An undecidability result for Separation Logic with theory reasoning

Mnacho Echenim, Nicolas Peltier

An Undecidability Result for Separation Logic With Theory Reasoning.

Mnacho Echenim[a], Nicolas Peltier[a]

[a] *Univ. Grenoble Alpes, CNRS, LIG, 700 Av. Centrale, 38000 Grenoble France*

Abstract

We show that the entailment problem is undecidable in Separation Logic with inductively defined spatial predicates, for rules satisfying the conditions given in [1], if theory reasoning is considered. The result holds for a wide class of theories, even those with a very low expressive power. For instance it applies to natural numbers with the successor function, or with the usual order relation.

Keywords: Separation Logic, Inductive Definition, Decidability

1. Introduction

Separation Logic (SL) [2, 3] is a formal system that was designed to reason about programs that manipulate pointer-based data structures such as, e.g., linked lists or trees. Intuitively, this logic permits to state properties of the heap on which a program dynamically allocates objects, in a natural and concise way. Verifying the correctness of a program then boils down to testing whether one formula in this logic entails another one. However, the considered entailment problem is undecidable [4], and several works have been devoted to devising proof procedures for decidable fragments of this logic. One such fragment was defined in [1], and it was recently established that the entailment problem in this fragment is in 2-EXPTIME [5].

When working on the verification of actual programs, it is often necessary to combine reasoning tasks about the heap *and* about the data stored within the considered data structures. Such a case could occur for example when considering a program that modifies binary trees with integers labeling each node. The combination of SL with data constraints has already been considered by several authors, see, e.g., [6, 7, 8]. On the other hand, the fragment defined in [1] contains no theory other than equality, it is thus natural to investigate whether the above decidability result extends to the case where theory reasoning is considered. In this paper, we show this is not the case: the entailment problem is undecidable for a wide class of theories, including some with a very low expressive power.

2. Separation Logic with Inductive Definitions and Theory Predicates

We consider a countably infinite set of *variables* denoted by $\mathcal{V}$, along with two disjoint sets of predicates: a set of $\mathcal{T}$*-predicates* (or *theory predicates*) $\mathcal{P}_\mathcal{T}$, denoting relations in an underlying theory of locations, and a set of *spatial predicates* $\mathcal{P}_S$. Each symbol $p \in \mathcal{P}_\mathcal{T} \cup \mathcal{P}_S$ is associated with

a unique arity $\#(p)$. Let κ be a fixed natural number. The set of *SL-formulas* (or simply formulas) ϕ is inductively defined as follows:

$$\phi := \texttt{emp} \parallel x \mapsto (y_1, \ldots, y_\kappa) \parallel \phi_1 \vee \phi_2 \parallel \phi_1 * \phi_2 \parallel p(x_1, \ldots, x_{\#(p)}) \parallel \exists x.\ \phi_1$$

where ϕ_1, ϕ_2 are *SL*-formulas, $p \in \mathcal{P}_\mathcal{T} \cup \mathcal{P}_\mathcal{S}$ and $x, x_1, \ldots, x_{\#(p)}, y_1, \ldots, y_\kappa$ are variables. The symbol $*$ is called the *separating conjunction*. In the following, formulas are considered modulo the associativity and commutativity of $\vee$ and $*$, modulo the commutativity of existential quantifications, modulo the neutrality of emp w.r.t. the separating conjunction, and modulo prenex form. A formula of the form emp, $p(\mathbf{x})$ or $x \mapsto (\mathbf{y})$ is called an *atom*. A *substitution* σ is a function mapping variables to variables. For any expression (variable, tuple of variables or formula) e, we denote by $e\sigma$ the expression obtained from e by replacing every free occurrence of a variable x by $\sigma(x)$. The semantics of the predicates in $\mathcal{P}_\mathcal{S}$ is given by user-defined inductive rules. To ensure decidability in the case where the theory only contains the equality predicate, these rules must satisfy some additional conditions[1] [1]:

Definition 1. *A progressing and connected set of inductive rules (pc-SID) $\mathcal{R}$ is a finite set of rules of the form $p(x_1, \ldots, x_n) \Leftarrow x_1 \mapsto (y_1, \ldots, y_\kappa) * \phi$, where ϕ is an SL formula and for all predicate atoms $q(z_1, \ldots, z_{\#(q)})$ in ϕ, we have $z_1 \in \{y_1, \ldots, y_\kappa\}$.*

Definition 2. *For every formula ϕ, we write $\phi \Leftarrow_\mathcal{R} \psi$ if ψ is obtained from ϕ by replacing an atom $p(\mathbf{x})$ with $p \in \mathcal{P}_\mathcal{S}$ occurring in ϕ by a formula $\gamma\sigma$, and there exist a rule $p(\mathbf{y}) \Leftarrow \gamma$ in $\mathcal{R}$ and a substitution σ such that $\mathbf{y}\sigma = \mathbf{x}$.*

Let $\mathcal{L}$ be a countably infinite set of *locations*. An *SL-structure* is a pair $(\mathfrak{s}, \mathfrak{h})$, where $\mathfrak{s}$ is a *store*, i.e. a total function from $\mathcal{V}$ to $\mathcal{L}$, and $\mathfrak{h}$ is a *heap*, i.e. a partial finite function from $\mathcal{L}$ to $\mathcal{L}^\kappa$ (written as a relation: $\mathfrak{h}(\ell) = (\ell_1, \ldots, \ell_\kappa)$ iff $(\ell, \ell_1, \ldots, \ell_\kappa) \in \mathfrak{h}$). A location ℓ (resp. a variable x) is *allocated* in a heap $\mathfrak{h}$ (resp. in a structure $(\mathfrak{s}, \mathfrak{h})$) if $\ell \in dom(\mathfrak{h})$ (resp. $\mathfrak{s}(x) \in dom(\mathfrak{h})$). Two heaps $\mathfrak{h}_1, \mathfrak{h}_2$ are *disjoint* if $dom(\mathfrak{h}_1) \cap dom(\mathfrak{h}_2) = \emptyset$, and $\mathfrak{h}_1 \uplus \mathfrak{h}_2$ then denotes their union.

Definition 3. *Let $\models_\mathcal{T}$ be a satisfiability relation between stores and atoms $p(\mathbf{x})$ with $p \in \mathcal{P}_\mathcal{T}$. Given a formula ϕ, a pc-SID $\mathcal{R}$ and a structure $(\mathfrak{s}, \mathfrak{h})$, we write $(\mathfrak{s}, \mathfrak{h}) \models_\mathcal{R} \phi$ and say that $(\mathfrak{s}, \mathfrak{h})$ is an $\mathcal{R}$-*model *(or simply a model if $\mathcal{R}$ is clear from the context) of ϕ if one of the following conditions holds: (i) $\phi = x \mapsto (y_1, \ldots, y_\kappa)$ and $\mathfrak{h} = \{(\mathfrak{s}(x), \mathfrak{s}(y_1), \ldots, \mathfrak{s}(y_\kappa))\}$; (ii) $\phi = p(\mathbf{x})$ with $p \in \mathcal{P}_\mathcal{T}$ and $\mathfrak{s} \models_\mathcal{T} \phi$; (iii) $\phi = \phi_1 \vee \phi_2$ and $(\mathfrak{s}, \mathfrak{h}) \models_\mathcal{R} \phi_i$, for some $i = 1, 2$; (iv) $\phi = \phi_1 * \phi_2$ and there exist disjoint heaps $\mathfrak{h}_1, \mathfrak{h}_2$ such that $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$ and $(\mathfrak{s}, \mathfrak{h}_i) \models_\mathcal{R} \phi_i$, for all $i = 1, 2$; (v) $\phi = \exists x.\ \phi$ and $(\mathfrak{s}', \mathfrak{h}) \models_\mathcal{R} \phi$, for some store $\mathfrak{s}'$ coinciding with $\mathfrak{s}$ on all variables distinct from x; (vi) $\phi = p(\mathbf{x})$ with $p \in \mathcal{P}_\mathcal{S}$ and $(\mathfrak{s}, \mathfrak{h}) \models_\mathcal{R} \psi$ for some ψ such that $\phi \Leftarrow_\mathcal{R} \psi$.*

We emphasize that a $\mathcal{T}$-formula is satisfied only in structures with empty heaps. Note that Definition 3 is well-founded because every rule allocates at least one variable We write $\phi \models_\mathcal{R} \psi$ if every $\mathcal{R}$-model of ϕ is an $\mathcal{R}$-model of ψ. It is well-known that $*$ distributes over $\vee$, hence every formula ϕ can be transformed into a formula $\bigvee_{i=1}^n \phi_i$ where ϕ_i contains no disjunction. We denote by $dnf(\phi)$ the set $\{\phi_i \mid i = 1, \ldots, n\}$. We slightly adapt the notion of establishment [1]:

[1] For technical convenience we allow for disjunctions in the rules.

Definition 4. *A pc-SID is* established *if for every atom* α*, every predicate-free formula* ϕ *such that* $\alpha \Leftarrow^*_{\mathcal{R}} \phi$*, every* $\psi \in dnf(\phi)$ *and every existential variable* x *in* ψ*,* ψ *contains an atom of the form* $x \mapsto (y_1, \ldots, y_\kappa)$*.*

Note that in contrast to [1] equations are not taken into account; this yields a slightly more restrictive condition hence strengthens our undecidability result. The *entailment problem* (written $\phi \vdash_{\mathcal{R}} \psi$) consists in determining, given two formulas ϕ and ψ and an established pc-SID $\mathcal{R}$, whether $\phi \models_{\mathcal{R}} \psi$. A structure $(\mathfrak{s}, \mathfrak{h})$ is a *countermodel* of an entailment problem $\phi \vdash_{\mathcal{R}} \psi$ iff $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} \phi$ and $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathcal{R}} \psi$.

3. An Undecidability Result

The presented undecidability result relies on the use of a binary predicate symbol $S \in \mathcal{P}_{\mathcal{T}}$, interpreted as a relation $\mathfrak{S}$. Intuitively, this relation will be used to relate distant locations α, α' in a heap, so that α' is uniquely determined for a given α, which will permit to express conditions involving both cells. In the simplest case, one may use any injective function $\mathfrak{S}$ (e.g., the successor function on natural numbers) and state that $(\alpha, \alpha') \in \mathfrak{S}$. However, we also intend to capture theories for which no such injective function exists, e.g., the natural numbers with only the usual order $\leq$. To this aim, we provide a more complex condition. Instead of associating α' with α using an injective function, we associate α' with a couple (α, α'') in such a way that α' is uniquely determined when it satisfies the conditions $\alpha \neq \alpha'$, $(\alpha, \alpha') \in \mathfrak{S}$ and $(\alpha'', \alpha) \notin \mathfrak{S}$. For example, if $\mathcal{L} = \mathbb{N}$ and $\mathfrak{S}$ is $\leq$, the conditions entail $\alpha < \alpha' < \alpha''$ and α' is uniquely determined by letting $\alpha'' = \alpha + 2$. The conditions in Theorem 5 are meant to ensure that this is feasible and that there exists an infinite set of locations $\alpha, \alpha', \alpha''$ such that (α, α'') uniquely defines α'. The fact that the set of locations is infinite is essential because it ensures that data structures of unbounded size may be constructed.

Theorem 5. *Assume that* $\mathcal{P}_{\mathcal{T}}$ *contains predicates* S *and* $\overline{S}$*, where:*

- *S is interpreted as a relation $\mathfrak{S}$ such that there exist 3 infinite sequences of locations α_i, α'_i and α''_i (with $i \in \mathbb{N}$) where:*
  - *all the locations $\alpha_i, \alpha'_i, \alpha''_i$ (for $i \in \mathbb{N}$) are pairwise distinct, i.e., we have, for all $i, j \in \mathbb{N}$: $\alpha_i \neq \alpha'_j \wedge \alpha_i \neq \alpha''_j \wedge \alpha'_i \neq \alpha''_j \wedge (i \neq j \implies \alpha_i \neq \alpha_j \wedge \alpha'_i \neq \alpha'_j \wedge \alpha''_i \neq \alpha''_j)$;*
  - *$\forall i \in \mathbb{N}, (\alpha_i, \alpha'_i) \in \mathfrak{S}$;*
  - *$\forall i \in \mathbb{N}, (\alpha''_i, \alpha'_i) \notin \mathfrak{S}$;*
  - *for all $i \in \mathbb{N}$ and for all locations $\ell \in \{\alpha_j \mid j \in \mathbb{N}\} \cup \{\alpha'_j \mid j \in \mathbb{N}\} \cup \{\alpha''_j \mid j \in \mathbb{N}\}$, if $\alpha_i \neq \ell$, $(\alpha_i, \ell) \in \mathfrak{S}$ and $(\alpha''_i, \ell) \notin \mathfrak{S}$, then $\ell = \alpha'_i$;*
- *$\overline{S}(x, y)$ and $\neg S(x, y)$ are interpreted equivalently when x and y refer to distinct locations.*

Then the entailment problem is undecidable for established pc-SID.

For instance, the hypotheses of Theorem 5 are satisfied if $\mathcal{L} = \mathbb{N}$ and if $\mathfrak{S}$ is either the successor function or the usual order relation $\leq$ (in which case $\overline{S}$ is $\geq$). Indeed, it is sufficient to take $\alpha_i = 3 \cdot i$, $\alpha'_i = 3 \cdot i + 1$, $\alpha''_i = 3 \cdot i + 2$ in both cases, since $\alpha'_i = \ell$ exactly when ℓ is the successor of α_i, and $\alpha_i \leq \ell \wedge \alpha''_i > \ell \wedge \ell \neq \alpha_i \Rightarrow \alpha'_i = \ell$. More generally, the conditions hold if the domain is infinite and $\mathfrak{S}$ is any injective function f such that $f(x) \neq x$. In this case, the sequences $\alpha_i, \alpha'_i, \alpha''_i$ may

be constructed inductively: for every $i \in \mathbb{N}$, α_i is any element e such that both e and $f(e)$ do not occur in $\{\alpha_j, \alpha'_j, \alpha''_j \mid j < i\}$, α'_i is $f(\alpha_i)$ and α''_i is any element not occurring in $\{\alpha_j, \alpha'_j, \alpha''_j \mid j < i\} \cup \{\alpha_i, \alpha'_i\}$. Note that in this case the locations α''_i are actually irrelevant, but these locations play an essential rôle in the undecidability proof when $\mathfrak{S}$ is $\leq$. We have $\{\alpha_i \mid i \in \mathbb{N}\} \cup \{\alpha'_i \mid i \in \mathbb{N}\} \cup \{\alpha''_i \mid i \in \mathbb{N}\} \subseteq \mathcal{L}$, but we do not assume that the inclusion is strict.

The rest of the section is devoted to the proof of Theorem 5. It goes by a reduction from the Post Correspondence Problem (PCP).

Definitions and Notations. We recall that the PCP consists in determining, given two sequences of words $\lambda = (\lambda_1, \ldots, \lambda_n)$ and $\gamma = (\gamma_1, \ldots, \gamma_n)$, whether there exists a nonempty sequence $(i_1, \ldots, i_k) \in \{1, \ldots, n\}^k$ such that $\lambda_{i_1} \ldots, \lambda_{i_k} = \gamma_{i_1} \ldots, \gamma_{i_k}$. It is well-known that this problem is undecidable. We assume, w.l.o.g., that $\|\lambda_i\| > 1$ and $\|\gamma_i\| > 1$ for all $i \in \{1, \ldots, n\}$. A word w such that $w = \lambda_{i_1} \ldots, \lambda_{i_k} = \gamma_{i_1} \ldots, \gamma_{i_k}$ is called a *witness*. Positions within words of the sequences $(\lambda_1, \ldots, \lambda_n)$ or $(\gamma_1, \ldots, \gamma_n)$ will be denoted by pairs (i, j), encoding the j-nth character of the word λ_i or γ_i. More formally, if $p = (i, j)$, and $w \in \{\lambda, \gamma\}$, then we denote by $w(p)$ the j-th symbol of the word w_i, provided this symbol is defined. We write $p \lhd q$ if both $\lambda(p)$ and $\gamma(q)$ are defined and $\lambda(p) = \gamma(q)$. Let $m = \max\{\|\lambda_i\|, \|\gamma_i\| \mid i \in \{1, \ldots, n\}\}$. We denote by $\mathtt{P}$ the set of pairs (i, j) where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, and by $\mathtt{B}$ the set of pairs of the form $(i, 1)$. For $w \in \{\lambda, \gamma\}$, we denote by $\mathtt{E}_w$ the set of pairs of the form $(i, \|w_i\|)$, where $i \in \{1, \ldots, n\}$, and we write $(i, j) \rightarrow^w (i', j')$ when either $i' = i$, $j < \|w_i\|$ and $j' = j + 1$, or $j = \|w_i\|$, $i' \in \{1, \ldots, n\}$ and $j' = 1$. Note that i' is arbitrary in the latter case (intuitively $(i, j) \rightarrow^w (i', j')$ states that the character corresponding to the position (i, j) may be followed in a witness by the character at position (i', j')).

Overview. We shall reduce the PCP to an entaillment problem $\phi \vdash_{\mathcal{R}} \psi$, where ϕ encodes a potential solution of the problem, i.e., a pair of sequences $(i_1, \ldots, i_k)$ and $(j_1, \ldots, j_l)$ such that $\lambda_{i_1} \ldots, \lambda_{i_k} = \gamma_{j_1} \ldots, \gamma_{j_{k'}}$, and ψ holds if both sequences $(i_1, \ldots, i_k)$ and $(j_1, \ldots, j_{k'})$ are distinct. Thus $\phi \vdash_{\mathcal{R}} \psi$ admits a countermodel iff the PCP admits a solution.

Left-hand side. This part of the encoding is straightforward, since the set of words w such that $w = \lambda_{i_1} \ldots, \lambda_{i_k} = \gamma_{j_1} \ldots, \gamma_{j_{k'}}$, for some sequences $(i_1, \ldots, i_k)$ and $(j_1, \ldots, j_l)$ is regular. Let $\mathbf{v}$ be a vector of variables. We assume that every $p \in \mathtt{P}$ is associated with a variable x_p in $\mathbf{v}$, in such a way that the mapping $p \mapsto x_p$ is injective (this is possible since $\mathtt{P}$ is finite). To simplify notations, we will simply denote the variable x_p by p. The vector $\mathbf{v}$ also contains a special variable $\perp$, distinct from the variables corresponding to $p \in \mathtt{P}$. Potential witnesses are encoded by linked lists, with links on the last argument, starting with a dummy element. Except for the first dummy element, each location in the list refers to two locations associated with pairs $p, q \in \mathtt{P}$ denoting positions within both sequences $\lambda_1, \ldots, \lambda_n$ and $\gamma_1, \ldots, \gamma_n$ respectively, and to three additional allocated locations the rôles of which will be detailed below. Let $\phi = W(x, \mathbf{v})$, and consider the rules:

$$
\begin{array}{rcl}
W(x, \mathbf{v}) & \Leftarrow & \exists x'.\, x \mapsto (\perp, \perp, \perp, \perp, \perp, x') * W_{p,p}(x', \mathbf{v}) \\
 & & \quad \text{where } p \in \mathtt{B} \\
W_{p,q}(x, \mathbf{v}) & \Leftarrow & \exists x', y, z, u.\, x \mapsto (p, q, y, z, u, x') * W_{p',q'}(x', \mathbf{v}) * P(y, \perp) * P(z, \perp) \\
 & & \quad * P(u, \perp) \qquad \text{where } p \lhd q,\ p \rightarrow^u p' \text{ and } q \rightarrow^v q' \\
W_{p,q}(x, \mathbf{v}) & \Leftarrow & \exists y, z, u.\, x \mapsto (p, q, y, z, u, \perp) * P(y, \perp) * P(z, \perp) * P(u, \perp) \\
 & & \quad \text{where } p = (i, \|\lambda_i\|),\ q = (i, \|\gamma_i\|), \text{ and } p \lhd q \\
P(x, y) & \Leftarrow & x \mapsto (y, y, y, y, y, y)
\end{array}
$$

The following result follows immediately from the definition of the rules defining W:

Lemma 6. *The structures that validate ϕ are of the form $(\mathfrak{s},\mathfrak{h})$, where: (i) $\mathfrak{s}(x)=\ell$ and $\mathfrak{s}(\bot)=\ell'$; (ii) for all $i=1,\dots,m'$, $\mathfrak{s}(p_i)=\ell_i^p$ and $\mathfrak{s}(q_i)=\ell_i^q$; (iii) $p_i,q_i\in\mathtt{P}$ are such that $p_i\lhd q_i$, $p_i\to^{\lambda}p_{i+1}$ and $q_i\to^{\gamma}q_{i+1}$; (iv) the heap $\mathfrak{h}$ is of the form (with $\ell_{m'+1}=\ell'$):*

$$\begin{array}{rcl}\mathfrak{h} = \{(\ell,\ell',\ell',\ell',\ell',\ell',\ell_1)\} & \cup & \{(\ell_i,\ell_i^p,\ell_i^q,\ell_i^y,\ell_i^z,\ell_i^u,\ell_{i+1}) \mid i=1,\dots,m'\}\\ \cup\{(\ell_i^y,\ell',\ell',\ell',\ell',\ell',\ell') \mid i=1,\dots,m'\} & \cup & \{(\ell_i^z,\ell',\ell',\ell',\ell',\ell',\ell') \mid i=1,\dots,m'\}\\ & \cup & \{(\ell_i^u,\ell',\ell',\ell',\ell',\ell',\ell') \mid i=1,\dots,m'\}.\end{array}$$

Furthermore, $p_1=q_1\in\mathtt{B}$ and there exists i such that $p_{m'}=(i,\|\lambda_i\|)$ and $q_{m'}=(i,\|\gamma_i\|)$. Thus, the words $\lambda(p_1)\cdots\lambda(p_{m'})$ and $\gamma(p_1)\cdots\gamma(p_{m'})$ are of the form $\lambda_{i_1}.\dots.\lambda_{i_k}$ and $\gamma_{j_1}.\dots.\gamma_{j_{k'}}$, for some sequences $(i_1,\dots,i_k)$ and $(j_1,\dots,j_{k'})$ of elements in $\{1,\dots,n\}$.

Note that we have $\lambda(p_1)\cdots\lambda(p_{m'})=\gamma(p_1)\cdots\gamma(p_{m'})$, however the sequences $(i_1,\dots,i_k)$ and $(j_1,\dots,j_{k'})$ may be distinct (but we still have $i_1=j_1$ and $i_k=j_{k'}$).

Right-hand side. We introduce a formula ψ that is satisfied when $(i_1,\dots,i_k)\neq(j_1,\dots,j_{k'})$, i.e., for which either $k\neq k'$ or $i_l\neq j_l$ for some $l\in\{2,\dots,k-1\}$. This is done by using the additional locations ℓ_i^y, ℓ_i^z and ℓ_i^u to relate the indices $I_l=\|\lambda_{i_1}.\dots.\lambda_{i_{l-1}}\|+1$ and $J_l=\|\gamma_{j_1}.\dots.\gamma_{j_{l-1}}\|+1$, corresponding to the beginning of the words λ_{i_l} and γ_{j_l} respectively in $\lambda_{i_1}.\dots.\lambda_{i_k}$ and $\gamma_{j_1}.\dots.\gamma_{j_{k'}}$, with the convention that $I_1=J_1=1$. We introduce predicates relating the locations ℓ_i^y, ℓ_i^z and ℓ_i^u using the relation $\mathfrak{S}$. They are associated with rules that guarantee that all the models of the left-hand side that are countermodels of the right-hand side of the sequent will satisfy the following properties: (i) $k=k'$ and for all $l\in\{1,\dots,k\}$, $(\ell_{I_l}^y,\ell_{J_l}^z)\in\mathfrak{S}$ and $(\ell_{I_l}^u,\ell_{J_l}^z)\notin\mathfrak{S}$; (ii) $i_l=j_l$ for $1\leq l\leq k$. Predicates A and B introduced below are used to guarantee that the first condition holds for these countermodels. Predicate A is satisfied by those structures that start with a dummy node and for which the condition does not hold when $l=1$, and B is satisfied by those structures for which either $k\neq k'$ or there is an $l\in\{1,\dots,k-1\}$ such that the condition is satisfied at l, but not at $l+1$. Thus the structures that satisfy $W(x,\mathbf{v})$ and that are countermodels of the disjunction of A and B are exactly the structures for which $k=k'$ and $(\ell_{I_l}^y,\ell_{J_l}^z)\in\mathfrak{S}\wedge(\ell_{I_l}^u,\ell_{J_l}^z)\notin\mathfrak{S}$ for $l=1,\dots,k$. Predicate C is then used to guarantee that for all $1\leq l\leq k$, $p_{I_l}=q_{J_l}$: this predicate is satisfied by the considered structures for which there is an l such that $(\ell_{I_l}^y,\ell_{J_l}^z)\in\mathfrak{S}\wedge(\ell_{I_l}^u,\ell_{J_l}^z)\notin\mathfrak{S}$ holds but $p_{I_l}\neq q_{J_l}$. We begin with the rules for predicate A.

$$\begin{array}{rcl}A(x,\mathbf{v}) & \Leftarrow & \exists x'.\, x\mapsto(\bot,\bot,\bot,\bot,\bot,x')*A'(x',\mathbf{v})\\ A'(x,\mathbf{v}) & \Leftarrow & \exists x',y,z,u.\, x\mapsto(p,q,y,z,u,x')*W_{p',q'}(x,\mathbf{v})*P(y,\bot)\\ & & \quad *P(z,\bot)*P(u,\bot)*(\overline{S}(y,z)\vee S(u,z)),\text{ for every } p,q,p',q'\in\mathtt{P}\end{array}$$

Note that since y,z,u are allocated in distinct predicates, they must be distinct, hence $\overline{S}(y,z)$ is equivalent to $\neg S(y,z)$ and $S(u,z)$ is equivalent to $\neg\overline{S}(u,z)$. The following property is a straightforward consequence of the definition of A (note that, by definition, $I_1=J_1=1$):

Lemma 7. *A model $(\mathfrak{s},\mathfrak{h})$ of ϕ validates $A(x,\mathbf{v})$ iff (using the notations of Lemma 6) either $(\ell_{I_1}^y,\ell_{J_1}^z)\notin\mathfrak{S}$ or $(\ell_{I_1}^u,\ell_{J_1}^z)\in\mathfrak{S}$.*

We now define the rules for predicate B, which is meant to ensure that if the condition "$(\ell^y_{I_l}, \ell^z_{J_l}) \in \mathfrak{S}$ and $(\ell^u_{I_l}, \ell^z_{J_l}) \notin \mathfrak{S}$" (†) holds for some l then it also holds for $l+1$. This predicate has additional parameters y, y', z, z', u, u' corresponding to the locations $\ell^y_{I_l}, \ell^y_{I_{l+1}}, \ell^z_{J_j}, \ell^z_{J_{j+1}}, \ell^u_{I_l}, \ell^u_{I_{l+1}}$ which "break" the propagation of (†). The locations y, y', z, z', u, u' must be chosen in such a way that the $\mathcal{T}$-formula $S(y,z) * \overline{S}(u,z) * (\overline{S}(y',z') \vee S(u',z'))$ holds. The predicates $B_{a,b}$ with $a, b \in \{0,1,2\}$ allocate all the locations $\ell_1, \ldots, \ell_{m'}$ and in particular the "faulty" locations associated with y, y', z, z', u, u'. Intuitively, a (resp. b) denotes the number of variables in $\{y, y'\}$ (resp. $\{z, z'\}$) that have already been allocated. The rules for predicates $B_{a,b}$ are meant to guarantee that the following conditions hold for variables y and y' (similar constraints hold for z and z'): (i) y is allocated before y', (ii) y is allocated for a variable p corresponding to the beginning of a word ($p \in \mathtt{B}$), (iii) when y has been allocated, no variable $p \in \mathtt{B}$ can occur on the right-hand side of an atom of the form $u \mapsto \mathbf{v}$ until y' is allocated. Several cases are distinguished depending on whether the locations associated with y and z (resp. y' and z') are in the same heap image of a location or not. Note that u and u' are allocated in the same rules as y and y' respectively. Predicate B also tackles the case where $k \neq k'$. This corresponds to the case where the recursive calls end with $B_{1,2}$ or $B_{2,1}$ in the last rule below, meaning that (†) holds for some l, with either $l = k$ and $l < k'$ or $l = k'$ and $l < k$. For the sake of conciseness and readability, we denote by $\mathbf{w}$ the vector of variables $\mathbf{v}, y, y', z, z', u, u'$ in the rules below. We also denote by $\phi'(y,z,u)$ the formula $P(y,\bot) * P(z,\bot) * P(u,\bot)$.

$$
\begin{array}{rcl}
B(x,\mathbf{w}) & \Leftarrow & x \mapsto (\bot,\bot,\bot,\bot,\bot,x') * B_{0,0}(x',\mathbf{w}) \\
& & \quad * S(y,z) * \overline{S}(u,z) * (\overline{S}(y',z') \vee S(u',z')) \\
B_{a,b}(x,\mathbf{w}) & \Leftarrow & \exists x', y'', z'', u''.x \mapsto (p,q,y'',z'',u'',x') * B_{a,b}(x',\mathbf{w}) * \phi'(y'',z'',u'') \\
& & \quad \text{if } (a \neq 1 \text{ or } p \notin \mathtt{B}) \text{ and } (b \neq 1 \text{ or } q \notin \mathtt{B}) \\
B_{0,0}(x,\mathbf{w}) & \Leftarrow & \exists x'.x \mapsto (p,q,y,z,u,x') * B_{1,1}(x',\mathbf{w}) * \phi'(y,z,u) \quad \text{if } p,q \in \mathtt{B} \\
B_{0,1}(x,\mathbf{w}) & \Leftarrow & \exists x'.x \mapsto (p,q,y,z',u,x') * B_{1,2}(x',\mathbf{w}) * \phi'(y,z',u) \quad \text{if } p,q \in \mathtt{B} \\
B_{1,0}(x,\mathbf{w}) & \Leftarrow & \exists x'.x \mapsto (p,q,y',z,u',x') * B_{2,1}(x',\mathbf{w}) * \phi'(y',z,u') \quad \text{if } p,q \in \mathtt{B} \\
B_{1,1}(x,\mathbf{w}) & \Leftarrow & \exists x'.x \mapsto (p,q,y',z',u',x') * B_{2,2}(x',\mathbf{w}) * \phi'(y',z',u') \quad \text{if } p,q \in \mathtt{B} \\
B_{0,b}(x,\mathbf{w}) & \Leftarrow & \exists x', z''.x \mapsto (p,q,y,z'',u,x') * B_{1,b}(x',\mathbf{w}) * \phi'(y,z'',u) \\
& & \quad \text{if } p \in \mathtt{B} \text{ and } (b \neq 1 \text{ or } q \notin \mathtt{B}) \\
B_{1,b}(x,\mathbf{w}) & \Leftarrow & \exists x', z''.x \mapsto (p,q,y',z'',u',x') * B_{2,b}(x',\mathbf{w}) * \phi'(y',z'',u') \\
& & \quad \text{if } p \in \mathtt{B} \text{ and } (b \neq 1 \text{ or } q \notin \mathtt{B}) \\
B_{a,0}(x,\mathbf{w}) & \Leftarrow & \exists x', y'', u''.x \mapsto (p,q,y'',z,u'',x') * B_{a,1}(x',\mathbf{w}) * \phi'(y'',z,u'') \\
& & \quad \text{if } q \in \mathtt{B} \text{ and } (a \neq 1 \text{ or } p \notin \mathtt{B}) \\
B_{a,1}(x,\mathbf{w}) & \Leftarrow & \exists x', y'', u''.x \mapsto (p,q,y'',z',u'',x') * B_{a,2}(x',\mathbf{w}) * \phi'(y'',z',u'') \\
& & \quad \text{if } q \in \mathtt{B} \text{ and } (a \neq 1 \text{ or } p \notin \mathtt{B}) \\
B_{a,b}(x,\mathbf{w}) & \Leftarrow & \exists y'', z'', u''.x \mapsto (p,q,y'',z'',u'',\bot) * \phi'(y'',z'',u'') \\
& & \text{if } (a,b) \in \{(2,2),(2,1),(1,2)\}
\end{array}
$$

By imposing that all words in the PCP instance are of length at least 2, we guarantee that the last rule above is the only terminating one that is necessary. It is possible to get rid of the condition on the word lengths at the cost of requiring additional rules for the case where the recursive calls should end at the beginning of a word. The following lemma states that models of ϕ validate $\exists y, z, y', z', u, u'.B(x,\mathbf{w})$ iff Property (†) does not propagate:

Lemma 8. *A model $(\mathfrak{s},\mathfrak{h})$ of ϕ validates $\exists y, z, y', z', u, u'.B(x,\mathbf{w})$ iff (using the notations of Lemma 6) there exists $l \in \{1,\ldots,k\}$ and $l' \in \{1,\ldots,k'\}$ such that $(\ell^y_{I_l}, \ell^z_{J_{l'}}) \in \mathfrak{S}$, $(\ell^u_{I_l}, \ell^z_{J_{l'}}) \notin \mathfrak{S}$ and one of the*

following holds: ($l = k$ and $l' \neq k'$) or ($l \neq k$ and $l' = k'$) or $(\ell^y_{I_{l+1}}, \ell^z_{J_{l'+1}}) \notin \mathfrak{S}$ or $(\ell^u_{I_{l+1}}, \ell^z_{J_{l'+1}}) \in \mathfrak{S}$.

Proof. Consider a store $\mathfrak{s}'$, coinciding with $\mathfrak{s}$ on all variables other than y, z, y', z', u, u' and such that $(\mathfrak{s}, \mathfrak{h})$ satisfies $B(x, \mathbf{w})$. The rules defining B impose that the last rule of B is applied at some point, as it is the only rule with no predicate on its right-hand side. Since this rule applies only on predicates $B_{a,b}$ with $a > 0$ and $b > 0$, this entails that there are cells ℓ^y_i, ℓ^u_i and ℓ^z_j, possibly with $i = j$, that must have been allocated and are such that $\mathfrak{s}'(y) = \ell^y_i$, $\mathfrak{s}'(u) = \ell^u_i$ and $\mathfrak{s}'(z) = \ell^z_j$. By the conditions on the rules, i and j must correspond to the start of a word in λ or γ respectively, thus we have $i = I_l$ and $j = J_{l'}$, for some $l \in \{1, \ldots, k\}$ and $l' \in \{1, \ldots, k'\}$. If the last rule is applied with $a = b = 2$, then there exist cells $\ell^y_{i'}$, $\ell^u_{i'}$ and $\ell^z_{j'}$ that must have been allocated, and are such that $\mathfrak{s}'(y') = \ell^y_{i'}$, $\mathfrak{s}'(u') = \ell^u_{i'}$ and $\mathfrak{s}'(z') = \ell^z_{j'}$. Still by the conditions on the rules, none of the cells $\ell_{i+1}, \ldots, \ell_{i'-1}, \ell_{j+1}, \ldots, \ell_{j'-1}$ may correspond to the start of a word, thus we have $i' = I_{l+1}$ and $j' = J_{l'+1}$. Because of the first rule we have $(\mathfrak{s}'(y), \mathfrak{s}'(z)) \in \mathfrak{S}$, $(\mathfrak{s}'(u), \mathfrak{s}'(z)) \notin \mathfrak{S}$, and either $(\mathfrak{s}'(y'), \mathfrak{s}'(z')) \notin \mathfrak{S}$, or $(\mathfrak{s}'(u'), \mathfrak{s}'(z')) \in \mathfrak{S}$, hence we get the result. If the rule is applied with $(a, b) = (1, 2)$ then $\ell^z_{j'}$ is allocated but not $\ell^y_{i'}$, which entails that $l' \neq k'$ and $l = k$. The proof is similar if $(a, b) = (2, 1)$. □

We derive the following result.

Lemma 9. *With the notations of Lemma 6, a model $(\mathfrak{s}, \mathfrak{h})$ of ϕ falsifies the formula $A(x, \mathbf{v}) \vee \exists y, z, y', z', u, u'.B(x, \mathbf{w})$, iff $k = k'$ and for all $l \in \{1, \ldots, k\}$, $(\ell^y_{I_l}, \ell^z_{J_l}) \in \mathfrak{S} \wedge (\ell^u_{I_l}, \ell^z_{J_l}) \notin \mathfrak{S}$.*

Proof. The proof is by induction on l. Lemmata 7 and 8 give the base and inductive cases, respectively. □

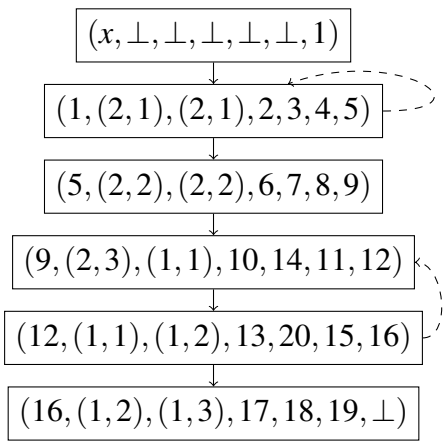
Finally, we consider an atom $C(x, \mathbf{v})$ that will be satisfied by structures such that $p_{I_i} \neq q_{J_i}$, for some $i = 1, \ldots, k$, assuming the condition (†) above is fulfilled. This predicate allocates the location ℓ and introduces existential variables y, z, u denoting the faulty locations $\ell^y_{I_i}, \ell^z_{J_i}$ and $\ell^u_{I_i}$, i.e., the locations corresponding to the index i such that $p_{I_i} \neq q_{J_i}$. By (†), these locations must be chosen in such a way that the constraints $S(y, z)$ and $\overline{S}(u, z)$are satisfied. The predicate $C(x, \mathbf{v})$ also guesses pairs p, q such that $p \neq q$ (denoting the distinct pairs p_{I_i} and q_{J_i}) and invokes the predicate $C^{(0,0)}_{p,q}$ to allocate all the remaining locations. As for the previous rules, the predicates $C^{a,b}_{p,q}$, for $p, q \in \mathtt{B}$ and $a, b \in \{0, 1\}$ allocate $\ell_1, \ldots, \ell_{m'}$, where a (resp. b) denotes the number of variables in $\{y\}$ (resp. $\{z\}$) that have already been allocated. In the rules below, we denote by $\mathbf{u}$ the vector $\mathbf{v}, y, z, u$. In all the rules we have $p', q' \in \mathtt{P}$.

$$
\begin{array}{rcl}
C(x, \mathbf{v}) & \Leftarrow & \exists y, z, u.\ x \mapsto (\bot, \bot, \bot, \bot, \bot, x') * C^{0,0}_{p,q}(x', \mathbf{u}) * S(y, z) * \overline{S}(u, z) \\
 & & \quad \text{if } p \neq q \text{ and } p, q \in \mathtt{B} \\
C^{a,b}_{p,q}(x, \mathbf{u}) & \Leftarrow & \exists x', y'', z'', u''.x \mapsto (p', q', y'', z'', u'', x') * C^{a,b}_{p,q}(x, \mathbf{u}) \\
 & & \quad * \phi'(y'', z'', u'') \\
C^{0,0}_{p,q}(x, \mathbf{u}) & \Leftarrow & \exists x'.x \mapsto (p, q, y, z, u, x') * C^{1,1}_{p,q}(x, \mathbf{u}) * \phi'(y, z, u) \\
C^{0,b}_{p,q}(x, \mathbf{u}) & \Leftarrow & \exists x', z''.x \mapsto (p, q', y, z'', u, x') * C^{1,b}_{p,q}(x, \mathbf{u}) * \phi'(y, z'', u) \\
C^{a,0}_{p,q}(x, \mathbf{u}) & \Leftarrow & \exists x', y'', u''.x \mapsto (p', q, y'', z, u'', x') * C^{a,1}_{p,q}(x, \mathbf{u}) * \phi'(y'', z, u'') \\
C^{1,1}_{p,q}(x, \mathbf{u}) & \Leftarrow & \exists y'', z'', u''.x \mapsto (p', q', y'', z'', u'', \bot) * \phi'(y'', z'', u'')
\end{array}
$$

Let $\psi = A(x,\mathbf{v}), \exists y,z,y',z',u,u'.B(x,\mathbf{w}), C(x,\mathbf{u})$. The entailment problem $\phi \vdash_{\mathcal{R}} \psi$ satisfies the expected property:

Lemma 10. *The PCP admits a solution iff the entailment problem $\phi \vdash_{\mathcal{R}} \psi$ has a countermodel.*

Proof. Assume a structure satisfies the atom $W(x,\mathbf{v})$ but not the disjunction $A(x,\mathbf{v}) \vee \exists y,z,y',z',u,u'.B(x,\mathbf{w}) \vee C(x,\mathbf{u})$. Then by Lemma 6, there exists a word $\lambda_{i_1}.\ldots.\lambda_{i_k} = \gamma_{j_1}.\ldots.\gamma_{j_{k'}}$. Since $C(x,\mathbf{u})$ is not satisfied, $i_l = j_{l'}$ holds if $(\ell^y_{I_l}, \ell^z_{J_{l'}}) \in \mathfrak{S} \wedge (\ell^u_{I_l}, \ell^z_{J_{l'}}) \notin \mathfrak{S}$. By Lemma 9, we have $k = k'$ and Property (†) holds, hence we deduce that $i_l = j_l$ for all $l \in \{1,\ldots,k\}$, i.e., that $(i_1,\ldots,i_k) = (j_1,\ldots,j_{k'})$. Conversely, if a solution of the PCP exists, then by using the locations $\alpha_l, \alpha'_l, \alpha''_l$ guaranteed to exist by the hypothesis of the theorem as the respective locations $\ell^y_{I_l}, \ell^z_{J_l}, \ell^u_{I_l}$, it is easy to construct a structure satisfying $W(x,\mathbf{v})$. Also, by hypothesis, since $(\alpha_l, \alpha'_l) \in \mathfrak{S}$ and $(\alpha''_l, \alpha'_l) \notin \mathfrak{S}$, we have $(\ell^y_{I_l}, \ell^z_{J_l}) \in \mathfrak{S}$ and $(\ell^u_{I_l}, \ell^z_{J_l}) \notin \mathfrak{S}$ for all $l = 1,\ldots,k$. Thus, both $A(x,\mathbf{v})$ and $\exists y,z,y',z',u,u'.B(x,\mathbf{w})$ do not hold. To fulfill $\neg C(x,\mathbf{u})$ we have to ensure that, for all $i,j \in \{1,\ldots,k\}$, we have $(\ell^y_{I_i}, \ell^z_{J_j}) \in \mathfrak{S} \wedge (\ell^u_{I_i}, \ell^z_{J_j}) \notin \mathfrak{S} \implies p_{I_i} = q_{J_i}$. Since the considered word is a solution of the PCP, we have $p_{I_i} = q_{J_i}$ for all $i = 1,\ldots,k$, hence $\neg C(x,\mathbf{u})$ is satisfied. □



We provide a heap encoding a witness *abcde* for the PCP, with $\lambda = (de, abc)$ and $\gamma = (cde, ab)$. The solution is $i_1 = 2$ and $i_2 = 1$, with $k = 2$. We assume that $\mathcal{L} = \mathbb{N}$ and that S is interpreted as the successor function. For readability the interpretation of the variables $x, \bot$, $(1,1)$, $(1,2)$, $(1,3)$, $(2,1)$, $(2,2)$ and $(2,3)$ are left unspecified and the mappings from the locations $2,3,4,6,7,8,10,11,13,14,15,17,18,19,20$ to $(\bot,\bot,\bot,\bot,\bot,\bot)$ are not depicted. The sequence $(\ell_1,\ell_2,\ell_3,\ell_4,\ell_5)$ is $(1,5,9,12,16)$. We have $I_1 = 1$, $I_2 = 4$, $J_1 = 1$ and $J_2 = 3$. The dashed arrows indicate the connections between cells ℓ_{I_l} and ℓ_{J_l}, for $l = 1,2$. For $l = 1$, $\ell_{I_l} = \ell_{J_l} = \ell_1 = 1$ is connected to itself since we have $\ell^y_1 + 1 = 2 + 1 = 3 = \ell^z_1$. For $l = 2$, $\ell_{I_l} = \ell_4 = 12$ is connected to $\ell_{J_l} = \ell_3 = 9$, as $\ell^y_4 + 1 = 13 + 1 = 14 = \ell^z_3$. This example also covers the case where S is interpreted as the usual order on natural numbers, as we have $\ell^y_1 < \ell^z_1 < \ell^u_1$ and $\ell^y_4 < \ell^z_3 < \ell^u_4$.

4. Discussion

The presented result is very tight. Theorem 5 applies to most non trivial theories and the proof only uses very simple data structures (simply linked lists). The proof could be adapted (at the cost of cluttering the presentation) to handle quantifier-free entailments and even simpler inductive systems with at most one predicate atom on the right-hand side of each rule, in the spirit of word automata. The reduction is given for $\kappa = 6$ but it may also be defined with $\kappa = 2$ by encoding tuples as binary trees. Our logic has only one sort of variables, denoting locations, thus one cannot directly describe structures in which the heap maps locations to tuples containing both locations

and data, ranging over disjoint domains. However, data can be easily encoded in our framework by considering a non-injective function $\mathtt{d}(x)$ mapping locations to data, and adding theory predicates constructed on this function, such as $\mathtt{d}(x) \approx \mathtt{d}(y)$ to state that two possibly distinct locations x, y are mapped to the same element. The obtained theory falls within the scope of Theorem 5 (using $\mathtt{d}(x) \approx \mathtt{d}(y)$ as the relation $S(x,y)$), provided the domain of the data is infinite. This shows that entailments with data disjoint from locations are undecidable, even if the theory only contains equations and disequations, except when the data domain is finite.

Acknowledgments. This work has been partially funded by the the French National Research Agency (ANR-21-CE48-0011). The authors wish to thank Radu Iosif for his comments on an earlier version of the paper. The result presented in this paper is also described in a paper accepted for presentation at ASL2022 (workshop on Advancing Separation Logic, with no formal proceedings).

References

[1] R. Iosif, A. Rogalewicz, J. Simacek, The tree width of separation logic with recursive definitions, in: Proc. of CADE-24, Vol. 7898 of LNCS, 2013.

[2] S. S. Ishtiaq, P. W. O'Hearn, Bi as an assertion language for mutable data structures, in: ACM SIGPLAN Notices, Vol. 36, 2001, pp. 14–26.

[3] J. Reynolds, Separation Logic: A Logic for Shared Mutable Data Structures, in: Proc. of LICS'02, 2002.

[4] R. Iosif, A. Rogalewicz, T. Vojnar, Deciding entailments in inductive separation logic with tree automata, in: F. Cassez, J. Raskin (Eds.), ATVA 2014, Proceedings, Vol. 8837 of LNCS, Springer, 2014, pp. 201–218.

[5] J. Pagel, F. Zuleger, Beyond symbolic heaps: Deciding separation logic with inductive definitions, in: LPAR-23, Vol. 73 of EPiC Series in Computing, EasyChair, 2020, pp. 390–408.

[6] R. Piskac, T. Wies, D. Zufferey, Automating Separation Logic using SMT, in: CAV 2013, Proceedings, 2013, pp. 773–789.

[7] X. Qiu, P. Garg, A. Stefanescu, P. Madhusudan, Natural proofs for structure, data, and separation, in: H. Boehm, C. Flanagan (Eds.), ACM SIGPLAN PLDI '13, ACM, 2013, pp. 231–242.

[8] Z. Xu, T. Chen, Z. Wu, Satisfiability of compositional separation logic with tree predicates and data constraints, in: L. de Moura (Ed.), CADE 26, Vol. 10395 of LNCS, Springer, 2017, pp. 509–527.