Entailment is Undecidable for Symbolic Heap Separation Logic Formulae with Non-Established Inductive Rules

Mnacho Echenim, Radu Iosif, Nicolas Peltier

Entailment is Undecidable for Symbolic Heap Separation Logic Formulæ with Non-Established Inductive Rules

Mnacho Echenim, Radu Iosif and Nicolas Peltier

September 30, 2020

Abstract

Entailment is undecidable in general for Separation (SL) Logic formulæ with inductive definitions, but it has been shown to be decidable [1] if the inductive rules satisfy three conditions, namely *progress*, *connectivity* and *establishment*. We show that entailment is undecidable if the latter condition is dropped, thus drawing a much clearer frontier for (un)decidability.

1 Introduction

Separation Logic (SL) is widely used in program verification to reason about programs manipulating dynamically allocated data structures [2].It forms the basis of several industrial-scale static program analyzers [3, 4, 5]. SL formulæ describe *heaps*, i.e. finite partial functions mapping locations (memory addresses) to tuples of locations (records) of a fixed size. A formula $x \mapsto (y_1, \ldots, y_\kappa)$ states that the image of the location associated with x in the heap is the tuple of locations associated with $(y_1, \ldots, y_\kappa)$, whereas $\phi_1 * \phi_2$ states that the heap can be split into two disjoint parts satisfying ϕ_1 and ϕ_2, respectively. Describing unbounded data structures (lists, trees, etc.) is possible by the use of predicate symbols, the interpretation of which is given by a set of user-defined inductive rules. These SL formulæ, in which boolean conjunction does not occur and negation is restricted to disequalities, are commonly called *symbolic heaps*.

Testing entailment between symbolic heaps is known to be decidable for a class of inductive definitions satisfying three natural conditions [1]:(1) *progress*, stating that every inductive rule allocates exactly one memory cell, (2) *connectivity*, ensuring that the locations allocated during the unfolding of a predicate form a tree-like connected structure and (3) *establishment*, stating that all the existentially quantified variables introduced during the unfolding of inductive predicates must be eventually allocated.

These conditions play very different rôles. Condition (1) is actually not restrictive and is considered for simplicity only. In contrast, Condition (2) is crucial for decidability: entailments are undecidable for inductive definitions that are not connected, because then the inductive definitions may be used to encode context-free languages, the inclusion of which is known to be undecidable. The remaining question is whether

condition (3) is actually required for decidability or whether it can be ignored. We show that discarding this condition leads to undecidability.

The precise rôle of establishment for decidability of entailments between symbolic heaps has been an open problem since establishment was identified as a condition ensuring, with progress and connectivity, decidability of entailments between symbolic heaps [1]. The proof of [1] used a reduction of the entailment between symbolic heaps to the (decidable) satisfiability of monadic second order logic on graphs with bounded treewidth [6] and establishment was crucial to ensure that the considered heaps have a bounded treewidth. We recently showed that establishment can be replaced by a strictly more general condition restricting the form of the disequations in the system [7]. The undecidability proof for non-established definitions given here implies the undecidability of entailments for definitions that cannot be transformed into the restricted disequations form [7].

2 Separation Logic

Let $\kappa \geq 0$ be a natural number(denoting the number of record fields), $\mathcal{V}$ and $\mathcal{C}$ be two disjoint sets of symbols (denoting respectively *variables* and *constants*[1], and $\mathcal{P}$ be a set of *predicate symbols*. Each symbol $p \in \mathcal{P}$ is associated with a unique *arity* $\#(p) \geq 1$. The set of *separation logic symbolic heaps* ϕ is inductively defined as follows, where $x, x', y_1, \ldots, y_\kappa, z_1, \ldots, z_{\#(p)} \in \mathcal{V} \cup \mathcal{C}$, $u \in \mathcal{V}$ and $p \in \mathcal{P}$:

$$\phi ::= x \approx x' \mid x \not\approx x' \mid x \mapsto (y_1, \ldots, y_\kappa) \mid \phi * \phi \mid p(z_1, \ldots, z_{\#(p)}) \mid \exists u \,.\, \phi.$$

A formula is *predicate-free* if it contains no symbols from $\mathcal{P}$. We denote by $\mathrm{var}(\phi)$ the set of variables freely occurring in ϕ. A *substitution* is a partial mapping from $\mathcal{V}$ to $\mathcal{V} \cup \mathcal{C}$. The substitution of domain $\{x_1, \ldots, x_n\}$ mapping every variable x_i to t_i is denoted by $[x_i \leftarrow t_i \mid 1 \leq i \leq n]$. For every formula ϕ and substitution σ, $\phi\sigma$ denotes the formula obtained by replacing each free occurrence of a variable x in ϕ by $\sigma(x)$ (bound variables are renamed to avoid collisions).

Let $\mathcal{U}$ be a countably infinite set of *locations*, with $\mathcal{C} \subseteq \mathcal{U}$. A *structure* is a pair $(\mathfrak{s}, \mathfrak{h})$ where $\mathfrak{s}$ is a *store*, i.e., a total mapping from $\mathcal{V} \cup \mathcal{C}$ to $\mathcal{U}$ that is the identity on $\mathcal{C}$; and $\mathfrak{h}$ is a *heap*, i.e., a partial finite mapping from $\mathcal{U}$ to $\mathcal{U}^\kappa$. Two heaps $\mathfrak{h}_1$ and $\mathfrak{h}_2$ are *disjoint* iff $\mathrm{dom}(\mathfrak{h}_1) \cap \mathrm{dom}(\mathfrak{h}_2) = \emptyset$, and in this case $\mathfrak{h}_1 \uplus \mathfrak{h}_2$ denotes the union of disjoint heaps.

A *system of inductive definitions* (SID) $\mathcal{R}$ is a set of rules $p(x_1, \ldots, x_{\#(p)}) \Leftarrow \phi$, where $x_1, \ldots, x_{\#(p)}$ are pairwise distinct variables and ϕ is a symbolic heap with $\mathrm{var}(\phi) \subseteq \{x_1, \ldots, x_{\#(p)}\}$. A system $\mathcal{R}$ is *progressing* and *connected* if every rule in $\mathcal{R}$ is of the form $p(x_1, \ldots, x_{\#(p)}) \Leftarrow \exists y_1, \ldots, y_m \,.\, x_1 \mapsto (u_1, \ldots, u_{\#(p)}) * \ast_{i=1}^{m} p_i(y_1^i, \ldots, y_{\#(p_i)}^i)$, where for all $i \in [\![1, m]\!]$, we have $y_1^i \in \{u_1, \ldots, u_{\#(p)}\}$.

For any formula ϕ, we write $\phi \rightarrow_{\mathcal{R}} \psi$ if ψ is obtained from ϕ by replacing an occurrence of a predicate atom $p(x_1, \ldots, x_{\#(p)})$ by $\rho[y_i \leftarrow x_i \mid 1 \leq i \leq \#(p)]$, where $\mathcal{R}$

[1] Constants are introduced only to improve readability. One could replace them by variables added as parameters to every predicate symbol, and adding disequality constraints $x \not\approx y$ for all $x, y \in \mathcal{C}$, stating that all constants are mapped to distinct locations. Hence the undecidability result holds also for $\mathcal{C} = \emptyset$.

contains a rule $p(y_1,\ldots,y_{\#(p)}) \Leftarrow \rho$ and the existential variables of ρ are α-renamed to avoid collisions. As usual, $\rightarrow^*_{\mathcal{R}}$ denotes the reflexive and transitive closure of $\rightarrow_{\mathcal{R}}$. A formula ψ such that $\phi \rightarrow^*_{\mathcal{R}} \psi$ is called an $\mathcal{R}$-*unfolding* of ϕ. A SID $\mathcal{R}$ is *established* if, in every predicate-free $\mathcal{R}$-unfolding ψ of a predicate atom $p(x_1,\ldots,x_{\#(p)})$, each existentially quantified variable in ψ is asserted equal to a variable x, where $x \mapsto (y_1,\ldots,y_\kappa)$ is a an atom of ψ.

For all structures $(\mathfrak{s},\mathfrak{h})$, symbolic heaps ϕ containing no predicate atom and SIDs $\mathcal{R}$, we write $(\mathfrak{s},\mathfrak{h}) \models \phi$ iff one of the following holds: (i) $\phi \equiv x \approx x'$ (resp. $\phi \equiv x \not\approx x'$), $\mathrm{dom}(\mathfrak{h}) = \emptyset$ and $\mathfrak{s}(x) = \mathfrak{s}(x')$ (resp. $\mathfrak{s}(x) \neq \mathfrak{s}(x')$); (ii) $\phi \equiv x \mapsto (y_1,\ldots,y_\kappa)$, $\mathrm{dom}(\mathfrak{h}) = \{\mathfrak{s}(x)\}$ and $\mathfrak{h}(\mathfrak{s}(x)) = (\mathfrak{s}(y_1),\ldots,\mathfrak{s}(y_\kappa))$; (iii) $\phi \equiv \phi_1 * \phi_2$ and $\mathfrak{h} = \mathfrak{h}_1 \uplus \mathfrak{h}_2$ where $(\mathfrak{s},\mathfrak{h}_i) \models \phi_i$, for all $i \in \{1,2\}$; (iv) $\phi \equiv \exists x \,.\, \phi'$ and there exists a store $\mathfrak{s}'$ coinciding with $\mathfrak{s}$ on all variables and constants distinct from x such that $(\mathfrak{s}',\mathfrak{h}) \models \phi'$. This definition is extended to formulæ containing predicate symbols as follows: $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \phi$ iff $(\mathfrak{s},\mathfrak{h}) \models_{\mathcal{R}} \psi$, for some predicate-free $\mathcal{R}$-unfolding ψ of ϕ.

3 Encoding the Post Correspondence Problem

The undecidability proof uses a reduction from a variant of Post's Correspondence Problem (PCP). We denote by $w[i]$ the i-th character[2] of the word w, and by $|w|$ its length, so that $w = (w[1],\ldots,w[|w|])$. For readability, we shall use the following convention: when a capital letter (say, A) is used to denote a natural number, then the corresponding lower case letter a will denote a natural number ranging over the interval $[\![1,A]\!]$. Further, the variable i will range over the set $\{1,2\}$. We recall that the PCP asks, given two sequences $\mathbf{u}^i = (\mathbf{u}^i_1,\ldots,\mathbf{u}^i_N)$ (with $i = 1,2$) of words on the same alphabet $\mathbb{V}$, whether there exists a nonempty sequence $\mathbf{s} = (\mathbf{s}_1,\ldots,\mathbf{s}_K)$ of elements of $[\![1,N]\!]$ (called a *solution* of the PCP) such that $\mathbf{u}^1_{\mathbf{s}_1}.\ldots.\mathbf{u}^1_{\mathbf{s}_K} = \mathbf{u}^2_{\mathbf{s}_1}.\ldots.\mathbf{u}^2_{\mathbf{s}_K}$. The latter word is called a *witness* of the PCP and its length will be denoted by L. PCP is a well-known undecidable problem. We assume that all the words in $\mathbf{u}^i$ are nonempty (which does not impact the undecidability proof of the PCP). We consider a slightly adapted version of the PCP, in which: (i) The last element of both sequences of words is a special character $\$ = \mathbf{u}^1_N = \mathbf{u}^2_N$, not occurring in the words $\mathbf{u}^i_n$, for $i \in \{1,2\}$, $n \in [\![1,N-1]\!]$; and (ii) any solution must be such that $K > 1$, $\mathbf{s}_K = N$ and $\mathbf{s}_k \neq N$ if $k < K$. In other words, in this adapted version, the witness must end with the special element \$. It is clear that the standard PCP can be reduced to the adapted version, which is thus also undecidable. We denote by M^i the maximal length of the words in $\mathbf{u}^i$, i.e. $M^i \stackrel{\text{def}}{=} \max\{|\mathbf{u}^i_n| \mid n \in [\![1,N]\!]\}$, for $i \in \{1,2\}$ and let $M \stackrel{\text{def}}{=} \max(\{M_1,M_2\})$.

We consider two special constants `nil` and $\bot$, and we associate all natural numbers $m \in [\![1,M]\!]$ and all elements $a \in \mathbb{V}$ with pairwise distinct constants also distinct from `nil` and $\bot$. These constants will be, for the sake of readability, also denoted by m or a (assuming, w.l.o.g., that $\mathbb{V} \cup \mathbb{N} \cup \{\bot,\mathtt{nil}\} \subseteq \mathcal{C}$ and $\mathbb{V} \cap \mathbb{N} = \emptyset$). We assume next that $\kappa = 6$, i.e. we fix the number of record fields. The encoding could easily be adapted for any $\kappa \geq 2$, using binary trees to represent tuples of elements, but this would greatly hinder readability, because of the progress condition which requires that

[2]Since words will often be indexed, using the notation w_i would be confusing.

every rule allocates exactly one location. For readability, we shall actually consider elements referring to tuples of any length $n \leq 6$, where missing elements are implicitly replaced by the constant $\bot$. Hence the notation $x \mapsto (y_1, \ldots, y_n)$ for $n \leq 6$ stands for $x \mapsto (y_1, \ldots, y_n, \bot^{6-n})$. Similarly, $\mathfrak{h}(\ell) = (\ell_1, \ldots, \ell_n)$ stands for $\mathfrak{h}(\ell) = (\ell_1, \ldots, \ell_n, \bot^{6-n})$. For the sake of conciseness, we allow disjunctions in the right-hand side of the inductive rules: a rule $p(\vec{x}) \Leftarrow \phi * (\psi_1 \vee \psi_2)$ stands for $\{p(\vec{x}) \Leftarrow \phi * \psi_1,\ p(\vec{x}) \Leftarrow \phi * \psi_2\}$.

We begin by showing that the solutions of the PCP can be encoded by structures of a specific form.

Definition 1. *A heap $\mathfrak{h}$ is a* PCP-encoding *if there exists a word w with $L \stackrel{\text{def}}{=} |w| > 1$, a natural number $K > 1$ and pairwise distinct locations $\ell_1, \ldots, \ell_{L+1}$ and $\ell'_1, \ldots, \ell'_{K+1}$ with $\ell_{L+1} = \ell'_{K+1} = \mathtt{nil}$, such that:*

1. $\mathrm{dom}(\mathfrak{h}) = \{\ell_1, \ldots, \ell_L, \ell'_1, \ldots, \ell'_K\}$.
2. *For every $l \in [\![1, L]\!]$, $\mathfrak{h}(\ell_l)$ is of the form $(\ell_{l+1}, w[l], \ell^1_l, p^1_l, \ell^2_l, p^2_l)$, for locations ℓ^1_l, ℓ^2_l and integers $p^1_l, p^2_l \in [\![1, M]\!]$, with $\ell^1_1 = \ell^2_1 = \ell'_1$, $\ell^1_L = \ell^2_L = \ell'_K$ and $p^1_1 = p^2_1 = p^1_L = p^2_L = 1$.*
3. *For every $k \in [\![1, K]\!]$, $\mathfrak{h}(\ell'_k)$ is of the form $(\ell'_{k+1}, \mathbf{s}_k)$ for some $\mathbf{s}_k \in [\![1, N]\!]$, with $\mathbf{s}_K = N$ and $\mathbf{s}_k \neq N$ if $k < K$.*

The location ℓ_1 is called the root *of $\mathfrak{h}$. We denote $w(\mathfrak{h}) \stackrel{\text{def}}{=} w$ and $\mathbf{s}(\mathfrak{h}) \stackrel{\text{def}}{=} (\mathbf{s}_1, \ldots, \mathbf{s}_K)$; clearly $\mathfrak{h}$ uniquely defines $w(\mathfrak{h})$ and $\mathbf{s}(\mathfrak{h})$.*

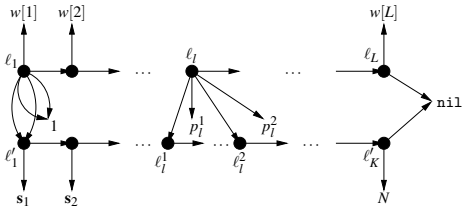

Figure 1: Encoding of a PCP Solution by a Heap

The sequence $\ell_1, \ldots, \ell_L$ encodes the linked sequence of characters $w[1], \ldots, w[L]$ in a witness w. Each location ℓ_l refers to a tuple containing the next element of the sequence ℓ_{l+1} as well as the character $w[l]$, and locations $\ell^1_l, p^1_l, \ell^2_l, p^2_l$, which will be used to denote the position of the character within the words $\mathbf{u}^1_{\mathbf{s}_1} \ldots . \mathbf{u}^1_{\mathbf{s}_K}$ and $\mathbf{u}^2_{\mathbf{s}_1} \ldots . \mathbf{u}^2_{\mathbf{s}_K}$ (see Definition 2). The sequence $\ell'_1, \ldots, \ell'_K$ represents the solution $\mathbf{s}$. Each location ℓ'_k points to a tuple containing the next element of the sequence ℓ'_{k+1} and the k-th component of $\mathbf{s}$ (see Fig. 1). The next definition states conditions ensuring that a PCP-encoding indeed encodes a solution of the PCP.

Definition 2. *A PCP-encoding $\mathfrak{h}$ is i-*consistent *for $i = 1, 2$ if, for all $l \in [\![1, L]\!]$, there exists $k \in [\![1, K]\!]$ such that, with the notations of Definition 1, $\ell^i_l = \ell'_k$ and:*

1. *$w[l]$ is the p^i_l-th character of the word $\mathbf{u}^i_{\mathbf{s}_k}$ (i.e., $w[l] = \mathbf{u}^i_{\mathbf{s}_k}[p^i_l]$).*
2. *If $l \neq L$ and $p^i_l < |\mathbf{u}^i_{\mathbf{s}_k}|$ then $\ell^i_{l+1} = \ell'_k$ and $p^i_{l+1} = p^i_l + 1$.*
3. *If $l \neq L$ and $p^i_l = |\mathbf{u}^i_{\mathbf{s}_k}|$ then $k < K$, $\ell^i_{l+1} = \ell'_{k+1}$ and $p^i_{l+1} = 1$.*

Example 3. *We consider the following instance of the PCP:* $N = 3$, $\mathbf{u}_1^1 = ab$, $\mathbf{u}_2^2 = c$, $\mathbf{u}_1^2 = a$, $\mathbf{u}_2^2 = bc$ *and* $\mathbf{u}_3^1 = \mathbf{u}_3^2 = \$$. *The following heap, of domain* $\{\ell_1, \ell_2, \ell_3, \ell_4, \ell'_1, \ell'_2, \ell'_3\}$ *is a PCP-encoding that is* i*-consistent for* $i = 1, 2$. *It encodes the solution* $(1,2,3)$ *of the above PCP, with witness* $abc\$$. *Locations* $\ell_1, \ell_2, \ell_3, \ell_4$ *encode the witness whereas locations* $\ell'_1, \ell'_2, \ell'_3$ *encode the solution.*

$$\begin{array}{lcllcl} \ell_1 & \mapsto & (\ell_2, a, \ell'_1, 1, \ell'_1, 1) & \ell'_1 & \mapsto & (\ell'_2, 1) \\ \ell_2 & \mapsto & (\ell_3, b, \ell'_1, 2, \ell'_2, 1) & \ell'_2 & \mapsto & (\ell'_3, 2) \\ \ell_3 & \mapsto & (\ell_4, c, \ell'_2, 1, \ell'_2, 2) & \ell'_3 & \mapsto & (\mathtt{nil}, 3) \\ \ell_4 & \mapsto & (\mathtt{nil}, \$, \ell'_3, 1, \ell'_3, 1) & & & \end{array}$$

Lemma 4. *A PCP admits a solution iff there exists a PCP-encoding that is both* 1*-consistent and* 2*-consistent.*

Proof. Assume that the PCP admits a solution $\mathbf{s}$ of length K, with witness w and let $L = |w|$. We construct a PCP-encoding as follows. Let $\ell_1, \ldots, \ell_L, \ell'_1, \ldots, \ell'_K$ be pairwise distinct locations distinct from all constants. Let $\ell_{L+1} \stackrel{\mathsf{def}}{=} \ell'_{K'+1} \stackrel{\mathsf{def}}{=} \mathtt{nil}$, and let $\mathfrak{h}$ be a heap of domain $\{\ell_1, \ldots, \ell_L, \ell'_1, \ldots, \ell'_K\}$, such that: (i) For every $l \in [\![1, L]\!]$, $\mathfrak{h}(\ell_l) \stackrel{\mathsf{def}}{=} (\ell_{l+1}, w[l], \ell'_{k_l^1}, p_l^1, \ell'_{k_l^2}, p_l^2)$, where k_l^i is the greatest element in $[\![1, K]\!]$ with $|\mathbf{u}^i_{\mathbf{s}_1} \ldots . \mathbf{u}^i_{\mathbf{s}_{k_l^i - 1}}| < l$, and $p_l^i \stackrel{\mathsf{def}}{=} l - |\mathbf{u}^i_{\mathbf{s}_1} \ldots . \mathbf{u}^i_{\mathbf{s}_{k_l^i - 1}}|$. (ii) For every $k \in [\![1, K]\!]$, $\mathfrak{h}(\ell'_k) \stackrel{\mathsf{def}}{=} (\ell'_{k+1}, \mathbf{s}_k)$. It is straightforward to check that $\mathfrak{h}$ is a PCP-encoding. Conversely, let $\mathfrak{h}$ be a PCP-encoding that is i-consistent for all $i = 1, 2$, and let $w \stackrel{\mathsf{def}}{=} w(\mathfrak{h})$ and $\mathbf{s} \stackrel{\mathsf{def}}{=} \mathbf{s}(\mathfrak{h})$. By induction on l, we can show, by a case analysis on whether $p_l^i < |\mathbf{u}^i_{\mathbf{s}_{k_l^i}}|$ or not, and using Conditions 2 and 3 of Def. 2 that for every $i \in \{1, 2\}$ and $l \in [\![1, L]\!]$, the following equality holds: $w[1] \ldots . w[l] = \mathbf{u}^i_{\mathbf{s}_1} \ldots . \mathbf{u}^i_{\mathbf{s}_{k_l^i - 1}} . \mathbf{u}^i_{\mathbf{s}_{k_l^i}}[1] \ldots . \mathbf{u}^i_{\mathbf{s}_{k_l^i}}[p_l^i]$ (1). Note that, in particular, for $l = L$, Equation 1 entails that $w = \mathbf{u}^i_{\mathbf{s}_1} \ldots . \mathbf{u}^i_{\mathbf{s}_K}$ (i.e. $\mathbf{s}$ is indeed a solution of the PCP). Indeed, we have $p_L^i = 1$ and $\ell_L^i = \ell'_K$, so that $k_L^i = K$; moreover $\mathbf{s}_K = N$ and $\mathbf{u}_N^i = \$$ is of length 1. The proof goes by induction on l.

- If $l = 1$ then we have $k_1^i = 1$ and $p_1^i = 1$, by Definition 1 (2). Further, $w[1] = \mathbf{u}^i_{\mathbf{s}_1}[1]$, by Condition 1 in Definition 2.
- Assume that Equation 1 holds for some $l < K$. We distinguish two cases.
  - If $p_l^i < |\mathbf{u}^i_{k_l^i}|$, then by Definition 2 (2), applied on l with $k = k_l^i$, this entails that $\ell_{l+1}^i = \ell_l^i$ (hence $k_{l+1}^i = k_l^i$), and $p_{l+1}^i = p_l^i + 1$. Moreover:

    $$w[1] \ldots . w[l+1] = \mathbf{u}^i_{\mathbf{s}_1} \ldots . \mathbf{u}^i_{\mathbf{s}_{k_l^i - 1}} . \mathbf{u}^i_{\mathbf{s}_{k_l^i}}[1] \ldots . \mathbf{u}^i_{\mathbf{s}_{k_l^i}}[p_l^i] . w[l+1].$$

    By Definition 2 (1), applied on $l+1$ with $k = k_{l+1}^i$, we have $w[l+1] = \mathbf{u}^i_{k_{l+1}^i}[p_{l+1}^i] = \mathbf{u}^i_{k_l^i}[p_l^i + 1]$, hence

    $$w[1] \ldots . w[l+1] = \mathbf{u}^i_{\mathbf{s}_1} \ldots . \mathbf{u}^i_{\mathbf{s}_{k_l^i - 1}} . \mathbf{u}^i_{\mathbf{s}_{k_l^i}}[1] \ldots . \mathbf{u}^i_{\mathbf{s}_{k_l^i}}[p_l^i + 1],$$

    and the proof is completed.

– Otherwise, by Definition 2 (3) applied on l with $k = k^i_l$, we have that $k^i_{l+1} = k^i_l + 1$ and $p^i_{l+1} = 1$. By Equation (1), since $p^i_l = |\mathbf{u}^i_{k^i_l}|$, we get:

$$w[1].....w[l] = \mathbf{u}^i_{\mathbf{s}_1}.....\mathbf{u}^i_{\mathbf{s}_{k^i_l}}$$

By Definition 2 (1), applied on $l+1$ with $k = k^i_{l+1}$, we have $w[l+1] = \mathbf{u}^i_{k^i_{l+1}}[p^i_{l+1}] = \mathbf{u}^i_{k^i_l+1}[1]$. Thus:

$$w[1].....w[l+1] = \mathbf{u}^i_{\mathbf{s}_1}.....\mathbf{u}^i_{\mathbf{s}_{k^i_l}}.\mathbf{u}^i_{\mathbf{s}_{k^i_l}}[1],$$

and the proof is completed.

□

Let $\pi = (\mathbf{u}^1, \mathbf{u}^2)$ be an instance of PCP, where $\mathbf{u}^i = (\mathbf{u}^i_1, \ldots, \mathbf{u}^i_N)$, $i = 1,2$. We show how to transform i-consistent PCP-encodings into SL formulæ. Let $\mathcal{R}^\pi$ be the set of rules in Fig. 2, we show that I defines the PCP-encodings of π (Def. 1):

Lemma 5. $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}^\pi} I(x_1)$ *iff* $\mathfrak{h}$ *is a PCP-encoding of root* $\mathfrak{s}(x_1)$.

Proof. $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}^\pi} I(x_1)$ iff $(\mathfrak{s}, \mathfrak{h}) \models \phi$ for some predicate-free formula ϕ with $I(x_1) \to^*_{\mathcal{R}^\pi} \phi$. By Rule (2), $I(x_1) \to^*_{\mathcal{R}^\pi} \phi$ iff there exist $a_1 \in \mathbb{V}$ such that $\exists x_2, y_1, z.\, x_1 \mapsto (x_2, a_1, y_1, 1, y_1, 1) * W(x_2, z) * S(y_1, z) \to^*_{\mathcal{R}^\pi} \phi$. Any $\to_{\mathcal{R}^\pi}$-derivation from $W(x_2, z)$ starts by a sequence of application of Rule (3) followed by one application of (4). Thus $I(x_1) \to^*_{\mathcal{R}^\pi} \phi$ iff there exists $L \in \mathbb{N}, a_1, \ldots, a_{L-1} \in \mathbb{V}$, $p^1_2, p^2_2, \ldots, p^1_{L-1}, p^2_{L-1} \in [\![1, M]\!]$ s.t. $\exists x_2, \ldots, x_L, y_1, z.\, x_1 \mapsto (x_2, a_1, y_1, 1, y_1, 1) * \psi * S(y_1, z) \to^*_{\mathcal{R}^\pi} \phi$, where $\psi \stackrel{\text{def}}{=} *^{L-1}_{l=2} x_l \mapsto (x_{l+1}, a_l, y^1_l, p^1_l, y^2_l, p^2_l) * x_L \mapsto (\mathtt{nil}, \$, z, 1, z, 1)$. Applying Rule (5) $K-1$ times and then Rule (6), we deduce that $I(x_1) \to^*_{\mathcal{R}^\pi} \phi$ iff there exists $L \in \mathbb{N}, K \in \mathbb{N}, a_1, \ldots, a_{L-1} \in \mathbb{V}$, $p^1_2, p^2_2, \ldots, p^1_{L-1}, p^2_{L-1} \in [\![1, M]\!]$, $n_1, \ldots, n_{K-1} \in [\![1, N-1]\!]$ s.t. $\exists x_2, \ldots, x_L, y_1, \ldots, y_K.\, x_1 \mapsto (y, a_1, y_1, 1, y_1, 1) * \psi * \psi' \to^*_{\mathcal{R}} \phi$ where $\psi' \stackrel{\text{def}}{=} *^{K-1}_{k=1} y_k \mapsto (y_{k+1}, n_k) * y_K \mapsto (\mathtt{nil}, N) * y_K \approx z$. Using the previous equivalence, it is straightforward to check that $(\mathfrak{s}, \mathfrak{h}) \models I(x_1)$ iff $\mathfrak{h}$ is a PCP-encoding with root $\mathfrak{s}(x_1)$, where, using the notations of Definition 1, ℓ_l and ℓ'_k are the locations associated to x_l and y_k, respectively. □

Predicate B defines the PCP-encodings that are *not* i-consistent, for some $i = 1,2$ (Def. 2). Such structures necessarily contain locations ℓ_l and ℓ'_k contradicting one of Condition 1, 2 or 3 from Definition 2. Rule (11) is applied when ℓ'_k is such that $k > 1$. It allocates the first element ℓ_1 of the witness, then invokes $A(y)$ to allocate all elements $\ell'_1, \ldots, \ell'_{k-1}$, and the predicate $B(x')$ to allocate the remaining elements. Note that A allocates arbitrary linked structures, with the link on the first record field (the resulting list does not necessarily end with $\mathtt{nil}$). Predicate C is similar to A, but has 6 additional parameters denoting the heap image of the first allocated location. Rule (12) is applied $l-2$ times to allocate locations $\ell_2, \ldots, \ell_{l-1}$ and also to allocate ℓ_1 in the case where $k = 1$ (this case must be considered apart since the sequence $\ell'_1, \ldots, \ell'_{k-1}$ is empty, thus there is no element to allocate). The remaining rules allocate the faulty location ℓ_l. Rules (13-15) handle the case where the heap is not 1-consistent. Rule (13) allocates

a location contradicting Condition 1 in Definition 2. Afterwards, the predicates $A(y)$ and $C(z_1, y', n, \bot, \bot, \bot, \bot)$ are used to allocate locations $\ell_{l+1}, \ldots, \ell_L$, and $\ell'_k, \ldots, \ell'_K$, respectively. Similarly, Rules (14) and (15) allocate a location contradicting Condition 2 or 3 in Definition 2, respectively. The rules corresponding to the case where the heap is 2-consistent are defined in a similar way (they are omitted for conciseness).

Lemma 6. *Let $\mathfrak{h}$ be a PCP-encoding and let $\mathfrak{s}$ be a store such that $\mathfrak{s}(x)$ is the root of $\mathfrak{h}$. The following equivalence statement holds: $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}} B(x)$ iff $\mathfrak{h}$ is not i-consistent, for some $i = 1, 2$.*

Proof. We use the same notations as in Definition 1. **$\Rightarrow$:** Assume that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}^\pi} B(x)$. Then there exists a predicate-free unfolding ϕ of $B(x)$ such that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}^\pi} \phi$. By definition of the rules defining B, this entails that one of rules (13), (14) or (15) (or the corresponding rules for $i = 2$) must have been applied at some point in the derivation, since these are the only rules that can eliminate the predicate. By symmetry we assume that one of the rules corresponding to $i = 1$ is applied. If Rule (13) is applied, then there exist locations $\mathfrak{l}, \mathfrak{l}', \mathfrak{l}'', \mathfrak{l}_1$ and $\mathfrak{l}_2$ such that $\mathfrak{h}(\mathfrak{l}) = (\mathfrak{l}', a, \mathfrak{l}_1, p_1, \mathfrak{l}_2, p_2)$ and $\mathfrak{h}(\mathfrak{l}_1) = (\mathfrak{l}'', n)$, where $a \in \mathbb{V}$, $n \in [\![1, N]\!]$, $a \neq \mathbf{u}_n^1[p_1]$. Since $\mathfrak{h}$ is a PCP-encoding, this is possible only if $\mathfrak{l} = \ell_l$, for some $l \in [\![1, L]\!]$ and $\mathfrak{l}'_1 = \ell'_k$, for some $k \in [\![1, K]\!]$. Then $\mathfrak{h}$ falsifies Condition 1 of Definition 2, hence $\mathfrak{h}$ is not 1-consistent. If Rule (14) is applied then there exist locations $\mathfrak{l}, \mathfrak{l}', \mathfrak{l}'', \mathfrak{l}_1, \mathfrak{l}_2, \mathfrak{l}'_1, \mathfrak{l}'_2$ and $\mathfrak{l}''_1$ such that $\mathfrak{h}(\mathfrak{l}) = (\mathfrak{l}', a, \mathfrak{l}_1, p_1, \mathfrak{l}_2, p_2)$, $\mathfrak{h}(\mathfrak{l}') = (\mathfrak{l}'', a', \mathfrak{l}'_1, p'_1, \mathfrak{l}'_2, p'_2)$ and $\mathfrak{h}(\mathfrak{l}_1) = (\mathfrak{l}''_1, n)$, where $1 \leq p_1 < |\mathbf{u}_n^1|$, and either $\mathfrak{l}_1 \not\approx \mathfrak{l}'_1$ or $p'_1 \neq p_1 + 1$. Since $\mathfrak{h}$ is a PCP-encoding, we must have $\mathfrak{l} = \ell_l$ and $\mathfrak{l}' = \ell_{l+1}$ for some $l \in [\![1, L]\!]$, and $\mathfrak{l}_1 = \ell'_k$, for some $k \in [\![1, K]\!]$. Therefore, Condition 2 of Definition 2 is falsified. The proof is similar if Rule (15) is applied: we have $p_1 = |\mathbf{u}_n^1|$, and either $\mathfrak{l}'_1 \not\approx \mathfrak{l}''_1$ or $p'_1 \neq 1$, which entails that Condition 3 of Definition 2 is falsified.

$\Leftarrow$: Assume that $\mathfrak{h}$ is not i-consistent, for some $i = 1, 2$. We assume that $i = 1$, the proof for $i = 2$ is symmetric. By definition of a PCP-encoding, there exists a store $\mathfrak{s}'$ such that $\mathfrak{s}(x) = \mathfrak{s}'(x)$ and:

$$(\mathfrak{s}', \mathfrak{h}) \models_{\mathcal{R}^\pi} *_{l=1}^{L} x_l \mapsto (x_{l+1}, a_l, y_l^1, p_l^1, y_l^2, p_l^2) * *_{k=1}^{K} y_k \mapsto (y_{k+1}, n_k),$$

where $x_l, y_l^i, y_k \in \mathcal{V}$, $a_l \in \mathbb{V}$, $p_l^i \in [\![1, M]\!]$ and $n_k \in [\![1, N]\!]$, for all $l \in [\![1, L]\!]$, $k \in [\![1, K]\!]$ and $i \in \{1, 2\}$. The location $\mathfrak{s}(x_1)$ is the root of the heap, thus we have $\mathfrak{s}(x) = \mathfrak{s}(x_1)$, and we may assume, w.l.o.g., that $x = x_1$. Since by hypothesis $\mathfrak{h}$ is not 1-consistent, there exist $l \in [\![1, L]\!]$ and $k \in [\![1, K]\!]$ such that one of the conditions in Definition 2 is falsified. Observe that, according to Definition 2, this entails that $\mathfrak{s}(y_l^1) = \mathfrak{s}(y_k)$. We also have $l < L$; indeed, it is easy to check that the faulty location cannot be the last one, since, by definition of a PCP-encoding, this location is associated with the special character \$ and index $n_K = N$, with $\mathbf{u}_N^i = \$$ and $p_L^i = 1$. We therefore have:

$$(\mathfrak{s}', \mathfrak{h}) \models_{\mathcal{R}^\pi} \phi_1 * x_l \mapsto (x_{l+1}, a_l, y_l^1, p_l^1, y_l^2, p_l^2) * \phi_2 * \psi_1 * y_k \mapsto (y_{k+1}, n_k) * \psi_2,$$

where:

$$\phi_1 \stackrel{\text{def}}{=} *_{l'=1}^{l-1} x_{l'} \mapsto (x_{l'+1}, a_{l'}, y_{l'}^1, p_{l'}^1, y_{l'}^2, p_{l'}^2) \qquad \psi_1 \stackrel{\text{def}}{=} *_{k'=1}^{k-1} y_{k'} \mapsto (y_{k'+1}, n_{k'})$$
$$\phi_2 \stackrel{\text{def}}{=} *_{l'=l+1}^{L} x_{l'} \mapsto (x_{l'+1}, a_{l'}, y_{l'}^1, p_{l'}^1, y_{l'}^2, p_{l'}^2) \qquad \psi_2 \stackrel{\text{def}}{=} *_{k'=k+1}^{K} y_{k'} \mapsto (y_{k'+1}, n_{k'})$$

It is clear that $\phi_2 \models_{\mathcal{R}^\pi} A(x_{l+1})$ $(\star)$ using Rule (7) $L-l$ times and Rule (8) once. Further, if $l < L-1$ then $\phi_2 \models_{\mathcal{R}^\pi} x_{l+1} \mapsto (x_{l+2}, a_{l+1}, y_1^{l+1}, p_1^{l+1}, y_2^{l+1}, p_2^{l+1}) * A(x_{l+2})$. We also have $\phi_2 \models_{\mathcal{R}^\pi} C(x_{l+1}, x_{l+2}, a_l+1, y_1^{l+1}, p_1^{l+1}, y_2^{l+1}, p_2^{l+1})$ $(\dagger)$, using Rule (9) if $l < L-1$ and Rule (10) otherwise. Similarly, $\psi_2 \models_{\mathcal{R}^\pi} A(y_{k+1})$ and $y_k \mapsto (y_{k+1}, n_k) * \psi_2 \models_{\mathcal{R}^\pi} C(y_k, y_{k+1}, n_k, \bot, \bot, \bot, \bot)$ $(\ddagger)$. Furthermore, if $k > 1$, then $\psi_1 \models_{\mathcal{R}^\pi} A(y_k)$.

Assume first that Condition 1 is falsified. Then $a_l \neq \mathbf{u}_k^1[p_l^1]$, hence, by Rule (13), using the relations $(\star)$ and $(\ddagger)$, we deduce that $x_l \mapsto (x_{l+1}, a_l, y_1^l, p_1^l, y_2^l, p_2^l) * \phi_2 * y_k \mapsto (y_{k+1}, n_k) * \psi_2 \models_{\mathcal{R}^\pi} B(x_l)$. If Condition 2 or 3 are falsified, then we get the same result, using the rules (14) and (15), instead of (13), and Relation $(\dagger)$ instead of $(\star)$. If $k = 1$, then, by applying Rule (12) $k-1$ times, we deduce that $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}^\pi} B(x_1)$. If $k > 1$, then we get the same result by applying Rule (11) once and Rule (12) $k-1$ times, using the fact that $\psi_1 \models_{\mathcal{R}^\pi} A(y_k)$. □

Putting everything together, we get the stated result:

Theorem 7. *The entailment problem is undecidable for systems of inductive definitions satisfying the progress and connectivity condition, but not the establishment condition.*

Proof. Consider an instance π of the (adapted) PCP. By Lemma 4, the PCP has a solution iff there exists a PCP-encoding that is i-consistent, for $i = 1, 2$. By Lemmas 5 and 6, we can compute a system of inductive rules $\mathcal{R}^\pi$ such that $\mathfrak{h}$ is a PCP-encoding iff $(\mathfrak{s}, \mathfrak{h}) \models_{\mathcal{R}^\pi} I(x_1)$, and $\mathfrak{h}$ is i-consistent for all $i = 1, 2$ iff $(\mathfrak{s}, \mathfrak{h}) \not\models_{\mathcal{R}^\pi} B(x_1)$; i.e., such that the PCP has a solution iff $I(x_1) \not\models_{\mathcal{R}^\pi} B(x_1)$. It is easy to check that $\mathcal{R}^\pi$ is progressing and connected but not established, see for instance Rules (7), (8) and (12) in Fig. 2. Since the PCP is undecidable, the proof is completed. □

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

[3] C. Calcagno, D. Distefano, J. Dubreil, D. Gabi, P. Hooimeijer, M. Luca, P. W. O'Hearn, I. Papakonstantinou, J. Purbrick, D. Rodriguez, Moving fast with software verification, in: K. Havelund, G. J. Holzmann, R. Joshi (Eds.), NASA Formal Methods - 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings, Vol. 9058 of LNCS, Springer, 2015, pp. 3–11.

[4] J. Berdine, B. Cook, S. Ishtiaq, Slayer: Memory safety for systems-level code, in: G. G. andShaz Qadeer (Ed.), Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings, Vol. 6806 of LNCS, Springer, 2011, pp. 178–183.

[5] K. Dudka, P. Peringer, T. Vojnar, Predator: A practical tool for checking manipulation of dynamic data structures using separation logic, in: G. Gopalakrishnan, S. Qadeer (Eds.), Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings, Vol. 6806 of LNCS, Springer, 2011, pp. 372–378.

[6] B. Courcelle, The monadic second-order logic of graphs. I. Recognizable sets of finite graphs, Information and Computation 85 (1) (1990) 12 – 75.

[7] M. Echenim, R. Iosif, N. Peltier, Decidable entailments in separation logic with inductive definitions: Beyond established systems, CoRR abs/2007.00502.
URL https://arxiv.org/abs/2007.00502

$$I(x) \Leftarrow \exists x', y, z \,.\, x \mapsto (x', \mathtt{a}, y, 1, y, 1) * W(x', z) * S(y, z) \tag{2}$$

$$W(x, z) \Leftarrow \exists x', y_1, y_2 \,.\, x \mapsto (x', \mathtt{a}, y_1, \mathtt{p}_1, y_2, \mathtt{p}_2) * W(x', z) \tag{3}$$
$$W(x, z) \Leftarrow x \mapsto (\mathtt{nil}, \$, z, 1, z, 1) \tag{4}$$

$$S(y, z) \Leftarrow \exists y' \,.\, y \mapsto (y', \mathtt{n}) * S(y, z) \quad \text{if } \mathtt{n} < N \tag{5}$$
$$S(y, z) \Leftarrow y \mapsto (\mathtt{nil}, N) * y \approx z \tag{6}$$

$$A(x) \Leftarrow \exists x', a, z_1, p_1, z_2, p_2 \,.\, x \mapsto (x', a, z_1, p_1, z_2, p_2) * A(x') \tag{7}$$
$$A(x) \Leftarrow \exists x', a, z_1, p_1, z_2, p_2 \,.\, x \mapsto (x', a, z_1, p_1, z_2, p_2) \tag{8}$$

$$C(x, x', a, z_1, p_1, z_2, p_2) \Leftarrow x \mapsto (x', a, z_1, p_1, z_2, p_2) * A(x') \tag{9}$$
$$C(x, x', a, z_1, p_1, z_2, p_2) \Leftarrow x \mapsto (x', a, z_1, p_1, z_2, p_2) \tag{10}$$

$$B(x) \Leftarrow \exists x', a, y \,.\, x \mapsto (x', a, y, 1, y, 1) * B(x') * A(y) \tag{11}$$
$$B(x) \Leftarrow \exists x', a, z_1, p_1, z_2, p_2 \,.\, x \mapsto (x', a, z_1, p_1, z_2, p_2) * B(x') \tag{12}$$
$$\begin{aligned} B(x) \Leftarrow\ & \exists x', a, z_1, z_2, p_2, z_1' \,.\, x \mapsto (x', a, z_1, \mathtt{p}_1, z_2, p_2) * A(x') \\ & * C(z_1, z_1', \mathtt{n}, \bot, \bot, \bot, \bot) * a \not\approx \mathbf{u}^1_n[p_1] \ \text{ if } p_1 \leq |\mathbf{u}^1_{\mathtt{n}}| \end{aligned} \tag{13}$$
$$\begin{aligned} B(x) \Leftarrow\ & \exists x', a, z_1, z_2, p_2, a', z_1', p_1', z_2', p_2', z_1'' \,.\, x \mapsto (x', a, z_1, \mathtt{p}_1, z_2, p_2) \\ & * C(x', a', z_1', p_1', z_2', p_2') * C(z_1, z_1'', \mathtt{n}, \bot, \bot, \bot, \bot) \\ & * (z_1' \not\approx z_1 \vee p_1' \not\approx p_1 + 1) \ \text{ if } p_1 < |\mathbf{u}^1_{\mathtt{n}}| \end{aligned} \tag{14}$$
$$\begin{aligned} B(x) \Leftarrow\ & \exists x', a, z_1, z_2, p_1', p_2, a', z_1', z_2', p_2', z_1'' \,.\, x \mapsto (x', a, z_1, \mathtt{p}_1, z_2, p_2) \\ & * C(x', a', z_1', p_1', z_2', p_2') * C(z_1, z_1'', \mathtt{n}, \bot, \bot, \bot, \bot) \\ & * (p_1' \not\approx 1 \vee z_1' \not\approx z_1'') \ \text{ if } p_1 = |\mathbf{u}^1_{\mathtt{n}}| \end{aligned} \tag{15}$$
$$\begin{aligned} B(x) \Leftarrow\ & \exists x', a, z_1, p_1, z_2, z_2' \,.\, x \mapsto (x', a, z_1, p_1, z_2, \mathtt{p}_2) * A(x') \\ & * C(z_2, z_2', \mathtt{n}, \bot, \bot, \bot, \bot) * a \not\approx \mathbf{u}^2_n[p_2] \ \text{ if } \mathtt{p}_2 \leq |\mathbf{u}^2_{\mathtt{n}}| \end{aligned} \tag{16}$$
$$\begin{aligned} B(x) \Leftarrow\ & \exists x', a, z_1, z_2, p_2', p_2, a', z_1', p_1', z_2', p_2', z_2'' \,.\, x \mapsto (x', a, z_1, p_1, z_2, \mathtt{p}_2) \\ & * C(x', a', z_1', p_1', z_2', p_2') * C(z_2, z_2'', \mathtt{n}, \bot, \bot, \bot, \bot) \\ & * (z_2' \not\approx z_2 \vee p_2' \not\approx p_2 + 1) \ \text{ if } \mathtt{p}_2 < |\mathbf{u}^2_{\mathtt{n}}| \end{aligned} \tag{17}$$
$$\begin{aligned} B(x) \Leftarrow\ & \exists x', a, z_1, z_2, a', z_1', z_2', p_2', z_2'' \,.\, x \mapsto (x', a, z_1, p_1, z_2, \mathtt{p}_2) \\ & * C(x', a', z_1', p_1', z_2', p_2') * C(z_2, z_2'', \mathtt{n}, \bot, \bot, \bot, \bot) \\ & * (p_2' \not\approx 1 \vee z_2' \not\approx z_2'') \ \text{ if } \mathtt{p}_2 = |\mathbf{u}^2_{\mathtt{n}}| \end{aligned} \tag{18}$$

Figure 2: The SID $\mathcal{R}^{\pi}$ for the PCP Instance $\pi = (\mathbf{u}^1, \mathbf{u}^2)$ (for all $\mathtt{a} \in \mathbb{V}$, $\mathtt{p}_1, \mathtt{p}_2 \in [\![1, M]\!]$, $\mathtt{n} \in [\![1, N]\!]$)