On Symbolic Heaps Modulo Permission Theories

S. Demri[1], E. Lozes[1], and D. Lugiez[1,2]

1 **LSV, ENS Paris-Saclay, CNRS, Université Paris-Saclay, France**
{demri,lozes}@lsv.fr
2 **Aix-Marseille Univ, LIF, CNRS, Marseille, France**
denis.lugiez@univ-amu.fr

Abstract

We address the entailment problem for separation logic with symbolic heaps admitting list predicates and permissions for memory cells that are essential to express ownership of a heap region. In the permission-free case, the entailment problem is known to be in P. Herein, we design new decision procedures for solving the satisfiability and entailment problems that are parameterised by the permission theories. This permits the use of solvers dealing with the permission theory at hand, independently of the shape analysis. We also show that the entailment problem without list predicates is coNP-complete for several permission models, such as counting permissions and binary tree shares but the problem is in P for fractional permissions. Furthermore, when list predicates are added, we prove that the entailment problem is coNP-complete when the entailment problem for permission formulae is in coNP, assuming the write permission can be split into as many read permissions as desired. Finally, we show that the entailment problem for any Boolean permission model with infinite width is coNP-complete.

1998 ACM Subject Classification D.2.4, Software/Program Verification, F.3. Logics and Meaning of Programs

Keywords and phrases separation logic, entailment, permission, reasoning modulo theories

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2017.25

1 Introduction

Separation logics with permissions. In program verification, proving properties of the memory is one of the most difficult tasks and separation logic has been devised for this goal [14]. Separation logic with permissions [4] can express that the ownership of a given heap region is shared with other threads. A permission can be thought of as a "quantity of ownership" associated to each cell of the heap. This quantity prescribes whether write accesses are allowed or not on this cell and how such a write access may be restored in the future. This abstract notion has lead to many permission theories and separation logics, including fractional permissions [5], token-based permissions [4], combinations of the two, binary tree shares [7], and yet some other models. Separation logic with permissions is supported by several tools like VeriFast [12], Hip/Sleek [11], or Heap-Hop [16]. Usually, these tools support only one permission model and demand that permissions are explicit values. For instance, in a tool that supports fractional permissions, to express that a cell x is shared by two threads for read access, one may write $x \overset{0.3}{\mapsto} y$ and $x \overset{0.7}{\mapsto} y$ making an arbitrary choice for permissions (0.3 and 0.7) when a better approach would use $x \overset{\alpha}{\mapsto} y$ and $x \overset{\beta}{\mapsto} y$ and the constraint $\mathbf{1} = \alpha + \beta$ (as it is done in the paper). This hides the logical structure of the proof and ties it to a specific arbitrary permission model.

Our motivations. We wish to get rid of the use of explicit permission models and to provide a separation logic with permissions which can use symbolic permission expressions


© S. Demri, E. Lozes and D. Lugiez;
licensed under Creative Commons License CC-BY



such as $\mathbf{1} = \alpha + \beta$. Furthermore, we aim at lifting the results obtained so far for separation logic with lists but without permissions to separation logic with lists and symbolic permissions.

Our contributions. We devise a separation logic based on symbolic heaps with list predicates [3] modulo an unspecified permission theory (containing separation logic without permission as an instance). As far as we know, a uniform treatment with both features is new. We give generic decision procedures modulo a permission theory $\mathfrak{P}$ for the satisfiability problem SATSH($\mathfrak{P}$), and for the entailment problem ENTSH($\mathfrak{P}$). Then we simply instantiate the permission theory by the desired theory (fractional models, token model, binary tree-share model,...). This approach has many advantages: (a) the reasoning on the spatial part is separated from the reasoning on permissions, (b) the latter part can be discharged to a dedicated solver, for instance any SMT specialised in the relevant permission theory (see e.g. [1]; and also [13] for the fractional case), and (c) we obtain optimal worst-case complexity results (obviously, the whole complexity depends on the complexity of the permission theory of interest). Since our logic contains the constant $\top$, we can treat both the intuitionistic and the non-intuitionistic case of the entailment problem in a uniform setting, see e.g. [6, 10]. Let us detail more precisely the technical contributions as well as the plan of the paper.

Outline of the paper. Permissions and separation logic with lists and permissions are introduced in Section 2. In Section 3, we treat separation logic with permissions but without lists and we give PTIME algorithms for SATSH($\mathfrak{P}$) and ENTSH($\mathfrak{P}$) using an oracle for the corresponding problems on permission theories. As a byproduct, SATSH($\mathfrak{P}_{\mathsf{Boy}}$) and ENTSH($\mathfrak{P}_{\mathsf{Boy}}$) without list predicates are in PTIME for the fractional model $\mathfrak{P}_{\mathsf{Boy}}$. In Section 4, we prove that SATSH($\mathfrak{P}$) is NP-hard and that ENTSH($\mathfrak{P}$) is coNP-hard even for a permission theory $\mathfrak{P}$ which is in PTIME, showing a complexity gap between the logic without permissions and the logic with permissions. At the end of Section 4, we design a non-deterministic polynomial-time procedure solving ENTSH($\mathfrak{P}$) (fully parametrised by the entailment problem for the permission theory $\mathfrak{P}$). A key ingredient is the notion of SL-graphs that are used to abstract formulae and several variants of homomorphisms between graphs used to prove the entailment property. This approach is clearly inspired by [6] but on one hand we can take advantage of nondeterminism since there is little hope for a PTIME algorithm, and on the other hand permissions lead to technical complications (such as the need to respect the linearisation induced by an SL-graph). In Section 5, we give our results on permission theories: (i) the fractional model $\mathfrak{P}_{\mathsf{Boy}}$ has PTIME satisfiability and entailment problems, (ii) we introduce the notion of Boolean permission models $\mathfrak{P}_{\mathbb{B}}$ that encompasses all classical permission models but the trivial one and $\mathfrak{P}_{\mathsf{Boy}}$ and (iii) we prove that SAT($\mathfrak{P}_{\mathbb{B}}$) is NP-complete and ENT($\mathfrak{P}_{\mathbb{B}}$) is coNP-complete in Boolean permission models $\mathfrak{P}_{\mathbb{B}}$ that have an infinite width (which is the case for the aforementionned models). Section 6 concludes the paper.

2 Preliminaries

We introduce permission formulae and permission models which are the building blocks for defining symbolic heaps with permissions and their related decision problems.

2.1 Permission models

Permission formulae are defined by the grammar below:

$$\begin{array}{lll} p ::= & \mathbf{1} \mid \alpha \mid p \oplus p & \text{(permission term)} \\ A ::= & \top \mid p = p \mid p \leq p \mid \mathsf{defined}(p) \mid A \wedge A & \text{(permission formula)} \end{array}$$

where $\mathtt{PVar} = \{\alpha, \beta, \dots\}$ is a countably infinite set of **permission variables**. Permission formulae are interpreted in permission models, defined below.

▶ **Definition 1.** A **permission model** is a tuple $\mathfrak{P} = (P_\mathfrak{P}, \mathbf{1}_\mathfrak{P}, \oplus_\mathfrak{P})$ such that

- $P_\mathfrak{P} = \{\pi, \dots\}$ is a set of **permissions**,
- $\mathbf{1}_\mathfrak{P} \in P_\mathfrak{P}$ is a distinguished permission called the **write** permission or the **total** permission,
- $\oplus_\mathfrak{P} : P_\mathfrak{P} \times P_\mathfrak{P} \to P_\mathfrak{P}$ is a partial composition that is cancellative, commutative and associative,[1]
- the relation $<_\mathfrak{P} \stackrel{\text{def}}{=} \{(\pi', \pi) \mid \pi = \pi' \oplus_\mathfrak{P} \pi'' \text{ for some } \pi''\}$ is irreflexive and transitive, with maximum element $\mathbf{1}_\mathfrak{P}$.

An example of permission model is Boyland's fractional model $\mathfrak{P}_{\mathsf{Boy}} = ((0,1], 1, \oplus_{\mathfrak{P}_{\mathsf{Boy}}})$ [5], where $\pi \oplus_{\mathfrak{P}_{\mathsf{Boy}}} \pi' \stackrel{\text{def}}{=} \pi + \pi'$ is defined when the sum is at most 1. The **width** of a permission model $\mathfrak{P}$ is $\mathsf{width}(\mathfrak{P}) \in \mathbb{N} \cup \{\omega\}$ such that $\mathsf{width}(\mathfrak{P}) \stackrel{\text{def}}{=} \sup\{n \geq 1 \mid \exists\, \pi_1, \dots, \pi_n \in P_\mathfrak{P} \text{ such that } \pi_1 \oplus \dots \oplus \pi_n = \mathbf{1}_\mathfrak{P}\}$.

Given $\mathfrak{P} = (P_\mathfrak{P}, \mathbf{1}_\mathfrak{P}, \oplus_\mathfrak{P})$, a **$\mathfrak{P}$-interpretation** is a map $\iota : \mathtt{PVar} \to P_\mathfrak{P}$. The map ι is extended to a partial map from the set of permission terms to $P_\mathfrak{P}$ so that $\iota(\mathbf{1}) \stackrel{\text{def}}{=} \mathbf{1}_\mathfrak{P}$ and $\iota(p \oplus p') \stackrel{\text{def}}{=} \iota(p) \oplus_\mathfrak{P} \iota(p')$ if $\iota(p)$, $\iota(p')$ and $\iota(p) \oplus_\mathfrak{P} \iota(p')$ are defined. Otherwise $\iota(p \oplus p')$ is undefined. We may write $[\![p]\!]_\iota$ for $\iota(p)$ and $\iota \models A$ to denote that ι satisfies the permission formula A, following the clauses below:

- always $\iota \models \top$; $\iota \models \mathsf{defined}(p) \stackrel{\text{def}}{\Leftrightarrow} \iota(p)$ is defined; $\iota \models A \wedge A' \stackrel{\text{def}}{\Leftrightarrow} \iota \models A$ and $\iota \models A'$.
- $\iota \models p = p' \stackrel{\text{def}}{\Leftrightarrow} \iota$ is defined for both p and p' and $\iota(p) = \iota(p')$.
- $\iota \models p \leq p' \stackrel{\text{def}}{\Leftrightarrow} \iota$ is defined for both p and p' and, either $\iota(p) = \iota(p')$ or $\iota(p) <_\mathfrak{P} \iota(p')$.

For example, the permission formula $\alpha \oplus \alpha = \mathbf{1}$, is satisfied by the $\mathfrak{P}_{\mathsf{Boy}}$-interpretation ι defined by $\iota(\alpha) = 0.5$. We write $\bot$ to denote $\mathbf{1} \oplus \mathbf{1} = \mathbf{1}$. Observe that $\iota \not\models \bot$ for all ι (by irreflexivity of $<_\mathfrak{P}$).

2.2 Separation logic with permissions

A **symbolic heap** with list predicates and symbolic permissions is a formula (Π, Σ) where Π is a **pure formula** and Σ a **spatial formula** according to the grammar below:

$$\begin{array}{lll} \Pi ::= & \top \mid x = y \mid x \neq y \mid A \mid \Pi \wedge \Pi & \text{(pure formula)} \\ \Sigma ::= & \mathsf{emp} \mid \top \mid x \overset{p}{\mapsto} y \mid \mathsf{lseg}_p(x, y) \mid \Sigma * \Sigma & \text{(spatial formula)} \end{array}$$

where $\mathtt{LVAR} = \{x, y, \dots\}$ is a countably infinite set of **location/program variables**. We write Π_{pe} and Π_{pv} to denote respectively the permission constraints and the program variable constraints that appear in Π, so that Π is logically equivalent to $\Pi_{pe} \wedge \Pi_{pv}$. We write $\mathtt{LVAR}(\varphi)$ [resp. $\mathtt{PVar}(\varphi)$] to denote the set of location [resp. permission] variables occurring in φ.

▶ **Example 2.** The following symbolic heaps are used throughout the paper.
$(\Pi_1, \Sigma_1) \stackrel{\text{def}}{=} (\bigwedge_{1 \leq i < j \leq 3} x_i \neq x_j, \mathsf{lseg}_\alpha(x_1, x_3))$
$(\Pi_2, \Sigma_2) \stackrel{\text{def}}{=} (\bigwedge_{1 \leq i < j \leq 3} x_i \neq x_j, \mathsf{lseg}_\alpha(x_1, x_2) * \mathsf{lseg}_\alpha(x_2, x_3))$
$(\Pi_3, \Sigma_3) \stackrel{\text{def}}{=} (\bigwedge_{1 \leq i < j \leq 3} x_i \neq x_j, \mathsf{lseg}_{\alpha \oplus \alpha}(x_1, x_3) * \mathsf{lseg}_\alpha(x_3, x_2) * \mathsf{lseg}_\alpha(x_2, x_1))$.

Let $\mathfrak{P} = (P_\mathfrak{P}, \mathbf{1}_\mathfrak{P}, \oplus_\mathfrak{P})$ be a fixed permission model and let $\mathsf{Loc} = \{\ell, \dots\}$ be a countably infinite set of locations (by default, $\mathsf{Loc} = \mathbb{N}$). A **$\mathfrak{P}$-memory state** is a triple (s, h, ι) where:

[1] in particular, whenever a sum $\pi_1 \oplus_\mathfrak{P} \pi_2 \dots \oplus_\mathfrak{P} \pi_n$ is defined, each subsum $\pi_{i_1} \oplus_\mathfrak{P} \dots \oplus_\mathfrak{P} \pi_{i_k}$ for each $\{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$ is defined.

- s is a store i.e. a function $s : \mathtt{LVAR} \to \mathsf{Loc}$ that assigns to each variable a location,
- h is a $\mathfrak{P}$**-heap** i.e. a partial function with a finite domain $h : \mathsf{Loc} \rightharpoonup_{\mathsf{fin}} P_{\mathfrak{P}} \times \mathsf{Loc}$,
- ι is a $\mathfrak{P}$-interpretation.

Intuitively, $h(\ell) = (\pi, \ell')$ holds if the cell at address ℓ is allocated and points to the location ℓ', and that the thread that owns ℓ has permission π on the cell ℓ.

Before defining the semantics of symbolic heaps, we define the composition of $\mathfrak{P}$-heaps. The **composition** $h_1 \bullet h_2$ of two $\mathfrak{P}$-heaps h_1 and h_2 is defined whenever there is no $\ell \in \mathrm{dom}(h_1) \cap \mathrm{dom}(h_2)$ with $h_1(\ell) = (\pi_1, \ell_1)$ and $h_2(\ell) = (\pi_2, \ell_2)$ such that either $\ell_1 \neq \ell_2$ or $\pi_1 \oplus_{\mathfrak{P}} \pi_2$ is undefined. When $h_1 \bullet h_2$ is defined, say equal to the $\mathfrak{P}$-heap h, it takes the unique value satisfying the conditions below:

- if $\ell \notin \mathrm{dom}(h_1) \cup \mathrm{dom}(h_2)$, then $\ell \notin \mathrm{dom}(h)$,
- if $\ell \in \mathrm{dom}(h_i) \setminus \mathrm{dom}(h_j)$, then $\ell \in \mathrm{dom}(h)$ and $h(\ell) = h_i(\ell)$ (for all $i \neq j \in \{1, 2\}$),
- if $\ell \in \mathrm{dom}(h_1) \cap \mathrm{dom}(h_2)$, then $\ell \in \mathrm{dom}(h)$ and $h(\ell) = (\pi_1 \oplus_{\mathfrak{P}} \pi_2, \ell')$ with $h_1(\ell) = (\pi_1, \ell')$, $h_2(\ell) = (\pi_2, \ell')$ and $\pi_1 \oplus_{\mathfrak{P}} \pi_2$ is defined (otherwise $\ell \notin \mathrm{dom}(h)$).

The composition of heaps is partial, commutative, associative, and cancellative. We write $h' \sqsubseteq h$ if there is h'' so that $h = h' \bullet h''$ and we also write $h' \sqsubset h$ whenever $h' \sqsubseteq h$ and $h' \neq h$. The satisfaction relations $s, h, \iota \models_{\mathfrak{P}} \Sigma$ or $s, h, \iota \models_{\mathfrak{P}} \Pi$ are defined below:

$s, h, \iota \models_{\mathfrak{P}} \top$	always	
$s, h, \iota \models_{\mathfrak{P}} x = y$	iff	$s(x) = s(y)$
$s, h, \iota \models_{\mathfrak{P}} x \neq y$	iff	$s(x) \neq s(y)$
$s, h, \iota \models_{\mathfrak{P}} A$	iff	$\iota \models A$
$s, h, \iota \models_{\mathfrak{P}} \Pi_1 \wedge \Pi_2$	iff	$s, h, \iota \models_{\mathfrak{P}} \Pi_1$ and $s, h, \iota \models_{\mathfrak{P}} \Pi_2$
$s, h, \iota \models_{\mathfrak{P}} \mathtt{emp}$	iff	$\mathrm{dom}(h) = \emptyset$
$s, h, \iota \models_{\mathfrak{P}} x \overset{p}{\mapsto} y$	iff	$\mathrm{dom}(h) = \{s(x)\}$, $[\![p]\!]_\iota$ is defined, and $h(s(x)) = ([\![p]\!]_\iota, s(y))$
$s, h, \iota \models_{\mathfrak{P}} \mathsf{lseg}_p(x, y)$	iff	$[\![p]\!]_\iota$ is defined, and either $(s(x) = s(y)$ and $\mathrm{dom}(h) = \emptyset)$
		or $h = \{\ell_0 \mapsto ([\![p]\!]_\iota, \ell_1), \ell_1 \mapsto ([\![p]\!]_\iota, \ell_2), \ldots, \ell_{n-1} \mapsto ([\![p]\!]_\iota, \ell_n)\}$
		with $n \geq 1$, $\ell_0 = s(x)$, $\ell_n = s(y)$ and for all $i \neq j \in [0, n]$, $\ell_i \neq \ell_j$
$s, h, \iota \models_{\mathfrak{P}} \Sigma_1 * \Sigma_2$	iff	there are subheaps h_1, h_2 such that
		$h = h_1 \bullet h_2$, $s, h_1, \iota \models_{\mathfrak{P}} \Sigma_1$, and $s, h_2, \iota \models_{\mathfrak{P}} \Sigma_2$.

Our definitions of symbolic heaps and models encompass the standard definitions without permissions: choose $\mathfrak{P}_{\mathbf{1}} = (\{\mathbf{1}\}, \mathbf{1}, \oplus_{\mathfrak{P}_{\mathbf{1}}})$ that has only the write permission and the always undefined composition. By way of example, given the permission model $\mathfrak{P}_{\mathsf{Boy}}$, let (s, h, ι) be defined by $s(x_1) = 1, s(x_2) = 2, s(x_3) = 3$, $h = \{1 \mapsto (0.5, 3), 3 \mapsto (0.25, 2), 2 \mapsto (0.25, 1)\}$ and $\iota(\alpha) = 0.25$. Then, taking the (Π_i, Σ_i)'s from Example 2, we have $s, h, \iota \not\models_{\mathfrak{P}_{\mathsf{Boy}}} (\Pi_1, \Sigma_1)$ but $s, h, \iota \models_{\mathfrak{P}_{\mathsf{Boy}}} (\Pi_2, \Sigma_2)$ (since 0.5 can be split into $0.25 + 0.25$) and $s, h, \iota \models_{\mathfrak{P}_{\mathsf{Boy}}} (\Pi_3, \Sigma_3)$.

Satisfiability and entailment problems. A symbolic heap (Π, Σ) is $\mathfrak{P}$**-satisfiable** if there is a $\mathfrak{P}$-memory state (s, h, ι) such that $s, h, \iota \models_{\mathfrak{P}} \Pi$ and $s, h, \iota \models_{\mathfrak{P}} \Sigma$ and we say that (s, h, ι) is a $\mathfrak{P}$**-model** of (Π, Σ). Two symbolic heaps (Π, Σ) and (Π', Σ') are equivalent, written $(\Pi, \Sigma) \equiv_{\mathfrak{P}} (\Pi', \Sigma')$, if they have the same $\mathfrak{P}$-models. The **satisfiability problem w.r.t.** $\mathfrak{P}$, written SATSH($\mathfrak{P}$), takes as input (Π, Σ) and asks whether (Π, Σ) has a $\mathfrak{P}$-model. The **entailment problem w.r.t.** $\mathfrak{P}$ written ENTSH($\mathfrak{P}$)) takes as input two symbolic heaps (Π, Σ) and (Π', Σ') and asks whether every $\mathfrak{P}$-model of (Π, Σ) is a $\mathfrak{P}$-model of (Π', Σ') (written $(\Pi, \Sigma) \models_{\mathfrak{P}} (\Pi', \Sigma')$). In the paper, the decision procedures are parameterised by the corresponding problems in the permission model $\mathfrak{P}$, written SAT($\mathfrak{P}$) and ENT($\mathfrak{P}$).

$$
\begin{array}{lllll}
(\textsc{Subst}) & (\Pi, \Sigma) & \Longrightarrow & (\Pi, \Sigma[y/x]) & \text{if } \Pi \models x = y, \{x, y\} \subseteq \mathtt{LVAR}(\Sigma) \\
(\textsc{Merge}) & (\Pi, \Sigma * x \overset{p}{\mapsto} y * x \overset{p'}{\mapsto} z) & \Longrightarrow & (\Pi \wedge y = z, \Sigma * x \overset{p \oplus p'}{\mapsto} y) & \\
(\textsc{Fail}) & (\Pi, \Sigma) & \Longrightarrow & \bot \quad \text{if } \Pi_{pv} \models x \neq y \text{ and } \Pi_{pv} \models x = y & \\
(\textsc{Empty}) & (\Pi, \Sigma * \mathtt{emp}) & \Longrightarrow & (\Pi, \Sigma) \quad \text{if non-empty } \Sigma & \\
(\textsc{True}) & (\Pi, \Sigma * \top * \top) & \Longrightarrow & (\Pi, \Sigma * \top) & \\
(\textsc{Mergelist}) & (\Pi, \Sigma * \mathsf{lseg}_p(x, y) * \mathsf{lseg}_{p'}(x, y) & \Longrightarrow & (\Pi, \Sigma * \mathsf{lseg}_{p \oplus p'}(x, y)) &
\end{array}
$$

Figure 1 Rewrite system R

3 Reasoning Modulo Permission Theories Without List Predicates

3.1 Normalising formulae

In Figure 2, we present a set R of rewrite rules that are used to normalise formulae. The reduction $\Longrightarrow$ is the rewrite relation associated to R and $\Longrightarrow^*$ is its reflexive and transitive closure. Note that if a rewrite sequence starts from a symbolic heap not containing an expression $\mathsf{lseg}_p(x, y)$, then the rule Mergelist never applies. We write $|(\Pi, \Sigma)|$ to denote the **size** of the symbolic heap for some reasonably succinct encoding.

▶ **Lemma 3.** *The rewrite relation $\Longrightarrow$ has the following properties.*
- *If $(\Pi, \Sigma) \Longrightarrow (\Pi', \Sigma')$ then $(\Pi, \Sigma) \equiv (\Pi', \Sigma')$.*
- *Any rewrite sequence starting from (Π, Σ) terminates in time $\mathcal{O}(|(\Pi, \Sigma)|)$.*

The proof of the first part is a direct analysis of the rules according to the semantics of symbolic heaps and the termination proof is straightforward. Note that the rule Subst cannot be applied indefinitely because of the second side-condition. From now on, unless otherwise stated, **normal form** refers to a normal form with respect to $\Longrightarrow$.

3.2 Satisfiability and entailment for symbolic heaps without lists

We give our first results for symbolic heaps with permission but without lists. In the rest of this section, we consider symbolic heaps without lists. Given a spatial formula Σ, we denote by $\mathsf{defined}(\Sigma)$ the conjunction of formulae $\mathsf{defined}(p)$ for all p occuring in Σ.

▶ **Lemma 4.** *Given (Π, Σ) in normal form, Π, Σ is satisfiable iff $\Pi_{pe} \wedge \mathsf{defined}(\Sigma)$ is satisfiable.*

Consequently, we can provide complexity upper bounds for SATSH($\mathfrak{P}$).

▶ **Theorem 5.** *Let $\mathfrak{P}$ be a permission model and $\mathcal{C} \supseteq$ PTime be a complexity class such that* SAT($\mathfrak{P}$) *is in $\mathcal{C}$. Then* SATSH($\mathfrak{P}$) *restricted to symbolic heaps without list predicates is in $\mathcal{C}$.*

Let us now address the entailment problem $(\Pi_l, \Sigma_l) \models (\Pi_r, \Sigma_r)$. First, we restrict our attention to instances of the form $(\Pi_l, \Sigma_l) \models (\top, \Sigma_r)$, where (Π_l, Σ_l) *is in normal form.* An entailment $(\Pi_l, \Sigma_l) \models (\top, \Sigma_r)$ holds if there is a map from the points-to predicates $x' \overset{p'}{\mapsto} y'$ that occur in Σ_r to the $x \overset{p}{\mapsto} y$ that occur in Σ_l, such that the sum of all permissions terms p' mapped to a given $x \overset{p}{\mapsto} y$ is smaller or equal to p, with an equality required if $\top$ does not occur in Σ_r. We represent $(\Pi_l, \Sigma_l) \models (\top, \Sigma_r)$ by a triple $(\Pi_l, \Sigma_l, \Sigma_r)$, and we check the existence of such a map by means of the rewrite rules Align and Substract of Figure 2. Intuitively, we remove each points-to predicate of Σ_r, one by one, until Σ_r is trivial. This new rewrite relation, denoted by $\Longrightarrow_{AS}$, terminates in at most $|\Sigma_r|$ steps, and it

preserves the entailment validity : if $(\Pi_l, \Sigma_l, \Sigma_r) \Longrightarrow_{AS} (\Pi'_l, \Sigma'_l, \Sigma'_r)$ then $(\Pi_l, \Sigma_l) \models (\top, \Sigma_r)$ iff $(\Pi'_l, \Sigma'_l) \models (\top, \Sigma'_r)$.

$$\text{(Align)} \quad (\Pi, \Sigma_l * x \overset{p}{\mapsto} y, \Sigma_r * x' \overset{p'}{\mapsto} y') \implies (\Pi \wedge \mathsf{defined}(p), \Sigma_l, \Sigma_r) \quad \text{if } \Pi \models p = p' \wedge x = x' \wedge y = y'$$

$$\text{(Substract)} \quad \big(\Pi\ ,\ \Sigma_l * x \overset{p}{\mapsto} y\ ,\ \Sigma_r * x' \overset{p'}{\mapsto} y'\big) \implies \big(\Pi \wedge p = p' \oplus \alpha\ ,\ \Sigma_l * x \overset{\alpha}{\mapsto} y\ ,\ \Sigma_r\big) \quad \text{if } \Pi \models x = x' \wedge y = y', \Pi \wedge \mathsf{defined}(\Sigma_l) \wedge p = p' \oplus \alpha \text{ is satisfiable, } \alpha \notin \mathtt{PVar}(\Pi, \Sigma_l, \Sigma_r)$$

Figure 2 Rewrite rules for triples (Π, Σ, Σ')

In order to check an entailment $(\Pi_l, \Sigma_l) \models (\Pi_r, \Sigma_r)$ where Π_r is not necessarily $\top$, we check on the one hand whether $(\Pi_l, \Sigma_l) \models \Pi_r$, and on the other hand, using rules Align and Substract, whether $(\Pi_l, \Sigma_l) \models \Sigma_r$, which is implemented by the algorithm below.

▶ **Lemma 6.** *The algorithm terminates, and returns* **true** *iff* $(\Pi, \Sigma) \models (\Pi', \Sigma')$.

▶ **Theorem 7.** *The problem* ENTSH($\mathfrak{P}$) *restricted to symbolic heaps without list predicates can be decided in polynomial time with an oracle for* ENT($\mathfrak{P}$).

Entailment checking without lists
input two symbolic heaps (Π, Σ) and (Π', Σ') without list predicates
output returns **true** if $(\Pi, \Sigma) \models (\Pi', \Sigma')$, and returns **false** otherwise
  put (Π, Σ) in normal form
  if $(\Pi, \Sigma) = \bot$ or $\Pi_{pe} \wedge \mathsf{defined}(\Sigma)$ is not $\mathfrak{P}$-satisfiable **then return true**
  if $\Pi_{pe} \wedge \mathsf{defined}(\Sigma) \not\models \Pi'_{pe}$ **then return false**
  for all atomic formulae φ of Π'_{pv} **do**
    let (Π'', Σ'') be the normal form of $(\Pi \wedge \neg\varphi, \Sigma)$
    if $(\Pi'', \Sigma'') \neq \bot$ and $\Pi''_{pe} \wedge \mathsf{defined}(\Sigma'')$ is $\mathfrak{P}$-satisfiable **then return false**
  end for
  put $\big(\Pi, \Sigma, \Sigma'\big)$ in normal form w.r.t. $\Longrightarrow_{AS}$
  put (Π, Σ') in normal form w.r.t. $\Longrightarrow$
  return $\big(\Sigma' = \top\big)$ or $\big(\Sigma = \Sigma' = \mathsf{emp}\big)$

Using the results from Section 5, it follows that ENTSH($\mathfrak{P}_{\mathsf{Boy}}$) restricted to symbolic heaps without list predicates is in PTime.

4 Reasoning on Symbolic Heaps with Lists and Permissions

Below, we design algorithms for SATSH($\mathfrak{P}$) and ENTSH($\mathfrak{P}$) respectively, parameterised by decision problems for $\mathfrak{P}$. Assuming that Σ and Σ' are $\top$-free and $\mathtt{emp}$-free spatial formulae, we introduce the following subproblems of ENTSH($\mathfrak{P}$):

- $(\Pi, \Sigma) \models_I (\Pi', \Sigma') \overset{\text{def}}{\Leftrightarrow} (\Pi, \Sigma * \top) \models (\Pi', \Sigma' * \top)$. (intuitionistic entailment)
- $(\Pi, \Sigma) \models_{NI} (\Pi', \Sigma') \overset{\text{def}}{\Leftrightarrow} (\Pi, \Sigma) \models (\Pi', \Sigma')$. (non-intuitionistic entailment)

Below, we provide the developments for $\models_{NI}$ only but all our results can be adapted for the full problems SATSH($\mathfrak{P}$) and ENTSH($\mathfrak{P}$). In the permission model $\mathfrak{P}_1$, SATSH($\mathfrak{P}_1$) and ENTSH($\mathfrak{P}_1$) are in PTime [6] but untractability of SATSH($\mathfrak{P}$) and ENTSH($\mathfrak{P}$) happens quite quickly, even with the rather simple permission model $\mathfrak{P}_{\mathsf{Boy}}$. A positive consequence of Theorem 9 is that we can use non-determinism to get optimal complexity bounds.

4.1 Lower bounds

In this short section, we explain how the combination of list predicates and permission leads to NP/coNP-hardness. Let $G = (V, E)$ be an instance of the three-colorability problem, known to be NP-complete. W.l.o.g., we can assume that $V = \{x_1, \ldots, x_n\}$ for some $n \geq 1$ and E is a set of edges of the form $\{x_i, x_j\}$ with $i \neq j$. The hardness proof is by reduction from the three-colorability problem following a similar treatment when conjunctions are added to spatial formulae in the standard symbolic heap fragment, see [6, Section 5]. The main difference below rests on the replacement of Boolean conjunctions by separating conjunctions. Let Π_G be the pure formula $(\bigwedge_{\{x_i,x_j\}\in E}(x_i \neq x_j)) \wedge y_1 \neq y_2 \wedge y_1 \neq y_3 \wedge y_2 \neq y_3$. Let Σ_G be the spatial formula $y_1 \overset{\alpha_0}{\mapsto} y_2 * y_2 \overset{\alpha'_0}{\mapsto} y_3 * y_3 \overset{\mathbf{1}}{\mapsto} y_1 \mathbin{\text{✳}}_{x_i\in V}(\mathsf{lseg}_{\alpha_i}(y_1, x_i) * \mathsf{lseg}_{\alpha'_i}(x_i, y_3))$, where the α_i's and α'_i's are distinct permission variables.

▶ **Lemma 8.** *Assuming that* $\mathsf{width}(\mathfrak{P}) = \omega$, *$G$ has a three-coloring iff* (Π_G, Σ_G) *is satisfiable.*

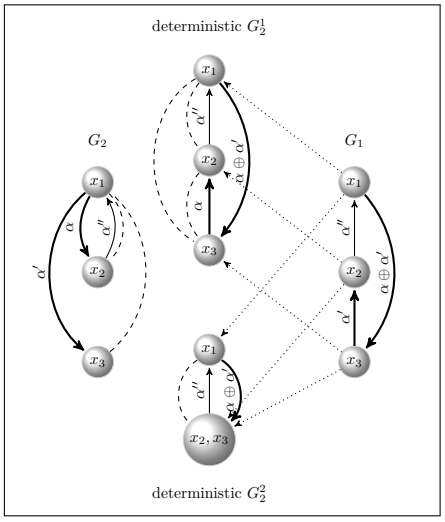
Consequently, we get the following hardness results.

▶ **Theorem 9.** *If* $\mathsf{width}(\mathfrak{P}) = \omega$ *then* $\mathrm{SATSH}(\mathfrak{P})$ *is* NP*-hard and* $\mathrm{ENTSH}(\mathfrak{P})$ *is* coNP*-hard.*

Observe that Π, Σ is not $\mathfrak{P}$-satisfiable iff $\Pi, \Sigma \models_{\mathfrak{P}} x \neq x$. So, the NP-hardness of $\mathrm{SATSH}(\mathfrak{P})$ entails the coNP-hardness of $\mathrm{ENTSH}(\mathfrak{P})$.

4.2 SL-graphs and homomorphisms

We assume a fixed permission model $\mathfrak{P}$ and a fixed set of variables $\{x_1, \ldots, x_q\}$ with its ordering $x_1 < \ldots < x_q$. All the symbolic heaps are assumed to be built from $\{x_1, \ldots, x_q\}$. Given $\emptyset \neq X \subseteq \{x_1, \ldots, x_q\}$, $\min(X)$ denotes the variable in X with the minimal index.



Below, we introduce a notion of SL-graph that can be understood as a graphical representation of a symbolic heap in which the permission part of the pure formula is encoded directly by a permission formula, the location variable part of a pure formula is encoded by an inequality relation ($\overset{\neq}{\leftrightarrow}$) and by a labelling ($L$). Besides, the atomic spatial formulae are encoded by the two relations $\rightarrow$ and $\Rightarrow$. Such structures are quite convenient to characterise entailment between symbolic heaps via homomorphisms, as it is done in the permission-free setting in [6]. Contrary to the developments in [6] that aim to reach a PTime upper bound, nondeterminism will be essential below since this is the best we can hope for, see Theorem 9.

Figure 3 SL-graphs

An **SL-graph** G is either $\bot$ or a tuple $(A, V, \rightarrow, \Rightarrow, \overset{\neq}{\leftrightarrow}, L)$ such that

- A is a permission formula and V is a non-empty finite subset of $\mathbb{N}$ (the nodes).
- $\rightarrow$ and $\Rightarrow$ are finite subsets of $V \times PT \times V$ where PT is the set of permission terms. We also write $v \overset{p}{\rightarrow} v'$ [resp. $v \overset{p}{\Rightarrow} v'$] instead $(v, p, v') \in \rightarrow$ [resp. $(v, p, v') \in \Rightarrow$]. We require a functionality condition: $v \overset{p}{\rightarrow} v'$ implies the unicity of p and v'.
- $\overset{\neq}{\leftrightarrow}$ is an irreflexive and symmetric binary relation on V.

- $L : \{x_1, \ldots, x_q\} \to V$ is a surjective labelling.

Given $v \in V$, we write $vars(v)$ to denote the (non-empty) set $\{x \mid L(x) = v\}$ and $var(v) \stackrel{\text{def}}{=} min(vars(v))$. As expected, the arrow $v \stackrel{p}{\to} v'$ [resp. $v \stackrel{p}{\Rightarrow} v'$] is intended to represent the atomic formula $x \stackrel{p}{\mapsto} x'$ [resp. $\mathsf{lseg}_p(x, x')$] whenever $L(x) = v$ and $L(x') = v'$. Moreover, we write $v \stackrel{p}{\rightsquigarrow} v'$ whenever $v \stackrel{p}{\to} v'$ or $v \stackrel{p}{\Rightarrow} v'$. Our notion of SL-graph shares in spirit the one from [6] but there are essential differences (permission formulae and terms as well as slight simplifications). Figure 3 presents SL-graphs where dashed lines encode the inequality relation, thick arrows encode $\Rightarrow$, normal arrows encode $\to$ and the permission formulae are omitted.

Below, we provide a semantics to the SL-graphs by defining a symbolic heap for each SL-graph. Given $G = (A, V, \to, \Rightarrow, \stackrel{\neq}{\leftrightarrow}, L)$, $(\mathsf{pure}(G), \mathsf{spatial}(G))$ denotes the symbolic heap defined from G:

$$\mathsf{pure}(G) \stackrel{\text{def}}{=} A \wedge \Big(\bigwedge_{x_i, x_j, L(x_i)=L(x_j)} x_i = x_j\Big) \wedge \Big(\bigwedge_{v \stackrel{\neq}{\leftrightarrow} v'} var(v) \neq var(v')\Big).$$

$$\mathsf{spatial}(G) \stackrel{\text{def}}{=} \Big(\mathop{\Large *}_{v \stackrel{p}{\to} v'} var(v) \stackrel{p}{\mapsto} var(v')\Big) * \Big(\mathop{\Large *}_{v \stackrel{p}{\Rightarrow} v'} \mathsf{lseg}_p(var(v), var(v'))\Big).$$

An empty separating conjunction is understood as emp. If $G = \bot$, then the corresponding symbolic heap is $(x \neq x, \top)$. In the sequel, $\mathsf{spatial}(G)$ is also written $\mathop{\Large *}_{v \stackrel{p}{\rightsquigarrow} v' \in G} \chi(v \stackrel{p}{\rightsquigarrow} v')$, where $\chi(v \stackrel{p}{\to} v') \stackrel{\text{def}}{=} var(v) \stackrel{p}{\mapsto} var(v')$ and $\chi(v \stackrel{p}{\Rightarrow} v') \stackrel{\text{def}}{=} \mathsf{lseg}_p(var(v), var(v'))$.

An SL-graph $(A, V, \to, \Rightarrow, \stackrel{\neq}{\leftrightarrow}, L)$ is **deterministic** iff for all $v \in V$, $\text{card}(\{v' \mid v \stackrel{p}{\rightsquigarrow} v'\}) \leq 1$ and, for all $v \neq v' \in V$, we have $v \stackrel{\neq}{\leftrightarrow} v'$ (G_2^1 and G_2^2 are deterministic in Figure 3 unlike G_1 and G_2). A deterministic SL-graph can be viewed as a syntactic structure whose interpretation stands between the $\mathfrak{P}$-memory states (thanks to the determinism and the syntactic nature of the inequality relation) and the SL-graphs (no $\mathfrak{P}$-interpretation and no permission values are involved). Each deterministic SL-graph carries a lot of structural properties about the $\mathfrak{P}$-memory states that satisfy it, which explains why this is a crucial structure to consider (see also the dependency graphs in [8]).

Let $G_1 = (A_1, V_1, \to_1, \Rightarrow_1, \stackrel{\neq}{\leftrightarrow}_1, L_1)$ and $G_2 = (A_2, V_2, \to_2, \Rightarrow_2, \stackrel{\neq}{\leftrightarrow}_2, L_2)$ be two SL-graphs with G_2 *being deterministic*. We introduce below a notion of precise homomorphism that will admit a counterpart in terms of entailment, see e.g. Theorem 11. A map $\mathfrak{f} : V_1 \to V_2$ is a **precise homomorphism** from G_1 to G_2 whenever the conditions below are satisfied:

(H0) $\mathfrak{f}$ is surjective.

(H1) $A^\star \models A_1$ where $A^\star = A_2 \wedge \bigwedge_i \mathsf{defined}(p_i) \wedge \bigwedge_j \mathsf{defined}(p'_j)$, $\to_2 = \{v_1 \stackrel{p_1}{\to}_2 v'_1, \ldots, v_n \stackrel{p_n}{\to}_2 v'_n\}$ and $\Rightarrow_2 = \{\{u_1 \stackrel{p'_1}{\Rightarrow}_2 u'_1, \ldots, u_m \stackrel{p'_m}{\Rightarrow}_2 u'_m\}\}$ (multiset notation).

(H2) For all $v \in V_1$, we have $vars(v) \subseteq vars(\mathfrak{f}(v))$.

(H3) For all $v, v' \in V_1$, $v \stackrel{\neq}{\leftrightarrow}_1 v'$ implies $\mathfrak{f}(v) \stackrel{\neq}{\leftrightarrow}_2 \mathfrak{f}(v')$.

(H4) For all $v, v' \in V_1$, $v \stackrel{p}{\to}_1 v'$ implies there is some permission term p' such that $\mathfrak{f}(v) \stackrel{p'}{\to}_2 \mathfrak{f}(v')$. We say that the edge $\mathfrak{f}(v) \stackrel{p'}{\to}_2 \mathfrak{f}(v')$ **contributes** to the edge $v \stackrel{p}{\to}_1 v'$.

(H5) For all $v, v' \in V_1$, $v \stackrel{p}{\Rightarrow}_1 v'$ implies either ($\mathfrak{f}(v) = \mathfrak{f}(v')$ and $A^\star \models \mathsf{defined}(p)$) or there is a path $v_0 \stackrel{p_0}{\rightsquigarrow}_2 v_1 \stackrel{p_1}{\rightsquigarrow}_2 v_2 \cdots \stackrel{p_{n-1}}{\rightsquigarrow}_2 v_n$ with $n \geq 1$, $v_0 = \mathfrak{f}(v)$, $v_n = \mathfrak{f}(v')$ and for all $i < j \in [0, n]$, $v_i \neq v_j$ (equivalent to $v_i \stackrel{\neq}{\leftrightarrow}_2 v_j$ since G_2 is deterministic). For each edge $v_k \stackrel{p_k}{\rightsquigarrow}_2 v_{k+1}$, we say that it **contributes** to the edge $v \stackrel{p}{\Rightarrow}_1 v'$. Since G_2 is deterministic, there is a unique path satisfying the above condition (if any).

(H6′) Each edge $v \overset{p'}{\rightsquigarrow}_2 v'$ in G_2 contributes to at least one edge of G_1, and we have $A^\star \models \oplus\{\{p \mid v \overset{p'}{\rightsquigarrow}_2 v' \text{ contributes to } u \overset{p}{\rightsquigarrow}_1 u' \in G_1\}\} = p'$.

Above, given a non-empty and finite multiset $T = \{\{p_1, \ldots, p_k\}\}$ of permission terms, we write $\oplus T$ instead of $p_1 \oplus \cdots \oplus p_k$ (the ordering of the terms is irrelevant because $\oplus$ is AC). Precise homomorphisms could be defined between two arbitrary SL-graphs (as done in [6]) but the unicity of the path in the condition (H5) is not anymore guaranteed. We assume that G_2 is deterministic to have unicity, which also leads to the right upper bounds for the complexity. The existence of a precise homomorphism $\mathfrak{f}$ implies that for all $u \overset{p}{\rightsquigarrow}_1 u' \in G_1$, we have $A^\star \models \mathsf{defined}(p)$, which is partly justified by $\mathsf{defined}(p_1 \oplus p_2) \models_{\mathfrak{P}} \mathsf{defined}(p_1) \wedge \mathsf{defined}(p_2)$. The dotted arrows in Figure 3 partly materialize two precise homomorphisms from G_1 to G_2^1 or to G_2^2 (assuming the permission formulae match).

A precise homomorphism $\mathfrak{f}$ is **strongly precise** $\overset{\text{def}}{\Leftrightarrow}$

(H1′) $A_2 = A_1 \wedge \bigwedge_{v \overset{p}{\rightsquigarrow} v' \text{in } G_1} \mathsf{defined}(p)$ (which implies (H1)),

(H5′) is equal to (H5) except that in the case $\mathfrak{f}(v) = \mathfrak{f}(v')$, we do not require that $A^\star \models \mathsf{defined}(p)$,

(H6″) each edge $v \overset{p'}{\rightsquigarrow}_2 v'$ in G_2 contributes to at least one edge of G_1, and we have $\oplus\{\{p \mid v \overset{p'}{\rightsquigarrow}_2 v' \text{ contributes to } u \overset{p}{\rightsquigarrow}_1 u' \in G_1\}\}$ equal to p' modulo AC (implying (H6′)).

In Figure 3, there is a strongly precise homomorphism from G_2 to G_2^1 [resp. to G_2^2] with the adequate permission formulae. Notably, checking whether $\mathfrak{f} : V_1 \to V_2$ is a [resp. strongly] precise homomorphism can be checked in coNP [resp. PTIME] when $\mathrm{ENT}(\mathfrak{P})$ is in coNP, which is useful to establish the complexity upper bounds.

4.3 From symbolic heaps to SL-graphs

Let (Π, Σ) be a symbolic heap, possibly with list predicates and, Σ is $\mathtt{emp}$-free and $\top$-free. Let us define the SL-graph $\mathsf{slg}(\Pi, \Sigma)$ as follows. If $(\Pi, \Sigma) \Longrightarrow^* \bot$, then $\mathsf{slg}(\Pi, \Sigma)$ is defined as the inconsistent SL-graph $\bot$ (see Section 3 for the definition of $\Longrightarrow$).

Otherwise, assume that $(\Pi, \Sigma) \Longrightarrow^* (\Pi', \Sigma')$ and (Π', Σ') is in normal form (i.e., no rule can be further fired from (Π', Σ')). Let us define $\mathsf{slg}(\Pi, \Sigma) \overset{\text{def}}{=} (A, V, \to, \Rightarrow, \overset{\neq}{\leftrightarrow}, L)$ as follows:

- $A \overset{\text{def}}{=} \Pi'_{pe}$, $V \overset{\text{def}}{=} \{i \in [1, q] \mid \text{there is no } j < i \text{ such that } \Pi' \models x_i = x_j\}$.
- $L(x_i) \overset{\text{def}}{=} \min\{j \in V \mid \Pi' \models x_i = x_j\}$.
- If $x \overset{p}{\mapsto} y$ occurs in Σ', then $L(x) \overset{p}{\to} L(y)$; if $\mathsf{lseg}_p(x, y)$ occurs in Σ', then $L(x) \overset{p}{\Rightarrow} L(y)$.
- For all $i \neq j \in [1, q]$, we have $\{L(x_i), L(x_j)\} \in \overset{\neq}{\leftrightarrow} \overset{\text{def}}{\Leftrightarrow} \Pi' \models x_i \neq x_j$.

In Figure 3, $G_1 = \mathtt{slg}(\top, \mathsf{lseg}_{\alpha \oplus \alpha'}(x_1, x_3) * \mathsf{lseg}_{\alpha'}(x_3, x_2) * x_2 \overset{\alpha''}{\mapsto} x_1)$. The soundness of the constructions between symbolic heaps and SL-graphs is best illustrated by Lemma 10.

▶ **Lemma 10.** $(\Pi, \Sigma) \equiv_{\mathfrak{P}} (\mathsf{pure}(\mathsf{slg}(\Pi, \Sigma)), \mathsf{spatial}(\mathsf{slg}(\Pi, \Sigma)))$.

4.4 Relating memory states and deterministic SL-graphs

Given G_1 and G_2, we write $\mathfrak{f}_{G_1,G_2} : V_1 \to V_2$ to denote the map s.t. $\mathfrak{f}_{G_1,G_2}(v) \overset{\text{def}}{=} L_2(var(v))$. Without any further assumption, note that $\mathfrak{f}_{G_1,G_2}$ is not necessarily a precise homomorphism.

Given a $\mathfrak{P}$-memory state (s, h, ι) and a deterministic SL-graph $G = (A, V, \to, \Rightarrow, \overset{\neq}{\leftrightarrow}, L)$, we write $(s, h, \iota) \overset{\mathsf{lin}}{\approx} G$ whenever for all $v, v', v'' \in V$, if there are non-empty paths from v to v' and from v' to v'' in G, then for all variables x, x', x'' such that $L(x) = v$, $L(x') = v'$ and $L(x'') = v''$, one of the conditions below holds:

1. there is no non-empty sequence of memory cells in (s,h,ι) from $s(x)$ to $s(x'')$,
2. there is no non-empty sequence of memory cells in (s,h,ι) from $s(x'')$ to $s(x')$.

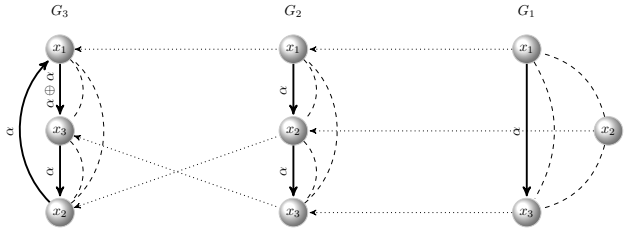
Roughly speaking, $(s,h,\iota) \overset{\mathsf{lin}}{\approx} G$ holds when (s,h,ι) respects the linearisation induced by the graphical part of G.

Below, we establish an equivalence between the existence of a precise homomorphism and the entailment $\models_{NI}$. A similar statement can be found in [6] but herein we deal with permissions and with the deterministic SL-graphs. This is a key result at the heart of our whole entreprise. An auxiliary definition is needed. Given a deterministic SL-graph G' and a symbolic heap (Π,Σ), we write $(\mathsf{pure}(G'),\mathsf{spatial}(G')) \models^{\mathsf{lin}} \Pi,\Sigma$ iff for all $\mathfrak{P}$-memory states (s,h,ι) such that $(s,h,\iota) \overset{\mathsf{lin}}{\approx} G$, if $(s,h,\iota) \models (\mathsf{pure}(G'),\mathsf{spatial}(G'))$, then $(s,h,\iota) \models \Pi,\Sigma$.

▶ **Lemma 11.** *Let G' be a deterministic SL-graph such that $(\mathsf{pure}(G'),\mathsf{spatial}(G'))$ is satisfiable. For all SL-graphs G, the statements below are equivalent:*

- *The map $\mathfrak{f}_{G,G'}$ is a precise homomorphism from G to G'.*
- $(\mathsf{pure}(G'),\mathsf{spatial}(G')) \models^{\mathsf{lin}} (\mathsf{pure}(G),\mathsf{spatial}(G))$.

Let us briefly explain below why in Lemma 11, $\models^{\mathsf{lin}}$ cannot be replaced by $\models$. Let us consider the deterministic SL-graphs G_1, G_2 and G_3 with respective permission formulae $\top$, $\top \wedge \mathsf{defined}(\alpha)$ and $\top \wedge \mathsf{defined}(\alpha) \wedge \mathsf{defined}(\alpha) \wedge \mathsf{defined}(\alpha)$ (all of them equivalent to $\top$).



Note that $\mathfrak{f}_{G_1,G_2}$ and $\mathfrak{f}_{G_2,G_3}$ are strongly precise homomorphisms and for $i \in \{1,2,3\}$, $(\mathsf{pure}(G_i),\mathsf{spatial}(G_i))$ is logically equivalent to (Π_i,Σ_i) from Example 2. However, not $(\mathsf{pure}(G_2),\mathsf{spatial}(G_2)) \models_{\mathfrak{P}_{\mathsf{Boy}}} (\mathsf{pure}(G_1),\mathsf{spatial}(G_1))$. Indeed, let us consider the $\mathfrak{P}_{\mathsf{Boy}}$-memory state (s,h,ι) defined in Section 2. We have $(s,h,\iota) \models (\mathsf{pure}(G_2),\mathsf{spatial}(G_2))$ and $(s,h,\iota) \not\models (\mathsf{pure}(G_1),\mathsf{spatial}(G_1))$. Actually, $(s,h,\iota) \overset{\mathsf{lin}}{\approx} G_2$ is false. By contrast, $(s,h,\iota) \overset{\mathsf{lin}}{\approx} G_3$. By Lemma 11, we conclude however that $(\mathsf{pure}(G_3),\mathsf{spatial}(G_3)) \models^{\mathsf{lin}} (\mathsf{pure}(G_2),\mathsf{spatial}(G_2))$ and $(\mathsf{pure}(G_2),\mathsf{spatial}(G_2)) \models^{\mathsf{lin}} (\mathsf{pure}(G_1),\mathsf{spatial}(G_1))$, which is sufficient for our needs.

4.5 Decision procedures modulo permission theories

Let us characterise non-entailment between two symbolic heaps in the non-intuitionistic setting (leading to entailment of symbolic heaps from Figure 3 with $G_2 = \mathtt{slg}(x_1 \neq x_3 \wedge x_1 \neq x_2 \wedge \alpha = \alpha', \mathsf{lseg}_{\alpha}(x_1,x_2) * \mathsf{lseg}_{\alpha'}(x_1,x_3) * x_2 \overset{\alpha''}{\mapsto} x_1)$. This induces a nondeterministic algorithm by guessing the appropriate G (see below). Such a guess could be formalised in a proof system, as done for a fragment in Section 3, but herein we focus on the characterisation. In Theorem 12 below, note the use of $\models_{NI}$ (instead of $\models^{\mathsf{lin}}$).

▶ **Theorem 12.** *Let (Π,Σ), (Π',Σ') be symbolic heaps s.t. neither $\mathsf{slg}(\Pi,\Sigma)$ nor $\mathsf{slg}(\Pi',\Sigma')$ is equal to $\bot$. $(\Pi,\Sigma) \not\models_{NI} (\Pi',\Sigma')$ iff there is a deterministic SL-graph G that satisfies:*

(SMALL) *For every $v \overset{p}{\rightsquigarrow} v'$ in G, the permission term p is a sum of at most $|\Sigma|$ terms from $\mathsf{slg}(\Pi,\Sigma)$. Moreover, the permission formula in G is precisely the permission part of Π.*

(SAT) *$(\mathsf{pure}(G),\mathsf{spatial}(G))$ is satisfiable.*

(PRE) *$\mathfrak{f}_{\mathsf{slg}(\Pi,\Sigma),G}$ is a strongly precise homomorphism.*

(NOTPRE) $\mathfrak{f}_{\mathsf{slg}(\Pi',\Sigma'),G}$ *is not a precise homomorphism.*

Here is our characterisation for satisfiability checking.

▶ **Theorem 13.** *Let* (Π, Σ) *be a symbolic heap so that* Σ *is* $\top$*-free and* $\mathtt{emp}$*-free and* $\mathsf{slg}(\Pi, \Sigma)$ *is not equal to* $\bot$*. Then,* (Π, Σ) *is satisfiable iff there is a deterministic SL-graph* G *such that (SMALL), (SAT) and (PRE) hold.*

The previous characterisations allow us to conclude to optimal complexity bounds.

▶ **Theorem 14.** ***(I)*** $\mathrm{SATSH}(\mathfrak{P}_{\mathsf{Boy}})$ *is* NP*-complete and* $\mathrm{ENTSH}(\mathfrak{P}_{\mathsf{Boy}})$ *is* coNP*-complete.* ***(II)*** *For any permission model* $\mathfrak{P}$ *such that* $\mathsf{width}(\mathfrak{P}) = \omega$ *and,* $\mathrm{ENT}(\mathfrak{P})$ *is in* coNP*,* $\mathrm{SATSH}(\mathfrak{P})$ *is* NP*-complete and* $\mathrm{ENTSH}(\mathfrak{P})$ *is* coNP*-complete.*

Currently, the permission terms do not allow constants other than $\mathbf{1}$ but for many permission models, constants can be easily added. For instance, in $\mathfrak{P}_{\mathsf{Boy}}$, one can deal with $\frac{3}{4}$ by using the variable α thanks to $(\alpha' \oplus \alpha' \oplus \alpha' \oplus \alpha' = \mathbf{1}) \wedge (\alpha = \alpha' \oplus \alpha' \oplus \alpha')$ (generalisation to other rational numbers is obvious). Of course, depending on the permission models in mind, constants should be either introduced in the language of permission terms (according to a specified encoding) or can be enforced directly in the language, as it is the case for $\mathfrak{P}_{\mathsf{Boy}}$.

5 Solving Permission Constraints

Herein, we review permission models introduced for separation logic and we classify these models according to the complexity of decision problems. In this process, we introduce Boolean permission models that generalise all existing permission models but the trivial model $\mathfrak{P}_{\mathbf{1}}$ and the fractional model $\mathfrak{P}_{\mathsf{Boy}}$ that admit PTime decision problems. We give optimal complexity results for the satisfiability and entailment problems. This is motivated by Theorem 15 below, which is a consequence of the parameterised decision procedures from Section 4 that separate the reasoning on the memory shapes from the one on permissions.

▶ **Theorem 15.** *Let* $\mathfrak{P}$ *and* $\mathfrak{P}'$ *be permission models. (I)* $\mathrm{SAT}(\mathfrak{P}) = \mathrm{SAT}(\mathfrak{P}')$ *implies* $\mathrm{SATSH}(\mathfrak{P}) = \mathrm{SATSH}(\mathfrak{P}')$*. (II)* $\mathrm{ENT}(\mathfrak{P}) = \mathrm{ENT}(\mathfrak{P}')$ *implies* $\mathrm{ENTSH}(\mathfrak{P}) = \mathrm{ENTSH}(\mathfrak{P}')$*.*

We briefly review the permission models that have been introduced in the literature. The singleton model $\mathfrak{P}_{\mathbf{1}}$ presented in Section 2.2 is simply an artefact used to show that separation logic without permission is a special case of the general case. The fractional model $\mathfrak{P}_{\mathsf{Boy}}$ already presented in Section 2, is one of the most popular permission models and it enjoys nice complexity properties stated below.

▶ **Theorem 16.** $\mathrm{SAT}(\mathfrak{P}_{\mathsf{Boy}})$ *and* $\mathrm{ENT}(\mathfrak{P}_{\mathsf{Boy}})$ *are in* PTime*.*

The PTime bound is obtained by reduction to the satisfiability of a system of linear inequalities. Other classical permission models are :

- Bornat-Parkinson's permission model [4] with tokens is $\mathfrak{P}_{\mathsf{Tok}} = (P_{\mathfrak{P}_{\mathsf{Tok}}}, \mathbf{1}_{\mathfrak{P}_{\mathsf{Tok}}}, \oplus_{\mathfrak{P}_{\mathsf{Tok}}})$, where a permission $\pi \in P_{\mathfrak{P}_{\mathsf{Tok}}}$ is either a finite or a co-finite, non-empty subset of $\mathbb{N}$, $\mathbf{1}_{\mathfrak{P}_{\mathsf{Tok}}}$ is $\mathbb{N}$, and $\pi \oplus_{\mathfrak{P}_{\mathsf{Tok}}} \pi'$ is defined if $\pi \cap \pi' = \emptyset$ and is then equal to $\pi \cup \pi'$.
- Dockins-Hobor model, a.k.a binary shares [7], is $\mathfrak{P}_{\mathsf{Bin}} = ((\mathcal{T}(\mathcal{F})/\equiv) \setminus \{\mathbf{0}\}, \oplus, \mathbf{1})$ where $\mathcal{T}(\mathcal{F})$ is the set of closed terms constructed over the function symbols $\mathcal{F} = \{f : 2, \mathbf{0} : 0, \mathbf{1} : 0\}$, $\equiv$ is the least congruence such that $f(\mathbf{0}, \mathbf{0}) \equiv \mathbf{0}$ and $f(\mathbf{1}, \mathbf{1}) \equiv \mathbf{1}$, $\mathcal{T}(\mathcal{F})/\equiv$ denotes the quotient, and $\oplus$ is defined by $\mathbf{0} \oplus \mathbf{1} \equiv \mathbf{1} \oplus \mathbf{0} \equiv \mathbf{1}$, $\mathbf{0} \oplus \mathbf{0} \equiv \mathbf{0}$, $\mathbf{1} \oplus \mathbf{1}$ is undefined, and $f(\pi_1, \pi_2) \oplus f(\pi'_1, \pi'_2) \equiv f(\pi_1 \oplus \pi'_1, \pi_2 \oplus \pi'_2)$.

Note that in these two permission models a permission term $\alpha \oplus \alpha$ has no interpretation since the partial function $\oplus$ is not defined for identical elements. As a consequence, it holds for instance that $\mathsf{defined}(\alpha \oplus \alpha)$ is unsatisfiable and $\mathsf{lseg}_{\alpha\oplus\alpha}(x, y) \models_{\mathfrak{P}_{\mathsf{Tok}}} x \neq x$.

These two permission models are particular instances of what we call Boolean permission models, i.e. permission models defined on top of a given Boolean algebra, as explained below. Let $\mathbb{B} = (B_{\mathbb{B}}, \wedge_{\mathbb{B}}, \vee_{\mathbb{B}}, \top_{\mathbb{B}}, \bot_{\mathbb{B}}, \neg_{\mathbb{B}})$ be a Boolean algebra. The permission model $\mathfrak{P}_{\mathbb{B}}$ associated to $\mathbb{B}$ is $\mathfrak{P}_{\mathbb{B}} \stackrel{\text{def}}{=} (P_{\mathbb{B}}, \oplus_{\mathbb{B}}, \top_{\mathbb{B}})$ where $P_{\mathbb{B}} = B_{\mathbb{B}} \setminus \{\bot_{\mathbb{B}}\}$ and $\pi \oplus_{\mathbb{B}} \pi'$ is defined when $\pi \wedge_{\mathbb{B}} \pi' = \bot_{\mathbb{B}}$, and in that case $\pi \oplus_{\mathbb{B}} \pi' \stackrel{\text{def}}{=} \pi \vee_{\mathbb{B}} \pi'$. A permission model is **Boolean** if it is isomorphic to $\mathfrak{P}_{\mathbb{B}}$ for some Boolean algebra $\mathbb{B}$. Both $\mathfrak{P}_{\mathsf{Tok}}$ and $\mathfrak{P}_{\mathsf{Bin}}$ are Boolean, the first one through the Boolean algebra of finite or co-finite subsets of $\mathbb{N}$, the second one through the Boolean algebra of open-closed sets of $\{0,1\}^\omega$, see e.g. [7]. As stated below, Boolean permission models are canonical in some sense.

▶ **Lemma 17.** *Let A_L, A_R be permission formulae. Let $\mathfrak{P}$ be a Boolean permission model with at least two elements such that $A_L \models_{\mathfrak{P}} A_R$, and $\mathsf{width}(\mathfrak{P}) \geq$ card$(\mathtt{PVar}(A_L))$. Then for all Boolean permission models $\mathfrak{P}'$, we have $A_L \models_{\mathfrak{P}'} A_R$.*

Lemma 17 entails $\mathrm{ENT}(\mathfrak{P}_{\mathsf{Tok}}) = \mathrm{ENT}(\mathfrak{P}_{\mathsf{Bin}})$. The case $\mathrm{card}(P_{\mathfrak{P}}) = 1$ cannot be added to Lemma 17 since $\top \models_{\mathfrak{P}_1} \alpha_1 = \alpha_2$ but $\top \not\models_{\mathfrak{P}_{\mathsf{Tok}}} \alpha_1 = \alpha_2$ with α_1 different from α_2. Note also that if we could express the atomicity of a permission, the two models could be distinguished.

▶ **Theorem 18.** $\mathrm{ENT}(\mathfrak{P}_{\mathsf{Tok}}) = \mathrm{ENT}(\mathfrak{P}_{\mathsf{Bin}})$ *and* $\mathrm{SAT}(\mathfrak{P}_{\mathsf{Tok}}) = \mathrm{SAT}(\mathfrak{P}_{\mathsf{Bin}})$.

From now on, we consider an arbitrary Boolean permission model $\mathfrak{P}_{\mathbb{B}}$ associated to a Boolean algebra $\mathbb{B}$ such that $\mathsf{width}(\mathfrak{P}_{\mathbb{B}}) = \omega$. Boolean permission models behave quite nicely and below we establish that their decision problems are in coNP.

▶ **Theorem 19.** *Let $\mathfrak{P}_{\mathbb{B}}$ be a Boolean permission model such that $\mathsf{width}(\mathfrak{P}_{\mathbb{B}}) = \omega$. $\mathrm{SAT}(\mathfrak{P}_{\mathbb{B}})$ is* NP*-complete and, $\mathrm{ENT}(\mathfrak{P}_{\mathbb{B}})$ is* coNP*-complete.*

A reduction from the NP-complete problem 1-in-3 SAT [15] gives the hardness result.

We give below the proof idea for $\mathrm{SAT}(\mathfrak{P}_{\mathbb{B}})$ is in NP. Actually, we can consider only permission terms equal to $\mathbf{1}$ and of the form $\bigoplus \alpha_i$ and atomic permission formulae of the form $\bigoplus \alpha_i = \bigoplus \alpha_j$, $\bigoplus \alpha_i \leq \bigoplus \alpha_j$, $\bigoplus \alpha_i = \mathbf{1}$ or $\mathsf{defined}(\bigoplus \alpha_i)$ and we can assume that each A contains a conjunct $\mathsf{defined}(p)$ for each permission term p. Let A be a permission formula built on $\alpha_1, \ldots, \alpha_n$. We introduce an arithmetical formula ψ_A such that A is satisfiable iff ψ_A is satisfiable. The Boolean/arithmetical variables of ψ_A taking their values in $\{0,1\}$ are precisely $\mathtt{X}_1^1, \ldots, \mathtt{X}_1^n, \ldots, \mathtt{X}_n^1, \ldots, \mathtt{X}_n^n$. The formula ψ_A is a conjunction of formulae $\psi_1 \wedge \cdots \wedge \psi_n$ where each ψ_i is built on the variables $\mathtt{X}_1^i, \ldots, \mathtt{X}_n^i$. For each $i \in [1, n]$, we define $\mathfrak{t}^i(\alpha_j) \stackrel{\text{def}}{=} \mathtt{X}_j^i$, and $\mathfrak{t}^i(p)$ replaces each occurrence of α_j by $\mathfrak{t}^i(\alpha_j)$ and each occurrence of $\oplus$ by $+$, each occurrence of $\mathbf{1}$ by 1. Each ψ_i is a conjunction of constraints defined by: (a) for each $p = p'$ [resp. $p \leq p'$] in A, ψ_i contains $\mathfrak{t}^i(p) \leq \mathfrak{t}^i(p') \wedge \mathfrak{t}^i(p') \leq \mathfrak{t}^i(p)$ [resp. $\mathfrak{t}^i(p) \leq \mathfrak{t}^i(p')$] and (b) for each formula $\mathsf{defined}(p)$ in A, ψ_i contains $\mathfrak{t}^i(p) \leq 1$, and $\mathtt{X}_i^i = 1$ belongs to ψ_i. The next lemma relates A and ψ_A.

▶ **Lemma 20.** *A is satisfiable iff ψ_A is satisfiable.*

This lemma entails that $\mathrm{SAT}(\mathfrak{P}_{\mathbb{B}})$ is in NP.

6 Conclusion

Our results provide optimal complexity results about several standard permission models and are summarized by the following table. The algorithms can be implemented using any

checker following the standard viewpoint for SMT solvers [2] for reasoning on permission. Besides, this work could be continued in several directions, for instance to consider enriched permission theories (e.g., adding inequalities between permission terms), permission models without infinite width, to allow existential quantifications or to design sequent-style proof systems for checking entailment based on our characterisations.

	$\mathfrak{P}_1$	$\mathfrak{P}_{\mathsf{Boy}}$	Boolean $\mathfrak{P}$, $\mathsf{width}(\mathfrak{P}) = \omega$ $(\mathfrak{P}_{\mathsf{Tok}}, \mathfrak{P}_{\mathsf{Bin}})$
SAT($\mathfrak{P}$)	in PTIME	in PTIME (Th. 16)	NP-C. (Th. 19)
ENT($\mathfrak{P}$)	in PTIME	in PTIME (Th. 16)	coNP-C. (Th. 19)
SATSH($\mathfrak{P}$)	in PTIME [6]	NP-C. (Th. 9,14(I))	NP-C. (Th. 9,14(II))
ENTSH($\mathfrak{P}$)\ list pred.	in PTIME [6]	in PTIME (Th. 7)	coNP-C. (Th. 19,14(II))
ENTSH($\mathfrak{P}$)	in PTIME [6]	coNP-C. (Th. 9,14(I))	coNP-C. (Th. 9,14(II))

Acknowledgements. We thanks the anonymous referees for their remarks and suggestions.

References

1. C. Barrett, C. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, and C. Tinelli. CVC4. In *CAV'11*, volume 8606 of *LNCS*, pages 171–177. Springer, 2011.
2. C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli. *Satisfiability Modulo Theories*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. IOS Press, 2008.
3. J. Berdine, C. Calcagno, and P. O'Hearn. A decidable fragment of separation logic. In *FSTTCS'04*, volume 3328 of *LNCS*, pages 97–109. Springer, 2004.
4. R. Bornat, C. Calcagno, P. O'Hearn, and M. Parkinson. Permission accounting in separation logic. In *POPL'05*, pages 259–270. ACM, 2005.
5. J. Boyland. Checking interference with fractional permissions. In *SAS'03*, number 2694 in LNCS, pages 55–72. Springer, 2003.
6. B. Cook, C. Haase, J. Ouaknine, M. Parkinson, and J. Worrell. Tractable reasoning in a fragment of separation logic. In *CONCUR'11*, volume 6901 of *LNCS*, pages 235–249, 2011.
7. R. Dockins, A. Hobor, and A.W. Appel. A fresh look at separation algebras and share accounting. In *APLAS'09*, volume 5904 of *LNCS*, pages 161–177. Springer, 2009.
8. D. Galmiche, D. Mery, and D. Pym. Resource tableaux (extended abstract). In *CSL'02*, volume 2471 of *LNCS*, pages 183–199. Springer, 2002.
9. M. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, San Francisco, California, 1979.
10. C. Haase, S. Ishtiaq, J. Ouaknine, and M. Parkinson. SeLoger: A tool for graph-based reasoning in separation logic. In *CAV'13*, volume 8044 of *LNCS*, pages 790–795, 2013.
11. G. He, S. Qin, C. Luo, and W.N. Chin. Memory Usage Verification Using Hip/Sleek. In *ATVA'09*, number 5799 in LNCS, pages 166–181. Springer, 2009.
12. B. Jacobs, J. Smans, P. Philippaerts, F. Vogels, W. Penninckx, and F. Piessens. Verifast: A powerful, sound, predictable, fast verifier for C and Java. In *NFM'11*, volume 6617 of *LNCS*, pages 41–55. Springer, 2011.
13. X. Bach Le, C. Gherghina, and A. Hobor. Decision procedures over sophisticated fractional permissions. In *APLAS'12*, pages 368–385, 2012.
14. J.C. Reynolds. Separation logic: a logic for shared mutable data structures. In *LICS'02*, pages 55–74. IEEE, 2002.
15. Th. Schaefer. The complexity of satisfiability problems. In *STOC'78*, pages 216–226, 1978.
16. J. Villard, E. Lozes, and C. Calcagno. Tracking heaps that hop with Heap-Hop. In *TACAS'10*, volume 6015 of *LNCS*, pages 275–279. Springer, 2010.