The Lower Bound of Decidable Entailments in Separation Logic with Inductive Definitions

Mnacho Echenim, Radu Iosif, Nicolas Peltier

The Lower Bound of Decidable Entailments in Separation Logic with Inductive Definitions

Mnacho Echenim[1], Radu Iosif[2] and Nicolas Peltier[1]

[1] Univ. Grenoble Alpes, CNRS, LIG, F-38000 Grenoble France
[2] Univ. Grenoble Alpes, CNRS, VERIMAG, F-38000 Grenoble France

Abstract. The entailment between separation logic formulæ with inductive predicates (also known as symbolic heaps) has been shown to be decidable for a large class of inductive definitions [4]. Recently, a 2EXPTIME algorithm has been proposed [7], however no precise lower bound is known (although a EXPTIME-hard bound for this problem has been established in [5]). In this paper, we show that deciding entailment between predicate atoms is 2EXPTIME-hard. The proof is based on a reduction from the membership problem for exponential-space bounded alternating Turing machines [3].

1 Introduction

In this paper, we consider the fragment of separation logic formulæ known as *symbolic heaps*, defined as (separated) conjunctions of atoms. Such atoms may be either equational atoms, asserting equalities or disequalities between memory locations, points-to atoms asserting that some location refers to a given record, or may be built on additional predicates that assert that a part of the memory has a specific shape. For genericity, such predicates are associated with user-provided inductive definitions that allow one to describe custom data-structures. Satisfiability is EXPTIME-complete for such formulæ [2], but entailment is not decidable in general[3] [5,1]. However, the entailment problem was proven to be decidable for a large class of inductive definitions, whose syntactical restrictions ensure that the generated heap structures have a bounded-tree width [4], using a reduction to monadic second order logic interpreted over graphs. A EXPTIME-hard bound was established in [5], and very recently, a 2EXPTIME algorithm has been proposed [7]. However, personal communication with the authors of [7] revealed several problems related to quantifier instantiation that are potential sources of incompleteness. Thus we believe that their decision procedure might not cover the entire class of entailments between symbolic heaps and is complete for a strict subclass in which the existential quantifiers of the right-hand side of entailments can be instantiated by the free variables on the left. In this paper, we show that the problem is 2EXPTIME-hard, even if only entailment between predicate atoms is considered. The proof uses a reduction from the membership problem for alternating Turing machines [3] whose working tape is exponentially bounded in the size of the input. Proving a matching 2EXPTIME upper bound reduces essentially to fixing the issues of the algorithm proposed in [7] and is considered for future work.

[3] Entailment does not reduce to satisfiability since the considered logic has no negation.

2 Separation Logic with Inductive Definitions

For a finite set S, we denote by $\|S\| \in \mathbb{N}$ its cardinality. For a partial mapping $f : A \rightharpoonup B$, let $\mathrm{dom}(f) \stackrel{\text{def}}{=} \{x \in A \mid f(x) \in B\}$ and $\mathrm{img}(f) \stackrel{\text{def}}{=} \{f(x) \mid x \in A\}$ be its domain and range, respectively, and we write $f : A \rightharpoonup_{fin} B$ if $\|\mathrm{dom}(f)\| < \infty$. Given integers $n \leq m$, we denote by $[\![n \,..\, m]\!]$ the set $\{n, n+1, \ldots, m\}$. For a tuple $\boldsymbol{t} = (t_1, \ldots, t_n)$ and $i \in [\![1 \,..\, n]\!]$, we denote by $\boldsymbol{t}_i$ the element on the i-th position in $\boldsymbol{t}$. By slight abuse of notation, we write $t \in \boldsymbol{t}$ if $t = t_i$, for some $i \in [\![1 \,..\, n]\!]$.

Let $\mathsf{Var} = \{x, y, \ldots\}$ be an infinite countable set of logical first-order variables and $\mathsf{Pred} = \{p, q, \ldots\}$ be an infinite countable set of uninterpreted relation symbols, called *predicates*. Each predicate p has an arity $\#p \geq 1$, denoting the number of arguments. In addition, we consider a special function symbol nil, of arity zero. A *term* is an element of the set $\mathsf{Term} \stackrel{\text{def}}{=} \mathsf{Var} \cup \{\mathsf{nil}\}$. Let $\mathsf{k} \geq 1$ be an integer constant fixed throughout this paper. The logic SL^{k} is the set of formulæ generated inductively as follows:

$$\phi := \mathsf{emp} \mid t_0 \mapsto (t_1, \ldots, t_{\mathsf{k}}) \mid p(t_1, \ldots, t_{\#p}) \mid t_1 \doteq t_2 \mid t_1 \dot{\neq} t_2 \mid \phi_1 * \phi_2 \mid \exists x \,.\, \phi_1$$

where $p \in \mathsf{Pred}$, $t_1, t_2, \ldots, t_{\#p} \in \mathsf{Term}$ and $x \in \mathsf{Var}$. We write $\bot$ for $t \dot{\neq} t$, for any $t \in \mathsf{Term}$.

A *ground formula* is a formula of SL^{k} in which no predicates occur. We write $\mathsf{fv}(\phi)$ for the set of *free* variables, that occur in ϕ not within the scope of an existential quantifier. By writing $\phi(x_1, \ldots, x_n)$ we mean $x_1, \ldots, x_n \in \mathsf{fv}(\phi)$ and $\phi[y_1/x_1, \ldots, y_n/x_n]$ is the formula obtained from ϕ by simultaneously substituting each x_i with y_i, for $i \in [\![1 \,..\, n]\!]$. A *substitution* is a mapping $\sigma : \mathsf{Var} \to \mathsf{Term}$ and we denote by $\phi\sigma$ the formula $\phi[\sigma(x_1)/x_1, \ldots, \sigma(x_n)/x_n]$.

To interpret SL^{k} formulæ, we consider an infinite countable set Loc of *locations* and a designated location $nil \in \mathsf{Loc}$. Let $\mathsf{Loc}^i \stackrel{\text{def}}{=} \{(\ell_1, \ldots, \ell_i) \mid \ell_1, \ldots, \ell_i \in \mathsf{Loc}\}$. The semantics of SL^{k} formulæ is defined in terms of *structures* $(\mathfrak{s}, \mathfrak{h}, \mathfrak{I})$, where:

- $\mathfrak{s} : \mathsf{Term} \to \mathsf{Loc}$ is a total mapping of terms into locations, called *store*, such that always $\mathfrak{s}(\mathsf{nil}) = nil$,
- $\mathfrak{h} : \mathsf{Loc} \rightharpoonup_{fin} \mathsf{Loc}^{\mathsf{k}}$ is a finite partial mapping of locations into k-tuples of locations, called *heap*, such that $nil \notin \mathrm{dom}(\mathfrak{h})$. Let Heaps denotes the set of heaps, and
- $\mathfrak{I} : \mathsf{Pred} \to \bigcup_{i \geq 1} 2^{\mathsf{Loc}^i \times \mathsf{Heaps}}$ is an *interpretation* associating each predicate p a set of pairs $\left\langle (\ell_1, \ldots, \ell_{\#p}), \mathfrak{h}' \right\rangle \in \mathsf{Loc}^{\#p} \times \mathsf{Heaps}$.

Two heaps $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are *disjoint* iff $\mathrm{dom}(\mathfrak{h}_1) \cap \mathrm{dom}(\mathfrak{h}_2) = \emptyset$, in which case their *disjoint union* is denoted $\mathfrak{h}_1 \uplus \mathfrak{h}_2$, undefined otherwise. The satisfaction relation $\models$ between structures and SL^{k} formulæ is defined, as usual, recursively on the syntax of formulæ:

$$\begin{array}{ll}
(\mathfrak{s}, \mathfrak{h}, \mathfrak{I}) \models t_1 \doteq t_2 & \Leftrightarrow \mathfrak{h} = \emptyset \text{ and } \mathfrak{s}(t_1) = \mathfrak{s}(t_2) \\
(\mathfrak{s}, \mathfrak{h}, \mathfrak{I}) \models t_1 \dot{\neq} t_2 & \Leftrightarrow \mathfrak{h} = \emptyset \text{ and } \mathfrak{s}(t_1) \neq \mathfrak{s}(t_2) \\
(\mathfrak{s}, \mathfrak{h}, \mathfrak{I}) \models \mathsf{emp} & \Leftrightarrow \mathfrak{h} = \emptyset \\
(\mathfrak{s}, \mathfrak{h}, \mathfrak{I}) \models t_0 \mapsto (t_1, \ldots, t_{\mathsf{k}}) & \Leftrightarrow \mathrm{dom}(\mathfrak{h}) = \{\mathfrak{s}(t_0)\} \text{ and } \mathfrak{h}(\mathfrak{s}(t_0)) = (\mathfrak{s}(t_1), \ldots, \mathfrak{s}(t_{\mathsf{k}})) \\
(\mathfrak{s}, \mathfrak{h}, \mathfrak{I}) \models p(t_1, \ldots, t_{\#p}) & \Leftrightarrow \left\langle (\mathfrak{s}(t_1), \ldots, \mathfrak{s}(t_{\#p})), \mathfrak{h} \right\rangle \in \mathfrak{I}(p) \\
(\mathfrak{s}, \mathfrak{h}, \mathfrak{I}) \models \phi_1 * \phi_2 & \Leftrightarrow \text{there are disjoint heaps } \mathfrak{h}_1 \text{ and } \mathfrak{h}_2, \text{ such that } \mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2 \\
 & \quad \text{and } (\mathfrak{s}, \mathfrak{h}_i, \mathfrak{I}) \models \phi_i, \text{ for each } i = 1, 2 \\
(\mathfrak{s}, \mathfrak{h}, \mathfrak{I}) \models \exists x \,.\, \phi & \Leftrightarrow (\mathfrak{s}[x \leftarrow \ell], \mathfrak{h}, \mathfrak{I}) \models \phi, \text{ for some } \ell \in \mathsf{Loc}
\end{array}$$

where $\mathfrak{s}[x \leftarrow \ell]$ is the store mapping x into ℓ and behaving like $\mathfrak{s}$ for all $t \in \mathsf{Term} \setminus \{x\}$.

Given formulæ ϕ and ψ, we write $\phi \stackrel{\mathsf{ac}}{=} \psi$ if ψ is obtained from ϕ by a reordering of $*$-connected subformulæ. Since $*$ is associative and commutative, any two such formulæ are equivalent. It is known that, for any formulæ ϕ and ψ, if $x \notin \mathsf{fv}(\psi)$ then the formulæ $(\exists x \,.\, \phi) * \psi$ and $\exists x \,.\, \phi * \psi$ are equivalent [6]. Thus, each SL^{k} formula ϕ can be written as $\exists x_1 \ldots \exists x_n \,.\, \psi * \circledast_{j=1}^{m} p_j(u_1^j, \ldots, u_{\#p_j}^j)$, where $\widehat{\phi} \stackrel{\mathsf{def}}{=} \psi$ is a ground formula, called the *hat* of ϕ. Note that, moreover, $\widehat{\phi}$ is unique, up to the $\stackrel{\mathsf{ac}}{=}$ relation.

In the rest of this paper, we consider that the predicates are intepreted by a set $\mathcal{S}$ of *rules* $p(x_1, \ldots, x_{\#p}) \Leftarrow \rho$, where ρ is an SL^{k} formula, such that $\mathsf{fv}(\rho) \subseteq \{x_1, \ldots, x_{\#_p}\}$. We refer to $p(x_1, \ldots, x_{\#p})$ as to the *head* and to ρ as to the *body* of the rule. A rule is a *base rule* if its body is a ground formula. W.l.o.g., we assume that any two heads with the predicate symbol p share the same tuple of variables $\boldsymbol{x}_p$ (this can be achieved by variable renaming) and by writing $p(\boldsymbol{x}_p)$ we mean that $\boldsymbol{x}_p$ is the unique tuple of variables associated with p. We write $p(\boldsymbol{x}_p) \Leftarrow_{\mathcal{S}} \rho$ if the rule $p(\boldsymbol{x}_p) \Leftarrow \rho$ belongs to $\mathcal{S}$.

A set $\mathcal{S}$ of rules defines a transformer function $\mathfrak{T}_{\mathcal{S}}$ on interpretations of predicates. For an interpretation $\mathfrak{I}$, the interpretation $\mathfrak{T}_{\mathcal{S}}(\mathfrak{I})$ is defined as:

$$\mathfrak{T}_{\mathcal{S}}(\mathfrak{I}) \stackrel{\mathsf{def}}{=} \lambda p \,.\, \left\{ \langle \boldsymbol{\ell}, \mathfrak{h} \rangle \mid ([\boldsymbol{x}_p \leftarrow \boldsymbol{\ell}], \mathfrak{h}, \mathfrak{I}) \models \rho,\ p(\boldsymbol{x}_p) \Leftarrow_{\mathcal{S}} \rho \right\}$$

where $\boldsymbol{x}_p = (x_1, \ldots, x_{\#p})$, $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_{\#p})$ and $[\boldsymbol{x}_p \leftarrow \boldsymbol{\ell}]$ is any store mapping x_i into ℓ_i, for all $i \in [\![1 \,..\, \#p]\!]$. It is not hard to prove that the set of interpretations forms a complete lattice, equipped with the pointwise inclusion as a partial order, union as join and intersection as meet. Moreover, the transformer function is monotonic, because no predicate symbol occurs under negation in a rule body. Consequently, $\mathfrak{T}_{\mathcal{S}}$ has a unique least fixed point $\mathfrak{T}_{\mathcal{S}}^{\mu}$. In the rest of this paper, given a set of rules $\mathcal{S}$, we write $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} \phi$ for $(\mathfrak{s}, \mathfrak{h}, \mathfrak{T}_{\mathcal{S}}^{\mu}) \models \phi$ and refer also to the pairs $(\mathfrak{s}, \mathfrak{h})$ as structures. We are now ready to define the class of entailment problems, which are the concern of this paper:

Definition 1 (Entailment Problem). *Given a set of rules $\mathcal{S}$ and two SL^{k} formulæ ϕ and ψ is it the case that for every store $\mathfrak{s}$ and every heap $\mathfrak{h}$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} \phi$, we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} \psi$? Instances of the entailment problem are denoted $\phi \models_{\mathcal{S}} \psi$.*

2.1 Unfolding Trees

In many cases it is useful to think of heaps from the least fixed point interpretation of a predicate symbol as obtained by a finite unfolding of the predicate, which produces a ground formula in which all the predicate symbols have been successively replaced with the bodies of their defining rules. In this section, let $\mathcal{S}$ be a given set of rules.

Definition 2 (Unfolding). *A formula ψ is a* 1-unfolding *of a formula ϕ, written $\phi \Longrightarrow_{\mathcal{S}} \psi$, if ψ is obtained by substituting an occurrence of an atom $p(y_1, \ldots, y_{\#p})$ in ϕ with $\rho[y_1/x_1, \ldots, y_{\#p}/x_{\#p}]$, where $(p(x_1, \ldots, x_{\#p}), \rho) \in \mathcal{S}$ is a rule. An* unfolding *of ϕ is a formula ψ such that $\phi \Longrightarrow_{\mathcal{S}}^{*} \psi$, where $\Longrightarrow_{\mathcal{S}}^{*}$ is the reflexive and transitive closure of $\Longrightarrow_{\mathcal{S}}$. Moreover, ψ is a* ground unfolding *of ϕ if ψ is a ground formula.*

It is often convenient to place the steps of an unfolding sequence in a tree that records the partial order in which these steps occur. We assume the set of nodes in a tree t, denoted as $\mathrm{nodes}(t)$, to be a finite prefix-closed subset of $\mathbb{N}^*$, where $\mathbb{N}^*$ is the set of finite sequences of positive integers, such that if w and wi are nodes of t, for some $i \in \mathbb{N} \setminus \{0\}$, then so are wj, for all $j \in [\![0 \,..\, i-1]\!]$. We write $|w|$ for the lenght of the sequence w and λ for the empty sequence ($|\lambda| = 0$). The label of a node $w \in \mathrm{nodes}(t)$ is denoted by $t(w)$ and the subtree of t rooted at w by $t_{|w}$. The height of a tree is $\mathrm{height}(t) \stackrel{\text{def}}{=} \max_{w \in \mathrm{nodes}(t)} |w|$. A *traversal* of a tree t is an enumeration of $\mathrm{nodes}(t)$ such that the index of a node is smaller than all indices of its successors.

Definition 3 (Unfolding Tree). *Given a formula ϕ, an* unfolding tree *for ϕ is a tree u, labeled with pairs (p, ψ), where $p \in \mathsf{Pred} \cup \{\bot\}$ and ψ is a formula, such that:*

- *$t(\lambda) = (\bot, \phi)$, and*
- *if $u(w) = (p, \psi)$, then there exists a bijective mapping from the set of occurrences of atoms $q(t_1, \ldots, t_{\#q})$ in ψ and the children of w, such that if $q(t_1, \ldots, t_{\#q})$ is mapped to wi, then $u(wi) = (q(t_1, \ldots, t_{\#q}), \rho[t_1/x_1, \ldots, t_{\#q}/x_{\#q}])$, where $q(x_1, \ldots, x_{\#q}) \Leftarrow_{\mathcal{S}} \rho$.*

We denote by $\mathcal{T}_{\mathcal{S}}(\phi)$ the set of unfolding trees for ϕ.

Clearly, every traversal of an unfolding tree $u \in \mathcal{T}_{\mathcal{S}}(\phi_0)$ corresponds to a ground unfolding sequence $\sigma : \phi_0 \Longrightarrow_{\mathcal{S}} \phi_1 \Longrightarrow_{\mathcal{S}} \ldots \Longrightarrow_{\mathcal{S}} \psi$ where, for each node w_i of label $(q(\boldsymbol{t}), \psi)$ in σ, ϕ_i is obtained from ϕ_{i-1} by replacing $q(\boldsymbol{t})$ by ψ. It is easy to check that the outcome ψ only depends on u, not on σ. In other words, all traversals of u yield the same ground formula, denoted by $\mathcal{F}(t)$. By convention, we let $\mathcal{F}(u) \stackrel{\text{def}}{=} \phi$ if $\mathrm{nodes}(u) = \{\lambda\}$. The following lemma provides an equivalent definition for the semantics of SL^{k} formulæ:

Lemma 1. *Given a SL^{k} formula ϕ, for any store $\mathfrak{s}$ and any heap $\mathfrak{h}$, we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} \phi$ if and only if $(\mathfrak{s}, \mathfrak{h}) \models \mathcal{F}(u)$, for some unfolding tree $u \in \mathcal{T}_{\mathcal{S}}(\phi)$.*

Proof: It is sufficient to give the proof only in the case $\phi = p(t_1, \ldots, t_{\#p})$, for a predicate symbol $p \in \mathsf{Pred}$ and terms $t_1, \ldots, t_{\#p} \in \mathsf{Term}$, the generalization to the case where ϕ is any SL^{k} formula being immediate. We start by proving the following fact:

Fact 1 *Let $u \in \mathcal{T}_{\mathcal{S}}\Big(p(t_1, \ldots, t_{\#p})\Big)$ be an unfolding tree, such that (necessarily) $u(0) = (p(t_1, \ldots, t_{\#p}), \rho\sigma)$, for some rule:*

$$p(x_1, \ldots, x_{\#p}) \Leftarrow \underbrace{\exists y_1 \ldots \exists y_n \,.\, \hat{\rho} * \mathop{\Large *}_{j=1}^{m} p_j(t_1^j, \ldots, t_{\#p_j}^j)}_{\rho}$$

and some substitution σ such that $\sigma(x_k) = t_k$, for all $k \in [\![1 \,..\, \#p]\!]$, then $\mathcal{F}(u) \stackrel{\text{ac}}{=} \exists y_1 \ldots \exists y_n \,.\, \hat{\rho}\sigma * * \mathop{\Large *}_{j=1}^{m} \mathcal{F}(u_j)$, where $u_j \in \mathcal{T}_{\mathcal{S}}\Big(p_j(t_1^j, \ldots, t_{\#p_j}^j)\sigma\Big)$, for all $j \in [\![1 \,..\, m]\!]$.

Proof: Let $s : \phi_0 = p(t_1, \ldots, t_{\#p}) \Longrightarrow_{\mathcal{S}} \phi_1 = \rho\sigma \Longrightarrow_{\mathcal{S}} \ldots \Longrightarrow_{\mathcal{S}} \phi_r = \mathcal{F}(u)$ be a ground unfolding sequence. Then there exists $\ell \in [\![1 \,..\, m]\!]$ and a position $j \in [\![1 \,..\, r]\!]$ in this sequence such that the sequence $\phi_1 \Longrightarrow_{\mathcal{S}} \ldots \Longrightarrow_{\mathcal{S}} \phi_j$ traverses all nodes from $u_{|\ell}$. W.l.o.g., let ℓ and j be the least such numbers, respectively. We re-organize the unfolding sequence s into s^1 by moving all steps between 1 and j, corresponding to visits of nodes

from $u_{|\ell'}$, for $\ell' \in [\![1 \,..\, m]\!] \setminus \{\ell\}$, after j and let $k \leq j$ be the new position of ϕ_j in s^1. Clearly this re-organization does not change the outcome ϕ_r, who is the same for s and s_1. Because all nodes from $u_{|j}$ are visited by $s_1 : \phi_0^1 = \phi_0 \Longrightarrow_{\mathcal{S}} \phi_1^1 = \phi_1 \Longrightarrow_{\mathcal{S}} \ldots \Longrightarrow_{\mathcal{S}} \phi_k^1 \Longrightarrow_{\mathcal{S}} \ldots \Longrightarrow_{\mathcal{S}} \phi_r^1 = \phi_r$, we obtain:

$$\phi_k^1 \stackrel{\text{ac}}{=} \exists y_1 \ldots \exists y_n \,.\, \hat{\rho}\sigma * \mathop{\Large *}_{j \in [\![1..m]\!] \setminus \ell} p_j(t_1^j, \ldots, t_{\#p_j}^j) * \mathcal{F}(u_\ell)$$

The unfolding sequence $\phi_k^1 \Longrightarrow_{\mathcal{S}} \ldots \Longrightarrow_{\mathcal{S}} \phi_r$ is strictly smaller than s and we apply the same argument inductively until the ground formula $\mathcal{F}(u)$ is derived. □

Let $\boldsymbol{t} \stackrel{\text{def}}{=} (t_1, \ldots, t_{\#p})$ and $\mathfrak{s}(\boldsymbol{t}) \stackrel{\text{def}}{=} (\mathfrak{s}(t_1), \ldots, \mathfrak{s}(t_{\#p}))$. Let $\mathfrak{T}_{\mathcal{S}}^0 \stackrel{\text{def}}{=} \lambda p.\emptyset$ be the interpretation mapping each predicate symbol into the empty set. We have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} p(\boldsymbol{t})$ if and only if $\langle \mathfrak{s}(\boldsymbol{t}), \mathfrak{h} \rangle \in \mathfrak{T}_{\mathcal{S}}^{\mu}(p) = \left(\bigcup_{i=0}^{\infty} \mathfrak{T}_{\mathcal{S}}^i\right)(p)$. The last equality uses the fact that $\mathfrak{T}_{\mathcal{S}}$ is monotone and continuous which, by Kleene's Fixpoint Theorem, guarantees that its least fixpoint is the limit of the increasing sequence of approximants $\left\{\mathfrak{T}_{\mathcal{S}}^i\right\}_{i \geq 0}$.

For any predicate symbol $p \in \mathsf{Pred}$ and any tuple of terms $\boldsymbol{t} = (t_1, \ldots, t_{\#p})$, we prove the following equivalence by induction on $i \geq 0$:

$$\langle \mathfrak{s}(\boldsymbol{t}), \mathfrak{h} \rangle \in \mathfrak{T}_{\mathcal{S}}^i(p) \Leftrightarrow (\mathfrak{s}, \mathfrak{h}) \models \mathcal{F}(u), \text{ for some } u \in \mathcal{T}_{\mathcal{S}}(p(\boldsymbol{t})), \text{ such that } \text{height}(u) \leq i$$

In the base case $i = 0$, the equivalence holds because both $\langle \mathfrak{s}(\boldsymbol{t}), \mathfrak{h} \rangle \in \mathfrak{T}_{\mathcal{S}}^0(p)$ is false and $\mathcal{T}_{\mathcal{S}}(p(\boldsymbol{t}))$ contains no unfolding tree of height 0. For the induction step $i \geq 1$, we consider two directions:

"⇒" If $\langle \mathfrak{s}(\boldsymbol{t}), \mathfrak{h} \rangle \in \mathfrak{T}_{\mathcal{S}}^i(p)$ then $\left(\mathfrak{s}, \mathfrak{h}, \mathfrak{T}_{\mathcal{S}}^{i-1}\right) \models \rho\sigma$, for some rule:

$$p(x_1, \ldots, x_{\#p}) \Leftarrow_{\mathcal{S}} \underbrace{\exists y_1 \ldots \exists y_n \,.\, \hat{\rho} * \mathop{\Large *}_{j=1}^{m} p_j(t_1^j, \ldots, t_{\#p_j}^j)}_{\rho}$$

and substitution σ such that $\sigma(x_k) = t_k$, for all $k \in [\![1 \,..\, \#p]\!]$. Then there exists locations $\ell_1, \ldots, \ell_n \in \mathsf{Loc}$ and heaps $\mathfrak{h}_0, \mathfrak{h}_1, \ldots, \mathfrak{h}_m$ such that $\mathfrak{h} = \biguplus_{j=0}^{m} \mathfrak{h}_j$ and the following hold:

- $(\mathfrak{s}', \mathfrak{h}_0) \models \hat{\rho}\sigma$, and
- $\left\langle \mathfrak{s}'(\boldsymbol{t}_j\sigma), \mathfrak{h}_j \right\rangle \in \mathfrak{T}_{\mathcal{S}}^{i-1}(p_j)$, for all $j \in [\![1 \,..\, m]\!]$,

where $\mathfrak{s}' \stackrel{\text{def}}{=} \mathfrak{s}[y_1 \leftarrow \ell_1, \ldots, y_n \leftarrow \ell_n]$ and $\boldsymbol{t}_j \stackrel{\text{def}}{=} (t_1^j, \ldots, t_{\#p_j}^j)$, for all $j \in [\![1 \,..\, m]\!]$. By the induction hypothesis, we obtain unfolding trees $u_j \in \mathcal{T}_{\mathcal{S}}\left(p_j(\boldsymbol{t}_j\sigma)\right)$, of height at most $i-1$, such that $(\mathfrak{s}', \mathfrak{h}_j) \models \mathcal{F}(u_j)$, for all $j \in [\![1 \,..\, m]\!]$. Note that, by Definition 3, the root of each u_j has a single child and let $u_{j|0}$ be the subtree of u_j rooted at the child of λ. Then we define an unfolding tree u, such that $u(\lambda) \stackrel{\text{def}}{=} (\bot, p(\boldsymbol{t}))$, $u(0) \stackrel{\text{def}}{=} (p, \rho\sigma)$ and $u_{|j} \stackrel{\text{def}}{=} u_{j|0}$, for all $j \in [\![1 \,..\, m]\!]$. It is straightforward to check that $u \in \mathcal{T}_{\mathcal{S}}(p(\boldsymbol{t}))$ and that the height of u is at most i. We are left with proving that $(\mathfrak{s}, \mathfrak{h}) \models \mathcal{F}(u)$, which follows from Fact 1 and the inductive hypothesis $(\mathfrak{s}', \mathfrak{h}_j) \models \mathcal{F}(u_j)$, for all $j \in [\![1 \,..\, m]\!]$.

"⇐" If $(\mathfrak{s}, \mathfrak{h}) \models \mathcal{F}(u)$ then, by Definition 3, the root of u has a single child 0 whose label is $u(0) = (p(\boldsymbol{t}), \rho\sigma)$, for a rule:

$$p(x_1, \ldots, x_{\#p}) \Leftarrow_{\mathcal{S}} \underbrace{\exists y_1 \ldots \exists y_n \,.\, \hat{\rho} * \mathop{\Large *}_{j=1}^{m} p_j(t_1^j, \ldots, t_{\#p_j}^j)}_{\rho}$$

and a substitution $\sigma(x_k) = t_k$, for all $k \in [\![1 \,..\, \#p]\!]$. By Fact 1, we obtain:

$$\mathcal{F}(u) \stackrel{\text{ac}}{=} \exists y_1 \ldots \exists y_n \,.\, \hat{\rho}\sigma * \circledast_{j=1}^{m} \mathcal{F}(u_j)$$

for some unfolding trees $u_j \in \mathcal{T}_{\mathcal{S}}\Big(p_j(t_1^j, \ldots, t_{\#p_j}^j)\Big)$, of height at most $i-1$, for $j \in [\![1 \,..\, m]\!]$. Since $(\mathfrak{s}, \mathfrak{h}) \models \mathcal{F}(u)$, there exists locations $\ell_1, \ldots, \ell_n \in \mathsf{Loc}$, such that:

- $(\mathfrak{s}', \mathfrak{h}_0) \models \hat{\rho}\sigma$ and
- $(\mathfrak{s}', \mathfrak{h}_j) \models \mathcal{F}(u_j)$, for all $j \in [\![1 \,..\, m]\!]$,

where $\mathfrak{s}' \stackrel{\text{def}}{=} \mathfrak{s}[y_1 \leftarrow \ell_1, \ldots, y_n \leftarrow \ell_n]$ and $\mathfrak{h} = \biguplus_{j=0}^{m} \mathfrak{h}_j$. By the inductive hypothesis, we obtain:

$$\Big\langle \mathfrak{s}'(\boldsymbol{t}_j\sigma), \mathfrak{h}_j \Big\rangle \in \mathfrak{T}_{\mathcal{S}}^{i-1}(p_j) \subseteq \mathfrak{T}_{\mathcal{S}}^{\mu}(p_j)$$

where $\boldsymbol{t}_j \stackrel{\text{def}}{=} (t_1^j, \ldots, t_{\#p}^j)$, for all $j \in [\![1 \,..\, m]\!]$. But then $(\mathfrak{s}', \mathfrak{h}_j) \models_{\mathcal{S}} p_j(\boldsymbol{t}_j\sigma)$, for all $j \in [\![1 \,..\, m]\!]$, thus $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} \rho\sigma$ and $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} p(\boldsymbol{t})$ follows. □

3 A Decidable Class of Entailments

In general, the entailment problem is undecidable, proofs can be found in [5,1]. Thus we consider a subclass of entailments for which decidability (with elementary recursive complexity) was proved in [4] and provide a 2EXPTIME lower bound for this problem. The decidable class is given by three restrictions of the systems used for the interpretation of predicates, namely *progress*, *connectivity* and *establishment*, defined in the rest of this section.

First, the *progress* condition requires that each rule adds to the heap exactly one location, namely the one associated to the first parameter of the head. Second, the *connectivity* condition requires that all locations added during an unfolding of a predicate atom $p(\boldsymbol{t})$ form a connected tree-like structure.

Definition 4 (Progress & Connectivity). *A set of rules $\mathcal{S}$ is* progressing *if and only if, the body ρ of each rule $p(x_1, \ldots, x_{\#p}) \Leftarrow_{\mathcal{S}} \rho$ is of the form $\exists z_1 \ldots \exists z_m \,.\, x_1 \mapsto (y_1, \ldots, y_k) * \psi$ and ψ contains no occurrence of a $\mapsto$ atom. If, moreover, each occurrence of a predicate atom in ψ is of the form $q(y_i, u_1, \ldots, u_{\#q-1})$, for some $i \in [\![1 \,..\, k]\!]$, then $\mathcal{S}$ is* connected.

For upcoming developments, we make the connection between heaps and unfolding trees more precise, in the context of progressing connected sets of rules.

Lemma 2. *Assume that $\mathcal{S}$ is a progressing and connected set of rules and that $(\mathfrak{s}, \mathfrak{h})$ is a structure such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} p(t_1, \ldots, t_{\#p})$, for some predicate symbol $p \in \mathsf{Pred}$ and terms $t_1, \ldots, t_{\#p} \in \mathsf{Term}$. Then there exists an unfolding tree $u \in \mathcal{T}_{\mathcal{S}}\Big(p(t_1, \ldots, t_{\#p})\Big)$ and a bijection $\Lambda : \mathrm{dom}(\mathfrak{h}) \to \mathrm{nodes}(u) \setminus \{\lambda\}$ such that $(\mathfrak{s}, \mathfrak{h}) \models \mathcal{F}(u)$ and for each node $wi \in \mathrm{nodes}(u) \setminus \{\lambda\}$, where $i \in \mathbb{N}$, we have $\Lambda^{-1}(wi) \in \mathfrak{h}(\Lambda^{-1}(w))$.*

Proof: If $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} p(t_1, \ldots, t_{\#p})$, by Lemma 1, there exists $u \in \mathcal{T}_{\mathcal{S}}\Big(p(t_1, \ldots, t_{\#p})\Big)$, such that $(\mathfrak{s}, \mathfrak{h}) \models \mathcal{F}(u)$. Note that, by Definition 3, the root of u is labeled by $u(\lambda) = (\bot, p(t_1, \ldots, t_{\#p}))$

and has a single child, namely 0. The bijection Λ is built inductively on the structure of the subtree $u_{|0}$ of u, rooted at 0, taking into account Definition 4. □

Given a structure $(\mathfrak{s}, \mathfrak{h})$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} p(t_1, \ldots, t_{\#p})$, for any location $\ell \in \mathrm{dom}(\mathfrak{h})$, necessarily ℓ must be allocated by a 1-unfolding of some atom $q(\boldsymbol{t})$, which corresponds to a node in an unfolding tree $\Lambda(\ell) \in \mathcal{T}_{\mathcal{S}}\Big(p(t_1, \ldots, t_{\#p})\Big)$. In the following, we shall denote by $\mathsf{Pr}_{\mathfrak{s},\mathfrak{h},\Lambda}(\ell)$ the predicate symbol q. Note that the existence of the unfolding tree u and bijection $\Lambda : \mathrm{dom}(\mathfrak{h}) \rightarrow \mathrm{nodes}(u) \setminus \{\lambda\}$ are guaranteed by Lemmas 1 and 2, respectively. The notation $\mathsf{Pr}(\ell)$ stands for $\mathsf{Pr}_{\mathfrak{s},\mathfrak{h},\Lambda}(\ell)$, for some Λ as in Lemma 2, where the structure $(\mathfrak{s}, \mathfrak{h})$ is clear from the context.

The third condition requires that all the existentially quantified variables introduced during a can only be associated with locations from the heap, in every ground unfolding of a formula. To formalize this condition, for a ground formula ϕ, we define the *partial satisfaction* relation as $(\mathfrak{s}, \mathfrak{h}) \Vdash \phi$ iff $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$ and $(\mathfrak{s}, \mathfrak{h}_1) \models \phi$ and partial entailment as $\phi \Vdash \psi$ iff $(\mathfrak{s}, \mathfrak{h}) \Vdash \psi$ for each structure $(\mathfrak{s}, \mathfrak{h})$, such that $(\mathfrak{s}, \mathfrak{h}) \models \phi$. If ϕ is ground, its set of allocated variables is $\mathsf{alloc}(\phi) \stackrel{\text{def}}{=} \{x \in \mathsf{fv}(\phi) \mid x' \mapsto (y_1, \ldots, y_k) \text{ occurs in } \phi \text{ and } \phi \Vdash x \doteq x'\}$. Note that $\mathsf{alloc}(\phi) = \mathsf{fv}(\phi)$ if ϕ is unsatisfiable. Extending this notion to formulæ that are not necessarily ground, we define $\mathsf{alloc}_{\mathcal{S}}(\phi) \stackrel{\text{def}}{=} \bigcap_{u \in \mathcal{T}_{\mathcal{S}}(\phi)} \mathsf{alloc}(\mathcal{F}(u))$, i.e. a variable is allocated in ϕ iff it is allocated in every ground unfolding of ϕ w.r.t. a set of rules $\mathcal{S}$.

Definition 5 (Establishment). *A set of rules $\mathcal{S}$ is* established *if and only if, for each rule $(p(\boldsymbol{x}_p), \exists z_1 \ldots \exists z_m \,.\, \psi) \in \mathcal{S}$, where ψ is quantifier-free, we have $z_1, \ldots, z_m \in \mathsf{alloc}_{\mathcal{S}}(\psi)$.*

In the following, we consider only sets of rules that are progressing, connected and established (PCE). The interest for PCE sets of rules is motivated by the following decidability result, proved in [4]:

Theorem 1 (Decidability). *Given a PCE set of rules $\mathcal{S}$ and two formulæ ϕ and ψ such that the free variables of ψ are also free in ϕ, the problem $\phi \models_{\mathcal{S}} \psi$ belongs to* ELEMENTARY.

The rest of this paper is concerned with proving that the entailment problem $\phi \models_{\mathcal{S}} \psi$, for PCE sets of rules $\mathcal{S}$, is 2EXPTIME-hard. Previously, a EXPTIME-hard bound for this problem was established in [5].

3.1 Syntactic Shorthands

Before proving 2EXPTIME-hardness by reduction from the membership problem of alternating Turing machines running in exponential space, we define several syntactical shorthands that simplify the presentation.

Given a term t and $n \geq 0$, we denote by t^n the tuple consisting of n occurrences of t. In the rest of the paper we silently assume that any heap entry of the form $\mathfrak{h}(\ell) = (\ell_1, \ldots, \ell_{\mathsf{k}})$ is represented by a binary heap $\mathfrak{h}_2 : \mathsf{Loc} \rightharpoonup_{fin} \mathsf{Loc}^2$ such that $\mathrm{dom}(\mathfrak{h}_2) = \{\ell\} \cup \left\{\ell_0^i \mid i \in [\![1 \,..\, \mathsf{k}-1]\!]\right\}$, $\mathfrak{h}_2(\ell) = (\ell_0^1, \ell_1)$ and $\mathfrak{h}_2(\ell_0^i) = (\ell_0^{i+1}, \ell_{i+1})$, for all $i \in [\![1 \,..\, \mathsf{k}-1]\!]$, with $\ell_0^{\mathsf{k}} = \mathsf{nil}$. This allows one to encode records of non constant length (for instance tuples of length $\mathfrak{N}$) by using only a constant number of record fields (i.e., $\mathsf{k} = 2$).

Deep Hats For a tuple $\boldsymbol{t} = (t_1, \ldots, t_j) \in \mathsf{Term}^j$, $j \geq 1$ and $n \geq 0$, by writing:

$$p(x_1, \ldots, x_{\#p}) \Leftarrow \exists y_1 \ldots \exists y_r \,.\, x_1 \mapsto (u_1, \ldots, u_{i-1}, [\boldsymbol{t}]^n, u_{i+1}, \ldots, u_m) * \psi$$

we denote the rules:

$$p(x_1, \ldots, x_{\#p}) \Leftarrow \exists y_1 \ldots \exists y_r \,.\, x_1 \mapsto (u_1, \ldots, u_{i-1}, t_1, \ldots, t_j, u_{i+1}, \ldots, u_m) * \psi, \text{ if } n = 0$$

and, recursively, for all $n > 0$:

$$\begin{array}{l} p(x_1, \ldots, x_{\#p}) \Leftarrow \exists y_1 \ldots \exists y_r \exists z \,.\, x_1 \mapsto (u_1, \ldots, u_{i-1}, z, u_{i+1}, \ldots, u_m) * \widetilde{p}_{n-1}(z, x_1, \ldots, x_{\#p}, y_1, \ldots, y_r) * \psi' \\ \widetilde{p}_\ell(z_1, \ldots, z_{\#p+r+1}) \Leftarrow \exists z'_1 \,.\, z_1 \mapsto (z'_1) * \widetilde{p}_{\ell-1}(z'_1, z_2, \ldots, z_{\#p+r+1}), \text{ for } \ell \in [\![1 \,..\, n-1]\!] \\ \widetilde{p}_0(z_1, \ldots, z_{\#p+r+1}) \Leftarrow z_1 \mapsto (t_1, \ldots, t_j) * \psi'' \end{array}$$

where $\widetilde{p}_\ell$, for $\ell \in [\![0 \,..\, n-1]\!]$ are fresh predicate symbols of arity $\#\widetilde{p}_\ell = \#p + r + 1$ and ψ'' (resp. ψ') is the separating conjunction of all atoms of ψ whose first argument is in $\boldsymbol{t}$ (resp. is not in $\boldsymbol{t}$). Clearly, $\psi' * \psi'' \stackrel{\mathsf{ac}}{=} \psi$, and the obtained rules are connected.

Special Variables We assume the existence of the following special variables that occur free in each formula: $\mathbf{0}, \mathbf{1}, \gamma_1, \ldots, \gamma_N$, for some fixed constant $N \geq 2$, used throughout the paper. These variables will always be allocated and denote separate locations, as required by the following rule:

$$\mathsf{Const}(x) \Leftarrow x \mapsto (\mathbf{0}, \mathbf{1}, \gamma_1, \ldots, \gamma_N) * \mathbf{0} \mapsto (\mathsf{nil}) * \mathbf{1} \mapsto (\mathsf{nil}) * \circledast_{i=1}^{N} \gamma_i \mapsto (\mathsf{nil}) \qquad (1)$$

Considering special variables is without loss of generality in the following, because these variables can be added to the parameter list of each head in the system, at the expense of cluttering the presentation.

Binary Choices The symbol $\bullet$ occurring in the body of a rule ranges over the special variables 0 and 1. Thus any rule of the form:

$$p(x_1, \ldots, x_{\#p}) \Leftarrow \exists z_1 \ldots \exists z_n \,.\, x_1 \mapsto (y_1, \ldots, y_{i-1}, \bullet, y_{i+1}, \ldots, y_m) * \psi$$

stands for the following two rules:

$$\begin{array}{l} p(x_1, \ldots, x_{\#p}) \Leftarrow \exists z_1 \ldots \exists z_n \,.\, x_1 \mapsto (y_1, \ldots, y_{i-1}, \mathbf{0}, y_{i+1}, \ldots, y_m) * \psi \\ p(x_1, \ldots, x_{\#p}) \Leftarrow \exists z_1 \ldots \exists z_n \,.\, x_1 \mapsto (y_1, \ldots, y_{i-1}, \mathbf{1}, y_{i+1}, \ldots, y_m) * \psi \end{array}$$

Note that eliminating the occurrences of $\bullet$ will increase the number of rules in $\mathcal{S}$ by a constant at most 2^{k}, because k is assumed to be constant (i.e. independent of the input of a decision procedure).

Binary Variables A binary variable b is understood as ranging over the domain of the interpretation of $\mathbf{0}$ and $\mathbf{1}$, namely the locations assigned to $\mathbf{0}$ and $\mathbf{1}$ by the formula Const (1). Additionally, for each binary variable b, we consider the associated variable

$\overline{b}$, intended to denote the complement of b. More precisely, the formula $\exists b \,.\, \psi$ is understood as $\psi[\mathbf{0}/b, \mathbf{1}/\overline{b}] \vee \psi[\mathbf{1}/b, \mathbf{0}/\overline{b}]$. However, this direct substitution of the (existentially quantified) binary variables by $\mathbf{0}$ and $\mathbf{1}$ within the rules of an established system would break the establishment condition (Definition 5), because $\mathbf{0}$ and $\mathbf{1}$ are not necessarily allocated within the body of the rule[4]. This problem can be overcome by passing $\mathbf{0}$ and $\mathbf{1}$ as parameters to a fresh predicate. More precisely, a rule of the form:

$$p(x_1, \ldots, x_{\#p}) \Leftarrow \exists b_1 \ldots \exists b_i \exists y_1 \ldots \exists y_n \,.\, x_1 \mapsto ([\boldsymbol{t}]^m) * \psi \tag{2}$$

where $1 \leq i \leq m$, is a shorthand for the following set of rules:

$$\begin{array}{rl} p(x_1, \ldots, x_{\#p}) & \Leftarrow \exists y \,.\, x_1 \mapsto (y) * p'(y, x_1, \ldots, x_{\#p}, \mathbf{0}, \mathbf{1}) \\ p(x_1, \ldots, x_{\#p}) & \Leftarrow \exists y \,.\, x_1 \mapsto (y) * p'(y, x_1, \ldots, x_{\#p}, \mathbf{1}, \mathbf{0}) \\ p'(y, x_1, \ldots, x_{\#p}, b_1, \overline{b}_1) & \Leftarrow \exists b_2 \ldots \exists b_i \exists y_1 \ldots \exists y_n \,.\, y \mapsto ([\boldsymbol{t}]^{m-1}) * \psi \end{array}$$

Clearly, the elimination of the binary existential quantifiers from the rule (2) will add $2i$ rules to the set. Note that the hat $[\boldsymbol{t}]^m$, of height $m \geq i$ decreases at each step of the elimination. This guarantees that rules resulting from the elimination of $\exists b_1, \ldots, \exists b_i$ respectively, are progressing (Definition 4).

Next, for a vector $\boldsymbol{b} = (b_1, \ldots, b_n)$ of binary variables, we denote by $\overline{\boldsymbol{b}}$ the vector $(\overline{b}_1, \ldots, \overline{b}_n)$. The following rule:

$$p(x_1, \ldots, x_{\#p}) \Leftarrow \exists y_1 \ldots \exists y_m \,.\, x_1 \mapsto \boldsymbol{t} * \psi \mid (b_1, \ldots, b_n) \not\approx \overline{(c_1, \ldots, c_n)} \tag{3}$$

where $b_1, \ldots, b_n \in \{x_2, \ldots, x_{\#p}, y_1, \ldots, y_m\}$ occur only once in $\boldsymbol{t}$ and do not occur in ψ and $c_1, \ldots, c_n \in \{x_2, \ldots, x_{\#p}, y_1, \ldots, y_m\}$, is a shorthand for the following set of rules:

$$p(x_1, \ldots, x_{\#p}) \Leftarrow \exists y_1 \ldots \exists y_m \,.\, x_1 \mapsto (\boldsymbol{t}[c_i/b_i])\,[\bullet/b_j]_{j \in [\![1..n]\!] \setminus \{i\}} * \psi,\ i \in [\![1 \,..\, n]\!] \tag{4}$$

Intuitively, the rule (3) introduces new binary variables $b_1, \ldots, b_n$, such that not all of them are equal to the complement of $c_1, \ldots, c_n$, respectively. In other words, at least one b_i must be equal to c_i, for some $i \in [\![1 \,..\, n]\!]$. Note that expanding the rule (3) as described above results in at most n rules of the form (2), hence the full elimination of binary variables from the system is possible in polynomial time.

4 Alternating Turing Machines

An *Alternating Turing Machine* (ATM) is a tuple $M = (Q, \Gamma, \delta, q_0, g)$ where:

- Q is a finite set of control *states*,
- $\Gamma = \{\gamma_1, \ldots, \gamma_N, \mathsf{b}\}$ is a finite alphabet, b is the *blank* symbol and each symbol from $\Gamma \setminus \{\mathsf{b}\}$ is named after a special variable,
- $\delta \subseteq Q \times \Gamma \times Q \times (\Gamma \setminus \{\mathsf{b}\}) \times \{\leftarrow, \rightarrow\}$ is the *transition relation*, $(q, a, q', b, \mu) \in \delta$ meaning that, in state q, upon reading symbol a, the machine moves to state q', writes $b \neq \mathsf{b}$ to the tape[5] and moves the head by one to the left (resp. right) if $\mu = \leftarrow$ (resp. $\mu = \rightarrow$),

[4] In fact they are allocated by the side condition Const.

[5] A machine never writes blank symbols, that are used only for the initially empty tape cells.

– $q_0 \in Q$ is the *initial state*, and
– $g : Q \to \{\vee, \wedge\}$ partitions the set of states into *existential* ($g(q) = \vee$) and *universal* ($g(q) = \wedge$) states.

A *configuration* of M is a tuple (q, w, i) where $q \in Q$ is the current state, $w : \mathbb{N} \to \Gamma$ is (the contents of) the *tape*, such that $\|\{j \in \mathbb{N} \mid w(j) \neq \mathsf{b}\}\| < \infty$ and $i \in [\![0 \,..\, \max\{j \in \mathbb{N} \mid w(j) \neq \mathsf{b}\} + 1]\!]$ is the current position of the head on the tape. We denote by ϵ the empty word over Γ. For any tape w and integer i, we denote by $w[i \leftarrow a]$ the tape w' such that $w'(i) = a$ and $w'(j) = w(j)$ for all $j \neq i$. In the following, we write $w_i \stackrel{\text{def}}{=} w(i)$, $i^{\leftarrow} \stackrel{\text{def}}{=} i - 1$ and $i^{\rightarrow} \stackrel{\text{def}}{=} i + 1$.

The step relation of M is the following relation between configurations: we write $(q, w, i) \xrightarrow{(q,a,q',b,\mu)} (q', w', j)$ if and only if there exists a transition $(q, a, q', b, \mu) \in \delta$ such that $w_i = a$, $w' = w[i \leftarrow b]$ and $j = i^{\mu}$. We write $(q, w, i) \to (q', w', j)$ when the applied transition is not important. An *execution* is a sequence $(q_0, w_0, 0) \xrightarrow{(q_0,a_0,q_1,b_0,\mu_0)} (q_1, w_1, i_1) \xrightarrow{(q_1,a_1,q_2,b_1,\mu_1)} \dots$ Note that an execution is entirely determined by the initial configuration $(q_0, w_0, 0)$ and the sequence $(q_0, a_0, q_1, b_0, \mu_0), (q_1, a_1, q_2, b_1, \mu_1), \dots$ of transition rules applied to it.

Given a function $f : \mathbb{N} \to \mathbb{N}$, an execution is *$f$-space bounded* if and only if there exists a constant $c > 0$ such that $|w_i| \leq c \cdot f(|w_0|)$, for all $i > 0$. The ATM M is *exponential-space bounded* if it admits only f-space bounded executions, where $f(x) = 2^{g(x)}$ and g is a univariate polynomial function.

Definition 6. *A* derivation *of an ATM $M = (Q, \Gamma, \delta, q_0, g)$, starting from a configuration $(q_0, w_0, 0)$, is a tree t, whose nodes are either:*

1. branching nodes *labeled with configurations $(q, w, i) \in Q \times \Gamma^* \times \mathbb{N}$, or*
2. action nodes *labeled with tuples $(a, b, \mu) \in \Gamma \times \Gamma \setminus \{\mathsf{b}\} \times \{\leftarrow, \rightarrow\}$, where a is the symbol read, b is the symbol written and μ is the move of the head at that step,*

such that the root of t is a branching node, $t(\lambda) = (q_0, w_0, 0)$ and, moreover:

a. each branching node labeled by (q, w, i), such that $g(q) = \vee$, has exactly one successor, that is an action node labeled by (a, b, μ), where $(q, a, q', b, \mu) \in \delta$, whose successor is a branching node labeled by (q', w', j) such that $(q, w, i) \xrightarrow{(q,a,q',b,\mu)} (q', w', j)$,
b. each branching node labeled by (q, w, i), such that $g(q) = \wedge$ has exactly one successor for each tuple $(q, a, q', b, \mu) \in \delta$ such that $a = w(i)$, and the successor associated with a transition (q, a, q', b, μ) is a branching node, labeled by (q', w', j) with $(q, w, i) \xrightarrow{(q,a,q',b,\mu)} (q', w', j)$.

We say that M accepts *(q_0, w_0, i_0) iff it has a derivation starting from (q_0, w_0, i_0).*

Definition 7. *The* membership problem *(M, w) asks the following: given an ATM $M = (Q, \Gamma, \delta, q_0, g)$ and an input word $w \in (\Gamma \setminus \{\mathsf{b}\})^*$ does M accept $(q_0, w, 0)$?*

The complexity class AEXPSPACE is the class of membership problems where M is exponential-space bounded. It is known that AEXPSPACE=co–AEXPSPACE=2EXPTIME [3], where co–AEXPSPACE is the complement class of AEXPSPACE[6]. In the following, we shall w.l.o.g. consider only the membership problem (M, ϵ). Indeed, let (M, w)

[6] Every ATM can be complemented in linear time, by interchanging the existential with the universal states, thus all alternating classes are closed under complement.

be any instance of the membership problem, and let c and g be the constant and polynomial function witnessing the fact that M is exponential-space bounded. Let M_w be the ATM that produces w starting from input ϵ. Clearly, M_w uses at most $|w|$ working space, thus the machine $M_w; M$ which runs M_w on the empty word and then continues with M runs in space $c \cdot 2^{g(|w|)}$ and accepts $(q_0, \epsilon, 0)$ if and only if M accepts $(q_0, w, 0)$. If $\mathfrak{N} \geq \log_2(c) + g(w)$, then $M_w; M$ runs in space $2^{\mathfrak{N}}$. Therefore, we assume from now on that $M = (Q, \Gamma, \delta, q_0, g)$ is an ATM started in the configuration $(q_0, \epsilon, 0)$ and that M runs in space at most $2^{\mathfrak{N}}$ on $(q_0, \epsilon, 0)$, where $\mathfrak{N}$ is polynomial w.r.t. the length of w.

4.1 Pseudo-derivations as Heaps

Given an ATM $M = (Q, \Gamma, \delta, q_0, g)$, we consider its *pseudo-derivations*, which are the trees of Definition 6 relaxed so that any sequence of three consecutive branching-action-branching node labels (q, w, i), (a, b, μ) and (q', w', i') only needs to ensure the existence of a transition $(q, a, q', b, \mu) \in \delta$, i.e. we drop the requirements $w_i = a$, $w' = w[i \leftarrow b]$ and $i' = i^{\mu}$ from the condition $(q, w, i) \xrightarrow{(q,a,q',b,\mu)} (q', w', i')$ in Definition 6. Clearly, the leaves of a pseudo-derivation are all labeled by universal states.

We represent the pseudo-derivations of M as tree-shaped heaps generated by a set of rules where, intuitively, each predicate $q(x)$ allocates a branching node labeled by a configuration (q, w, i), for some $w \in \Gamma^*$ and some $i \in [\![0 \,..\, 2^{\mathfrak{N}} - 1]\!]$, and each predicate $\overline{q}(x, a, b, \mu)$ allocates an action node labeled (a, b, μ). Importantly, since M starts on the empty word ϵ, the tape contents in a branching node can be derived from the sequence of actions along the path from the root to that node. For this reason, we shall not explicitly represent tape contents within the configurations and label branching nodes with pairs $(q, i) \in Q \times [\![0 \,..\, 2^{\mathfrak{N}} - 1]\!]$.

We represent each position $i \in [\![0 \,..\, 2^{\mathfrak{N}} - 1]\!]$ on the tape succintly, by an $\mathfrak{N}$-tuple of binary digits $\mathrm{bin}(i) \in \{\mathbf{0}, \mathbf{1}\}^{\mathfrak{N}}$ and encode the left and right moves as $\overset{\sim}{\leftarrow} \overset{\text{def}}{=} \mathbf{0}$ and $\overset{\sim}{\rightarrow} \overset{\text{def}}{=} \mathbf{1}$. Let $\tau(q, a) \overset{\text{def}}{=} \delta \cap (\{q\} \times \{a\} \times Q \times \Gamma \setminus \{\mathsf{b}\} \times \{\leftarrow, \rightarrow\})$ be the set of transitions of M with source state q, reading symbol a from the tape.

As previously mentioned, we simplify the presentation by considering atoms of the form $x \mapsto (y_1, \ldots, y_m)$ for an arbitrary $m \geq 1$, with the understanding that the heap corresponding to this atom is uniquely encoded by a binary heap. This is without loss of generality, because a rule containing such atoms can be transformed into a finite set of rules containing only atoms of the form $x \mapsto (y_1, y_2)$, resulting in a progressing, connected and established set of rules. With this in mind, we consider the following rules, for each state $q \in Q$:

$$q(x) \Leftarrow \exists x' \,.\, x \mapsto (\bullet^{\mathfrak{N}}, x') * \overline{q}'(x', a, b, \widetilde{\mu}) \quad \text{if } g(q) = \vee \text{ and } (q, a, q', b, \mu) \in \tau(q, a) \tag{5}$$

$$q(x) \Leftarrow \exists y_1 \ldots \exists y_n \,.\, x \mapsto (\bullet^{\mathfrak{N}}, y_1, \ldots, y_m) * \overset{m}{\underset{j=1}{\circledast}} \overline{q}_j(y_j, a, b_j, \widetilde{\mu}_j) \quad \text{if } g(q) = \wedge \text{ and } \tau(q, a) = \{(q, a, q_1, b_1, \mu_1), \ldots, (q, a, q_m, b_m, \mu_m)\} \tag{6}$$

$$\overline{q}(x, y, z, u) \Leftarrow \exists x' \,.\, x \mapsto (y, z, u, x') * q(x') \tag{7}$$

The heaps defined by the above rules ensure only that the control structure of a derivation of M is respected, namely that the branching and action nodes alternate correctly,

and that the sequence of control states labeling the branching nodes on any path is consistent with the transition relation of M. In other words, these trees encode pseudo-derivations of M. Further, we introduce a top-level predicate $p_M(x)$ that allocates the special variables $\mathbf{0}, \mathbf{1}, \gamma_1, \ldots, \gamma_N$ and ensures that the initial state q_0 of M is the first control state that occurs on an path of a pseudo-derivation:

$$p_M(x) \Leftarrow \exists y_0 \exists z_0 \ . \ x \mapsto ([y_0]^{\mathfrak{N}}, z_0) * q_0(y_0) * \mathsf{Const}(z_0) \tag{8}$$

Note that the hat $[y_0]^{\mathfrak{N}}$ above ensures that every heap in the least fixed point interpretation of p begins with a tree of height $\mathfrak{N}$. The use of this tree will be made clear below. For now, let $\mathcal{S}$ be the set consisting of the rules above. We formalize the encoding of a pseudo-derivation by a structure:

Definition 8. *A structure* $(\mathfrak{s}, \mathfrak{h})$ *such that* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} p_M(x)$ encodes *a pseudo-derivation* t *of* M*, written as* $(\mathfrak{s}, \mathfrak{h}) \rhd t$*, if and only if there exists a injective mapping* $f : \mathrm{nodes}(t) \rightarrow \mathrm{dom}(\mathfrak{h}) \setminus \mathfrak{s}(\Gamma \cup \{\mathbf{0}, \mathbf{1}\})$ *such that, for all* $w \in \mathrm{nodes}(t)$*, the following hold:*

1. *for all* $i \in \mathbb{N}$*, if* $wi \in \mathrm{nodes}(t)$ *then* $f(wi) \in \mathfrak{h}(f(w))$*,*
2. *if* w *is a branching node and* $t(w) = (q, i)$ *then* $\mathsf{Pr}(f(w)) = q$ *and* $\mathfrak{h}(f(w)) = (\ell_1, \ldots, \ell_{\mathfrak{N}+1})$*, where* $\ell_j = \mathfrak{s}(\mathrm{bin}(i)_j)$*, for all* $j \in [\![1 \,..\, \mathfrak{N}]\!]$ *and* $\ell_{\mathfrak{N}+1} \in \mathrm{dom}(\mathfrak{h})$*,*
3. *if* w *is an action node and* $t(w) = (a, b, \mu)$ *then* $\mathfrak{h}(f(w)) = (\mathfrak{s}(a), \mathfrak{s}(b), \mathfrak{s}(\widetilde{\mu}), \ell_4)$*, where* $\ell_4 \in \mathrm{dom}(\mathfrak{h})$*,*

We write $(\mathfrak{s}, \mathfrak{h}) > t$ *instead of* $(\mathfrak{s}, \mathfrak{h}) \rhd t$ *if condition (2) above is replaced by the weaker:*

2'. *if* w *is a branching node and* $t(w) = (q, i)$ *then* $\mathfrak{h}(f(w)) = (\ell_1, \ldots, \ell_{\mathfrak{N}+1})$*, where* $\ell_j = \mathfrak{s}(\mathrm{bin}(i)_j)$*, for all* $j \in [\![1 \,..\, \mathfrak{N}]\!]$ *and* $\ell_{\mathfrak{N}+1} \in \mathrm{dom}(\mathfrak{h})$*.*

Note that, by Lemma 1 $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} p_M(x)$ iff there exists an unfolding tree $u \in \mathcal{T}_{\mathcal{S}}(p_M(x))$ such that $(\mathfrak{s}, \mathfrak{h}) \models \mathcal{F}(u)$ and, by Lemma 2, there exists a bijection $\Lambda : \mathrm{dom}(\mathfrak{h}) \rightarrow \mathrm{nodes}(u) \setminus \{\lambda\}$, such that $\Lambda^{-1}(wi) \in \mathfrak{h}(\Lambda^{-1}(w))$, for all $wi \in \mathrm{nodes}(u)$. Then the occurrence of $\mathsf{Pr}(.)$ at point (2) of Definition 8 stands for $\mathsf{Pr}_{\mathfrak{s},\mathfrak{h},\Lambda}(.)$, the structure $(\mathfrak{s}, \mathfrak{h})$ being clear from the context and the existence of Λ being guaranteed by Lemma 2. The weaker relation $(\mathfrak{s}, \mathfrak{h}) > t$ shall be used next to encode pseudo-derivations by structures that are not necessarily models of the above rules (more rules will be added later).

Lemma 3. *(A) For each pseudo-derivation* t *of an ATM* $M = (Q, \Gamma, \delta, q_0, g)$*, there exists a structure* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} p_M(x)$ *such that* $(\mathfrak{s}, \mathfrak{h}) \rhd t$*. (B) Dually, for each structure* $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} p_M(x)$*, there exists an accepting pseudo-derivation* t *of* M *such that* $(\mathfrak{s}, \mathfrak{h}) \rhd t$*.*

Proof: (A) Let t be a pseudo-derivation of M. We build an isomorphic tree u, such that $\mathrm{nodes}(t) = \mathrm{nodes}(u)$ and the labels of u are of the form (q, ϕ), where $q \in \mathsf{Pred}$ and ϕ is a formula. The definition of u is top-down on the structure of t, for each node $w \in \mathrm{nodes}(t)$, with label $t(w) = (q, i)$:

– if $g(q) = \vee$, where $t(w0) = (a, b, \mu)$ and $t(w00) = (q', i')$ (we have necessarily $w0, w00 \in \mathrm{nodes}(t)$, where $w0$ and $w00$ are the only children of w and $w0$, respectively), then we define:

$$u(w) \stackrel{\mathsf{def}}{=} (q, \ \exists x' \ . \ x \mapsto (\mathrm{bin}(i), x') * \overline{q}(x', a, b, \widetilde{\mu})) \quad \text{(see rule 5)}$$
$$u(w0) \stackrel{\mathsf{def}}{=} (\overline{q}, \ \exists x'' \ . \ x' \mapsto (a, b, \widetilde{\mu}, x'') * q(x'')) \quad \text{(see rule 7)}$$

– otherwise, $g(q) = \wedge$ and $t(wj) = (a_j, b_j, \mu_j)$ for $j \in [\![1 \,..\, n]\!]$ are the children of w, and $t(wj0) = (q'_j, i'_j)$, for all $j \in [\![1 \,..\, n]\!]$, in which case we define:

$$u(w) \stackrel{\text{def}}{=} \Big(q,\ \exists y_1 \ldots \exists y_n \,.\, x \mapsto (\mathrm{bin}(i), y_1, \ldots, y_n) * \mathop{*}_{j=1}^{n} \overline{q}_j(y_j, a_j, b_j, \widetilde{\mu}_j)\Big) \quad \text{(see rule 6)}$$
$$u(wj) \stackrel{\text{def}}{=} \Big(\overline{q}_j,\ \exists x' \,.\, y_j \mapsto (a_j, b_j, \widetilde{\mu}_j, x') * q'_j(x')\Big) \quad \text{(see rule 7)}$$

Finally, either one of the following holds:

– $w = vi$, for some $v \in \mathrm{nodes}(t)$, $i \in \mathbb{N}$ and the atom $q(x)$ occurs exactly once in $u(v)$,
– $w = \lambda$ and the variable x in the definition of $u(w)$ is the same as y_0, in which case necessarily $t(\lambda) = (q_0, 0)$.

Next, we extend u to an unfolding tree $\hat{u} \in \mathcal{T}_{\mathcal{S}}(p_M(x))$ by adding to it a hat of height $\mathfrak{R}$ and a sibling tree $u' \in \mathcal{T}_{\mathcal{S}}(\mathsf{Const}(z_0))$. It is not hard to check that $\hat{u} \in \mathcal{T}_{\mathcal{S}}(p_M(x))$ and that $\mathcal{F}(\hat{u})$ is satisfiable, since there are no equality or disequality atoms and no variable occurs allocated in two different subtrees of $\hat{u}$. Then, let $(\mathfrak{s}, \mathfrak{h})$ be a structure such that $(\mathfrak{s}, \mathfrak{h}) \models \mathcal{F}(\hat{u})$. To check that $(\mathfrak{s}, \mathfrak{h}) \rhd t$, we need to exhibit an injective mapping $f : \mathrm{nodes}(t) \to \mathrm{dom}(\mathfrak{h})$ that meets the conditions (1), (2) and (3) from Definition 8. Because $\mathcal{S}$ is a progressing and connected set of rules and $(\mathfrak{s}, \mathfrak{h}) \models \mathcal{F}(\hat{u})$, by Lemma 2, there exists a bijective mapping $\Lambda : \mathrm{dom}(\mathfrak{h}) \to \mathrm{nodes}(\hat{u}) \setminus \{\lambda\}$ such that $\Lambda^{-1}(wi) \in \mathfrak{h}(\Lambda^{-1}(w))$, for all $w \in \mathrm{nodes}(\hat{u}) \setminus \{\lambda\}$ and all $i \in \mathbb{N}$, such that $wi \in \mathrm{nodes}(\hat{u})$. Let f be the restriction of Λ^{-1} to $\mathrm{nodes}(u)$. Point (1) follows by the definition of Λ, whereas points (2) and (3) are simple checks.

(B) Conversely, if $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} p_M(x)$ then, by Lemma 1, there exists an unfolding tree $u \in \mathcal{T}_{\mathcal{S}}(p_M(x))$ such that $(\mathfrak{s}, \mathfrak{h}) \models \mathcal{F}(u)$. Since $\mathcal{S}$ is progressing and connected, by Lemma 2, there exists a bijective mapping $\Lambda : \mathrm{dom}(\mathfrak{h}) \to \mathrm{nodes}(u) \setminus \{\lambda\}$, as before. By the definition of $\mathcal{S}$, all nodes $w \in \mathrm{nodes}(u) \setminus \{\lambda\}$ that are labeled with predicates q and $\overline{q}$, for some $q \in Q$, occur below a unique node $w_0 \in \mathrm{nodes}(u)$, such that $t(w_0) = (q_0, \phi)$, for some formula ϕ. We build a pseudo-derivation t of M such that $\mathrm{nodes}(t) = \mathrm{nodes}(u_{|w_0})$, by induction on the structure of $u_{|w_0}$. Namely, for each $w \in \mathrm{nodes}(u_{|w_0})$:

– If $u_{|w_0}(w) = (q, \phi)$ then ϕ is the body of a rule (5) or (6). In the case (5) (the other case is similar and left to the reader) we have $\mathfrak{h}(\Lambda^{-1}(w)) = (\ell_1, \ldots, \ell_{\mathfrak{R}+1})$, with $\ell_1, \ldots, \ell_{\mathfrak{R}} \in \{\mathfrak{s}(\mathbf{0}), \mathfrak{s}(\mathbf{1})\}$ and let i be the integer such that $\mathrm{bin}(i) = (\mathfrak{s}^{-1}(\ell_1), \ldots, \mathfrak{s}^{-1}(\ell_{\mathfrak{R}}))$. Note that, since $\mathbf{0}$ and $\mathbf{1}$ are always allocated separately in $\mathcal{F}(u)$, the restriction of $\mathfrak{s}$ to the set $\{\mathbf{0}, \mathbf{1}\}$ is a bijection. In this case, we define $t(w) \stackrel{\text{def}}{=} (q, i)$.
– Otherwise, $u_{|w_0}(w) = (\overline{q}, \phi)$ and ϕ is the body of a rule (7). In this case, we have $\mathfrak{h}(\Lambda^{-1}(w)) = (\ell_1, \ell_2, \ell_3, \ell_4)$, with $\ell_1 \in \mathfrak{s}(\Gamma)$, $\ell_2 \in \mathfrak{s}(\Gamma \setminus \{\mathsf{b}\})$ and $\ell_3 \in \{\mathfrak{s}(\mathbf{0}), \mathfrak{s}(\mathbf{1})\}$. Note that, since each $\gamma \in \Gamma \cup \{\mathbf{0}, \mathbf{1}\}$ is allocated separately in $\mathcal{F}(u)$, the restriction of $\mathfrak{s}$ to the set $\Gamma \cup \{\mathbf{0}, \mathbf{1}\}$ is a bijection. In this case, we define $t(w) \stackrel{\text{def}}{=} (\mathfrak{s}^{-1}(\ell_1), \mathfrak{s}^{-1}(\ell_2), \mu)$, where $\mu = \leftarrow$ if $\ell_3 = \mathfrak{s}(\mathbf{0})$ and $\mu = \rightarrow$ if $\ell_3 = \mathfrak{s}(\mathbf{1})$.

It is easy to check that, indeed t is a pseudo-derivation of M. To check that $(\mathfrak{s}, \mathfrak{h}) \rhd t$, we take $f : \mathrm{nodes}(t) \to \mathrm{dom}(\mathfrak{h})$ as the restriction of Λ^{-1} to the nodes of $u_{|w_0}$. Clearly, f is injective and the conditions (1), (2) and (3) of Definition 8 are easy checks. □

4.2 Encoding Complement Membership as Entailment Problems

Given an ATM $M = (Q, \Gamma, \delta, q_0, g)$, a pseudo-derivation of M is a derivation of M if the contents of the tape is consistent with the sequence of the actions applied, in particular the following conditions must hold:

I. If a branching node labeled (q,i) is followed by an action node labeled $(a,b,\rightarrow)$ [resp. $(a,b,\leftarrow)$], itself followed by a branching node labeled (q',i') then necessarily $i' = i+1$ [resp. $i = i'+1$], i.e. the position of the head changes according to the action executed between the adjacent configurations.

II. For every $i \in [\![0 \,..\, 2^{\mathfrak{N}} - 1]\!]$, if along a path from a branching node labeled (q,i), followed by an action node labeled (a,b,μ), to another branching node labeled (q',i), followed by an action node labeled (a',b',μ'), there is no branching node labeled (q'',i), then necessarily $a' = b$. Indeed, the symbol read on position i must be the one previously written, since it was not changed in the meantime.

III. For every $i \in [\![0 \,..\, 2^{\mathfrak{N}} - 1]\!]$, if along a path from the root to a branching node labeled (q,i), followed by an action node labeled (a,b,μ), there is no branching node labeled (q',i), then necessarily $a = \mathsf{b}$, i.e. the tape is initially empty.

In the following, we shall not check that the above conditions hold for some derivation of M, but rather the opposite: that for each derivation of M, at least one of the above conditions is broken. In other words, we reduce from the complement of the membership problem (M,ϵ) to an entailment problem, defined next. This does not change the final 2EXPTIME-hardness result, because 2EXPTIME=AEXPSPACE=co–AEXPSPACE, as previously mentioned.

In the following, we shall give the rules for a predicate $c_M(x)$ such that the entailment $p_M(x) \models_{\mathcal{S}} c_M(x)$ holds, for a suitable set of rules $\mathcal{S}$, containing the rules for $p_M(x)$ and $c_M(x)$, iff every pseudo-derivation of M breaks at least one of the conditions (I), (II) or (III), in other words, that M, started on input ϵ, has no derivation.

Let $\mathfrak{B} \stackrel{\text{def}}{=} \max_{q\in Q, a\in\Gamma} \|\tau(q,a)\|$ be the maximum branching degree of a derivation of M. First, we define an auxiliary predicate $r(x)$ that generates all tree-shaped heaps in which branching nodes correctly alternate with action nodes, with no regard to the labels of those nodes. In the following, we stick to the convention that predicate symbols p represent branching nodes, whereas $\overline{p}$ represent action nodes:

$$r(x) \Leftarrow \exists y_1 \ldots \exists y_n \,.\, x \mapsto (\bullet^{\mathfrak{N}}, y_1, \ldots, y_n) * \mathop{\ast}_{j=1}^{n} \overline{r}(y_j), \text{ for each } n \in [\![0 \,..\, \mathfrak{B}]\!]$$
$$\overline{r}(x) \Leftarrow \exists y \,.\, x \mapsto (a, b, \bullet, y) * r(y), \text{ for each } a \in \Gamma \text{ and } b \in \Gamma \setminus \{\mathsf{b}\}$$

Next, we define the heap encodings of those derivation trees that violate condition (I). To this end, we guess a vector $\boldsymbol{b}$ in $\{0,1\}^{\mathfrak{N}}$, encoding a position on the tape $i \in [\![0 \,..\, 2^{\mathfrak{N}} - 1]\!]$, a shift $\mu \in \{\leftarrow, \rightarrow\}$, encoded by $\widetilde{\mu} \in \{\mathbf{0}, \mathbf{1}\}$ and get the binary complement of the (encoding of the) position reached from $\boldsymbol{b}$ by applying μ. Here we distinguish two cases, depending on the choice of μ:

(a) if μ is $\rightarrow$ then $\text{bin}(i) = \boldsymbol{b} \stackrel{\text{def}}{=} (b_1, \ldots, b_n, \mathbf{0}, \mathbf{1}^{\mathfrak{N}-1-n})$ for some $n \in [\![0 \,..\, \mathfrak{N}-1]\!]$ and let $\boldsymbol{c} \stackrel{\text{def}}{=} (\overline{b}_1, \ldots, \overline{b}_n, \mathbf{0}, \mathbf{1}^{\mathfrak{N}-1-n})$ be the complement of $\text{bin}(i+1) = (b_1, \ldots, b_n, \mathbf{1}, \mathbf{0}^{\mathfrak{N}-1-n})$.

(b) otherwise, $\text{bin}(i) = \boldsymbol{b} \stackrel{\text{def}}{=} (b_1, \ldots, b_n, \mathbf{1}, \mathbf{0}^{\mathfrak{N}-1-n})$ and let $\boldsymbol{c} \stackrel{\text{def}}{=} (\overline{b}_1, \ldots, \overline{b}_n, \mathbf{1}, \mathbf{0}^{\mathfrak{N}-1-n})$ be the complement of $\text{bin}(i-1) = (b_1, \ldots, b_n, \mathbf{0}, \mathbf{1}^{\mathfrak{N}-1-n})$.

For every $n \in [\![0 \,..\, \mathfrak{R}-1]\!]$ and every $m \in [\![0 \,..\, \mathfrak{B}]\!]$ and $i \in [\![1 \,..\, m]\!]$, we consider the rules:

$$c_1(x) \Leftarrow \exists b_1 \ldots \exists b_n \exists y \,.\, x \mapsto ([y]^{\mathfrak{R}}) * d_1(y, \mathbf{1}, \underbrace{b_1 \ldots b_n, \mathbf{0}, \mathbf{1}^{\mathfrak{R}-n-1}}_{\boldsymbol{b}}, \underbrace{\overline{b}_1 \ldots \overline{b}_n, \mathbf{0}, \mathbf{1}^{\mathfrak{R}-n-1}}_{\boldsymbol{c}}) \quad (9)$$

$$c_1(x) \Leftarrow \exists b_1 \ldots \exists b_n \exists y \,.\, x \mapsto ([y]^{\mathfrak{R}}) * d_1(y, \mathbf{0}, \underbrace{b_1 \ldots b_n, \mathbf{1}, \mathbf{0}^{\mathfrak{R}-n-1}}_{\boldsymbol{b}}, \underbrace{\overline{b}_1 \ldots \overline{b}_n, \mathbf{1}, \mathbf{0}^{\mathfrak{R}-n-1}}_{\boldsymbol{c}}) \quad (10)$$

$$d_1(x,u,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y_1 \ldots \exists y_m . x \mapsto (\bullet^{\mathfrak{R}}, y_1, \ldots, y_m) * \mathop{\circledast}_{j \in [\![1..m]\!] \setminus \{i\}} \overline{r}(y_j) * \overline{d}_1(y_i, u, \boldsymbol{b}, \boldsymbol{c}) \quad (11)$$

$$d_1(x,u,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y_1 \ldots \exists y_m . x \mapsto (\boldsymbol{b}, y_1, \ldots, y_m) * \mathop{\circledast}_{j \in [\![1..m]\!] \setminus \{i\}} \overline{r}(y_j) * \overline{e}_1(y_i, u, \boldsymbol{b}, \boldsymbol{c}) \quad (12)$$

$$\overline{d}_1(x,u,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y \,.\, x \mapsto (a, b, \bullet, y) * d_1(y, u, \boldsymbol{b}, \boldsymbol{c}), \text{ for each } a \in \Gamma,\ b \in \Gamma \setminus \{\mathsf{b}\} \quad (13)$$

$$\overline{e}_1(x,u,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y \,.\, x \mapsto (a, b, \bullet, y) * f_1(y, u, \boldsymbol{b}, \boldsymbol{c}), \text{ for each } a \in \Gamma,\ b \in \Gamma \setminus \{\mathsf{b}\} \quad (14)$$

$$f_1(x,u,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y_1 \ldots \exists y_m \,.\, x \mapsto (\boldsymbol{e}, y_1, \ldots, y_m) * \mathop{\circledast}_{j=1}^{m} \overline{r}(y_j) \mid \boldsymbol{e} \not\approx \overline{\boldsymbol{c}} \quad (15)$$

Intuitively, rules (9) and (10) choose the move $u \in \{\mathbf{0}, \mathbf{1}\}$ and the binary vectors $\boldsymbol{b}, \boldsymbol{c} \in \{\mathbf{0}, \mathbf{1}\}^{\mathfrak{R}}$, according to the cases (a) and (b) above, respectively. Note that we use the hat $[y]^{\mathfrak{R}}$ to eliminate the binary variables $b_1, \ldots, b_n$, as $n \leq \mathfrak{R}$, according to the elimination procedure described in §3.1. Then a path to the branching node, labeled (q', i'), that violates condition (I) is chosen, by alternating the branching and action nodes allocated by rules (11) and (13), respectively. The offending branching node is allocated by rule (15) and its predecessors are the branching and the action nodes, labeled with (q, i) and (i, a, b, μ), such that $i' \neq i^{\mu}$. These latter nodes are allocated by rules (12) and (14), respectively.

The pseudo-derivations of M that violate condition (II) are encoded by the tree-structured heaps defined by the rules below. To this end, we guess a binary vector $\boldsymbol{b} \in \{\mathbf{0}, \mathbf{1}\}^{\mathfrak{R}}$ denoting the position of a write action that has an inconsistent read descendant and let $\boldsymbol{c}$ be its binary complement. Then, for every $m \in [\![0 \,..\, \mathfrak{B}]\!]$ and $i \in [\![1 \,..\, m]\!]$, we consider the rules below:

$$c_2(x) \Leftarrow \exists b_1 \ldots \exists b_{\mathfrak{R}} \exists y \,.\, x \mapsto ([y]^{\mathfrak{R}}) * d_2(y, \underbrace{b_1, \ldots, b_{\mathfrak{R}}}_{\boldsymbol{b}}, \underbrace{\overline{b}_1, \ldots, \overline{b}_{\mathfrak{R}}}_{\boldsymbol{c}}) \quad (16)$$

$$d_2(x,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y_1 \ldots \exists y_m \,.\, x \mapsto (\bullet^{\mathfrak{R}}, y_1, \ldots, y_m) * \mathop{\circledast}_{j \in [\![1..m]\!] \setminus \{i\}} \overline{r}(y_j) * \overline{d}_2(y_i, \boldsymbol{b}, \boldsymbol{c}) \quad (17)$$

$$\overline{d}_2(x,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y \,.\, x \mapsto (a, b, \bullet, y) * d_2(y, \boldsymbol{b}, \boldsymbol{c}), \text{ for each } a \in \Gamma,\ b \in \Gamma \setminus \{\mathsf{b}\} \quad (18)$$

$$\overline{d}_2(x,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y \,.\, x \mapsto (a, b, \bullet, y) * e_2(y, b, \boldsymbol{b}, \boldsymbol{c}), \text{ for each } a \in \Gamma,\ b \in \Gamma \setminus \{\mathsf{b}\} \quad (19)$$

$$e_2(x,\gamma,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y_1 \ldots \exists y_m \,.\, x \mapsto (\boldsymbol{b}, y_1, \ldots, y_m) * \mathop{\circledast}_{j \in [\![1..m]\!] \setminus \{i\}} \overline{r}(y_j) * \overline{e}_2(y_i, \gamma, \boldsymbol{b}, \boldsymbol{c}) \quad (20)$$

$$\overline{e}_2(x,\gamma,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y \,.\, x \mapsto (a, b, \bullet, y) * f_2(y, \gamma, \boldsymbol{b}, \boldsymbol{c}), \text{ for each } a \in \Gamma,\ b \in \Gamma \setminus \{\mathsf{b}\} \quad (21)$$

$$\overline{e}_2(x,\gamma,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y \,.\, x \mapsto (a, b, \bullet, y) * g_2(y, \gamma, \boldsymbol{b}, \boldsymbol{c}), \text{ for each } a \in \Gamma,\ b \in \Gamma \setminus \{\mathsf{b}\} \quad (22)$$

$$f_2(x,\gamma,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y_1 \ldots \exists y_m \,.\, x \mapsto (\boldsymbol{e}, y_1, \ldots, y_m) * \mathop{\circledast}_{j \in [\![1..m]\!] \setminus \{i\}} \overline{r}(y_j) * \overline{e}_2(y_i, \gamma, \boldsymbol{b}, \boldsymbol{c}) \mid \boldsymbol{e} \not\approx \overline{\boldsymbol{c}} \quad (23)$$

$$g_2(x,\gamma,\boldsymbol{b},\boldsymbol{c}) \Leftarrow \exists y_1 \ldots \exists y_m \,.\, x \mapsto (\boldsymbol{b}, y_1, \ldots, y_m) * \mathop{\circledast}_{j \in [\![1..m]\!] \setminus \{i\}} \overline{r}(y_j) * \overline{g}_2(y_i, \gamma) \quad (24)$$

$$\overline{g}_2(x,\gamma) \Leftarrow \exists y \,.\, x \mapsto (a, b, \bullet, y) * r(y), \text{ for each } a \in \Gamma \setminus \{\gamma\},\ b \in \Gamma \setminus \{\mathsf{b}\} \quad (25)$$

Intuitively, rule (16) uses the hat $[y]^{\mathfrak{N}}$ to choose the tuple of binary variables $\boldsymbol{b} = (b_1, \ldots, b_{\mathfrak{N}})$ and their complements $\boldsymbol{c} = (\overline{b}_1, \ldots, \overline{b}_{\mathfrak{N}})$. First, the path to a branching node labeled by the binary position $\boldsymbol{b}$ is non-deterministically chosen by an alternation of branching and action nodes allocated by the the rules (17) and (18), respectively, until the node and its predecessor are allocated by rules (20) and (19), respectively. The symbol written on the tape by this node is stored in the second parameter of $e_2(x, \gamma, \boldsymbol{b}, \boldsymbol{c})$. Next, a path to a second branching node labeled by the binary position $\boldsymbol{b}$ is non-deterministically chosen by an alternation of branching and action nodes allocated by the the rules (21) and (23) respectively, while checking that no branching node with the same position $\boldsymbol{b}$ occurs on this second path (23). At the end, we reach the offending branching node (24), whose predecessor is allocated by rule (22). At this point, we check that the symbol read by the last action node is different than the symbol previously written at position $\boldsymbol{b}$, by rule (19). This check is done by rules (24) and (25), thus ensuring that condition (II) is indeed violated.

Next, we define the tree-structured heap encodings of the derivation trees that violate condition (III). To this end, we guess a binary vector $\boldsymbol{b} \in \{\boldsymbol{0}, \boldsymbol{1}\}^{\mathfrak{N}}$ denoting the position where a symbol different from b has been read, with no previous write action at that position and let $\boldsymbol{c}$ be its complement. We consider the rules below, for every $m \in [\![0 \,..\, \mathfrak{B}]\!]$ and $i \in [\![1 \,..\, m]\!]$:

$$c_3(x) \Leftarrow \exists b_1 \ldots \exists b_{\mathfrak{N}} \exists y \,.\, x \mapsto ([y]^{\mathfrak{N}}) * d_3(y, \underbrace{b_1, \ldots, b_{\mathfrak{N}}}_{\boldsymbol{b}}, \underbrace{\overline{b}_1, \ldots, \overline{b}_{\mathfrak{N}}}_{\boldsymbol{c}}) \tag{26}$$

$$d_3(x, \boldsymbol{b}, \boldsymbol{c}) \Leftarrow \exists y_1 \ldots \exists y_m \,.\, x \mapsto (\boldsymbol{e}, y_1, \ldots, y_m) * \mathop{\mathbin{\ast}}_{j \in [\![1..m]\!] \setminus \{i\}} \overline{r}(y_j) * \overline{d}_3(y_i, \boldsymbol{b}, \boldsymbol{c}) \mid \boldsymbol{e} \not\approx \overline{\boldsymbol{c}} \tag{27}$$

$$\overline{d}_3(x, \boldsymbol{b}, \boldsymbol{c}) \Leftarrow \exists y \,.\, x \mapsto (a, b, \bullet, y) * d_3(y, \boldsymbol{b}, \boldsymbol{c}), \text{ for all } a \in \Gamma,\ b \in \Gamma \setminus \{\mathsf{b}\} \tag{28}$$

$$\overline{d}_3(x, \boldsymbol{b}, \boldsymbol{c}) \Leftarrow x \mapsto (a, b, \bullet, y) * e_3(y, \boldsymbol{b}, \boldsymbol{c}), \text{ for all } a \in \Gamma,\ b \in \Gamma \setminus \{\mathsf{b}\} \tag{29}$$

$$e_3(x, \boldsymbol{b}, \boldsymbol{c}) \Leftarrow \exists y_1 \ldots \exists y_m \,.\, x \mapsto (\boldsymbol{b}, y_1, \ldots, y_m) * \mathop{\mathbin{\ast}}_{j \in [\![1..m]\!] \setminus \{i\}} \overline{r}(y_j) * \overline{f}_3(y_i) \tag{30}$$

$$\overline{f}_3(x) \Leftarrow \exists y \,.\, x \mapsto (a, b, \bullet, y) * r(y), \text{ for all } a, b \in \Gamma \setminus \{\mathsf{b}\} \tag{31}$$

After the initial guess of the binary position $\boldsymbol{b}$, by rule (26), a path to a branching node labeled by $\boldsymbol{b}$ is nondeterministically guessed, by an alternation of branching and action nodes corresponding to the rules (27) and (28), respectively, while checking that no branching node labeled with position $\boldsymbol{b}$ occurs on this path. Once this node is reached, by rule (29), we check that its action node successor reads a symbol different than b, by rules (30) and (31), which is in violation of condition (III).

Finally, the predicate $c_M(x)$ that choses the condition (I), (II) or (III) to be violated, is defined by the following rules:

$$c_M(x) \Leftarrow \exists y_0 \exists z_0 \,.\, x \mapsto (y_0, z_0) * c_i(y_0) * \mathsf{Const}(z_0), \text{ for all } i \in \{1, 2, 3, \}\tag{32}$$

Let $\mathcal{S}$ denote the set of rules introduced so far. The following lemma states the property of the models of $c_M(x)$:

Lemma 4. *Given a pseudo-derivation t of M and a structure $(\mathfrak{s}, \mathfrak{h})$, such that $(\mathfrak{s}, \mathfrak{h}) > t$, we have $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} c_M(x)$ if and only if t is not a derivation of M.*

Proof: (*sketch*)

Fact 2 *Assume that* $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{S}} c_M(x)$. *For all* $\ell, \ell' \in \mathrm{dom}(\mathfrak{h})$ *and* $i, j \in [\![1 \,..\, \mathsf{k}]\!]$, $\mathfrak{h}(\ell)_i = \mathfrak{h}(\ell')_j$ *iff either (i)* $\ell = \mathfrak{h}(\ell)_i$, *(ii)* $\mathfrak{h}(\ell)_i \in \mathfrak{s}(\Gamma \cup \{\mathbf{0}, \mathbf{1}\})$, *or (iii)* $\ell = \ell'$ *and* $i = j$.

Proof: By close analysis of allocation within the rules of $\mathcal{S}$. □

"⇒" If $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{S}} c_M(x)$ then, by Lemma 1, there exists an unfolding tree $u \in \mathcal{T}_{\mathcal{S}}(c_M(x))$ such that $(\mathfrak{s},\mathfrak{h}) \models \mathcal{F}(u)$. Since $\mathcal{S}$ is a progressing and connected set of rules and, moreover $(\mathfrak{s},\mathfrak{h}) \models \mathcal{F}(u)$, by Lemma 2, there exists a bijection $\Lambda : \mathrm{dom}(\mathfrak{h}) \to \mathrm{nodes}(u) \setminus \{\lambda\}$ such that, for all $wi \in \mathrm{nodes}(u)$, where $i \in \mathbb{N}$, we have $\Lambda^{-1}(wi) \in \mathfrak{h}(\Lambda^{-1}(w))$. Moreover, since $(\mathfrak{s},\mathfrak{h}) > t$, there exists an injective mapping $f : \mathrm{nodes}(t) \to \mathrm{dom}(\mathfrak{h}) \setminus \mathfrak{s}(\Gamma \cup \{\mathbf{0}, \mathbf{1}\})$ such that, for all $w \in \mathrm{nodes}(t)$ such that $wi \in \mathrm{nodes}(t)$, we have $f(wi) \in \mathfrak{h}(f(w))$. Consequently, we obtain the bijection $\Lambda \circ f : \mathrm{nodes}(t) \to \mathrm{nodes}(u) \setminus \{\lambda\}$ where, for all $wi \in \mathrm{nodes}(t)$, such that $i \in \mathbb{N}$, there exists $j \in \mathbb{N}$ such that $f(wi) = \mathfrak{h}(f(w))_j$. Since Λ is bijective and $\lambda \notin \mathrm{img}(\Lambda)$, there exists a node $vk \in \mathrm{nodes}(u)$, for some $k \in \mathbb{N}$, such that $\mathfrak{h}(f(w))_j = \Lambda^{-1}(vk)$. By Lemma 2, there exists $m \in \mathbb{N}$ such that $\Lambda^{-1}(vk) = \mathfrak{h}(\Lambda^{-1}(v))_m$. Since f is injective, $\mathfrak{h}(f(w))_j = f(wi) \neq f(w)$. Moreover, $\mathfrak{h}(f(w))_j \notin \mathfrak{s}(\Gamma \cup \{\mathbf{0}, \mathbf{1}\})$ thus, by Fact 2, we obtain that $f(w) = \Lambda^{-1}(v)$, as a consequence of $\mathfrak{h}(f(w))_j = \mathfrak{h}(\Lambda^{-1}(v))_m$. Then we obtain that, for all $wi \in \mathrm{nodes}(t)$, such that $i \in \mathbb{N}$, there exists $k \in \mathbb{N}$, such that:

$$\Lambda(f(wi)) = \Lambda(\mathfrak{h}(f(w)_j)) = vk = \Lambda(f(w))_k$$

We can conclude that t and $u_{|0}$ are isomorphic, i.e. $\mathrm{nodes}(t) = \mathrm{nodes}(u) \setminus \{\lambda\}$.

By the definition of $\mathcal{S}$, namely rule (32), there exists a unique node $w_0 \in \mathrm{nodes}(u) \setminus \{\lambda\}$ such that $u(w_0) = (c_i, \phi_i)$, for some formula ϕ_i, where $i \in \{1,2,3\}$. Distinguishing the cases $i = 1,2,3$ and using the fact that $(\mathfrak{s},\mathfrak{h}) > t$, one shows that t breaks the one of the conditions (I), (II) or (III), respectively, thus t is not a derivation of M. The proof is along the lines of the second point of Lemma 3.

"⇐" If t is an accepting pseudo-derivation but not a derivation of M, then t violates one of the conditions (I), (II) or (III). In each case, we build an unfolding tree $u \in \mathcal{T}_{\mathcal{S}}(c_M(x))$, along the lines of the proof of the first point of Lemma 3. Using the fact that $(\mathfrak{s},\mathfrak{h}) > t$, we show that $(\mathfrak{s},\mathfrak{h}) \models \mathcal{F}(u)$, leading to $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{S}} c_M(x)$. □

Note that, akin to the rule for $p_M(x)$ (8), the rules (32) contain an occurrence of $\mathsf{Const}(z_0)$ as a sibling to a hat of height $\mathfrak{N}$, that occurs in $c_i(y_0)$, for all $i = 1,2,3$. Then the entailment $p_M(x) \models c_M(x)$ holds if and only if, for each structure $(\mathfrak{s},\mathfrak{h})$ such that $(\mathfrak{s},\mathfrak{h}) \models p_M(x)$ and each extension $\mathfrak{s}[y_0 \leftarrow \ell_0]$, for some location $\ell_0 \in \mathsf{Loc}$, the heap $\mathfrak{h}$ is matched by the unfolding of one of the rules with head $c_1(y_0)$, $c_2(y_0)$ or $c_3(y_0)$. This is possible because each such rule uses a hat $[y]^{\mathfrak{N}}$, matching the one from the rule with head $p_M(x)$ (8).

Lemma 5. *The entailment* $p_M(x) \models_{\mathcal{S}} c_M(x)$ *holds if and only if the membership problem* (M, ϵ) *has a negative answer.*

Proof: "⇒" Suppose, for a contradiction, that M accepts $(q_0, \epsilon, 0)$. By Definition 6 it has a derivation t. Since t is a derivation, it is also a pseudo-derivation of M and, by Lemma 3, there exists a structure $(\mathfrak{s},\mathfrak{h})$ such that $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{S}} p_M(x)$ and $(\mathfrak{s},\mathfrak{h}) \rhd t$. Moreover, $(\mathfrak{s},\mathfrak{h}) > t$ follows from $(\mathfrak{s},\mathfrak{h}) \rhd t$ and, by Lemma 4, we obtain $(\mathfrak{s},\mathfrak{h}) \not\models_{\mathcal{S}} c_M(x)$, thus $p_M(x) \not\models_{\mathcal{S}} c_M(x)$, contradiction.

"$\Leftarrow$" Suppose, for a contradiction, that $p_M(x) \not\models_{\mathcal{S}} c_M(x)$, hence there exists a structure $(\mathfrak{s}, \mathfrak{h})$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{S}} p_M(x)$ and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathcal{S}} c_M(x)$. By Lemma 3, there exists a pseudo-derivation t of M such that $(\mathfrak{s}, \mathfrak{h}) \rhd t$, hence $(\mathfrak{s}, \mathfrak{h}) > t$. By Lemma 4, t is a derivation of M, hence (M, ϵ) has a positive answer, contradiction. $\square$

We state the main result of this paper below:

Theorem 2. *The entailment problem $p(x) \models_{\mathcal{S}} q(x)$, where $\mathcal{S}$ is a progressing, connected and established set of rules and p, q are predicate symbols in* Pred *that occur as heads in $\mathcal{S}$, is* 2EXPTIME*-hard.*

Proof: Given an exponential-space bounded ATM M we define a set of rules $\mathcal{S}$, based on the description of M, such that $p_M(x) \models_{\mathcal{S}} c_M(x)$ if and only if (M, ϵ) has a negative answer (Lemma 5). Moreover, the set of rules is easy shown to be progressing, connected and established. The reduction is possible in time polynomial in the size of the standard encoding of M. Indeed, the number of rules in $\mathcal{S}$ is $O(\|Q\| \cdot \mathfrak{R} \cdot \mathfrak{B})$ and the succint representation of each rule, using deep hats, binary choices and binary variables can be generated in time $O(\|\Gamma\| \cdot \mathfrak{B} \cdot \mathfrak{R})$. Finally, the complete elimination of binary variables is possible in polynomial time. Since we reduce from the complement of a AEXPSPACE-complete membership problem and co-AEXPSPACE=AEXPSPACE=2EXPTIME, we obtain the 2EXPTIME-hardness result. $\square$

5 Conclusions

We show that the entailment problem, for symbolic heaps with inductively defined predicates, showed to be decidable (with elementary recursive time complexity) in [4] has an actual 2EXPTIME-hard lower bound. In the light of the recent results of [7], we expect 2EXPTIME to be the tight complexity for what is currently the most general decidable class of entailments for Separation Logic with inductive definitions.

References

1. Timos Antonopoulos, Nikos Gorogiannis, Christoph Haase, Max I. Kanovich, and Joël Ouaknine. Foundations for decision problems in separation logic with general inductive predicates. In Anca Muscholl, editor, *Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, volume 8412 of *Lecture Notes in Computer Science*, pages 411–425, 2014.
2. James Brotherston, Carsten Fuhs, Juan Antonio Navarro Pérez, and Nikos Gorogiannis. A decision procedure for satisfiability in separation logic with inductive predicates. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 25:1–25:10. ACM, 2014.
3. Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981. URL: https://doi.org/10.1145/322234.322243, doi:10.1145/322234.322243.

4. Radu Iosif, Adam Rogalewicz, and Jiri Simacek. The tree width of separation logic with recursive definitions. In *Proc. of CADE-24*, volume 7898 of *LNCS*, 2013.
5. Radu Iosif, Adam Rogalewicz, and Tomás Vojnar. Deciding entailments in inductive separation logic with tree automata. In Franck Cassez and Jean-François Raskin, editors, *Automated Technology for Verification and Analysis - 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014, Proceedings*, volume 8837 of *Lecture Notes in Computer Science*, pages 201–218. Springer, 2014.
6. Samin S Ishtiaq and Peter W O'Hearn. Bi as an assertion language for mutable data structures. In *ACM SIGPLAN Notices*, volume 36, pages 14–26, 2001.
7. Jens Katelaan, Christoph Matheja, and Florian Zuleger. Effective entailment checking for separation logic with inductive definitions. In Tomás Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part II*, volume 11428 of *Lecture Notes in Computer Science*, pages 319–336. Springer, 2019.