i_V}$. Hence, $\mathfrak{h}_B \models_{\mathfrak{V}_{\mathfrak{h} \uplus \mathfrak{h}^i_V}} \psi'$ and by the induction hypothesis, we get $\mathfrak{h} \uplus \mathfrak{h}^i_V \models$

$t(X \cup \{i\}, \psi')$. However, it is easy to conclude then that $\mathfrak{h} \models t(X, \psi)$. Indeed, $\mathfrak{h}$ satisfies the formula below

$$\exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ (\mathtt{indmin}(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}}) \wedge (\mathtt{localval}_i(\overline{\mathtt{u}}) \mathrel{-\!\!\!\!\!\rightarrow}$$

$$(\mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{Llp}_i(\overline{\mathtt{u}}) \wedge t(X \cup \{i\}, \psi'))))$$

whenever there are locations $\mathfrak{l}^\star$ and $\mathfrak{l}^{\star\star}$ and a disjoint heap $\mathfrak{h}^\star$ such that:

(1) $\mathfrak{l}^\star$ is the unique large 0-parenthesis in $\mathfrak{h}$ and $\mathfrak{l}^{\star\star}$ is isolated in $\mathfrak{h}$,
(2) $\mathfrak{h}^\star$ is an i-local valuation such that the index of the left i-parenthesis is $\mathfrak{l}^{\star\star}$, $\mathfrak{h} \uplus \mathfrak{h}^\star$ is $(X \cup \{i\})$-well-formed,
(3) $\mathfrak{l}^\star$ is the left 0-parenthesis in $\mathfrak{h} \uplus \mathfrak{h}^\star$ and $\mathfrak{l}^{\star\star}$ is the left i-parenthesis in $\mathfrak{h} \uplus \mathfrak{h}^\star$,
(4) $\mathfrak{h} \uplus \mathfrak{h}^\star \models_{[\mathtt{u} \mapsto \mathfrak{l}^\star, \overline{\mathtt{u}} \mapsto \mathfrak{l}^{\star\star}]} t(X \cup \{i\}, \psi')$.

It is clear that such objects exist by considering the above construction.

The proof in the other direction (i.e. $\mathfrak{h} \models t(X, \psi)$ implies $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \exists\, \mathtt{u}_i\ \psi'$) is actually very similar since most of the above implications are indeed equivalences.

Case 2: ψ is equal to $\exists\, \mathtt{P}_i\ \psi'$. Suppose that $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \exists\, \mathtt{P}_i\ \psi'$. By definition of the satisfaction relation $\models$, there is a finite binary relation $R \subseteq \mathbb{N}^2$ such that $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}[\mathtt{P}_i \mapsto R]} \psi'$. In case R involves locations in Y defined below,

$$Y = \mathrm{dom}(\mathfrak{h}_V) \cup \{\mathfrak{l} \in \mathbb{N} : \mathfrak{h} \models_{[\mathtt{u} \mapsto \mathfrak{l}]} \bigvee_{j \in X} (\mathtt{Llp}_j(\mathtt{u}) \vee \mathtt{Lrp}_j(\mathtt{u}))\}$$

(and therefore R involves some isolated locations in $\mathfrak{h}_B$), we pick another R' (of same cardinality β) that does not involve locations in Y. It is then easy to show that $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}[\mathtt{P}_i \mapsto R]} \psi'$ iff $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}[\mathtt{P}_i \mapsto R']} \psi'$. So, without any loss of generality, below we assume that R does not involve locations in Y (see also [Brochenin et al. 2012, Lemma 2.1]).

Let us build $\mathfrak{h}^i_V$ and an assignment $\mathfrak{f}$ such that:

(1) $\mathfrak{h}^i_V \models_\mathfrak{f} \mathtt{localval}_i(\overline{\mathtt{u}})$,
(2) $\mathfrak{h} \models_\mathfrak{f} \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}})$,
(3) $\mathfrak{h} \uplus \mathfrak{h}^i_V \models_\mathfrak{f} \mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{Llp}_i(\overline{\mathtt{u}})$.

Assume that $max(X) = j$ and m is the degree of the right j-parenthesis with greatest degree. It is easy to define a local i-valuation $\mathfrak{h}^i_V$ disjoint from $\mathfrak{h}$ such that

(1) the degree of the left i-parenthesis is $m + 1$,
(2) the degree of the right i-parenthesis is $(m + 1) + 3\beta + 1$ for some $\beta \geq 0$,
(3) there are 2β entries,
(4) for every pair $(\mathfrak{l}, \mathfrak{l}')$ in R, there are two entries of consecutive degrees whose elements are $\mathfrak{l}$ and $\mathfrak{l}'$ respectively.

This is always possible since $\mathrm{dom}(\mathfrak{h}) \cup \mathrm{ran}(\mathfrak{h})$ and R are finite.

It is not difficult to check that $\mathfrak{h}^i_V$ and $\mathfrak{f}$ satisfy the above conditions. Since $\mathfrak{h} \uplus \mathfrak{h}^i_V$ is $(X \cup \{i\})$-well-formed by construction, by Lemma 5.11, we have $\mathfrak{V}_\mathfrak{h}[\mathtt{P}_i \mapsto R]$ equal to $\mathfrak{V}_{\mathfrak{h} \uplus \mathfrak{h}^i_V}$. Hence, $\mathfrak{h}_B \models_{\mathfrak{V}_{\mathfrak{h} \uplus \mathfrak{h}^i_V}} \psi'$ and by the induction hypothesis, we get $\mathfrak{h} \uplus \mathfrak{h}^i_V \models t(X \cup \{i\}, \psi')$. However, it is easy to conclude then that $\mathfrak{h} \models t(X, \psi)$. Indeed, $\mathfrak{h}$ satisfies the formula below

$$\exists\, \mathtt{u}\ \exists\, \overline{\mathtt{u}}\ (\mathtt{indmin}(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}}) \wedge (\mathtt{localval}_i(\overline{\mathtt{u}}) \mathrel{-\!\!\!\!\!\rightarrow}$$

$$(\mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{Llp}_i(\overline{\mathtt{u}}) \wedge t(X \cup \{i\}, \psi'))))$$

whenever there are locations $\mathfrak{l}^\star$ and $\mathfrak{l}^{\star\star}$ and a disjoint heap $\mathfrak{h}^\star$ such that:

(1) $\mathfrak{l}^\star$ is the unique large 0-parenthesis in $\mathfrak{h}$ and $\mathfrak{l}^{\star\star}$ is isolated in $\mathfrak{h}$,
(2) $\mathfrak{h}^\star$ is an i-local valuation such that the index of the left i-parenthesis is $\mathfrak{l}^{\star\star}$, $\mathfrak{h} \uplus \mathfrak{h}^\star$ is $(X \cup \{i\})$-well-formed,
(3) $\mathfrak{l}^\star$ is the left 0-parenthesis in $\mathfrak{h} \uplus \mathfrak{h}^\star$ and $\mathfrak{l}^{\star\star}$ is the left i-parenthesis in $\mathfrak{h} \uplus \mathfrak{h}^\star$,
(4) $\mathfrak{h} \uplus \mathfrak{h}^\star \models_{[\mathtt{u} \mapsto \mathfrak{l}^\star, \overline{\mathtt{u}} \mapsto \mathfrak{l}^{\star\star}]} t(X \cup \{i\}, \psi')$.

It is clear that such objects exist by considering the above construction.

The proof in the other direction (i.e. $\mathfrak{h} \models t(X, \psi)$ implies $\mathfrak{h}_B \models_{\mathfrak{V}_\mathfrak{h}} \exists\, \mathtt{P}_i\ \psi'$) is actually very similar since most of the above implications are indeed equivalences. □

Here is the main result of the paper.

THEOREM 5.13. *For every sentence ϕ in DSOL, for every heap $\mathfrak{h}$, we have $\mathfrak{h} \models \phi$ iff $\mathfrak{h} \models \mathtt{T}(\phi)$, so WSOL and 1SL2(−∗) have the same expressive power.*

It is worth recalling that we already know that DSOL and WSOL have the same expressive power with 1SL [Brochenin et al. 2012].

PROOF. First, we can establish the following lemma:

LEMMA 5.14. *Let $\mathfrak{h}$ be some heap, ψ be some sentence in 1SL2(−∗). Then, the propositions below are equivalent:*

(I). $\mathfrak{h}$ satisfies the formula below:

$$\exists\, \mathtt{u}\ \mathtt{isoloc}(\mathtt{u}) \wedge (\mathtt{localval}_0(\mathtt{u}) \mathrel{\vec{-\!\!*}}$$

$$(\mathtt{wfh}_{\{0\}} \wedge \mathtt{indmin}(\mathtt{u}) \wedge (\forall\, \overline{\mathtt{u}}\ ((\mathtt{u} \neq \overline{\mathtt{u}}) \wedge \neg \mathtt{Lrp}_0(\overline{\mathtt{u}})) \Rightarrow (\sharp\overline{\mathtt{u}} < \sharp\mathtt{u}) \wedge \psi)).$$

(II). There is a heap $\mathfrak{h}_0$ disjoint from $\mathfrak{h}$, that is a local 0-valuation such that $\mathfrak{h} \uplus \mathfrak{h}_0$ is $\{0\}$-well-formed and its large left 0-parenthesis is precisely the left 0-parenthesis from $\mathfrak{h}_0$. Moreover, $\mathfrak{h} \uplus \mathfrak{h}_0$ satisfies ψ.

The proof is by an easy verification but it is helpful to show the correctness of the full translation.

Let ϕ be a sentence in DSOL. If ϕ does not satisfy the syntactic conditions defined at the beginning of Section 5.4, one can easily define an equivalent formula in DSOL satisfying those simple conditions. By Lemma 5.14, $\mathfrak{h} \models \mathtt{T}(\phi)$ iff there is a heap $\mathfrak{h}_0$ disjoint from $\mathfrak{h}$, that is a local 0-valuation such that $\mathfrak{h} \uplus \mathfrak{h}_0$ is $\{0\}$-well-formed and its large 0-parenthesis is precisely the left 0-parenthesis from $\mathfrak{h}_0$ and $\mathfrak{h} \uplus \mathfrak{h}_0$ satisfies $t(\{0\}, \phi)$. It is always possible to build a local 0-valuation $\mathfrak{h}_0$ disjoint from $\mathfrak{h}$ such that its large left 0-parenthesis is precisely the left 0-parenthesis from $\mathfrak{h}_0$. By Lemma 5.12, we have $\mathfrak{h} \models_{\mathfrak{V}_{\mathfrak{h}_0}} \phi$ iff $\mathfrak{h} \uplus \mathfrak{h}_0 \models t(\{0\}, \phi)$. By assumption, if $\mathfrak{h} \models \phi$, then $\mathfrak{h} \models \mathtt{T}(\phi)$ by using Lemma 5.14 and Lemma 5.12. Similarly, if $\mathfrak{h} \models \mathtt{T}(\phi)$, by using the equivalences of Lemma 5.14 and Lemma 5.12 in the other direction, we get that $\mathfrak{h} \models \phi$. □

Observe that $\mathtt{T}(\phi)$ is of cubic size in the size of ϕ thanks to the size properties of all the subformulae involved in the translation. Moreover, $\mathtt{T}(\phi)$ can be computed in logarithmic space in the size of ϕ but here we have to be a bit careful. Indeed, one of the parameters of the translation is the set X of indices and updating it through the translation requires linear space if no further observation is made. However, whenever a formula $t(X, \psi)$ needs to be constructed, X is not arbitrary. The set X is actually the set of variables indices whose variables are quantified above ψ and therefore, it would be possible to omit the first parameter and to reconstruct on demand the set X, which can be done in logarithmic space. Moreover, zero always belongs to X (even though this is not a variable index).

So, the restriction to two variables in 1SL2($-\!\!*$) does not reduce the expressive power, unlike restrictions in [Venema 1991; Etessami et al. 1997] but we know also other logics restricted to two variables that are expressively complete, see e.g. [Lutz et al. 2001; Marx and de Rijke 2005].

We get the ultimate undecidability result below (no separating conjunction, two quantified variables, one record field).

COROLLARY 5.15. *Satisfiability problem for 1SL2($-\!\!*$) is undecidable.*

The absence of program variables in 1SL2($-\!\!*$) makes the proof of Corollary 5.15 even more difficult to design, which is perfect to obtain the sharpest undecidability result. An expressiveness result with program variables is briefly presented in Section 6.1.

THEOREM 5.16. *The set of valid formulae in 1SL2($-\!\!*$) is not recursively enumerable.*

Indeed, finitary validity for classical predicate logic restricted to a single binary predicate is not recursively enumerable [Trakhtenbrot 1963], which implies a similar property for DSOL and therefore for 1SL2($-\!\!*$) by Theorem 5.13.

It is also possible to establish the following consequences.

COROLLARY 5.17.

(I). Let ϕ be a sentence in 1SL. There is an equivalent sentence in 1SL2($-\!\!*$) of polynomial size in the size of ϕ.
(II). 1SL2($-\!\!*$) is strictly more expressive than 1SL($*$).

Corollary 5.17(II) follows from Theorem 5.13 and 1SL($*$) is strictly less expressive than MSO [Antonopoulos and Dawar 2009, Corollary 5.3] (see also [Antonopoulos 2010]). Corollary 5.17(I) is a consequence of Theorem 5.13 and of the fact that 1SL can be translated into DSOL in polynomial time. The composition of two maps that increase respectively the formula size only polynomially, provides a formula of polynomial size too.

Our main results are Theorem 5.13 and Corollary 5.15, significantly improving previously known results (see the figure in Section 1). As far as the translation into 1SL2($-\!\!*$) is concerned (see the current section but it uses in essential ways the formulae of Section 4), the lack of variables is partially compensated by the introduction of left and right parentheses in order to constrain sufficiently the valuation heap. More importantly, we have shown that this is a viable solution in 1SL2($-\!\!*$) despite only having two variables (see the proof of Lemma 5.11). In particular, this means that we were able to apply the method of dividing the heap into two disjoint heaps: the original heap and the heap encoding the valuation without provoking any confusion and this is probably the hardest part of our technical developments (see the proof of Lemma 5.11). This was not at all clear at the outset, and of course, in view of the complexity of the final proof, this led to lengthy arguments to show correctness of the whole enterprise.

Probably, alternative definitions for local i-valuations and X-well-formed heaps are possible, while leading to slightly simpler structures. One may reach the same main results with a unique type of i-parenthesis (instead of having a left version and a right version); maybe only one type of parenthesis is also possible, for instance by repeating the parenthesis i times at the appropriate place. Our current encoding has more constraints and it is more explicit in view of the design of proofs. We believe our current version is a fair compromise between understandable valuation heaps and the complexity of the proofs.

6. EXTENSIONS

In this section, we present several variants of 1SL2($-\!\!*$) for which undecidability or expressive power can be established on the lines of the previous developments.

6.1. Adding an unbounded number of program variables

In this section, we consider 1SL with program variables, which is a strict extension of 1SL and therefore undecidability for 1SL2($-\!\!*$) is still valid in the presence of program variables. Adding program variables is the usual way to consider separation logic and below we show that the previous developments can be easily lifted to the case with program variables.

Let $\mathtt{PVAR} = \{\mathtt{x}_1, \mathtt{x}_2, \ldots\}$ be a countably infinite set of *program variables*. A *memory state* is a pair $(\mathfrak{s}, \mathfrak{h})$ such that $\mathfrak{s} : \mathtt{PVAR} \to \mathbb{N}$ and $\mathfrak{h}$ is a heap. Formulae of *1SL with program variables* are built from expressions of the form $e ::= \mathtt{x} \mid \mathtt{u}$ where $\mathtt{x} \in \mathtt{PVAR}$ and $\mathtt{u} \in \mathtt{FVAR}$, and atomic formulae of the form

$$\pi ::= e = e' \mid e \hookrightarrow e'.$$

Formulae are defined by the grammar

$$\phi ::= \pi \mid \phi \wedge \psi \mid \neg\phi \mid \phi * \psi \mid \phi -\!\!* \psi \mid \exists\, \mathtt{u}\ \phi,$$

where $\mathtt{u} \in \mathtt{FVAR}$. A *valuation* is a map $\mathfrak{f} : \mathtt{FVAR} \to \mathbb{N}$. The satisfaction relation $\models$ is extended as follows:

— $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{f}} e = e'$ iff $[\![e]\!] = [\![e']\!]$, with $[\![\mathtt{x}]\!] \stackrel{\text{def}}{=} \mathfrak{s}(\mathtt{x})$, $[\![\mathtt{u}]\!] \stackrel{\text{def}}{=} \mathfrak{f}(\mathtt{u})$. Obviously, program variables can be understood as free quantified variables interpreted rigidly.
— $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{f}} e \hookrightarrow e'$ iff $[\![e]\!] \in \mathrm{dom}(\mathfrak{h})$ and $\mathfrak{h}([\![e]\!]) = [\![e']\!]$.

The satisfiability problem takes as input a sentence from 1SL with program variables, in which the only free variables are program variables. The version of DSOL with program variables is defined similarly when models are memory states.

THEOREM 6.1. *There is a translation* $\mathtt{T}$ *such that for every sentence* ϕ *in DSOL with program variables, the sentence* $\mathtt{T}(\phi)$ *in 1SL2($-\!\!*$) with program variables (of polynomial size in the size of* ϕ*) is such that for all* $(\mathfrak{s}, \mathfrak{h})$*, we have* $(\mathfrak{s}, \mathfrak{h}) \models \phi$ *iff* $(\mathfrak{s}, \mathfrak{h}) \models \mathtt{T}(\phi)$.

Actually, the translation $\mathtt{T}$ is defined as a variant of the one in Section 5.4 that takes into account program variables. The variant is pretty natural since program variables do not require any encoding.

PROOF. Let us update the translation $\mathtt{T}$ from Section 5 as follows so that we can take into account program variables.

$$\begin{array}{lcl}
t(X, \mathtt{u}_i = \mathtt{x}) & \stackrel{\text{def}}{=} & \exists\, \mathtt{u}\ \mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{u} = \mathtt{x} \\
t(X, \mathtt{x} = \mathtt{x}') & \stackrel{\text{def}}{=} & \mathtt{x} = \mathtt{x}' \\
t(X, \mathtt{x} \hookrightarrow \mathtt{x}') & \stackrel{\text{def}}{=} & \mathtt{x} \hookrightarrow \mathtt{x}' \\
t(X, \mathtt{P}_i(\mathtt{x}, \mathtt{x}')) & \stackrel{\text{def}}{=} & \mathtt{P}_i(\mathtt{x}, \mathtt{x}') \\
t(X, \mathtt{u}_i \hookrightarrow \mathtt{x}) & \stackrel{\text{def}}{=} & \exists\, \mathtt{u}\ (\mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{u} \hookrightarrow \mathtt{x}) \\
t(X, \mathtt{x} \hookrightarrow \mathtt{u}_i) & \stackrel{\text{def}}{=} & \exists\, \mathtt{u}\ (\mathtt{elt}_i(\mathtt{u}) \wedge \mathtt{x} \hookrightarrow \mathtt{u}) \\
t(X, \mathtt{P}_i(\mathtt{u}_j, \mathtt{x})) & \stackrel{\text{def}}{=} & \exists\, \mathtt{u}\ (\mathtt{elt}_j(\mathtt{u}) \wedge \\
 & & \exists\, \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u} \wedge \mathtt{vind}_i(\overline{\mathtt{u}}) \wedge \exists\, \mathtt{u}\ \mathtt{vind}_i(\mathtt{u}) \wedge (\sharp\mathtt{u} = \sharp\overline{\mathtt{u}} + 1 \wedge \mathtt{u} \hookrightarrow \mathtt{x}))) \\
t(X, \mathtt{P}_i(\mathtt{x}, \mathtt{u}_j)) & \stackrel{\text{def}}{=} & \exists\, \mathtt{u}\ ((\mathtt{vind}_i(\mathtt{u}) \wedge \mathtt{u} \hookrightarrow \mathtt{x}) \wedge \\
 & & (\exists\, \overline{\mathtt{u}}\ \mathtt{vind}_i(\overline{\mathtt{u}}) \wedge ((\sharp\overline{\mathtt{u}} = \sharp\mathtt{u} + 1) \wedge \exists\, \mathtt{u}\ (\mathtt{elt}_j(\mathtt{u}) \wedge \overline{\mathtt{u}} \hookrightarrow \mathtt{u})))).
\end{array}$$

This is all that is required to update the translation, except that we should also guarantee that the allocated locations in the valuation heap do not correspond to the interpretation of program variables. Otherwise, this would lead to an unsound reduction.

To do this, we write $\mathtt{PVAR}(\phi)$ to denote the set of program variables occurring in ϕ. In the definition of $t_{top}(\phi)$, we replace $\mathtt{localval}_0(\mathtt{u})$ by the formula below:

$$\mathtt{localval}_0(\mathtt{u}) \wedge \left(\bigwedge_{\mathtt{x} \in \mathtt{PVAR}(\phi)} \neg \mathtt{alloc}(\mathtt{x}) \right) \wedge$$

$$\left(\bigwedge_{\mathtt{x} \in \mathtt{PVAR}(\phi)} (\forall\ \mathtt{u}\ (\mathtt{u} \hookrightarrow \mathtt{x}) \Rightarrow (\mathtt{Lindex}(\mathtt{u}) \wedge \mathtt{eindex}(\mathtt{u}))) \right).$$

Similarly, in the inductive definition of the translation for quantified formulae, we replace $\mathtt{localval}_i(\overline{\mathtt{u}})$ by

$$\mathtt{localval}_i(\overline{\mathtt{u}}) \wedge \left(\bigwedge_{\mathtt{x} \in \mathtt{PVAR}(\phi)} \neg \mathtt{alloc}(\mathtt{x}) \right) \wedge$$

$$\left(\bigwedge_{\mathtt{x} \in \mathtt{PVAR}(\phi)} (\forall\ \mathtt{u}\ (\mathtt{u} \hookrightarrow \mathtt{x}) \Rightarrow (\mathtt{Lindex}(\mathtt{u}) \wedge \mathtt{eindex}(\mathtt{u}))) \right).$$

The notion of X-well-formed heap is slightly updated in order to exclude the possibility that a location in the domain of the valuation heap corresponds to $\mathfrak{s}(\mathtt{x})$ for some $\mathtt{x} \in \mathtt{PVAR}(\phi)$ and moreover, $\mathfrak{s}(\mathtt{x})$ cannot be the index of some parenthesis. We have taken care of that in updating the formulae when subformulae of the form $\mathtt{localval}_i(\overline{\mathtt{u}})$ were present (see above). So, we can establish the lemma below. Actually, the proof is exactly the proof of Lemma 5.11 since we strengthen the assumptions.

LEMMA 6.2 (COMPOSITION). *Let $\mathfrak{f}$ be a valuation, $\mathfrak{h}$ be an X-well-formed heap with $\{0\} \subseteq X \subseteq [0, K]$, $i \in [1, K] \setminus X$ and $i > max(X)$, $\mathfrak{s}$ be a store and $\mathfrak{h}'$ be a disjoint heap such that:*

(1) $(\mathfrak{s}, \mathfrak{h}) \models_{\mathfrak{f}} \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{isoloc}(\overline{\mathtt{u}})$,

(2) $(\mathfrak{s}, \mathfrak{h}') \models_{\mathfrak{f}} \mathtt{localval}_i(\overline{\mathtt{u}}) \wedge \left(\bigwedge_{\mathtt{x} \in \mathtt{PVAR}(\phi)} \neg \mathtt{alloc}(\mathtt{x}) \right) \wedge \left(\bigwedge_{\mathtt{x} \in \mathtt{PVAR}(\phi)} (\forall\ \mathtt{u}\ (\mathtt{u} \hookrightarrow \mathtt{x}) \Rightarrow (\mathtt{Lindex}(\mathtt{u}) \wedge \mathtt{eindex}(\mathtt{u}))) \right)$,

(3) $(\mathfrak{s}, \mathfrak{h} \uplus \mathfrak{h}') \models_{\mathfrak{f}} \mathtt{wfh}_{X \cup \{i\}} \wedge \mathtt{indmin}(\mathtt{u}) \wedge \mathtt{Llp}_i(\overline{\mathtt{u}})$.

Then, we have $spect(\mathfrak{h} \uplus \mathfrak{h}') = spect(\mathfrak{h}) \uplus indspect(\mathfrak{h}')$.

With the above update of the translation, we can also establish correctness below, which leads to the proof of the statement.

LEMMA 6.3 (CORRECTNESS). *Let ϕ be a DSOL sentence with program variables, ψ be one of its subformulae and $(\mathtt{free}(\psi) \cup \{0\}) \subseteq X \subseteq [0, K]$. Let $\mathfrak{h} = \mathfrak{h}_B \uplus \mathfrak{h}_V$ be a X-well-formed heap, $\mathfrak{s}$ be a store and $\mathfrak{V}_{\mathfrak{h}}$ be the valuation extracted from $\mathfrak{h}$. Then, we have $(\mathfrak{s}, \mathfrak{h}_B) \models_{\mathfrak{V}_{\mathfrak{h}}} \psi$ iff $(\mathfrak{s}, \mathfrak{h}) \models t(X, \psi)$.*

The proof of Lemma 6.3 is similar to the proof of Lemma 5.12 except that there are additional base cases since there are new atomic formulae. By way of example, we present the proof for ψ equal to $\mathtt{u}_j = \mathtt{x}$.

Since $\mathfrak{h}$ is X-well-formed and $j \in X$, $\mathfrak{V}_{\mathfrak{h}}(\mathtt{u}_j)$ is equal to $\mathfrak{h}_V(\mathfrak{l})$ where $\mathfrak{l}$ is the unique index location such that $\widetilde{\sharp\mathfrak{l}} \in \mathit{degrees}(j, \mathfrak{h})$.

Let us recall that $\mathtt{elt}_j(\mathtt{u}) = \exists\ \overline{\mathtt{u}}\ (\overline{\mathtt{u}} \hookrightarrow \mathtt{u}) \wedge \mathtt{vind}_j(\overline{\mathtt{u}})$ with $\mathtt{vind}_j(\mathtt{u}) = \mathtt{Lindex}(\mathtt{u}) \wedge \mathtt{eindex}(\mathtt{u}) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{Llp}_j(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} < \sharp\mathtt{u}) \wedge (\exists\ \overline{\mathtt{u}}\ \mathtt{Lrp}_j(\overline{\mathtt{u}}) \wedge \sharp\overline{\mathtt{u}} > \sharp\mathtt{u})$.

First, let us suppose that $(\mathfrak{s}, \mathfrak{h}_B) \models_{\mathfrak{V}_{\mathfrak{h}}} \mathtt{u}_j = \mathtt{x}$. This means that $\mathfrak{V}_{\mathfrak{h}}(\mathtt{u}_j)$ and $\mathfrak{s}(\mathtt{x})$ are equal, say to $\mathfrak{l}'$ and therefore there is a unique index location $\mathfrak{l}_j$ such that $\mathfrak{h}_V(\mathfrak{l}_j) = \mathfrak{l}'$ and $\widetilde{\sharp\mathfrak{l}_j} \in \mathit{degrees}(j, \mathfrak{h})$. Since $\widetilde{\sharp\mathfrak{l}_j}$ belong to the index spectrum of $\mathfrak{h}_V$, the location $\mathfrak{l}_j$, has the same number of predecessors in $\mathfrak{h}$ and in $\mathfrak{h}_V$ and it is also an index in $\mathfrak{h}$ (see Lemma 6.2). Consequently, $(\mathfrak{s}, \mathfrak{h}) \models_{[\mathtt{u} \mapsto \mathfrak{l}']} \mathtt{elt}_j(\mathtt{u})$, whence $(\mathfrak{s}, \mathfrak{h}) \models \exists\ \mathtt{u}\ \mathtt{elt}_j(\mathtt{u}) \wedge \mathtt{u} = \mathtt{x}$.

Now suppose that $(\mathfrak{s}, \mathfrak{h}) \models \exists\ \mathtt{u}\ \mathtt{elt}_j(\mathtt{u}) \wedge \mathtt{u} = \mathtt{x}$. There is a location $\mathfrak{l}'$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{[\mathtt{u} \mapsto \mathfrak{l}']} \mathtt{elt}_j(\mathtt{u}) \wedge \mathtt{u} = \mathtt{x}$. Therefore there are an index $\mathfrak{l}_j$ such that $\widetilde{\sharp\mathfrak{l}_j} \in \mathit{degrees}(j, \mathfrak{h})$, $\mathfrak{h}(\mathfrak{l}_j) = \mathfrak{l}'$ and $\mathfrak{s}(\mathtt{x}) = \mathfrak{l}'$. By Lemma 6.2, $\mathfrak{h}_V(\mathfrak{l}_j) = \mathfrak{l}'$ and $\widetilde{\sharp\mathfrak{l}_j} \in \mathit{indspect}(\mathfrak{h}_V)$. By definition of $\mathfrak{V}_{\mathfrak{h}}$, this implies that $\mathfrak{V}_{\mathfrak{h}}(\mathtt{u}_j) = \mathfrak{s}(\mathtt{x})$ and therefore $(\mathfrak{s}, \mathfrak{h}_B) \models_{\mathfrak{V}_{\mathfrak{h}}} \mathtt{u}_j = \mathtt{x}$.

The base cases for the other atomic formulae are obtained from those in the proof of Lemma 5.12 in a similar fashion. The cases in the induction step are proved similarly. It is worth noting that we can guarantee that for all $\mathtt{x} \in \mathtt{PVAR}(\phi)$, $\mathfrak{s}(\mathtt{x})$ does not fall in the set Y (see the proof of Lemma 5.12) thanks to the addition of the conjunct

$$\Big(\bigwedge_{\mathtt{x} \in \mathtt{PVAR}(\phi)} \neg\mathtt{alloc}(\mathtt{x})\Big) \wedge \Big(\bigwedge_{\mathtt{x} \in \mathtt{PVAR}(\phi)} (\forall\ \mathtt{u}\ (\mathtt{u} \hookrightarrow \mathtt{x}) \Rightarrow (\mathtt{Lindex}(\mathtt{u}) \wedge \mathtt{eindex}(\mathtt{u})))\Big)$$

when a local i-valuation needs to be built. $\square$

6.2. Extension with k record fields

In this work, we have considered memory cells with a unique record field. It is possible to extend our results to $k > 1$ record fields, along the lines of [Brochenin et al. 2012, Section 7]. Let kSL be the (separation) logic in which heaps are partial functions $\mathfrak{h} : \mathbb{N} \rightharpoonup \mathbb{N}^k$ with finite domain and atomic formulae include $\mathtt{u}_j \overset{i}{\hookrightarrow} \mathtt{u}_{j'}$ ($i \in [1, k]$); kWSOL and kDSOL are defined similarly. Let kSL2($-\!\!*$) be the two-variable fragment of kSL with the magic wand as the only separating connective. One can show that every sentence in kDSOL has an equivalent sentence in kSL2($-\!\!*$). To do so, we need to adapt the definitions from Sections 3 and 4 so that memory cells involved in the valuation heap are relevant only with respect to $\overset{1}{\hookrightarrow}$ (and comparing numbers of predecessors is performed only with respect to $\overset{1}{\hookrightarrow}$). Details are tedious because many notions need to be redefined relatively to $\overset{1}{\hookrightarrow}$ but the encoding of valuations is based on the same general principles as for 1SL2($-\!\!*$).

6.3. Allowing infinite domains

Let $k\mathrm{SL}^\infty$ be the variant of kSL in which the heap domain can be either finite or infinite. Remember that in kSL, the heap domain is necessarily finite. The set of valid formulae for $k\mathrm{SL}^\infty$ without separating connectives is recursively enumerable, which contrasts with Theorem 5.16. Indeed, the heap can be encoded by a $(k+1)$-ary relation that is deterministic on its first argument. By contrast, $1\mathrm{SL}^\infty$ with the separating connectives does not admit a recursively enumerable set of valid formulae. Indeed, finiteness of the heap domain is a property that can be internalised in $1\mathrm{SL}^\infty$ with the quite simple formula $\mathtt{seg} \mathrel{-\!\!\!\overrightarrow{*}} \forall\ \mathtt{u}\ \mathtt{alloc}(\mathtt{u})$. By contrast, classical predicate logic cannot specify finiteness of the model.

LEMMA 6.4. *For every heap $\mathfrak{h}$, $\mathit{dom}(\mathfrak{h})$ is infinite iff $\mathfrak{h} \models \mathtt{seg} \mathrel{-\!\!\!\overrightarrow{*}} \forall\ \mathtt{u}\ \mathtt{alloc}(\mathtt{u})$.*

PROOF. Let $\mathfrak{h}$ be a heap such that $\mathrm{dom}(\mathfrak{h})$ is infinite. Let $\mathfrak{l}_1, \mathfrak{l}_2, \ldots$ be an arbitrary enumeration of the locations in $\mathrm{dom}(\mathfrak{h})$. Suppose that $\mathbb{N} \setminus \mathrm{dom}(\mathfrak{h})$ is infinite. Let $\mathfrak{l}'_1, \mathfrak{l}'_2, \ldots$ be an arbitrary enumeration of the locations in $\mathbb{N} \setminus \mathrm{dom}(\mathfrak{h})$. Let $\mathfrak{h}'$ be the heap such that $\mathrm{dom}(\mathfrak{h}') = \mathbb{N} \setminus \mathrm{dom}(\mathfrak{h})$ and for every $i \geq 1$, we set $\mathfrak{h}'(\mathfrak{l}'_i) = \mathfrak{l}_i$. Since $\{\mathfrak{l}_1, \mathfrak{l}_2, \ldots\}, \{\mathfrak{l}'_1, \mathfrak{l}'_2, \ldots\}$ is a partition of $\mathbb{N}$, we have $\mathfrak{h}' \models \mathtt{seg}$. As $\mathrm{dom}(\mathfrak{h}) \uplus \mathrm{dom}(\mathfrak{h}') = \mathbb{N}$, we conclude that $\mathfrak{h} \uplus \mathfrak{h}' \models \forall\, \mathtt{u}\ \mathtt{alloc(u)}$, whence $\mathfrak{h} \models \mathtt{seg} \mathrel{\overrightarrow{-\!\!*}} \forall\, \mathtt{u}\ \mathtt{alloc(u)}$. When $\mathbb{N} \setminus \mathrm{dom}(\mathfrak{h})$ is finite, we follow a similar and even simpler reasoning, the details are omitted.

Now, let $\mathfrak{h}$ be a heap such that $\mathrm{dom}(\mathfrak{h})$ is finite. Suppose that $\mathfrak{h} \models \mathtt{seg} \mathrel{\overrightarrow{-\!\!*}} \forall\, \mathtt{u}\ \mathtt{alloc(u)}$. So, there is a heap $\mathfrak{h}'$ such that $\mathfrak{h}' \models \mathtt{seg}$ and $\mathfrak{h} \uplus \mathfrak{h}' \models \forall\, \mathtt{u}\ \mathtt{alloc(u)}$. Let $\mathfrak{l}_1, \mathfrak{l}_2, \ldots$ be an arbitrary (infinite) enumeration of the locations in $\mathrm{dom}(\mathfrak{h}')$. Since $\mathfrak{h}' \models \mathtt{seg}$, $\mathfrak{h}'(\mathfrak{l}_1), \mathfrak{h}'(\mathfrak{l}_2), \ldots$ is an infinite sequence of distinct locations. Since $\mathrm{dom}(\mathfrak{h})$ is finite, there is i such that $\mathfrak{h}'(\mathfrak{l}_i) \notin \mathrm{dom}(\mathfrak{h})$, say $\mathfrak{h}'(\mathfrak{l}_i) = \mathfrak{l}_j$, which leads to a contradiction because $\mathfrak{h}'$ is supposed to be a segmented heap. □

Let SOL be the variant of WSOL such that quantification over infinite relations is allowed and the models are those of $1\mathrm{SL}^\infty$. The logic SOL is therefore not a 'weak' version of second-order logic since second-order variables are not required to be interpreted by finite relations. It is open whether there exist statements in SOL that cannot be expressed in $1\mathrm{SL}^\infty$. A possible candidate statement could be that there are an infinite number of non-empty connected components that are pairwise isomorphic. Indeed, this property can be expressed in SOL, in particular because infinity of a set can be expressed as well as isomorphism between two connected components. By contrast, it is unclear how to express it in $1\mathrm{SL}^\infty$. Note that if we attempted to adapt the proof from the previous sections to show that $1\mathrm{SL}^\infty$ and SOL have the same expressive power, we encounter a problem. For our encoding (see Section 5), we relied on having an infinite supply of isolated locations to draw from in order to build our valuation heap. If all the locations are "used up," so to speak, this is impossible.

7. CONCLUSION

We have shown that first-order separation logic with one record field, two quantified variables and no separating conjunction, $1\mathrm{SL2}(\mathrel{-\!\!*})$, is as expressive as weak second-order logic on concrete heaps (Theorem 5.13). As a consequence, the satisfiability problem for $1\mathrm{SL2}(\mathrel{-\!\!*})$ is undecidable (Corollary 5.15) and we have identified the undecidable core of separation logic, significantly improving several known results from the literature [Calcagno et al. 2001; Brochenin et al. 2012]. Moreover, this entails that $1\mathrm{SL2}(\mathrel{-\!\!*})$ has no finite axiomatization since weak second-order logic is not finitely axiomatizable.

On the positive side, we have provided means to express rich properties by only using a restricted amount of syntactic resources. Nevertheless, first-order separation logic with one record field, one quantified variable and an unbounded number of program variables, has recently been shown decidable and it admits a PSPACE-complete satisfiability problem [Demri et al. 2014]. Similarly, fragments using the list predicate `ls` and restricted use of the magic wand $\mathrel{-\!\!*}$ or the separation conjunction $*$ are known to admit decision procedures or semi-procedures, see e.g. [Cook et al. 2011; Thakur et al. 2014] and they are useful for formal verification. That is why, there are still some room to find fragments of separation logics that are tailored for verification and that have low computational cost, see also [Sighireanu and Cok 2014] where is described how the first competition of solvers for several fragments of separation logic has been run and which fragments have been considered.

We only use two variables, and our results also exclude separating conjunction, which is quite remarkable in view of the restricted number of variables. As far as the proofs of these results are concerned, we used first principles from [Brochenin et al. 2012] but we had to provide non-trivial adaptations to fit the restricted frag-

ment 1SL2($-\!\!*$). However, this illustrates the robustness of those principles since they could be applied by using the proof techniques developed in the present paper. Other semantical variants are possible; those variants may include the case when heap domain may be infinite (see Section 6.3) or when composition of heaps is allowed as soon as they agree on their common part. Finally, we believe that we identified a remarkable example of a two-variable logic that is equivalent to a queen logic, namely weak second-order logic, and that we have illustrated further the power of separating implication when interpreted on concrete heaps.

Certainly, this work can be completed or refined following several directions. We mention some of them below, sometimes inspired by remarks made by the anonymous referees. At the atomic level, the points-to formulae $\mathtt{x} \hookrightarrow \mathtt{y}$ could be replaced by its exact version $\mathtt{x} \mapsto \mathtt{y}$ or even by its undirected version (which is less relevant in the context of links). How does the undecidability status of 1SL2($-\!\!*$) evolve with such variants? Similarly, we have established undecidability of 1SL2($-\!\!*$) but without placing it precisely in the arithmetical hierarchy: what is an optimal upper bound? Finally, the formulae in 1SL2($-\!\!*$) used to encode the formulae from DSOL are constructed with the septraction (or the magic wand) often with restrictions on the antecedents of septraction. What would be an optimal restriction on antecedents of septraction (for instance) that would preserve undecidability? Answering to most of these questions, would refine further the results presented in the paper.

ACKNOWLEDGMENTS

We would like to thank the anonymous referees for many suggestions that help to improve the quality of the paper. We are also grateful to the anonymous referees of the CSL-LICS 2014 edition of this paper for their suggestions and remarks.

References

T. Antonopoulos. 2010. *Expressive power of query languages*. Ph.D. Dissertation. University of Cambridge.

T. Antonopoulos and A. Dawar. 2009. Separating Graph Logic from MSO. In *FOSSACS'09 (LNCS)*, Vol. 5504. Springer, 63–77.

T. Antonopoulos, N. Gorogiannis, C. Haase, M. Kanovich, and J. Ouaknine. 2014. Foundations for Decision Problems in Separation Logic with General Inductive Predicates. In *FOSSACS'14 (LNCS)*, Vol. 8412. Springer, 411–425.

K. Apt. 1981. Ten Years of Hoare's Logic. *ACM Transactions on Programming Languages and Systems* 3, 4 (1981), 431–483.

K. Bansal, A. Reynolds, T. King, C. Barrett, and Th. Wies. 2015. Deciding local theory extensions via E-matching. In *CAV'15 (LNCS)*, Vol. 9207. Springer, 87–105.

M. Bojanczyk, A. Muscholl, Th. Schwentick, and L. Segoufin. 2009. Two-variable logic on data trees and XML reasoning. *Journal of the Association for Computing Machinery* 56, 3 (2009).

E. Börger, E. Grädel, and Y. Gurevich. 1997. *The Classical Decision Problem*. Springer.

M. Bozga, R. Iosif, and S. Perarnau. 2010. Quantitative Separation Logic and Programs with Lists. *Journal of Automated Reasoning* 45, 2 (2010), 131–156.

D. Bresolin, D. Della Monica, V. Goranko, A. Montanari, and G. Sciavicco. 2010. Metric Propositional Neighborhood Logics: Expressiveness, Decidability, and Undecidability. In *ECAI'10 (Frontiers in Artificial Intelligence and Applications)*, Vol. 215. IOS Press, 695–700.

R. Brochenin. 2013. *Separation Logic: Expressiveness, Complexity, Temporal Extension*. Ph.D. Dissertation. LSV, ENS Cachan.

R. Brochenin, S. Demri, and E. Lozes. 2012. On the Almighty Wand. *Information and Computation* 211 (2012), 106–137.

J. Brotherston and M. Kanovich. 2014. Undecidability of Propositional Separation Logic and Its Neighbours. *Journal of the Association for Computing Machinery* 61, 2 (2014).

C. Calcagno, D. Distefano, P. O'Hearn, and H. Yang. 2011. Compositional Shape Analysis by Means of Bi-Abduction. *Journal of the Association for Computing Machinery* 58, 6 (2011), 26.

C. Calcagno, P. O'Hearn, and H. Yang. 2001. Computability and Complexity Results for a Spatial Assertion Language for Data Structures. In *FSTTCS'01 (LNCS)*, Vol. 2245. Springer, 108–119.

S. Chakraborty. 2012. Reasoning about heap manipulating programs using automata techniques. In *Modern Applications of Automata Theory*, D. D'Souza and P. Shankar (Eds.). IISc Research Monographs Series, Vol. 2. World Scientific, Chapter 7, 193–228.

B. Cook, C. Haase, J. Ouaknine, M. Parkinson, and J. Worrell. 2011. Tractable Reasoning in a Fragment of Separation Logic. In *CONCUR'11 (LNCS)*, Vol. 6901. Springer, 235–249.

A. Dawar, Ph. Gardner, and G. Ghelli. 2007. Expressiveness and complexity of graph logic. *Information and Computation* 205, 3 (2007), 263–310.

S. Demri and M. Deters. 2014. Expressive completeness of separation logic with two variables and no separating conjunction. In *CSL-LICS'14*. ACM, 37.

S. Demri and M. Deters. 2015. Two-variable separation logic and its inner circle. *ACM Transactions on Computational Logic* 16, 2 (2015), 15.

S. Demri, D. Galmiche, D. Larchey-Wendling, and D. Mery. 2014. Separation Logic with One Quantified Variable. In *CSR'14 (LNCS)*, Vol. 8476. Springer, 125–138.

K. Etessami, M. Vardi, and Th. Wilke. 1997. First-Order Logic with Two variables and Unary Temporal logics. In *LICS'97*. IEEE, 228–235.

D. Gabbay. 1981. Expressive Functional Completeness in Tense Logic. In *Aspects of Philosophical Logic*. Reidel, 91–117.

D. Gabbay, I. Hodkinson, and M. Reynolds. 1994. *Temporal Logic - Mathematical Foundations and Computational Aspects, Vol. 1*. Oxford University Press.

D. Galmiche and D. Méry. 2010. Tableaux and Resource Graphs for Separation Logic. *Journal of Logic and Computation* 20, 1 (2010), 189–231.

E. Grädel, Ph. Kolaitis, and M. Vardi. 1997. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic* 3, 1 (1997), 53–69.

E. Grädel, M. Otto, and E. Rosen. 1999. Undecidability Results on Two-Variable Logics. *Arch. Mathematical Logic* 38, 4–5 (1999), 313–354.

C. Haase, S. Ishtiaq, J. Ouaknine, and M. Parkinson. 2013. SeLoger: A Tool for Graph-Based Reasoning in Separation Logic. In *CAV'13 (LNCS)*, Vol. 8044. Springer, 790–795.

Z. Hou, R. Clouston, R. Goré, and A. Tiu. 2014. Proof search for propositional abstract separation logics via labelled sequents. In *POPL'14*. ACM, 465–476.

Z. Hou, R. Goré, and A. Tiu. 2015. Automated Theorem Proving for Assertions in Separation Logic with All Connectives. In *CADE'15 (LNCS)*, Vol. 9195. Springer, 501–516.

N. Immerman, A. Rabinovich, Th. Reps, M. Sagiv, and G. Yorsh. 2004. The Boundary Between Decidability and Undecidability for Transitive-Closure Logics. In *CSL'04 (LNCS)*, Vol. 3210. Springer, 160–174.

R. Iosif, A. Rogalewicz, and J. Simacek. 2013. The Tree Width of Separation Logic with Recursive Definitions. In *CADE'13 (LNCS)*, Vol. 7898. Springer, 21–38.

D. Janin and I. Walukiewicz. 1996. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *CONCUR'96 (LNCS)*, Vol. 1119. 263–277.

H. Kamp. 1968. *Tense Logic and the theory of linear order*. Ph.D. Dissertation. UCLA, USA.

V. Kuncak and M. Rinard. 2004. *On spatial conjunction as second-order logic*. Technical Report MIT–CSAIL–TR–2004–067. MIT CSAIL.

M. Lange. 2007. Linear Time Logics around PSL: Complexity, Expressiveness, and a little bit of Succinctness. In *CONCUR'07 (LNCS)*, Vol. 4703. Springer, 90–104.

D. Larchey-Wendling and D. Galmiche. 2013. Nondeterministic Phase Semantics and the Undecidability of Boolean BI. *ACM Transactions on Computational Logic* 14, 1 (2013).

W. Lee and S. Park. 2014. A proof system for separation logic with magic wand. In *POPL'14*. ACM, 477–490.

E. Lozes. 2012. Separation Logic: Expressiveness and Copyless Message-Passing. ENS Cachan. (2012). Habilitation thesis.

C. Lutz, U. Sattler, and F. Wolter. 2001. Modal Logic and the Two-Variable Fragment. In *CSL'01 (LNCS)*, Vol. 2142. Springer, 247–261.

J. Marcinkowski. 2006. On the expressive power of graph logic. In *CSL'06 (LNCS)*, Vol. 4207. Springer, 486–500.

M. Marx and M. de Rijke. 2005. Semantic characterizations of navigational XPath. *SIGMOD Record* 34, 2 (2005), 41–46.

J. Navarro Pérez and A. Rybalchenko. 2013. Separation Logic Modulo Theories. In *APLAS'13 (LNCS)*, Vol. 8301. 90–106.

R. Piskac, Th. Wies, and D. Zufferey. 2013. Automating Separation Logic using SMT. In *CAV'13 (LNCS)*, Vol. 8044. Springer, 773–789.

R. Piskac, Th. Wies, and D. Zufferey. 2014. GRASShopper - Complete Heap Verification with Mixed Specifications. In *TACAS'14 (LNCS)*, Vol. 8413. Springer, 124–139.

A. Rabinovich. 2014. A Proof of Kamp's theorem. *LMCS* 10, 1 (2014).

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

M. Schwerhoff and A. Summers. 2015. Lightweight support for magic wands in an automatic verifier. In *ECOOP'15*. Leibniz-Zentrum für Informatik, LIPICS, 999–1023.

M. Sighireanu and D. Cok. 2014. Report on SL-COMP 2014. *Journal of Satisfiability, Boolean Modeling and Computation* (2014). To appear.

A.V. Sreejith. 2013. *Regular Quantifiers in Logic*. Ph.D. Dissertation. The Institute of Mathematical Sciences, Chennai.

H. Straubing. 1994. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser.

A. Thakur, J. Breck, and Th. Reps. 2014. Satisfiability modulo abstraction for separation logic with linked lists. In *SPIN'14*. ACM, 58–67.

B. Trakhtenbrot. 1963. Impossibility of an algorithm for the decision problem in finite classes. *AMS Translations, Series 2* 23 (1963), 1–5.

V. Vafeiadis and M. Parkinson. 2007. A Marriage of Rely/Guarantee and Separation Logic. In *CONCUR'07 (LNCS)*, Vol. 4703. Springer, 256–271.

M.Y. Vardi. 1988. A temporal fixpoint calculus. In *15th Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, San Diego*. ACM, 250–259.

Y. Venema. 1991. A Modal Logic for Chopping Intervals. *Journal of Logic and Computation* 1, 4 (1991), 453–476.

Ph. Weis. 2011. *Expressiveness and Succinctness of First-Order Logic on Finite Words*. Ph.D. Dissertation. University of Massachussetts.

P. Wolper. 1983. Temporal logic can be more expressive. *Information and Computation* 56 (1983), 72–99.