Semipositivity in Separation Logic With Two Variables [⋆]

Zhilin Wu

State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences,
Beijing, P. R. China

Abstract. In a recent work by Demri and Deters (CSL-LICS 2014), first-order separation logic restricted to two variables and separating implication was shown undecidable, where it was shown that even with only two variables, if the use of negations is unrestricted, then they can be nested with separating implication in a complex way to get the undecidability result. In this paper, we revisit the decidability and complexity issues of first-order separation logic with two variables, and proposes *semi-positive* separation logic with two-variables (SPSL2), where the use of negations is restricted in the sense that negations can only occur in front of atomic formulae. We prove that satisfiability of the fragment of SPSL2 where neither separating conjunction nor septraction (the dual operator of separating implication) occurs in the scope of universal quantifiers, is NEXPTIME-complete. As a byproduct of the proof, we show that the finite satisfiability problem of first-order logic with two variables and a bounded number of function symbols is NEXPTIME-complete (the lower bound holds even with only one function symbol and without unary predicates), which may be of independent interest beyond separation logic community.

1 Introduction

Decidability and Separation Logics. Separation logic is a prominent logical formalism to verify programs with pointers and it comes in different flavours and many fragments and extensions exist. The decidability status of first-order separation logic with two record fields has been answered negatively quite early in [5] thanks to Trakhtenbrot's Theorem [23]: finitary satisfiability for predicate logic restricted to a single binary predicate symbol is undecidable and not recursively enumerable. The undecidability of first-order separation logic with a single record field was then established in [4] and a bit later in [8] with the further restriction that only two individual variables are permitted. Undecidability can be established in various ways: in [9] by reduction from the halting problem for Minsky machines [15] or from the satisfiability problem for FO2 on data words with a linear ordering on data [3]. Despite these negative results, many fragments of separation logic are known to be decidable and used in practice, mainly thanks to the absence of the separation implication, see e.g. [20, 7, 17, 1]. For instance, the symbolic-heap fragment is free of separating implication and the propositional fragment of separation logic can be decided in polynomial space [5]. Semi-decision procedures for fragments with separating implication can be found in [22]. First-order separation logic with all separating connectives but with a single variable is shown in PSPACE in [10].

[⋆] Partially supported by the NSFC grants (Nos. 61100062, 61272135, 61472474, and 61572478).

Our Motivations. Undecidable fragments of separation logic allow still too much freedom whereas the decidable fragments with relatively low complexity are still poorly expressive. The real question is how to reduce this gap by introducing restrictions based on negations (see also the related work [21] about restrictions on negations). In this paper, we consider semi-positive separation logic with two variables, denoted by SPSL2, where negation symbols only occur in front of the atomic formulae. Our goal is to understand the influence of the restricted use of negations on the decidability/complexity of the satisfiability problem.

We know that SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, *\right)$ (where $\mathbb{F}$ is a finite set of fields, $\overset{\mathfrak{f}}{\hookrightarrow}$ and $*$ represent the "points-to" and "separating conjunction" modality respectively), the fragment of SPSL2 where separating implication does not occur, admits a decidable satisfiability problem if $\mathbb{F} = \{\mathfrak{f}\}$ (that is, there is exactly one field), since SL2$\left(\overset{\mathfrak{f}}{\hookrightarrow}, *\right)$, the smallest extension of SPSL2$\left(\overset{\mathfrak{f}}{\hookrightarrow}, *\right)$ closed under negations, is decidable with a non-elementary computational complexity (cf. [4], the lower bound with only two variables was shown in [9]). Nevertheless, to the best of our knowledge, the decidability and complexity of various fragments of SPSL2 are still largely open.

Our contributions.

As a starting point towards a complete decidability/complexity charaterization of SPSL2, we show that the satisfiability of the following fragments of SPSL2 is NEXPTIME-complete (cf. Section 2 for the definition of these fragments).

1. SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}\right)$ and SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}\right)$, where separating operators do not occur, and $\mathcal{P}$ denotes a finite set of unary predicates. The NEXPTIME lower bound holds even if there is only one field, that is, $\mathbb{F} = \{\mathfrak{f}\}$. The upper bound proof is obtained by a reduction to the finite satisfiability of first-order logic with two variables and counting quantifiers, which is NEXPTIME-complete [18, 19]. The lower bound is shown by encoding the solutions of a given exponential-size tiling problem into a formula in SPSL2$\left(\overset{\mathfrak{f}}{\hookrightarrow}\right)$, where only one function symbol $\mathfrak{f}$ is used and no unary predicates are needed.
2. ESPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, *, \stackrel{\neg}{\twoheadrightarrow}\right)$, the extension of SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}\right)$ with separating conjunction $*$ and septraction $\stackrel{\neg}{\twoheadrightarrow}$ (the dual operator of magic wand $\twoheadrightarrow$), where neither $*$ nor $\stackrel{\neg}{\twoheadrightarrow}$ occurs in the scope of universal quantifiers. The result is obtained by a reduction to the satisfiability of SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}\right)$ formulae.

Related work. Logics with two variables are a classical topic in mathematical logic and theoretical computer science. Over arbitrary relational structures, first-order logic with two variables and its extensions have been investigated intensively, see [13, 14, 12, 16, 18, 19] (to cite a few). In [18, 19], it is well-known that the satisfiability and finite satisfiability problem of $\mathcal{C}^2$, first-order logic with two variables and counting quantifiers, are NEXPTIME-complete. Since function symbols can be encoded by relation symbols with the help of counting quantifiers, it follows that the satisfiability problem of SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}\right)$ is in NEXPTIME. In addition, it was shown that the (finite) satisfiability problem of first-order logic with two variables and unary predicates is already NEXPTIME-hard. The NEXPTIME lower bound we obtained is novel in the sense that in our reduction, only one function symbol but no unary predicates is used. First-order logic with two variables on special classes of structures, e.g. words and trees, has also been investigated [11, 2]. In [6], first-order logic with two variables and deterministic transitive closure over one binary relation was considered and its the satisfiability problem was shown to be EXPSPACE-complete.

Outline of the paper. Preliminaries are given in Section 2. In Section 3, the satisfiability of SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}\right)$ and SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}\right)$ is shown to be NEXPTIME-complete. Section 4 is devoted to ESPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, *, \overset{\neg}{\twoheadrightarrow}\right)$.

2 Preliminaries

For $n \in \mathbb{N}$, let $[n]$ denote the set $\{0, \ldots, n-1\}$. For $n, m \in \mathbb{N}$ such that $n \leq m$, let $[n, m] = \{n, n+1, \ldots, m\}$. Let $\mathbb{F}$ denote a finite set of *fields*. A *heap* $\mathfrak{h}$ over $\mathbb{F}$ is a collection of partial functions $(\mathfrak{h}_\mathfrak{f})_{\mathfrak{f}\in\mathbb{F}} : \mathbb{N} \rightharpoonup \mathbb{N}$ such that each of them has a finite domain. We write $\mathtt{dom}(\mathfrak{h}_\mathfrak{f})$ to denote the *domain* of $\mathfrak{h}_\mathfrak{f}$ and $\mathtt{ran}(\mathfrak{h}_\mathfrak{f})$ to denote its *range*. In addition, we use $\mathtt{loc}(\mathfrak{h}_\mathfrak{f})$ to denote $\mathtt{dom}(\mathfrak{h}_\mathfrak{f}) \cup \mathtt{ran}(\mathfrak{h}_\mathfrak{f})$. We also use $\mathtt{dom}(\mathfrak{h})$ to denote $\bigcup\limits_{\mathfrak{f}\in\mathbb{F}} \mathtt{dom}(\mathfrak{h}_\mathfrak{f})$, $\mathtt{ran}(\mathfrak{h})$ to denote $\bigcup\limits_{\mathfrak{f}\in\mathbb{F}} \mathtt{ran}(\mathfrak{h}_\mathfrak{f})$, and $\mathtt{loc}(\mathfrak{h})$ to denote $\bigcup\limits_{\mathfrak{f}\in\mathbb{F}} \mathtt{loc}(\mathfrak{h}_\mathfrak{f})$. Two heaps $\mathfrak{h}_1 = (\mathfrak{h}_{1,\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}$ and $\mathfrak{h}_2 = (\mathfrak{h}_{2,\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}$ are said to be *disjoint*, denoted $\mathfrak{h}_1 \perp \mathfrak{h}_2$, if $\mathtt{dom}(\mathfrak{h}_1) \cap \mathtt{dom}(\mathfrak{h}_2) = \emptyset$; when this holds, we write $\mathfrak{h}_1 \uplus \mathfrak{h}_2$ to denote the collection of partial functions $(\mathfrak{h}_\mathfrak{f})_{\mathfrak{f}\in\mathbb{F}}$ such that $\mathfrak{h}_\mathfrak{f}$ is obtained from $\mathfrak{h}_{1,\mathfrak{f}}$ and $\mathfrak{h}_{2,\mathfrak{f}}$ by taking their disjoint union.

We introduce some graph-theoretical notations for heaps. Let $\mathfrak{h} = (\mathfrak{h}_\mathfrak{f})_{\mathfrak{f}\in\mathbb{F}}$ and $\mathfrak{f} \in \mathbb{F}$.

- Let $\mathfrak{l}$ and $\mathfrak{l}'$ be two locations. If $\mathfrak{h}_\mathfrak{f}(\mathfrak{l}) = \mathfrak{l}'$, then $\mathfrak{l}'$ is said to be the $\mathfrak{f}$-*successor* of $\mathfrak{l}$ (resp. $\mathfrak{l}$ is said to be an $\mathfrak{f}$-*predecessor* of $\mathfrak{l}'$) in $\mathfrak{h}$.
- An $\mathfrak{f}$-*path* in $\mathfrak{h}$ is a sequence of locations, say $\mathfrak{l}_0\mathfrak{l}_1 \ldots \mathfrak{l}_k$ (where $k \geq 0$), such that for each $j : 0 \leq j < k$, $\mathfrak{l}_{j+1}$ is the $\mathfrak{f}$-successor of $\mathfrak{l}_j$. The location $\mathfrak{l}_0$ and $\mathfrak{l}_k$ are called the *start location* and *end location* of the $\mathfrak{f}$-path respectively. In addition, k is called the *length* of the $\mathfrak{f}$-path. An $\mathfrak{f}$-path $\mathfrak{l}_0\mathfrak{l}_1 \ldots \mathfrak{l}_k$ is called an $\mathfrak{f}$-*cycle* if $k \geq 1$ and $\mathfrak{l}_0 = \mathfrak{l}_k$. If $\mathfrak{l} \in \mathbb{N}$ is the end location of an $\mathfrak{f}$-path, then we also call the $\mathfrak{f}$-path as a *backward* $\mathfrak{f}$-*path* of $\mathfrak{l}$.
- Let $\mathcal{G}[\mathfrak{h}_\mathfrak{f}]$ be the directed graph corresponding to $\mathfrak{h}_\mathfrak{f}$, that is, the graph where the set of nodes is $\mathtt{loc}(\mathfrak{h}_\mathfrak{f})$, and for each pair of locations $\mathfrak{l}, \mathfrak{l}' \in \mathtt{loc}(\mathfrak{h}_\mathfrak{f})$, there is an arc from $\mathfrak{l}$ to $\mathfrak{l}'$ iff $\mathfrak{h}_\mathfrak{f}(\mathfrak{l}) = \mathfrak{l}'$. Note that $\mathcal{G}[\mathfrak{h}_\mathfrak{f}]$ has a special structure in the sense that each node has at most one successor (as a result of the fact that $\mathfrak{h}_\mathfrak{f}$ is a partial function). Suppose that $\mathcal{C}$ is a connected component of $\mathcal{G}[\mathfrak{h}_\mathfrak{f}]$, then the partial function $\mathfrak{h}'_\mathfrak{f}$ such that $\mathcal{G}[\mathfrak{h}'_\mathfrak{f}] = \mathcal{C}$ (this means that $\mathtt{loc}(\mathfrak{h}'_\mathfrak{f})$ is the set of nodes in $\mathcal{C}$, and $\mathfrak{h}'_\mathfrak{f}(\mathfrak{l}) = \mathfrak{l}'$ iff there is an arc from $\mathfrak{l}$ to $\mathfrak{l}'$ in $\mathcal{C}$ iff $\mathfrak{h}_\mathfrak{f}(\mathfrak{l}) = \mathfrak{l}'$), is called a *connected component* of $\mathfrak{h}$.

Formulae in $\mathrm{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, *, \mathrel{-\!\!*}\right)$, *semi-positive separation logic with two variables*, are defined by the following rules:

$$\begin{aligned}
&\mathtt{v} ::= \mathtt{x} \mid \mathtt{y},\\
&\alpha ::= \mathtt{v} = \mathtt{v} \mid \mathtt{v} \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{v},\\
&\phi ::= \alpha \mid \neg\alpha \mid \phi \vee \phi \mid \phi \wedge \phi \mid \exists \mathtt{v}.\phi \mid \forall \mathtt{v}.\phi \mid \phi * \phi \mid \phi \mathrel{-\!\!*} \phi,
\end{aligned}$$

where $\mathtt{x}$ and $\mathtt{y}$ are two distinguished first-order variables, and $\mathfrak{f} \in \mathbb{F}$.

We write $\mathrm{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}\right)$ to denote the fragment of $\mathrm{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, *, \mathrel{-\!\!*}\right)$ without separating connectives (remove the last two rules in the definition of ϕ). Note that $\mathrm{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}\right)$ can be seen as first-order logic with two variables and $|\mathbb{F}|$ function symbols (where $|\mathbb{F}|$ denote the cardinality of $\mathbb{F}$). Similarly, we write $\mathrm{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, *\right)$ to denote the fragment of $\mathrm{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, *, \mathrel{-\!\!*}\right)$ without separating implication (remove the last rule in the definition of ϕ). We use $|\phi|$ to denote the size of ϕ. In addition, the set of subformulae of ϕ, denoted by $\mathtt{Sub}(\phi)$, can be defined in a standard way.

An *assignment* is a map $\mathfrak{m} : \{\mathtt{x}, \mathtt{y}\} \to \mathbb{N}$. For an assignment $\mathfrak{m}$ and $\mathfrak{l} \in \mathbb{N}$, we use $\mathfrak{m}[\mathtt{v} \mapsto \mathfrak{l}]$ denote the assignment that is the same as $\mathfrak{m}$, except that it maps $\mathtt{v}$ to $\mathfrak{l}$ (where $\mathtt{v} = \mathtt{x}, \mathtt{y}$). The satisfaction relation $\models$ is parameterised by assignments and defined as follows (clauses are omitted when these can be obtained by permuting the two variables below):

- $\mathfrak{h} \models_{\mathfrak{m}} \mathtt{v}_1 = \mathtt{v}_2 \overset{\text{def}}{\Leftrightarrow} \mathfrak{m}(\mathtt{v}_1) = \mathfrak{m}(\mathtt{v}_2)$.
- $\mathfrak{h} \models_{\mathfrak{m}} \neg(\mathtt{v}_1 = \mathtt{v}_2) \overset{\text{def}}{\Leftrightarrow} \mathfrak{m}(\mathtt{v}_1) \neq \mathfrak{m}(\mathtt{v}_2)$.
- $\mathfrak{h} \models_{\mathfrak{m}} \mathtt{v}_1 \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{v}_2 \overset{\text{def}}{\Leftrightarrow} \mathfrak{m}(\mathtt{v}_1) \in \mathtt{dom}(\mathfrak{h}_{\mathfrak{f}})$ and $\mathfrak{h}_{\mathfrak{f}}(\mathfrak{m}(\mathtt{v}_1)) = \mathfrak{m}(\mathtt{v}_2)$.
- $\mathfrak{h} \models_{\mathfrak{m}} \neg(\mathtt{v}_1 \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{v}_2) \overset{\text{def}}{\Leftrightarrow} \mathfrak{m}(\mathtt{v}_1) \notin \mathtt{dom}(\mathfrak{h}_{\mathfrak{f}})$ or otherwise $\mathfrak{h}_{\mathfrak{f}}(\mathfrak{m}(\mathtt{v}_1)) \neq \mathfrak{m}(\mathtt{v}_2)$.
- $\mathfrak{h} \models_{\mathfrak{m}} \phi_1 \wedge \phi_2 \overset{\text{def}}{\Leftrightarrow} \mathfrak{h} \models_{\mathfrak{m}} \phi_1$ and $\mathfrak{h} \models_{\mathfrak{m}} \phi_2$.
- $\mathfrak{h} \models_{\mathfrak{m}} \phi_1 \vee \phi_2 \overset{\text{def}}{\Leftrightarrow} \mathfrak{h} \models_{\mathfrak{m}} \phi_1$ or $\mathfrak{h} \models_{\mathfrak{m}} \phi_2$.
- $\mathfrak{h} \models_{\mathfrak{m}} \phi_1 * \phi_2 \overset{\text{def}}{\Leftrightarrow}$ there exist $\mathfrak{h}_1, \mathfrak{h}_2$ such that $\mathfrak{h}_1 \perp \mathfrak{h}_2$, $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$, $\mathfrak{h}_1 \models_{\mathfrak{m}} \phi_1$ and $\mathfrak{h}_2 \models_{\mathfrak{m}} \phi_2$.
- $\mathfrak{h} \models_{\mathfrak{m}} \phi_1 \mathrel{-\!\!*} \phi_2 \overset{\text{def}}{\Leftrightarrow}$ for all $\mathfrak{h}'$, if $\mathfrak{h} \perp \mathfrak{h}'$ and $\mathfrak{h}' \models_{\mathfrak{m}} \phi_1$ then $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{m}} \phi_2$.
- $\mathfrak{h} \models_{\mathfrak{m}} \exists \mathtt{v}.\phi \overset{\text{def}}{\Leftrightarrow}$ there is $\mathfrak{l} \in \mathbb{N}$ such that $\mathfrak{h} \models_{\mathfrak{m}[\mathtt{v}\mapsto\mathfrak{l}]} \phi$.
- $\mathfrak{h} \models_{\mathfrak{m}} \forall \mathtt{v}.\phi \overset{\text{def}}{\Leftrightarrow}$ for every $\mathfrak{l} \in \mathbb{N}$, $\mathfrak{h} \models_{\mathfrak{m}[\mathtt{v}\mapsto\mathfrak{l}]} \phi$.

If ϕ is a sentence, we also omit $\mathfrak{m}$ and we write $\mathfrak{h} \models \phi$ since $\mathfrak{m}$ is irrelevant in this case. A formula ϕ is *satisfiable* if there is a pair $(\mathfrak{h}, \mathfrak{m})$ such that $\mathfrak{h} \models_{\mathfrak{m}} \phi$. The satisfiability problem asks whether ϕ is satisfiable, given a formula ϕ.

We are also interested another separating operator $\overset{\neg}{\mathrel{-\!\!*}}$, called "septraction", which is the dual operator of $\mathrel{-\!\!*}$, that is, $\phi_1 \overset{\neg}{\mathrel{-\!\!*}} \phi_2 \equiv \neg(\phi_1 \mathrel{-\!\!*} \neg\phi_2)$. More specifically, $\mathfrak{h} \models_{\mathfrak{m}} \phi_1 \overset{\neg}{\mathrel{-\!\!*}} \phi_2 \overset{\text{def}}{\Leftrightarrow}$ there is $\mathfrak{h}'$ such that $\mathfrak{h} \perp \mathfrak{h}'$, $\mathfrak{h}' \models_{\mathfrak{m}} \phi_1$, and $\mathfrak{h} \uplus \mathfrak{h}' \models_{\mathfrak{m}} \phi_2$. Then we can define the logic $\mathrm{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, *, \overset{\neg}{\mathrel{-\!\!*}}\right)$ by replacing the rule $\phi \mathrel{-\!\!*} \phi$ in the definition of

SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}, *, -\!\!*\right)$ with $\phi \mathrel{\overline{-\!\!*}} \phi$. Note that if the rule $\phi \mathrel{\overline{-\!\!*}} \phi$ was *added* to the definition of SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}, *, -\!\!*\right)$, then the resulting logic would become undecidable ([8]).

In this paper, we also consider an additional fragment whose definition is presented below. Let ESPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}, *, \overline{-\!\!*}\right)$ denote existential SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}, *, \overline{-\!\!*}\right)$, which is the extension of SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}\right)$ with $*$ and $\overline{-\!\!*}$ such that no occurrences of $*$ and $\overline{-\!\!*}$ are in the scope of universal quantifiers. More precisely, ESPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}, *, \overline{-\!\!*}\right)$ formulae ψ are defined by the following rules,

$$\psi ::= \phi \mid \psi \vee \psi \mid \psi \wedge \psi \mid \exists \mathtt{v}.\ \psi \mid \psi * \psi \mid \psi \mathrel{\overline{-\!\!*}} \psi,$$

where ϕ is an SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}\right)$ formula. Note that since SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}\right)$ formulae may contain universal quantifiers, we notice that universal quantifiers may still occur in the ESPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}, *, \overline{-\!\!*}\right)$ formulae.

Example 1. Let $\psi \overset{\text{def}}{=} (\neg\, \mathtt{x} = \mathtt{y}) \wedge (\psi'_1 \mathrel{\overline{-\!\!*}} \psi'_2) \mathrel{\overline{-\!\!*}} (\psi'_3 \mathrel{\overline{-\!\!*}} \psi'_4)$, where

- $\psi'_1 \overset{\text{def}}{=} \mathtt{x} \xhookrightarrow{\mathfrak{f}} \mathtt{y}$ expresses that $\mathtt{y}$ is the $\mathfrak{f}$-successor of $\mathtt{x}$,
- $\psi'_2 \overset{\text{def}}{=} (\exists \mathtt{y}.\ \mathtt{x} \xhookrightarrow{\mathfrak{f}} \mathtt{y}) \wedge (\exists \mathtt{y}.\ \mathtt{y} \xhookrightarrow{\mathfrak{f}} \mathtt{x})$ expresses that the $\mathfrak{f}$-successor of $\mathtt{x}$ exists and there is an $\mathfrak{f}$-predecessor of $\mathtt{x}$,
- $\psi'_3 \overset{\text{def}}{=} \neg\, \mathtt{x} \xhookrightarrow{\mathfrak{f}} \mathtt{y} \wedge \exists \mathtt{y}.\ (\mathtt{x} \xhookrightarrow{\mathfrak{f}} \mathtt{y} \wedge \forall \mathtt{x}.\ \neg \mathtt{y} \xhookrightarrow{\mathfrak{f}} \mathtt{x})$ expresses that $\mathtt{y}$ is not the $\mathfrak{f}$-successor of $\mathtt{x}$, but the $\mathfrak{f}$-successor of $\mathtt{x}$ exists, and the $\mathfrak{f}$-successor of $\mathtt{x}$ has no $\mathfrak{f}$-successor, and
- $\psi'_4 \overset{\text{def}}{=} ((\exists \mathtt{y}.\ \mathtt{y} \xhookrightarrow{\mathfrak{f}} \mathtt{x}) * (\exists \mathtt{y}.\ \mathtt{y} \xhookrightarrow{\mathfrak{f}} \mathtt{x})) \wedge \exists \mathtt{y}.\ (\mathtt{x} \xhookrightarrow{\mathfrak{f}} \mathtt{y} \wedge \exists \mathtt{x}.\ (\mathtt{y} \xhookrightarrow{\mathfrak{f}} \mathtt{x}))$ expresses that there are two distinct $\mathfrak{f}$-predecessors of $\mathtt{x}$ and a path of length at least two starting from $\mathtt{x}$.

Then ψ is an ESPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}, *, \overline{-\!\!*}\right)$ formula. It is not hard to see that $\mathfrak{h} \models_{\mathfrak{m}} \psi'_1 \mathrel{\overline{-\!\!*}} \psi'_2$ iff $\mathfrak{m}(\mathtt{x})$ has an $\mathfrak{f}$-predecessor in $\mathfrak{h}$, and $\mathfrak{h} \models_{\mathfrak{m}} \psi'_3 \mathrel{\overline{-\!\!*}} \psi'_4$ iff $\mathfrak{m}(\mathtt{x})$ has two $\mathfrak{f}$-predecessors in $\mathfrak{h}$ and there is a location $\mathfrak{l} \neq \mathfrak{m}(\mathtt{y})$ such that the $\mathfrak{f}$-successor of $\mathfrak{l}$ exists in $\mathfrak{h}$. Therefore, $\mathfrak{h} \models_{\mathfrak{m}} \psi$ iff $\mathfrak{m}(\mathtt{x}) \neq \mathfrak{m}(\mathtt{y})$, there is an $\mathfrak{f}$-predecessor of $\mathfrak{m}(\mathtt{x})$ in $\mathfrak{h}$, and there is a location $\mathfrak{l} \neq \mathfrak{m}(\mathtt{y})$ such that the $\mathfrak{f}$-successor of $\mathfrak{l}$ exists in $\mathfrak{h}$. □

For the presentation of the decision procedure for ESPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}, *, \overline{-\!\!*}\right)$ in Section 4, the extension of SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f}\in\mathbb{F}}\right)$ with *unary predicates* is also relevant.

Let $\mathcal{P}$ be a finite set of unary predicates. The extension of $\text{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}\right)$ with unary predicates from $\mathcal{P}$, denoted by $\text{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}\right)$, is defined by the syntax rules of $\text{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}\right)$, plus two new rules $\phi ::= P(\mathtt{v}) \mid \neg P(\mathtt{v})$, where $P \in \mathcal{P}$.

The semantics of $\text{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}\right)$ formulae are defined as a relation $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}} \phi$, where $\mathfrak{h}, \mathfrak{m}$ are as before, and $\mathfrak{I} : \mathbb{N} \to 2^{\mathcal{P}}$ is a function such that $\mathtt{dom}(\mathfrak{I}) = \{\mathfrak{l} \in \mathbb{N} \mid \mathfrak{I}(\mathfrak{l}) \neq \emptyset\}$ is finite. The relation $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}} \phi$ is a natural extension of the relation $\mathfrak{h} \models_{\mathfrak{m}} \phi$ defined above, where $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}} P(\mathtt{v}) \overset{\text{def}}{\Leftrightarrow} P \in \mathfrak{I}(\mathfrak{m}(\mathtt{v}))$, and $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}} \neg P(\mathtt{v}) \overset{\text{def}}{\Leftrightarrow} P \notin \mathfrak{I}(\mathfrak{m}(\mathtt{v}))$. The function $\mathfrak{I}$ can also be seen in another way: It assigns each $P \in \mathcal{P}$ a finite subset of $\mathbb{N}$, that is, the set $\{\mathfrak{l} \in \mathbb{N} \mid P \in \mathfrak{I}(\mathfrak{l})\}$. The pairs $(\mathfrak{h}, \mathfrak{I})$ are called *labeled heaps*. Note that in a labeled heap $(\mathfrak{h}, \mathfrak{I})$, there may exist $\mathfrak{l} \notin \mathtt{loc}(\mathfrak{h})$ such that $\mathfrak{I}(\mathfrak{l}) \neq \emptyset$, in other words, $\mathtt{dom}(\mathfrak{I})$ may not necessarily be a subset of $\mathtt{loc}(\mathfrak{h})$.

3 $\text{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}\right)$ and $\text{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}\right)$

This section is devoted to the proof of the following result.

Theorem 1. *The satisfiability of* $\text{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}\right)$ *and* $\text{SPSL2}\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}\right)$ *is* NEXPTIME-*complete.*

The rest of this section is devoted to the proof of Theorem 1. We consider the lower bound first, then the upper bound.

3.1 Lower bound

We show the lower bound for the special case that $\mathbb{F} = \{\mathfrak{f}\}$ and $\text{SPSL2}(\overset{\mathfrak{f}}{\hookrightarrow})$, that is, the satisfiability problem is NEXPTIME-hard even if there is only one field and there are no unary predicates. Since $\mathbb{F} = \{\mathfrak{f}\}$, in the following, for brevity, we will omit $\mathfrak{f}$ in the proof of the lower bound. The lower bound is obtained by a reduction from the exponential-size tiling problem. The problem is defined as follows: Given a tuple $(D, H, V, \mathfrak{u})$, where $D = \{\mathfrak{d}_1, \ldots, \mathfrak{d}_s\}$ is a finite set of *tiles*, $H, V \subseteq D \times D$, $\mathfrak{u} = \mathfrak{u}_0 \ldots \mathfrak{u}_{n-1} \in D^n$, decide whether there is a tiling $\mathfrak{t} : [2^n] \times [2^n] \to D$ such that

- horizontal constraint: for all $i, j \in [2^n]$, if $\mathfrak{t}(i, j) = \mathfrak{d}$ and $\mathfrak{t}(i+1, j) = \mathfrak{d}'$, then $(\mathfrak{d}, \mathfrak{d}') \in H$,
- vertical constraint: for all $i, j \in [2^n]$, if $\mathfrak{t}(i, j) = \mathfrak{d}$ and $\mathfrak{t}(i, j+1) = \mathfrak{d}'$, then $(\mathfrak{d}, \mathfrak{d}') \in V$,
- initial condition: for every $i \in [n]$, $\mathfrak{t}(i, 0) = \mathfrak{u}_i$.

A tiling $\mathfrak{t}$ is equivalent to the set $X_{\mathfrak{t}} = \{(i, j, \mathfrak{t}(i, j)) : i, j \in [2^n]\}$ and therefore we explain below how to encode such a set by a heap. Given a heap $\mathfrak{h}$, we say that a location $\mathfrak{l}$ has a backward path of length exactly $i \geq 1$ iff there are locations $\mathfrak{l}_0, \ldots, \mathfrak{l}_i$ such that $\mathfrak{l}_i = \mathfrak{l}$, $\mathfrak{l}_0$ has no predecessor and for every $j \in [1, i]$, $\mathfrak{h}(\mathfrak{l}_{j-1}) = \mathfrak{l}_j$. A triple $(i, j, \mathfrak{d})$ in $X_{\mathfrak{t}}$ is encoded by a connected component $\mathcal{C}$ satisfying the following constraints.

- There is a location $\mathfrak{l}$ in $\mathcal{C}$ without successor and with at least one predecessor, which is identified by the following formula $\mathtt{tile}(\mathtt{x}) \stackrel{\text{def}}{=} (\exists \mathtt{y}.\ (\mathtt{y} \hookrightarrow \mathtt{x})) \wedge \forall \mathtt{y}.\ \neg(\mathtt{x} \hookrightarrow \mathtt{y})$.
- For every $k \in [1, s]$, $\mathfrak{d} = \mathfrak{d}_k$ iff $\mathfrak{l}$ has a backward path of length exactly $2n + k$. It is not hard to construct an SPSL2($\hookrightarrow$) formula $\mathtt{d}_k$ to describe this property.
- For every $k \in [1, n]$, the kth bit in the binary representation of i (here the left-most bit is the first bit) is equal to 1 iff $\mathfrak{l}$ has a backward path of length exactly k. Similarly, this property can be described by an SPSL2($\hookrightarrow$) formula $\mathtt{h}_k$.
- For every $k \in [1, n]$, the kth bit in the binary representation of j is equal to 1 iff $\mathfrak{l}$ has a backward path of length exactly $n + k$. Similarly, this property can be described by an SPSL2($\hookrightarrow$) formula $\mathtt{v}_k$.

Then an SPSL2($\hookrightarrow$) formula ϕ can be constructed so that ϕ is satisfiable iff the tiling problem instance has a solution.

In the following, we first define the formulae in SPSL2($\hookrightarrow$) with a unique free variable, say $\mathtt{h}_1(\mathtt{x}), \ldots, \mathtt{h}_n(\mathtt{x}), \mathtt{v}_1(\mathtt{x}), \ldots, \mathtt{v}_n(\mathtt{x}), \mathtt{d}_1(\mathtt{x}), \ldots, \mathtt{d}_s(\mathtt{x})$, then ϕ. By swapping $\mathtt{x}$ and $\mathtt{y}$, we also get the formulae $\mathtt{h}_1(\mathtt{y}), \ldots, \mathtt{h}_n(\mathtt{y}), \mathtt{v}_1(\mathtt{y}), \ldots, \mathtt{v}_n(\mathtt{y}), \mathtt{d}_1(\mathtt{y}), \ldots, \mathtt{d}_s(\mathtt{y})$.

Let us start by defining some auxiliary formulae.

1. $\psi_1(\mathtt{x}) \stackrel{\text{def}}{=} \exists\, \mathtt{y}\ (\mathtt{y} \hookrightarrow \mathtt{x} \wedge \forall\, \mathtt{x}\ (\neg \mathtt{x} \hookrightarrow \mathtt{y}))$. The formula $\psi_1(\mathtt{x})$ simply states that $\mathtt{x}$ has a predecessor with no predecessors (but $\mathtt{x}$ may have other arbitrary predecessors).
2. $\psi_i(\mathtt{x}) \stackrel{\text{def}}{=} \exists\, \mathtt{y}\ (\mathtt{y} \hookrightarrow \mathtt{x} \wedge \wedge \psi_{i-1}(\mathtt{y}))$ for every $i \geq 2$. Assuming that $\mathtt{x}$ is interpreted by $\mathfrak{l}$, the formula $\psi_i(\mathtt{x})$ simply states that $\mathfrak{l}$ has a backward path of length exactly i.

Now let us define the formulae $\mathtt{h}_i(\mathtt{x})$, $\mathtt{v}_i(\mathtt{x})$ and $\mathtt{d}_i(\mathtt{x})$.

- For every $i \in [1, n]$, $\mathtt{h}_i(\mathtt{x}) \stackrel{\text{def}}{=} \psi_i(\mathtt{x})$.
- For every $i \in [1, n]$, $\mathtt{v}_i(\mathtt{x}) \stackrel{\text{def}}{=} \psi_{n+i}(\mathtt{x})$.
- For every $i \in [1, s]$, $\mathtt{d}_i(\mathtt{x}) \stackrel{\text{def}}{=} \psi_{2n+i}(\mathtt{x})$.

The three types of formulae are therefore only distinguished by path lengths.

Let ϕ be defined as the conjunction of the following formulae.

- Two locations encoding a position in the arena satisfy exactly the same formulae among $\mathtt{h}_1(\mathtt{x}), \ldots, \mathtt{h}_n(\mathtt{x}), \mathtt{v}_1(\mathtt{x}), \ldots, \mathtt{v}_n(\mathtt{x})$ are necessarily identical:

$$\phi_1 \stackrel{\text{def}}{=} \forall\, \mathtt{x}.\forall \mathtt{y}.\ \left(\begin{bmatrix} \mathtt{tile}(\mathtt{x}) \wedge \mathtt{tile}(\mathtt{y}) \wedge \\ \bigwedge_{i \in [1,n]} ((\mathtt{h}_i(\mathtt{x}) \leftrightarrow \mathtt{h}_i(\mathtt{y}))\ \wedge (\mathtt{v}_i(\mathtt{x}) \leftrightarrow \mathtt{v}_i(\mathtt{y}))) \end{bmatrix} \rightarrow \mathtt{x} = \mathtt{y} \right).$$

- There is a location that corresponds to the bottom left position:

$$\phi_2 \stackrel{\text{def}}{=} \exists\, \mathtt{x}.\ \left(\mathtt{tile}(\mathtt{x}) \wedge \bigwedge_{i \in [1,n]} (\neg \mathtt{h}_i(\mathtt{x}) \wedge \neg \mathtt{v}_i(\mathtt{x})) \right).$$

- Each location encoding a position in the arena satisfies a unique tile:

$$\phi_3 \stackrel{\text{def}}{=} \forall\, \mathtt{x}.\ \left(\mathtt{tile}(\mathtt{x}) \rightarrow \bigvee_{i \in [1,s]} \left(\mathtt{d}_i(\mathtt{x}) \wedge \bigwedge_{j \in [1,s] \setminus \{i\}} \neg \mathtt{d}_j(\mathtt{x}) \right) \right).$$

– Horizontal constraint for two consecutive positions within the same row:

$$\phi_4 \stackrel{\text{def}}{=} \forall\, \mathtt{x}. \left(\begin{array}{l} \left[\mathtt{tile}(\mathtt{x}) \wedge \bigwedge\limits_{i\in[1,n]} \left(\neg \mathtt{h}_i(\mathtt{x}) \wedge \bigwedge\limits_{i<j\leq n} \mathtt{h}_j(\mathtt{x})\right)\right] \rightarrow \\ \exists\, \mathtt{y}. \left[\begin{array}{l} \mathtt{tile}(\mathtt{y}) \wedge \bigvee\limits_{(\mathfrak{d}_l,\mathfrak{d}_m)\in H} (\mathtt{d}_l(\mathtt{x}) \wedge \mathtt{d}_m(\mathtt{y})) \wedge \\ \bigwedge\limits_{j\in[1,n]} (\mathtt{v}_j(\mathtt{x}) \leftrightarrow \mathtt{v}_j(\mathtt{y})) \wedge \bigwedge\limits_{1\leq j<i} (\mathtt{h}_j(\mathtt{x}) \leftrightarrow \mathtt{h}_j(\mathtt{y})) \wedge \\ \mathtt{h}_i(\mathtt{y}) \wedge \bigwedge\limits_{i<j\leq n} \neg \mathtt{h}_j(\mathtt{y}) \end{array}\right] \end{array} \right).$$

– The end of a row is immediately followed by the beginning of the next row, if any:

$$\phi_5 \stackrel{\text{def}}{=} \forall\, \mathtt{x}. \bigwedge_{i\in[1,n]} \left(\begin{array}{l} \left[\mathtt{tile}(\mathtt{x}) \wedge \left(\bigwedge\limits_{j\in[1,n]} \mathtt{h}_j(\mathtt{x})\right) \wedge \neg \mathtt{v}_i(\mathtt{x}) \wedge \bigwedge\limits_{i<j\leq n} \mathtt{v}_j(\mathtt{x})\right] \rightarrow \\ \exists\, \mathtt{y}. \left[\begin{array}{l} \mathtt{tile}(\mathtt{y}) \wedge \left(\bigwedge\limits_{j\in[1,n]} \neg \mathtt{h}_j(\mathtt{y})\right) \wedge \\ \left(\bigwedge\limits_{1\leq j<i} \mathtt{v}_j(\mathtt{x}) \leftrightarrow \mathtt{v}_j(\mathtt{y})\right) \wedge \mathtt{v}_i(\mathtt{y}) \wedge \bigwedge\limits_{i<j\leq n} \neg \mathtt{v}_j(\mathtt{y}) \end{array}\right] \end{array} \right).$$

Satisfaction of the formulae ϕ_2, ϕ_3, ϕ_4 and ϕ_5 guarantees that all the positions in the arena are encoded. Unicity of such an encoding is a consequence of the satisfaction of ϕ_1.

– Vertical constraint between two consecutive vertical positions:

$$\phi_6 \stackrel{\text{def}}{=} \forall\, \mathtt{x}. \bigwedge_{i\in[1,n]} \left(\begin{array}{l} \left[\mathtt{tile}(\mathtt{x}) \wedge (\neg \mathtt{v}_i(\mathtt{x})) \wedge \bigwedge\limits_{i<j\leq n} \mathtt{v}_j(\mathtt{x})\right] \rightarrow \\ \exists\, \mathtt{y}. \left[\begin{array}{l} \mathtt{tile}(\mathtt{y}) \wedge \bigvee\limits_{(\mathfrak{d}_l,\mathfrak{d}_m)\in V} (\mathtt{d}_l(\mathtt{x}) \wedge \mathtt{d}_m(\mathtt{y})) \wedge \\ \bigwedge\limits_{j\in[1,n]} (\mathtt{h}_j(\mathtt{x}) \leftrightarrow \mathtt{h}_j(\mathtt{y})) \wedge \bigwedge\limits_{1\leq j<i} (\mathtt{v}_j(\mathtt{x}) \leftrightarrow \mathtt{v}_j(\mathtt{y})) \wedge \\ \mathtt{v}_i(\mathtt{y}) \wedge \bigwedge\limits_{i<j\leq n} (\neg \mathtt{v}_j(\mathtt{y})) \end{array}\right] \end{array} \right).$$

– Suppose $\mathfrak{u}_0 \ldots \mathfrak{u}_{n-1} = \mathfrak{d}_{j_0} \ldots \mathfrak{d}_{j_{n-1}}$, then the initial condition is specified as follows:

$$\phi_7 \stackrel{\text{def}}{=} \bigwedge_{i\in[0,n-1]} \exists\, \mathtt{x}. \left(\begin{array}{l} \mathtt{tile}(\mathtt{x}) \wedge \bigwedge\limits_{j'\in[1,n]} (\neg \mathtt{v}_{j'}(\mathtt{x})) \wedge \mathtt{d}_{j_i}(\mathtt{x}) \wedge \\ \bigwedge\limits_{j'\in[1,n],\ j'\text{-th bit of } i \text{ is } 1} \mathtt{h}_{j'}(\mathtt{x}) \wedge \\ \bigwedge\limits_{j'\in[1,n],\ j'\text{-th bit of } i \text{ is } 0} \neg \mathtt{h}_{j'}(\mathtt{x}) \end{array} \right).$$

Note that for readability, we choose to write the formulae $\phi_1, \ldots, \phi_7$ above not in negation normal form as required in the definition of the logic SPSL2($\hookrightarrow$) (cf. Section 2). Nevertheless, since the logic SPSL2($\hookrightarrow$) is closed under negations, those formulae can be easily rewritten into the required form.

Lemma 1. *The formula ϕ is satisfiable iff the tiling problem instance has a solution.*

Proof. First, suppose the tiling problem instance has a solution $\mathfrak{t}$. Then we construct a heap $\mathfrak{h}$ from $\mathfrak{t}$ such that

- $\mathfrak{t}$ comprises $2^n \times 2^n$ connected components, one for each $(i, j) \in [2^n] \times [2^n]$ (denoted by $\mathcal{C}_{i,j}$),
- each $\mathcal{C}_{i,j}$ is a tree, which comprises the following backward paths from the root $\mathfrak{l}$,
  - for each $k \in [1, n]$, $\mathfrak{l}$ has a backward path of length exactly k (resp. $n + k$) iff the kth bit of the binary representation of i (resp. j) is equal to 1,
  - let $\mathfrak{t}(i, j) = \mathfrak{d}_k$, then $\mathfrak{l}$ has a backward path of length exactly $2n + k$.

Since $\mathfrak{t}$ satisfies the horizontal and vertical constraint, as well as the initial condition, $\mathfrak{h} \models \phi_4 \wedge \phi_5 \wedge \phi_6 \wedge \phi_7$. Moreover, from the construction of $\mathfrak{h}$, we know that $\mathfrak{h} \models \phi_1 \wedge \phi_2 \wedge \phi_3$. Therefore, we conclude that $\mathfrak{h} \models \phi$.

Let us establish the other direction. Suppose that ϕ is satisfiable. Then there is a heap $\mathfrak{h}$ satisfying ϕ. From the fact that $\mathfrak{h} \models \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4 \wedge \phi_5$, we know that for each $(i, j) \in [2^n] \times [2^n]$, there is exactly one connected component $\mathcal{C}_{i,j}$ such that for each $k \in [1, n]$, the root of $\mathcal{C}_{i,j}$ has a backward path of length exactly k (resp. $n + k$) iff the kth bit of the binary representation of i (resp. j) is equal to 1. We construct a tiling $\mathfrak{t} : [2^n] \times [2^n] \to D$ as follows: For each $(i, j) \in [2^n] \times [2^n]$, suppose $k \in [1, s]$ satisfies that the root of $\mathcal{C}_{i,j}$ has a backward path of length exactly $2n + k$ (such a k exists since $\mathfrak{h} \models \phi_3$), let $\mathfrak{t}(i, j) = \mathfrak{d}_k$. Because $\mathfrak{h} \models \phi_4 \wedge \phi_6 \wedge \phi_7$, we know that $\mathfrak{t}$ satisfies the horizontal and vertical constraint as well as the initial condition. Therefore, $\mathfrak{t}$ is a solution of the tiling problem instance. □

3.2 Upper bound

The upper bound is obtained by a linear time reduction to the finite satisfiability problem of first-order logic with two-variables and counting quantifiers (denoted by $\mathcal{C}^2$), which can be decided in NEXPTIME [18, 19]. Before presenting the reduction, we first recall the definition of $\mathcal{C}^2$. A *purely relational vocabulary* $\mathcal{V}$ comprises relational symbols, but no function symbols, nor constants. The logic $\mathcal{C}^2$ over a purely relational vocabulary $\mathcal{V}$ is defined by the following rules,

$$\begin{aligned} &\mathtt{v} ::= \mathtt{x} \mid \mathtt{y}, \\ &\varphi ::= \mathtt{v} = \mathtt{v} \mid R(\bar{\mathtt{v}}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists^{\odot C}\mathtt{v}.\ \varphi, \end{aligned}$$

where $R \in \mathcal{V}$ is of arity k, $\bar{\mathtt{v}} \in \{\mathtt{x}, \mathtt{y}\}^k$, $\odot \in \{<, >, \leq, \geq, =\}$, and $C \in \mathbb{N}$ is a constant. We assume that all the constants in $\mathcal{C}^2$ are encoded in binary. For a formula $\varphi \in \mathcal{C}^2$, let $|\varphi|$ denote the number of symbols occurring in φ. The formulae in $\mathcal{C}^2$ are interpreted on a triple $(A, \mathcal{I}, \mathfrak{m})$, where A is a *domain*, $\mathcal{I}$ is called an *interpretation function*, which assigns each k-ary relation symbol $R \in \mathcal{V}$ a subset of A^k, and $\mathfrak{m}$ is an *assignment* that maps x and y to A. The semantics of $\mathcal{C}^2$ formulae are defined by a relation $(A, \mathcal{I}) \models_{\mathfrak{m}} \varphi$. The semantics of the atomic formulae, the Boolean combination, the quantifiers are standard. For the $\mathcal{C}^2$ formulae $\varphi = \exists^{=C}\mathtt{x}.\ \varphi_1$, $(A, \mathcal{I}) \models_{\mathfrak{m}} \varphi$ iff there are exactly C elements of A, say $a_1, \ldots, a_C$, such that for each $i \in [C]$, $(A, \mathcal{I}) \models_{\mathfrak{m}[\mathtt{x} \mapsto a_i]} \varphi_1$. The semantics of the formulae $\exists^{=C}\mathtt{y}.\ \varphi_1$, $\exists^{>C}\mathtt{x}.\ \varphi_1$, $\exists^{<C}\mathtt{x}.\ \varphi_1$, etc. can be defined similarly. Let φ be a $\mathcal{C}^2$ formula. If $(A, \mathcal{I}) \models_{\mathfrak{m}} \varphi$, then $(A, \mathcal{I}, \mathfrak{m})$ is called a *model* of φ. Then φ is *satisfiable* if φ has a model, and φ is *finitely satisfiable* if φ has a model $(A, \mathcal{I}, \mathfrak{m})$ such that A is finite.

Lemma 2 ([18,19]). *The satisfiability and finite satisfiability problem of $\mathcal{C}^2$ are* NEXPTIME-*complete.*

We are ready to present the reduction.

For each $\mathfrak{f} \in \mathbb{F}$, introduce a fresh binary relation symbol $R_{\mathfrak{f}}$. Let $\mathcal{V} = \mathcal{P} \cup \{R_{\mathfrak{f}} \mid \mathfrak{f} \in \mathbb{F}\}$. In the following, for each formula ϕ in SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}\right)$, we construct a $\mathcal{C}^2$ formula $\mathfrak{trs}(\phi) = \phi_{\mathtt{fun}} \wedge \exists \mathtt{x}.\exists \mathtt{y}.\ (\neg \mathtt{x} = \mathtt{y} \wedge \phi_{\mathtt{rel}}(\mathtt{x}) \wedge \phi_{\mathtt{rel}}(\mathtt{y})) \wedge \phi'$ over the vocabulary $\mathcal{V}$, where

- $\phi_{\mathtt{fun}} = \bigwedge\limits_{\mathfrak{f}\in\mathbb{F}} \forall \mathtt{x}.\exists^{\leq 1}\mathtt{y}.\ R_{\mathfrak{f}}(\mathtt{x},\mathtt{y})$ expresses that each relation symbol $R_{\mathfrak{f}}$ is the image of a partial function,
- $\phi_{\mathtt{rel}}(\mathtt{v}) = \left(\bigwedge\limits_{P\in\mathcal{P}} \neg P(\mathtt{v})\right) \wedge \forall \mathtt{v}'.\ \bigwedge\limits_{\mathfrak{f}\in\mathbb{F}} (\neg R_{\mathfrak{f}}(\mathtt{v},\mathtt{v}') \wedge \neg R_{\mathfrak{f}}(\mathtt{v}',\mathtt{v}))$, where $\mathtt{v} = \mathtt{x}$ and $\mathtt{v}' = \mathtt{y}$, or vice versa, expresses that there is an element represented by $\mathtt{v}$ which does not occur in any tuple from the union of the relations $P \in \mathcal{P}$ and $R_{\mathfrak{f}}$ for $\mathfrak{f} \in \mathbb{F}$,
- ϕ' is obtained from ϕ by replacing each atomic formula of the form $\mathtt{v}_1 \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{v}_2$ with $R_{\mathfrak{f}}(\mathtt{v}_1,\mathtt{v}_2)$.

The formula $\exists \mathtt{x}.\exists \mathtt{y}.\ (\neg \mathtt{x} = \mathtt{y} \wedge \phi_{\mathtt{rel}}(\mathtt{x}) \wedge \phi_{\mathtt{rel}}(\mathtt{y}))$ expresses that there are two distinct elements satisfying the formula $\phi_{\mathtt{rel}}$.

The correctness of the reduction is guaranteed by the following result.

Proposition 1. *For each SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}\right)$ formula ϕ, ϕ is satisfiable iff $\mathfrak{trs}(\phi)$ is finitely satisfiable.*

Proof. Suppose that ϕ is an SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}\right)$ formula.

"Only if" direction: Suppose that ϕ is satisfiable. Then there are a labeled heap $(\mathfrak{h}, \mathfrak{I})$ and $\mathfrak{m}$ such that $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}} \phi$.

We construct a finite set $A = \mathtt{loc}(\mathfrak{h}) \cup (\bigcup_{P\in\mathcal{P}} \mathfrak{I}(P)) \cup \{\mathfrak{l}_1, \mathfrak{l}_2\}$, where

- if $\mathfrak{m}(\mathtt{x}), \mathfrak{m}(\mathtt{y}) \not\in \mathtt{loc}(\mathfrak{h}) \cup (\bigcup_{P\in\mathcal{P}} \mathfrak{I}(P))$ such that $\mathfrak{m}(\mathtt{x}) \neq \mathfrak{m}(\mathtt{y})$, then let $\mathfrak{l}_1 = \mathfrak{m}(\mathtt{x})$ and $\mathfrak{l}_2 = \mathfrak{m}(\mathtt{y})$,
- if $\mathfrak{m}(\mathtt{x}), \mathfrak{m}(\mathtt{y}) \not\in \mathtt{loc}(\mathfrak{h}) \cup (\bigcup_{P\in\mathcal{P}} \mathfrak{I}(P))$ such that $\mathfrak{m}(\mathtt{x}) = \mathfrak{m}(\mathtt{y})$, then let $\mathfrak{l}_1 = \mathfrak{m}(\mathtt{x})$ and $\mathfrak{l}_2$ be a location in $\mathbb{N} \setminus \left(\mathtt{loc}(\mathfrak{h}) \cup (\bigcup_{P\in\mathcal{P}} \mathfrak{I}(P)) \cup \{\mathfrak{m}(\mathtt{x})\}\right)$,
- if $\mathfrak{m}(\mathtt{v}) \not\in \mathtt{loc}(\mathfrak{h}) \cup (\bigcup_{P\in\mathcal{P}} \mathfrak{I}(P))$ and $\mathfrak{m}(\mathtt{v}') \in \mathtt{loc}(\mathfrak{h}) \cup (\bigcup_{P\in\mathcal{P}} \mathfrak{I}(P))$, then let $\mathfrak{l}_1 = \mathfrak{m}(\mathtt{v})$ and $\mathfrak{l}_2$ be a location in $\mathbb{N} \setminus \left(\mathtt{loc}(\mathfrak{h}) \cup (\bigcup_{P\in\mathcal{P}} \mathfrak{I}(P)) \cup \{\mathfrak{m}(\mathtt{v})\}\right)$, where $\mathtt{v} = \mathtt{x}$ and $\mathtt{v}' = \mathtt{y}$, or vice versa,
- if $\mathfrak{m}(\mathtt{x}) \in \mathtt{loc}(\mathfrak{h}) \cup (\bigcup_{P\in\mathcal{P}} \mathfrak{I}(P))$ and $\mathfrak{m}(\mathtt{y}) \in \mathtt{loc}(\mathfrak{h}) \cup (\bigcup_{P\in\mathcal{P}} \mathfrak{I}(P))$, then let $\mathfrak{l}_1$ and $\mathfrak{l}_2$ be two distinct locations in $\mathbb{N} \setminus \left(\mathtt{loc}(\mathfrak{h}) \cup (\bigcup_{P\in\mathcal{P}} \mathfrak{I}(P))\right)$.

Consider the triple $(A, \mathcal{I}, \mathfrak{m})$, where $\mathcal{I}(P) = \mathfrak{I}(P)$ for each $P \in \mathcal{P}$, and $\mathcal{I}(R_{\mathfrak{f}}) = \{(\mathfrak{l}, \mathfrak{l}') \in \mathtt{loc}(\mathfrak{h}) \times \mathtt{loc}(\mathfrak{h}) \mid \mathfrak{h}_{\mathfrak{f}}(\mathfrak{l}) = \mathfrak{l}'\}$. We claim that $(A, \mathcal{I}) \models_{\mathfrak{m}} \mathfrak{trs}(\phi)$. Since evidently $(A, \mathcal{I}) \models_{\mathfrak{m}} \psi_{\mathtt{fun}} \wedge \exists \mathtt{x}.\exists \mathtt{y}.\ (\neg \mathtt{x} = \mathtt{y} \wedge \phi_{\mathtt{rel}}(\mathtt{x}) \wedge \phi_{\mathtt{rel}}(\mathtt{y}))$, is sufficient to show that $(A, \mathcal{I}) \models_{\mathfrak{m}} \phi'$. In the following, we show $(A, \mathcal{I}) \models_{\mathfrak{m}} \phi'$ by proving that for each assignment $\mathfrak{m}'$ such that $\mathtt{ran}(\mathfrak{m}') \subseteq A$ and each subformula ϕ_1 of ϕ, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'} \phi_1$

iff $(A, \mathcal{I}) \models_{\mathfrak{m}'} \phi'_1$, where ϕ'_1 is obtained from ϕ_1 by replacing each atomic formula of the form $\mathtt{v}_1 \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{v}_2$ with $R_{\mathfrak{f}}(\mathtt{v}_1, \mathtt{v}_2)$. We show this fact by induction on the syntax of formulae.

- The cases $\phi_1 \overset{\text{def}}{=} \mathtt{v}_1 = \mathtt{v}_2$ and $\phi_1 \overset{\text{def}}{=} \neg \mathtt{v}_1 = \mathtt{v}_2$ are trivial.
- Case $\phi_1 \overset{\text{def}}{=} \mathtt{v}_1 \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{v}_2$: Since $\mathtt{ran}(\mathfrak{m}') \subseteq A$, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'} \mathtt{v}_1 \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{v}_2$ iff $\mathfrak{h}_{\mathfrak{f}}(\mathfrak{m}'(\mathtt{v}_1)) = \mathfrak{m}'(\mathtt{v}_2)$ iff $(\mathfrak{m}'(\mathtt{v}_1), \mathfrak{m}'(\mathtt{v}_2)) \in \mathcal{I}(R_{\mathfrak{f}})$. Similarly for $\phi_1 \overset{\text{def}}{=} \neg \mathtt{v}_1 \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{v}_2$.
- Case $\phi_1 \overset{\text{def}}{=} P(\mathtt{v})$: $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'} P(\mathtt{v})$ iff $\mathfrak{m}'(\mathtt{v}) \in \mathfrak{I}(P)$ iff $\mathfrak{m}'(\mathtt{v}) \in \mathcal{I}(P)$ iff $(A, \mathcal{I}) \models_{\mathfrak{m}'} P(\mathtt{v})$. Similarly for $\phi_1 \overset{\text{def}}{=} \neg P(\mathtt{v})$.
- Case $\phi_1 \overset{\text{def}}{=} \phi_2 \wedge \phi_3$ or $\phi_1 \overset{\text{def}}{=} \phi_2 \vee \phi_3$: The arguments are standard.
- Case $\phi_1 \overset{\text{def}}{=} \exists \mathtt{x}.\ \phi_2$: Our goal is to show $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'} \phi_1$ iff $(A, \mathcal{I}) \models_{\mathfrak{m}'} \phi'_1$. Since the "if" direction is easy, we focus on the "only if" direction below. Suppose $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'} \exists \mathtt{x}.\ \phi_2$. Then there is $\mathfrak{l}' \in \mathbb{N}$ such that $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi_2$.
  - If $\mathfrak{l}' \in A$, then according to the induction hypothesis, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi_2$ iff $(A, \mathcal{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi'_2$.
  - If $\mathfrak{l}' \notin A$, then $\mathfrak{l}' \neq \mathfrak{m}'(\mathtt{y})$ since $\mathfrak{m}'(\mathtt{y}) \in A$. Evidently, $\mathfrak{l}_1 \neq \mathfrak{m}'(\mathtt{y})$ or $\mathfrak{l}_2 \neq \mathfrak{m}'(\mathtt{y})$. Without loss of generality, we assume that $\mathfrak{l}_1 \neq \mathfrak{m}'(\mathtt{y})$. Because neither $\mathfrak{l}_1$ nor $\mathfrak{l}'$ belongs to $\mathtt{loc}(\mathfrak{h}) \cup \bigcup\limits_{P \in \mathcal{P}} \mathfrak{I}(P)$, we deduce that $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi_2$ iff $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}_1]} \phi_2$. From the induction hypothesis, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}_1]} \phi_2$ iff $(A, \mathcal{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}_1]} \phi'_2$. Therefore, $(A, \mathcal{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}_1]} \phi'_2$ and $(A, \mathcal{I}) \models_{\mathfrak{m}'} \exists \mathtt{x}.\ \phi'_2$.
- Case $\phi_1 \overset{\text{def}}{=} \exists \mathtt{y}.\ \phi_2$: Similarly to the previous case.
- Case $\phi_1 \overset{\text{def}}{=} \forall \mathtt{x}.\ \phi_2$: Suppose that $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'} \forall \mathtt{x}.\ \phi_2$. Then for each $\mathfrak{l}' \in \mathbb{N}$, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi_2$. For each $\mathfrak{l}' \in A$, according to the induction hypothesis, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi_2$ iff $(A, \mathcal{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi'_2$. Therefore, for each $\mathfrak{l}' \in A$, we have $(A, \mathcal{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi'_2$. We conclude that $(A, \mathcal{I}) \models_{\mathfrak{m}'} \forall \mathtt{x}.\ \phi'_2 = \phi'_1$. On the other hand, suppose that $(A, \mathcal{I}) \models_{\mathfrak{m}'} \phi'_1 = \forall \mathtt{x}.\ \phi'_2$. Then for each $\mathfrak{l}' \in A$, $(A, \mathcal{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi'_2$. By the induction hypothesis, for each $\mathfrak{l}' \in A$, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi_2$ iff $(A, \mathcal{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi'_2$. Therefore, for each $\mathfrak{l}' \in A$, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi_2$. Now suppose $\mathfrak{l}' \in \mathbb{N} \setminus A$. Without loss of generality, suppose that $\mathfrak{l}_1 \neq \mathfrak{m}'(\mathtt{y})$. Because neither $\mathfrak{l}_1$ nor $\mathfrak{l}'$ belongs to $\mathtt{loc}(\mathfrak{h}) \cup \bigcup\limits_{P \in \mathcal{P}} \mathfrak{I}(P)$, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi_2$ iff $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}_1]} \phi_2$. From this, we deduce that for each $\mathfrak{l}' \in \mathbb{N} \setminus A$, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi_2$. Thus for each $\mathfrak{l}' \in \mathbb{N}$, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \phi_2$, that is, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'[\mathtt{x} \mapsto \mathfrak{l}']} \forall \mathtt{x}.\ \phi_2 = \phi_1$.
- Case $\phi_1 \overset{\text{def}}{=} \forall \mathtt{y}.\ \phi_2$: Similarly to the previous case.

"If" direction: Suppose that $\mathfrak{trs}(\phi)$ is finitely satisfiable. Then there is a model $(A, \mathcal{I}, \mathfrak{m})$ of $\mathfrak{trs}(\phi)$ such that A is finite. Without loss of generality, we assume that A is a subset of $\mathbb{N}$.

We construct $(\mathfrak{h}, \mathfrak{I})$ such that $\mathfrak{h}_{\mathfrak{f}}(\mathfrak{l}) = \mathfrak{l}'$ iff $(\mathfrak{l}, \mathfrak{l}') \in \mathcal{I}(R_{\mathfrak{f}})$ for each $\mathfrak{f} \in \mathbb{F}$, and $\mathfrak{l} \in \mathfrak{I}(P)$ iff $\mathfrak{l} \in \mathcal{I}(P)$ for each $P \in \mathcal{P}$. Since $(A, \mathcal{I}) \models_{\mathfrak{m}} \psi_{\mathtt{fun}} \wedge \exists \mathtt{x}. \exists \mathtt{y}.\ (\neg \mathtt{x} = \mathtt{y} \wedge \phi_{\mathtt{rel}}(\mathtt{x}) \wedge \phi_{\mathtt{rel}}(\mathtt{y}))$, we know that each $\mathfrak{h}_{\mathfrak{f}}$ for $\mathfrak{f} \in \mathbb{F}$ is a partial function, and there are two distinct locations $\mathfrak{l}_1, \mathfrak{l}_2 \in A \setminus \left(\mathtt{loc}(\mathfrak{h}) \cup \bigcup\limits_{P \in \mathcal{P}} \mathfrak{I}(P) \right)$. Similarly to the arguments in the "Only if" direction, we can show that for each assignment $\mathfrak{m}'$ such that $\mathtt{ran}(\mathfrak{m}') \subseteq A$ and each subformula ϕ_1 of ϕ, $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}'} \phi_1$ iff $(A, \mathcal{I}) \models_{\mathfrak{m}'} \phi'_1$. From this, we deduce that $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}} \phi$ iff $(A, \mathcal{I}) \models_{\mathfrak{m}} \phi'$. We then conclude that $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}} \phi$. □

4 ESPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f} \in \mathbb{F}}, *, \neg\!\!\!*\right)$

In this section, we present the main result of this paper.

Theorem 2. *The satisfiability problem of ESPSL2*$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f} \in \mathbb{F}}, *, \neg\!\!\!*\right)$ *is* NEXPTIME*-complete.*

The rest of this section is devoted to the proof of Theorem 2. The basic idea of the proof is to reduce in polynomial time the satisfiability of ESPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f} \in \mathbb{F}}, *, \neg\!\!\!*\right)$ formulae to that of SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f} \in \mathbb{F}}, \mathcal{P}'\right)$ formulae (for some $\mathcal{P}'$), which is NEXPTIME-complete (cf. Theorem 1). The main idea of the reduction is as follows: For a heap $\mathfrak{h}$, an assignment $\mathfrak{m}$, and an ESPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f} \in \mathbb{F}}, *, \neg\!\!\!*\right)$ formula ψ such that $\mathfrak{h} \models_{\mathfrak{m}} \psi$, in order to witness the fact $\mathfrak{h} \models_{\mathfrak{m}} \psi$, some other heaps should be added to $\mathfrak{h}$. Nevertheless, these additional heaps may conflict with each other. For instance, in Example 1, two heaps corresponding to ψ'_1 and ψ'_3 should be added, but these two heaps conflict with each other, since ψ'_1 says that $\mathtt{y}$ is the $\mathfrak{f}$-successor of $\mathtt{x}$, while ψ'_3 says that the $\mathfrak{f}$-successor of $\mathtt{x}$ exists but is different from $\mathtt{y}$. In the reduction from the satisfiability of ψ to that of a SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f} \in \mathbb{F}}, \mathcal{P}'\right)$ formula below, the fields in the subformulae of ψ conflicting with each other are renamed, so that after the renaming, these subformulae refer to different fields. In addition, to guarantee the correctness of the reduction, some necessary constraints should be added to these fields as well as the unary predicates from $\mathcal{P}'$.

We introduce a concept of syntax trees which will be used in the reduction. Let ψ be an ESPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f} \in \mathbb{F}}, *, \neg\!\!\!*\right)$ formula. A *syntax tree* $\mathcal{T}_\psi = (T, E, L)$ can be constructed inductively as follows, where T is a set of nodes, E is the child-parent relation, and $L : T \to \mathtt{Sub}(\psi)$ is a labeling function.

- Case $\psi \overset{\text{def}}{=} \phi$, where ϕ is a SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f} \in \mathbb{F}}\right)$ formula: Then $\mathcal{T}_\psi = (\{t\}, \emptyset, L)$ such that $L(t) = \psi$,
- Case $\psi \overset{\text{def}}{=} \psi_1 \odot \psi_2$ (where $\odot = \vee, \wedge, *, \neg\!\!\!*$): Suppose two syntax trees $\mathcal{T}_{\psi_1} = (T_1, E_1, L_1)$ and $\mathcal{T}_{\psi_2} = (T_2, E_2, L_2)$ have been constructed for ψ_1 and ψ_2 respectively. Without loss of generality, suppose $T_1 \cap T_2 = \emptyset$ and the roots of $\mathcal{T}_{\psi_1}$ and $\mathcal{T}_{\psi_2}$ are t_1 and t_2 respectively. Then $\mathcal{T}_\psi = (T_1 \cup T_2 \cup \{t\}, E_1 \cup E_2 \cup \{(t_1, l, t), (t_2, r, t)\}, L_1 \cup L_2 \cup \{t \mapsto \psi\})$, where t is a new node not in T_1 or T_2, and the label sl, r denote the left and right child respectively.
- Case $\psi \overset{\text{def}}{=} \exists \mathtt{v}.\ \psi_1$: Suppose a syntax tree $\mathcal{T}_{\psi_1} = (T_1, E_1, L_1)$ has been constructed for ψ_1. Then $\mathcal{T}_\psi = (T_1 \cup \{t\}, E_1 \cup \{(t_1, l, t)\}, L_1 \cup \{t \mapsto \psi\})$, where t is a new node not in T_1.

Let ψ be an ESPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, *, \mathrel{-\!\!\circledast}\right)$ formula and $\mathcal{T}_\psi = (T, E, L)$. For each $t \in T$, we introduce a fresh unary predicate P'_t. In addition, for each node $t \in T$ and $\mathfrak{f} \in \mathbb{F}$, introduce a fresh field $\mathfrak{f}'_t$. Let $\mathcal{P}'$ be the set of freshly introduced unary predicates, and $\mathbb{F}'$ be the set of freshly introduced fields. Our goal is to use $\mathcal{T}_\psi$ to construct an SPSL2$\left((\overset{\mathfrak{f}'_t}{\hookrightarrow})_{\mathfrak{f}'_t\in\mathbb{F}'}, \mathcal{P}'\right)$ formula $\mathfrak{trs}(\psi)$ so that ψ is satisfiable iff $\mathfrak{trs}(\psi)$ is satisfiable. Toward this purpose, for each node $t \in T$, we construct an SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}, \mathcal{P}'\right)$ formula ϕ_t. Then let $\mathfrak{trs}(\psi) \overset{\text{def}}{=} \bigwedge\limits_{t\in T}\left(\forall \mathtt{x}.\ P'_t(\mathtt{x}) \leftrightarrow \bigvee\limits_{\mathfrak{f}\in\mathbb{F}} \exists \mathtt{y}.\ \mathtt{x} \overset{\mathfrak{f}'_t}{\hookrightarrow} \mathtt{y}\right) \wedge \phi_{t_0}$, where t_0 is the root of $\mathcal{T}_\psi$. The formulae ϕ_t for $t \in T$ are computed inductively as follows.

- If $L(t) = \phi$ for some SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f}\in\mathbb{F}}\right)$ formula ϕ, then $\phi_t \overset{\text{def}}{=} \phi'$, where ϕ' is obtained from ϕ by replacing each occurrence of $\mathfrak{f} \in \mathbb{F}$ with $\mathfrak{f}'_t$.
- If $L(t) = \psi_1 \odot \psi_2$ (where $\odot \in \{\vee, \wedge\}$) and t_1, t_2 are two children of t such that $L(t_1) = \psi_1$ and $L(t_2) = \psi_2$, suppose ϕ_{t_1} and ϕ_{t_2} have been computed from t_1 and t_2 respectively, then
$$\phi_t \overset{\text{def}}{=} P'_t = P'_{t_1} = P'_{t_2} \wedge \bigwedge_{\mathfrak{f}\in\mathbb{F}} \mathfrak{f}'_t = \mathfrak{f}'_{t_1} = \mathfrak{f}'_{t_2} \wedge (\phi_{t_1} \odot \phi_{t_2}),$$
where $P'_t = P'_{t_1} = P'_{t_2}$ is an abbreviation of $\forall \mathtt{x}.\ (P'_t(\mathtt{x}) \leftrightarrow P'_{t_1}(\mathtt{x})) \wedge (P'_t(\mathtt{x}) \leftrightarrow P'_{t_2}(\mathtt{x}))$ and $\mathfrak{f}'_t = \mathfrak{f}'_{t_1} = \mathfrak{f}'_{t_2}$ is an abbreviation of $\forall \mathtt{x}.\forall \mathtt{y}.\ \left(\mathtt{x} \overset{\mathfrak{f}'_t}{\hookrightarrow} \mathtt{y} \leftrightarrow \mathtt{x} \overset{\mathfrak{f}'_{t_1}}{\hookrightarrow} \mathtt{y}\right) \wedge \left(\mathtt{x} \overset{\mathfrak{f}'_t}{\hookrightarrow} \mathtt{y} \leftrightarrow \mathtt{x} \overset{\mathfrak{f}'_{t_2}}{\hookrightarrow} \mathtt{y}\right)$.
- If $L(t) = \exists \mathtt{v}.\ \psi_1$ and t_1 is the only child of t, suppose ϕ_{t_1} has been computed, then $\phi_t \overset{\text{def}}{=} P'_t = P'_{t_1} \wedge \bigwedge\limits_{\mathfrak{f}\in\mathbb{F}} \mathfrak{f}'_t = \mathfrak{f}'_{t_1} \wedge \exists \mathtt{v}.\ \phi_{t_1}$, where $P'_t = P'_{t_1}$ and $\mathfrak{f}'_t = \mathfrak{f}'_{t_1}$ are abbreviations of formulae defined similarly to the previous case.
- If $L(t) = \psi_1 * \psi_2$ and t_1, t_2 are two children of t such that $L(t_1) = \psi_1$ and $L(t_2) = \psi_2$, suppose ϕ_{t_1} and ϕ_{t_2} have been computed, then
$$\phi_t \overset{\text{def}}{=} P'_t = P'_{t_1} \uplus P'_{t_2} \wedge \bigwedge_{\mathfrak{f}\in\mathbb{F}} \mathfrak{f}'_t = \mathfrak{f}'_{t_1} \uplus \mathfrak{f}'_{t_2} \wedge \phi_{t_1} \wedge \phi_{t_2},$$
where $P'_t = P'_{t_1} \uplus P'_{t_2}$ is an abbreviation of $\forall \mathtt{x}.\ (P'_t(\mathtt{x}) \leftrightarrow (P'_{t_1}(\mathtt{x}) \vee P'_{t_2}(\mathtt{x}))) \wedge \forall \mathtt{x}.\ (\neg P'_{t_1}(\mathtt{x}) \vee \neg P'_{t_2}(\mathtt{x}))$ and $\mathfrak{f}'_t = \mathfrak{f}'_{t_1} \uplus \mathfrak{f}'_{t_2}$ is an abbreviation of $\forall \mathtt{x}.\ \forall \mathtt{y}.\ \mathtt{x} \overset{\mathfrak{f}'_t}{\hookrightarrow} \mathtt{y} \leftrightarrow \left(\mathtt{x} \overset{\mathfrak{f}'_{t_1}}{\hookrightarrow} \mathtt{y} \vee \mathtt{x} \overset{\mathfrak{f}'_{t_2}}{\hookrightarrow} \mathtt{y}\right) \wedge \forall \mathtt{x}.\ \forall \mathtt{y}.\ \left(\neg \mathtt{x} \overset{\mathfrak{f}'_{t_1}}{\hookrightarrow} \mathtt{y} \vee \neg \mathtt{x} \overset{\mathfrak{f}'_{t_2}}{\hookrightarrow} \mathtt{y}\right)$.
- If $L(t) = \psi_1 \mathrel{-\!\!\circledast} \psi_2$ and t_1, t_2 are two children of t such that $L(t_1) = \psi_1$ and $L(t_2) = \psi_2$, suppose ϕ_{t_1} and ϕ_{t_2} have been computed, then
$$\phi_t \overset{\text{def}}{=} P'_{t_2} = P'_t \uplus P'_{t_1} \wedge \bigwedge_{\mathfrak{f}\in\mathbb{F}} \mathfrak{f}'_{t_2} = \mathfrak{f}'_t \uplus \mathfrak{f}'_{t_1} \wedge \phi_{t_1} \wedge \phi_{t_2},$$

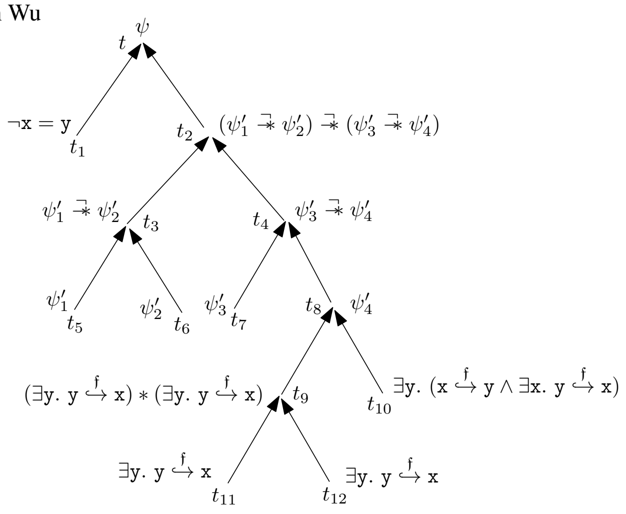
ψ
t
$\neg \mathtt{x} = \mathtt{y}$
t_1
t_2
$(\psi'_1 \overset{\neg}{-\!\!*} \psi'_2) \overset{\neg}{-\!\!*} (\psi'_3 \overset{\neg}{-\!\!*} \psi'_4)$
$\psi'_1 \overset{\neg}{-\!\!*} \psi'_2$
t_3
t_4
$\psi'_3 \overset{\neg}{-\!\!*} \psi'_4$
ψ'_1
t_5
ψ'_2
t_6
ψ'_3
t_7
t_8
ψ'_4
$(\exists \mathtt{y}.\ \mathtt{y} \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{x}) * (\exists \mathtt{y}.\ \mathtt{y} \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{x})$
t_9
t_{10}
$\exists \mathtt{y}.\ (\mathtt{x} \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{y} \wedge \exists \mathtt{x}.\ \mathtt{y} \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{x})$
$\exists \mathtt{y}.\ \mathtt{y} \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{x}$
t_{11}
t_{12}
$\exists \mathtt{y}.\ \mathtt{y} \overset{\mathfrak{f}}{\hookrightarrow} \mathtt{x}$

Fig. 1. The syntax tree $\mathcal{T}_\psi$: An example

where $P'_t = P'_{t_1} \uplus P'_{t_2}$ and $\mathfrak{f}'_{t_2} = \mathfrak{f}'_t \uplus \mathfrak{f}'_{t_1}$ are abbreviations of the formulae that can be defined similarly to the previous case.

Example 2. Let $\psi \overset{\text{def}}{=} (\neg\ \mathtt{x} = \mathtt{y}) \ \wedge\ (\psi'_1 \overset{\neg}{-\!\!*} \psi'_2) \overset{\neg}{-\!\!*} (\psi'_3 \overset{\neg}{-\!\!*} \psi'_4)$ be the formula in Example 1. Then $\mathcal{T}_\psi$ is illustrated in Fig. 1. By a bottom-up computation, we get $\phi_{t_5} \overset{\text{def}}{=} \mathtt{x} \overset{\mathfrak{f}_{t_5}}{\hookrightarrow} \mathtt{y}$,

$\phi_{t_3} \overset{\text{def}}{=} P'_{t_6} = P'_{t_5} \uplus P'_{t_3} \wedge \mathfrak{f}'_{t_6} = \mathfrak{f}'_{t_5} \uplus \mathfrak{f}'_{t_3} \wedge \mathtt{x} \overset{\mathfrak{f}'_{t_5}}{\hookrightarrow} \mathtt{y} \wedge \wedge \exists \mathtt{y}.\ \mathtt{x} \overset{\mathfrak{f}'_{t_6}}{\hookrightarrow} \mathtt{y} \wedge \exists \mathtt{y}.\ \mathtt{y} \overset{\mathfrak{f}'_{t_6}}{\hookrightarrow} \mathtt{x}$,

$\phi_{t_9} \overset{\text{def}}{=} P'_{t_9} = P'_{t_{11}} \uplus P'_{t_{12}} \wedge \mathfrak{f}'_{t_9} = \mathfrak{f}'_{t_{11}} \uplus \mathfrak{f}'_{t_{12}} \wedge \exists \mathtt{y}.\ \mathtt{y} \overset{\mathfrak{f}'_{t_{11}}}{\hookrightarrow} \mathtt{x} \wedge \exists \mathtt{y}.\ \mathtt{y} \overset{\mathfrak{f}'_{t_{12}}}{\hookrightarrow} \mathtt{x}$,

and $\phi_{t_8} \overset{\text{def}}{=} P'_{t_8} = P'_{t_9} = P'_{t_{10}} \wedge \bigwedge\limits_{\mathfrak{f} \in \mathbb{F}} \mathfrak{f}'_{t_8} = \mathfrak{f}'_{t_9} = \mathfrak{f}'_{t_{10}} \wedge \phi_{t_9} \wedge \phi_{t_{10}}$, where $\phi_{t_{10}}$ is the formula corresponding to t_{10}, and $\phi_{t_7} \overset{\text{def}}{=} \neg \mathtt{x} \overset{\mathfrak{f}'_{t_7}}{\hookrightarrow} \mathtt{y} \wedge \exists \mathtt{y}.\ \left(\mathtt{x} \overset{\mathfrak{f}'_{t_7}}{\hookrightarrow} \mathtt{y} \wedge \forall \mathtt{x}.\ \neg \mathtt{y} \overset{\mathfrak{f}'_{t_7}}{\hookrightarrow} \mathtt{x}\right)$, in addition, ϕ_{t_4} can be constructed from ϕ_{t_7} and ϕ_{t_8}, similarly to the construction of ϕ_{t_3}, and $\phi_{t_2} \overset{\text{def}}{=} P'_{t_4} = P'_{t_3} \uplus P'_{t_2} \wedge \mathfrak{f}'_{t_4} = \mathfrak{f}'_{t_3} \uplus \mathfrak{f}'_{t_2} \wedge \phi_{t_3} \wedge \phi_{t_4}$. The formula ϕ_{t_3} contains a conjunct $\phi_{t_5} = \mathtt{x} \overset{\mathfrak{f}'_{t_5}}{\hookrightarrow} \mathtt{y}$, while ϕ_{t_4} contains a conjunct $\phi_{t_7} = \neg \mathtt{x} \overset{\mathfrak{f}'_{t_7}}{\hookrightarrow} \mathtt{y} \wedge \exists \mathtt{y}.\ \left(\mathtt{x} \overset{\mathfrak{f}'_{t_7}}{\hookrightarrow} \mathtt{y} \wedge \forall \mathtt{x}.\ \neg \mathtt{y} \overset{\mathfrak{f}'_{t_7}}{\hookrightarrow} \mathtt{x}\right)$. Thus the conflict between ψ'_1 and ψ'_3 is resolved. □

Proposition 2. *For each ESPSL2*$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f} \in \mathbb{F}}, *, \overset{\neg}{-\!\!*}\right)$ *formula* ψ*,* ψ *is satisfiable iff* $\mathtt{trs}(\psi)$ *is satisfiable.*

Proof. Suppose ψ is an ESPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f} \in \mathbb{F}}, *, \overset{\neg}{-\!\!*}\right)$ formula.

"Only if" direction: Suppose that ψ is satisfiable, that is, there is a pair $(\mathfrak{h}, \mathfrak{m})$ such that $\mathfrak{h} \models_{\mathfrak{m}} \psi$.

Let $\mathtt{Leaves}(\mathcal{T}_\psi)$ denote the set of leaves of $\mathcal{T}_\psi$. Then $\{L(t) \mid t \in \mathtt{Leaves}(\mathcal{T}_\psi)\}$ is a subset of SPSL2$\left((\overset{\mathfrak{f}}{\hookrightarrow})_{\mathfrak{f} \in \mathbb{F}}\right)$ formulae. Since $\mathfrak{h} \models_{\mathfrak{m}} \psi$, we know that there is a subset of $\texttt{Leaves}(\mathcal{T}_\psi)$, say T', such that each $t' \in T'$ can be assigned a heap $\mathfrak{h}_{t'}$ with a nonempty-domain, in order to witness the fact $\mathfrak{h} \models_{\mathfrak{m}} \psi$. Then we construct a heap $\mathfrak{h}'$ to satisfy $\mathfrak{trs}(\psi)$ as follows:

1. For each $t' \in T'$, let $(\mathfrak{h}'_{t'}, \mathfrak{I}'_{t'})$ be the labeled heap such that for each $\mathfrak{f} \in \mathbb{F}$, $(\mathfrak{h}'_{t'})_{\mathfrak{f}'_{t'}}(\mathfrak{l}) = \mathfrak{l}'$ iff $(\mathfrak{h}_{t'})_{\mathfrak{f}}(\mathfrak{l}) = \mathfrak{l}'$, in addition, $\mathfrak{I}'_{t'}(P'_{t'}) = \texttt{dom}(\mathfrak{h}'_{t'})$ and $\mathfrak{I}'_{t'}(P') = \emptyset$ for each other unary predicate $P' \in \mathcal{P}'$. Moreover, for each leaf $t' \notin T'$, let $(\mathfrak{h}_{t'}, \mathfrak{I}'_{t'})$ be the labeled heap such that $\mathfrak{h}_{t'}$ has an empty domain and $\mathfrak{I}'_{t'}(P') = \emptyset$ for each $P' \in \mathcal{P}'$.
2. By induction on the structure of $\mathcal{T}_\psi$, we can construct bottom-up a labeled heap $(\mathfrak{h}'_t, \mathfrak{I}'_t)$ for each node $t \in T$. In the construction, we need trace the relationship between unary predicates in $\mathcal{P}'$ and the relationship between the fields in $\mathbb{F}'$ which are enforced by the nodes in $\mathcal{T}_\psi$. For instance, if t is a node such that $L(t) = \psi_1 * \psi_2$ and t has two children t_1 and t_2, suppose $(\mathfrak{h}'_{t_1}, \mathfrak{I}'_{t_1})$ and $(\mathfrak{h}'_{t_2}, \mathfrak{I}'_{t_2})$ have been computed, then $\mathfrak{h}'_t$ is computed as the domain-disjoint union of $\mathfrak{h}'_{t_1}$ and $\mathfrak{h}'_{t_2}$, in addition, for each $\mathfrak{f} \in \mathbb{F}$ and each pair of locations $(\mathfrak{l}, \mathfrak{l}')$, $(\mathfrak{h}'_t)_{\mathfrak{f}'_t}(\mathfrak{l}) = \mathfrak{l}'$ iff $(\mathfrak{h}'_{t_1})_{\mathfrak{f}'_{t_1}}(\mathfrak{l}) = \mathfrak{l}'$ or $(\mathfrak{h}'_{t_2})_{\mathfrak{f}'_{t_2}}(\mathfrak{l}) = \mathfrak{l}'$ (here $(\mathfrak{h}'_t)_{\mathfrak{f}'_t}$ is well-defined since $(\mathfrak{h}'_{t_1})_{\mathfrak{f}'_{t_1}}$ and $(\mathfrak{h}'_{t_2})_{\mathfrak{f}'_{t_2}}$ are domain-disjoint). Moreover, $\mathfrak{I}'_t(P'_t) = \mathfrak{I}'_{t_1}(P'_{t_1}) \cup \mathfrak{I}'_{t_2}(P'_{t_2})$, and for each other unary predicate $P' \in \mathcal{P}'$, $\mathfrak{I}'_t(P') = \mathfrak{I}'_{t_1}(P') \cup \mathfrak{I}'_{t_2}(P')$.

"If" direction: Suppose that $\mathfrak{trs}(\psi)$ is satisfiable. Then there is a labeled heap $(\mathfrak{h}, \mathfrak{I})$ and an assignment $\mathfrak{m}$ such that $(\mathfrak{h}, \mathfrak{I}) \models_{\mathfrak{m}} \mathfrak{trs}(\psi)$. The by induction on the structure of $\mathcal{T}_\psi$, we can compute bottom-up a heap $\mathfrak{h}'$ such that $\mathfrak{h}' \models_{\mathfrak{m}} \psi$. The construction is essentially just a renaming of the fields. □

5 Conclusion

In this paper, we proposed SPSL2, semi-positive first-order logic with two variables, and investigated the complexity of the satisfiability problem of several fragments of SPSL2. Our main result is that the satisfiability of ESPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f} \in \mathbb{F}}, *, \overset{\neg}{-\!\!*}\right)$, the fragment of SPSL2 where separating conjunction $*$ and septraction $\overset{\neg}{-\!\!*}$ (the dual operator of magic wand $-\!\!*$) may occur, but none of them occurs in the scope of universal quantifiers, is NEXPTIME-complete. The proof of this result relies on the NEXPTIME-completeness result of SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f} \in \mathbb{F}}, \mathcal{P}\right)$, the fragment of SPSL2 where separating operators do not occur, but unary predicates are available. A byproduct of this work is that the finite satisfiability of first order logic with two variables and one function symbol (without unary predicates) is NEXPTIME-complete. Although some interesting questions, e.g. the decidability of SPSL2$\left((\xhookrightarrow{\mathfrak{f}})_{\mathfrak{f} \in \mathbb{F}}, *, -\!\!*\right)$, are left open in this paper, we believe that this work can be seen as a substantial step towards solving them in the future.

Acknowledgements

This work was partially done when I was a visiting researcher at LIAFA, Université Paris Diderot, from June 2014 to June 2015, supported by China Scholarship Council. My great thanks go to Stéphane Demri for the numerous discussions with him when I did this work. At last, I would like to thank the reviewers for their valuable comments.

References

1. T. Antonopoulos, N. Gorogiannis, C. Haase, M. I. Kanovich, and J. Ouaknine. Foundations for decision problems in separation logic with general inductive predicates. In *FoSSaCS*, pages 411–425, 2014.
2. S. Benaim, M. Benedikt, W. Charatonik, E. Kieronski, R. Lenhardt, F. Mazowiecki, and J. Worrell. Complexity of two-variable logic on finite trees. In *ICALP*, pages 74–88, 2013.
3. M. Bojańczyk, C. David, A. Muscholl, Th. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Transactions on Computational Logic*, 12(4):27, 2011.
4. R. Brochenin, S. Demri, and E. Lozes. On the Almighty Wand. *Information and Computation*, 211:106–137, 2012.
5. C. Calcagno, P. O'Hearn, and H. Yang. Computability and complexity results for a spatial assertion language for data structures. In *FSTTCS*, pages 108–119, 2001.
6. W. Charatonik, E. Kieroński, and F. Mazowiecki. Decidability of Weak Logics with Deterministic Transitive Closure. In *CSL-LICS*, 2014.
7. B. Cook, C. Haase, J. Ouaknine, M. Parkinson, and J. Worrell. Tractable reasoning in a fragment of separation logic. In *CONCUR*, pages 235–249, 2011.
8. S. Demri and M. Deters. Expressive completeness of separation logic with two variables and no separating conjunction. In *CSL-LICS*, pages 1–37, 2014.
9. S. Demri and M. Deters. Two-variable separation logic and its inner circle. *ACM Transactions on Computational Logic*, 16(2):15, 2015.
10. Stéphane Demri, Didier Galmiche, Dominique Larchey-Wendling, and Daniel Méry. Separation logic with one quantified variable. In *CSR*, pages 125–138, 2014.
11. Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, 179(2):279–295, 2002.
12. Erich Grädel and Martin Otto. On logics with two variables. *Theor. Comput. Sci.*, 224(1-2):73–113, 1999.
13. Erich Grädel, Martin Otto, and Eric Rosen. Two-variable logic with counting is decidable. In *LICS*, pages 306–317, 1997.
14. Erich Grädel, Martin Otto, and Eric Rosen. Undecidability results on two-variable logics. *Arch. Math. Log.*, 38(4-5):313–354, 1999.
15. M. L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.
16. Leszek Pacholski, WiesL aw Szwast, and Lidia Tendera. Complexity results for first-order two-variable logic with counting. *SIAM J. Comput.*, 29(4):1083–1117, February 2000.
17. R. Piskac, Th. Wies, and D. Zufferey. Automating separation logic using SMT. In *CAV*, pages 773–789, 2013.
18. Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *J. of Logic, Lang. and Inf.*, 14(3):369–395, 2005.
19. Ian Pratt-Hartmann. The two-variable fragment with counting revisited. In *Proceedings of the 17th International Conference on Logic, Language, Information and Computation*, WoLLIC'10, pages 42–54, 2010.
20. J.C. Reynolds. Separation logic: a logic for shared mutable data structures. In *LICS*, pages 55–74, 2002.
21. L. Segoufin and B. ten Cate. Unary negation. *LMCS*, 9(3), 2013.
22. A. Thakur, J. Breck, and Th. Reps. Satisfiability modulo abstraction for separation logic with linked lists. In *SPIN*, pages 58–67, 2014.
23. B. Trakhtenbrot. Impossibility of an algorithm for the decision problem in finite classes. *AMS Translations, Series 2*, 23:1–5, 1963.