On Symbolic Heaps Modulo Permission Theories

S. Demri¹, E. Lozes¹, and D. Lugiez¹,²

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 [13]. 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 [11], Hip/Sleek [10], or Heap-Hop [15]. 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 $x^{0.3} \rightarrow y$ and $x^{0.7} \rightarrow y$ making an arbitrary choice for permissions (0.3 and 0.7) when a better approach would use $x^\alpha \rightarrow y$ and $x^\beta \rightarrow y$ and the constraint $1 = \alpha + \beta$. 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
such as $1 = \alpha + \beta$. 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 $\mathcal{P}$ for the satisfiability problem SATSH($\mathcal{P}$), and for the entailment problem ENTSH($\mathcal{P}$). 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 [12] 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, 9]. 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($\mathcal{P}$) and ENTSH($\mathcal{P}$) using an oracle for the corresponding problems on permission theories. As a byproduct, SATSH($\mathcal{P}_{\text{Boy}}$) and ENTSH($\mathcal{P}_{\text{Boy}}$) without list predicates are in PTIME for the fractional model $\mathcal{P}_{\text{Boy}}$. In Section 4, we prove that SATSH($\mathcal{P}$) is NP-hard and that ENTSH($\mathcal{P}$) is coNP-hard even for a permission theory $\mathcal{P}$ 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($\mathcal{P}$) (fully parametrised by the entailment problem for the permission theory $\mathcal{P}$). 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 $\mathcal{P}_{\text{Boy}}$ has PTIME satisfiability and entailment problems, (ii) we introduce the notion of Boolean permission models $\mathcal{P}_B$ that encompasses all classical permission models but the trivial one and $\mathcal{P}_{\text{Boy}}$ and (iii) we prove that SAT($\mathcal{P}_B$) is NP-complete and ENT($\mathcal{P}_B$) is coNP-complete in Boolean permission models $\mathcal{P}_B$ that have an infinite width (which is the case for the aforementioned 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:

$$
p ::= \ 1 \ | \ \alpha \ | \ p \oplus p$$

$$
A ::= \ \top \ | \ p = p \ | \ p \preceq p \ | \ \text{defined}(p) \ | \ A \land A \quad \text{(permission term)}$$

(.permission formula)
where $P\text{Var} = \{\alpha, \beta, \ldots\}$ is a countably infinite set of \textit{permission variables}. Permission formulae are interpreted in permission models, defined below.

\begin{definition}
A \textit{permission model} is a tuple $\mathcal{P} = (P_\mathcal{P}, 1_\mathcal{P}, \oplus_\mathcal{P})$ such that
\begin{itemize}
  \item $P_\mathcal{P} = \{\pi, \ldots\}$ is a set of \textit{permissions},
  \item $1_\mathcal{P} \in P_\mathcal{P}$ is a distinguished permission called the \textit{write} permission or the \textit{total} permission,
  \item $\oplus_\mathcal{P} : P_\mathcal{P} \times P_\mathcal{P} \rightarrow P_\mathcal{P}$ is a partial composition that is cancellative, commutative and associative,\footnote{In particular, whenever a sum $\pi_1 \oplus_\mathcal{P} \pi_2 \ldots \oplus_\mathcal{P} \pi_n$ is defined, each subsum $\pi_{i_1} \oplus_\mathcal{P} \ldots \oplus_\mathcal{P} \pi_{i_k}$ for each $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ is defined.}
\end{itemize}
\end{definition}

The relation $<_\mathcal{P} = \{(\pi', \pi) \mid \pi = \pi' \oplus_\mathcal{P} \pi'' \text{ for some } \pi''\}$ is irreflexive and transitive, with maximum element $1_\mathcal{P}$.

An example of permission model is Boyland’s fractional model $\mathcal{P}_{\text{Boy}} = ((0, 1], 1, \oplus_{\text{Boy}})$ \footnote{Such a model is an interpretation of a spatial formula and is used to model the spatial constraints of a program.}, where $\pi \oplus_{\text{Boy}} \pi' \equiv \pi + \pi'$ is defined when the sum is at most 1. The \textit{width} of a permission model $\mathcal{P}$ is $\text{width}(\mathcal{P}) \in \mathbb{N} \cup \{\omega\}$ such that $\text{width}(\mathcal{P}) \overset{\text{def}}{=} \sup\{n \geq 1 \mid \exists \pi_1, \ldots, \pi_n \in P_\mathcal{P} \text{ such that } \pi_1 \oplus \cdots \oplus \pi_n = 1_\mathcal{P}\}$.

Given $\mathcal{P} = (P_\mathcal{P}, 1_\mathcal{P}, \oplus_\mathcal{P})$, a $\mathcal{P}$-\textit{interpretation} is a map $\iota : P\text{Var} \rightarrow P_\mathcal{P}$. The map $\iota$ is extended to a partial map from the set of permission terms to $P_\mathcal{P}$ so that $\iota(1) \overset{\text{def}}{=} 1_\mathcal{P}$ and $\iota(p \oplus p') \overset{\text{def}}{=} \iota(p) \oplus_\mathcal{P} \iota(p')$ if $\iota(p)$ and $\iota(p')$ are defined. Otherwise $\iota(p \oplus p')$ is undefined. We may write $[p]_\mathcal{P}$ for $\iota(p)$ and $\iota \models A$ to denote that $\iota$ satisfies the permission formula $A$. Following the clauses below:
\begin{itemize}
  \item $\iota(A) = \top \iff \iota(p)$ is defined; $\iota(A \land A') \overset{\text{def}}{=} \iota(A) \land \iota(A')$.
  \item $\iota(p) = p' \iff$ both $\iota(p)$ and $\iota(p')$ are defined and $\iota(p) = \iota(p')$.
  \item $\iota(p) \leq p' \iff$ both $\iota(p)$ and $\iota(p')$ are defined and, either $\iota(p) = \iota(p')$ or $\iota(p) <_\mathcal{P} \iota(p')$.
\end{itemize}

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

\subsection{Separation logic with permissions}

A \textit{symbolic heap} with list predicates and symbolic permissions is a formula $(\Pi, \Sigma)$ where $\Pi$ is a \textit{pure formula} and $\Sigma$ a \textit{spatial formula} according to the grammar below:
\begin{align*}
\Pi &::= \top \quad | \quad x = y \quad | \quad x \neq y \quad | \quad A \quad | \quad \Pi \land \Pi \quad \text{(pure formula)} \\
\Sigma &::= \text{emp} \quad | \quad \top \quad | \quad x \overset{p}{\Rightarrow} y \quad | \quad \text{lseg}_p(x, y) \quad | \quad \Sigma \ast \Sigma \quad \text{(spatial formula)}
\end{align*}

where $\text{LVAR} = \{x, y, \ldots\}$ is a countably infinite set of \textit{location/program variables}. We write $\Pi_{\text{pe}}$ and $\Pi_{\text{pv}}$ to denote respectively the permission constraints and the program variable constraints that appear in $\Pi$, so that $\Pi$ is logically equivalent to $\Pi_{\text{pe}} \land \Pi_{\text{pv}}$. We write $\text{LVAR}(\varphi)$ [resp. $P\text{Var}(\varphi)$] to denote the set of location [resp. permission] variables occurring in $\varphi$.

\begin{example}
The following symbolic heaps are used throughout the paper.
\begin{align*}
(\Pi_1, \Sigma_1) &\overset{\text{def}}{=} (\land_{1 \leq i \leq 3} x_i \neq x_j, \text{lseg}_p(x_1, x_3)) \\
(\Pi_2, \Sigma_2) &\overset{\text{def}}{=} (\land_{1 \leq i < j \leq 3} x_i \neq x_j, \text{lseg}_p(x_1, x_2) \ast \text{lseg}_p(x_2, x_3)) \\
(\Pi_3, \Sigma_3) &\overset{\text{def}}{=} (\land_{1 \leq i \leq 3} x_i \neq x_j, \text{lseg}_{\alpha \oplus \alpha}(x_1, x_3) \ast \text{lseg}_{\alpha}(x_2, x_3) \ast \text{lseg}_p(x_2, x_1)).
\end{align*}

Let $\mathcal{P} = (P_\mathcal{P}, 1_\mathcal{P}, \oplus_\mathcal{P})$ be a fixed permission model and let $\text{Loc} = \{l, \ldots\}$ be a countably infinite set of locations (by default, $\text{Loc} = \mathbb{N}$). A $\mathcal{P}$-\textit{memory state} is a tuple $(s, h, i)$ where:
s is a store i.e. a function $s : \text{LVAR} \rightarrow \text{Loc}$ that assigns to each variable a location,

$h$ is a $\mathfrak{P}$-heap i.e. a partial function with a finite domain $h : \text{Loc} \rightarrow_{\text{fin}} P\mathfrak{P} \times \text{Loc},$

$\imath$ is a $\mathfrak{P}$-interpretation.

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

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 $\ell \notin \text{dom}(h_1) \cap \text{dom}(h_2)$ 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 $h_1, h_2$ is defined, so equal to the $\mathfrak{P}$-heap $h$, it takes the unique value satisfying the conditions below:

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

The composition of heaps is partial, commutative, associative, and cancellative. We write $h' \sqsubseteq h$ if there is an $h''$ so that $h = h' \bullet h''$ and we also write $h' \sqsubseteq h$ whenever $h' \sqsubseteq h$ and $h' \neq h$.

The satisfaction relations $s, h, \imath \models_{\mathfrak{P}} \Sigma$ or $s, h, \imath \models_{\mathfrak{P}} \Pi$ are defined below:

$s, h, \imath \models_{\mathfrak{P}} x = y$ always
$s, h, \imath \models_{\mathfrak{P}} x \neq y$ if $s(x) = s(y)$
$s, h, \imath \models_{\mathfrak{P}} A$ if $\imath \models A$
$s, h, \imath \models_{\mathfrak{P}} \Pi_1 \land \Pi_2$ if $s, h, \imath \models_{\mathfrak{P}} \Pi_1$ and $s, h, \imath \models_{\mathfrak{P}} \Pi_2$
$s, h, \imath \models_{\mathfrak{P}} \text{exp}$ if $\text{dom}(h) = \emptyset$
$s, h, \imath \models_{\mathfrak{P}} x \overset{p}{\rightarrow} y$ if $\text{dom}(h) = \{s(x)\}$, $[p]$, is defined, and $h(s(x)) = ([p], s(y))$
$s, h, \imath \models_{\mathfrak{P}} \text{bseg}_{p}(x, y)$ if $[p]$, is defined, and either $s(x) = s(y)$ and $\text{dom}(h) = \emptyset$ or $h = \{\ell_0 \mapsto ([p]_0, \ell_1), \ell_1 \mapsto ([p]_1, \ell_2), \ldots, \ell_{n-1} \mapsto ([p]_n, \ell_n)\}$ with $n \geq 1$, $\ell_0 = s(x)$, $\ell_n = s(y)$ and for all $i \neq j \in [0, n]$, $\ell_i \neq \ell_j$
$s, h, \imath \models_{\mathfrak{P}} \Sigma_1 \ast \Sigma_2$ if there are subheaps $h_1, h_2$ such that $h = h_1 \bullet h_2$, $s, h_1, \imath \models_{\mathfrak{P}} \Sigma_1$, and $s, h_2, \imath \models_{\mathfrak{P}} \Sigma_2$.

Our definitions of symbolic heaps and models encompass the standard definitions without permissions: choose $\mathfrak{P}_1 = (\{1\}, 1, \oplus_{\mathfrak{P}_1})$ that has only the write permission and the always undefined composition. By way of example, given the permission model $\mathfrak{P}_\text{Boy}$, let $(s, h, \imath)$ 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 $\imath(\alpha) = 0.25$. Then, taking the $(\Pi_1, \Sigma_1)$’s from Example 2, we have $s, h, \imath \not\models_{\mathfrak{P}_\text{Boy}} (\Pi_1, \Sigma_1)$ but $s, h, \imath \models_{\mathfrak{P}_\text{Boy}} (\Pi_2, \Sigma_2)$ (since 0.5 can be split into 0.25 + 0.25) and $s, h, \imath \models_{\mathfrak{P}_\text{Boy}} (\Pi_3, \Sigma_3)$.

Satisfiability and entailment problems. A symbolic heap $(\Pi, \Sigma)$ is $\mathfrak{P}$-satisfiable if there is a $\mathfrak{P}$-memory state $(s, h, \imath)$ such that $s, h, \imath \models_{\mathfrak{P}} \Pi$ and $s, h, \imath \models_{\mathfrak{P}} \Sigma$ and we say that $(s, h, \imath)$ is a $\mathfrak{P}$-model of $(\Pi, \Sigma)$. Two symbolic heaps $(\Pi, \Sigma)$ and $(\Pi', \Sigma')$ are equivalent, written $(\Pi, \Sigma) \equiv_{\mathfrak{P}} (\Pi', \Sigma')$, if they have the same $\mathfrak{P}$-models. The $\text{Satisfiability problem w.r.t.} \mathfrak{P}$, written SAT$(\mathfrak{P})$, takes as input $(\Pi, \Sigma)$ and asks whether $(\Pi, \Sigma)$ has a $\mathfrak{P}$-model. The $\text{Entailment problem w.r.t.} \mathfrak{P}$, written ENTS($\mathfrak{P}$)) takes as input two symbolic heaps $(\Pi, \Sigma)$ and $(\Pi', \Sigma')$ and asks whether every $\mathfrak{P}$-model of $(\Pi, \Sigma)$ is a $\mathfrak{P}$-model of $(\Pi', \Sigma')$ (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 ENTS($\mathfrak{P}$).
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 $\Sigma$, we denote by $\text{defined}(\Sigma)$ the conjunction of formulae $\text{defined}(p)$ for all $p$ occurring in $\Sigma$.

\begin{lemma}
The rewrite relation $\Rightarrow$ has the following properties.
\begin{itemize}
\item If $(\Pi, \Sigma) \Rightarrow (\Pi', \Sigma')$ then $(\Pi, \Sigma) \equiv (\Pi', \Sigma')$.
\item Any rewrite sequence starting from $(\Pi, \Sigma)$ terminates in time $O(|(\Pi, \Sigma)|)$.
\end{itemize}
\end{lemma}

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 $\Rightarrow$.

### 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 $\Sigma$, we denote by $\text{defined}(\Sigma)$ the conjunction of formulae $\text{defined}(p)$ for all $p$ occurring in $\Sigma$.

\begin{lemma}
Given $(\Pi, \Sigma)$ in normal form, $\Pi, \Sigma$ is satisfiable iff $(\Pi_{pc} \land \text{defined}(\Sigma))$ is satisfiable.
\end{lemma}

Consequently, we can provide complexity upper bounds for SATSH($\Psi$).

\begin{theorem}
Let $\Psi$ be a permission model and $C \supseteq \text{PTime}$ be a complexity class such that SAT($\Psi$) is in $C$. Then SATSH($\Psi$) restricted to symbolic heaps without list predicates is in $C$.
\end{theorem}

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 $(\Pi_l, \Sigma_l)$ 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 $x' \xrightarrow{p'} y'$ that occur in $\Sigma_r$ to the $x \xrightarrow{p} y$ that occur in $\Sigma_l$, such that the sum of all permissions terms $p'$ mapped to a given $x \xrightarrow{p} y$ is smaller or equal to $p$, with an equality required if $\top$ does not occur in $\Sigma_r$. 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 subst of Figure 2. Intuitively, we remove each points-to predicate of $\Sigma_r$, one by one, until $\Sigma_r$ is satisfiable.
trivial. This new rewrite relation, denoted by $\Rightarrow_{AS}$, terminates in at most $|\Sigma_r|$ steps, and it preserves the entailment validity: if $(\Pi_l, \Sigma_l, \Sigma_r) \Rightarrow_{AS} (\Pi'_l, \Sigma'_l, \Sigma'_r)$ then $(\Pi_l, \Sigma_l) \models (\top, \Sigma_r)$ iff $(\Pi'_l, \Sigma'_l) \models (\top, \Sigma'_r)$.

\begin{align}
\text{(ALIGN)} \quad (\Pi, \Sigma_l * x \xRightarrow{\Sigma} y, \Sigma_r * x' \xRightarrow{\Sigma} y') & \implies (\Pi \land \text{defined}(p), \Sigma_l, \Sigma_r) \\
\text{if } \Pi \models p = p' \land x = x' \land y = y' &
\end{align}

\begin{align}
\text{(SUBTRACT)} \quad (\Pi, \Sigma_l * x \xRightarrow{\Sigma} y, \Sigma_r * x' \xRightarrow{\Sigma} y') & \implies (\Pi \land \text{defined}(\Sigma_l) \land p = p' \lor \alpha \text{ is satisfiable}, \alpha \notin \text{PVar}(\Pi, \Sigma_l, \Sigma_r)) \\
\text{if } \Pi \models x = x' \land y = y' &
\end{align}

\begin{figure}[h]
\centering
\caption{Rewrite rules for triples $(\Pi, \Sigma, \Sigma')$}
\end{figure}

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

\begin{lemma}
The algorithm terminates, and returns \text{true} iff $(\Pi, \Sigma) \models (\Pi', \Sigma')$.
\end{lemma}

\begin{theorem}
The problem $\text{ENTSH}(\mathfrak{P})$ restricted to symbolic heaps without list predicates can be decided in polynomial time with an oracle for $\text{ENT}(\mathfrak{P})$.
\end{theorem}

\section{Reasoning on Symbolic Heaps with Lists and Permissions}

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

\begin{align}
(\Pi, \Sigma) \models_I (\Pi', \Sigma') & \iff (\Pi, \Sigma * \top) \models (\Pi', \Sigma' * \top). \quad \text{(intuitionistic entailment)} \\
(\Pi, \Sigma) \models_{NI} (\Pi', \Sigma') & \iff (\Pi, \Sigma) \models (\Pi', \Sigma'). \quad \text{(non-intuitionistic entailment)}
\end{align}

Below, we provide the developments for $\models_{NI}$ only but all our results can be adapted for the full problems $\text{SATSH}(\mathfrak{P})$ and $\text{ENTSH}(\mathfrak{P})$. In the permission model $\mathfrak{P}_l$, SATSH($\mathfrak{P}_l$) and
We assume a fixed permission model \( \mathcal{P}_1 \) are in PTIME \([6]\) but untractability of SATSH(\( \mathcal{P} \)) and ENTSH(\( \mathcal{P} \)) happens quite quickly, even with the rather simple permission model \( \mathcal{P}_\text{Boyer} \). 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 quite quickly, even with the rather simple permission model \( \mathcal{P}_\text{Boyer} \) the spatial formula

\[
\alpha \quad \text{the spatial formula}
\]

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 \([6, \text{Section 5}]\). The main difference below rests on the replacement of Boolean conjunctions by separating conjunctions. Let \( \Pi_G \) be the pure formula \( (\bigwedge_{(x_i, x_j) \in E}(x_i \neq x_j)) \land y_1 \neq y_2 \land y_1 \neq y_3 \land y_2 \neq y_3 \). Let \( \Sigma_G \) be the spatial formula \( y_1 \otimes y_2 \land y_2 \land y_3 \vdash y_1 \otimes \alpha \land \star \land \alpha \). All the symbolic heaps are assumed to be built from \( \{x_1, \ldots, x_n\} \). Given \( \emptyset \neq X \subseteq \{x_1, \ldots, x_n\} \), \( \min(X) \) denotes the variable in \( X \) with the minimal index.

\[\text{ENTSH}(\mathcal{P}_1) \text{ are in PTIME } [6] \text{ but untractability of SATSH}(\mathcal{P}) \text{ and ENTSH}(\mathcal{P}) \text{ happens quite quickly, even with the rather simple permission model } \mathcal{P}_\text{Boyer}. \text{ A positive consequence of Theorem 9 is that we can use non-determinism to get optimal complexity bounds.} \]

### 4.2 SL-graphs and homomorphisms

We assume a fixed permission model \( \mathcal{P} \) 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 \( \{x_1, \ldots, x_q\} \). 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 \( (\neq) \) 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.

\[\text{Figure 3 SL-graphs}\]

An SL-graph \( G \) is either \( \bot \) or a tuple \( (A, V, \rightarrow, \Rightarrow, \neq, L) \) such that

- \( A \) is a permission formula and \( V \) is a non-empty finite subset of \( \mathbb{N} \) (the nodes).
\[ \rightarrow \text{ and } \Rightarrow \text{ are finite subsets of } V \times PT \times V \text{ where } PT \text{ is the set of permission terms.} \]

We also write \( v \xrightarrow{\phi} v' \) [resp. \( v \xrightarrow{\phi} v' \)] instead of \( (v,p,v') \in \rightarrow \) [resp. \( (v,p,v') \in \Rightarrow \)]. We require a functionality condition: \( v \xrightarrow{\phi} v' \) implies the unicity of \( p \) and \( v' \).

\[ \xrightarrow{\phi} \text{ is an irreflexive and symmetric binary relation on } V. \]

\[ L : \{ x_1, \ldots, x_n \} \rightarrow V \text{ is a surjective labelling.} \]

Given \( v \in V \), we write \( \text{vars}(v) \) to denote the (non-empty) set \( \{ x \mid L(x) = v \} \) and \( \text{var}(v) \overset{\text{def}}{=} \text{min}(\text{vars}(v)) \). As expected, the arrow \( v \xrightarrow{\phi} v' \) [resp. \( v \xrightarrow{\phi} v' \)] is intended to represent the atomic formula \( x \xrightarrow{\phi} x' \) [resp. \( \xrightarrow{\phi} x' \)] whenever \( L(x) = v \) and \( L(x') = v' \).

Moreover, we write \( v \xrightarrow{\phi} v' \) whenever \( v \xrightarrow{\phi} v' \) or \( v \xrightarrow{\phi} v' \). 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 \( \rightarrow \) 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 \( G = (A,V,\rightarrow,\Rightarrow,\phi,L) \), \((\text{pure}(G),\text{spatial}(G))\) denotes the symbolic heap defined from \( G \):

\[
\text{pure}(G) \overset{\text{def}}{=} A \land ( \bigwedge_{x_i,x_j,L(x_i)=L(x_j)} x_i = x_j ) \land ( \bigwedge_{v \xrightarrow{\phi} v'} \text{var}(v) \neq \text{var}(v') ) .
\]

\[
\text{spatial}(G) \overset{\text{def}}{=} \bigwedge_{v \xrightarrow{\phi} v'} \chi(v) \overset{\text{def}}{=} \chi(v \xrightarrow{\phi} v') \overset{\text{def}}{=} \text{var}(v) \xrightarrow{\phi} \text{var}(v') \text{ and } \chi(v \xrightarrow{\phi} v') \overset{\text{def}}{=} \text{seg}_p(\text{var}(v),\text{var}(v')) .
\]

An empty separating conjunction is understood as \emph{emp}. If \( G = \bot \), then the corresponding symbolic heap is \( (x \neq x, T) \). In the sequel, \( \text{spatial}(G) \) is also written \( \bigwedge_{v \xrightarrow{\phi} v'} \chi(v) \overset{\text{def}}{=} \chi(v \xrightarrow{\phi} v') \), where \( \chi(v) \overset{\text{def}}{=} \text{var}(v) \xrightarrow{\phi} \text{var}(v') \text{ and } \chi(v \xrightarrow{\phi} v') \overset{\text{def}}{=} \text{seg}_p(\text{var}(v),\text{var}(v')) \).

An SL-graph \((A,V,\rightarrow,\Rightarrow,\phi,L)\) is \emph{deterministic} iff for all \( v \in V \), \( \text{card}(\{ v' \mid v \xrightarrow{\phi} v' \}) \leq 1 \) and, for all \( v \neq v' \in V \), we have \( v \xrightarrow{\phi} v' \) (\( G_1^2 \) 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 \( \mathcal{Q} \)-memory states (thanks to the determinism and the syntactic nature of the inequality relation) and the SL-graphs (no \( \mathcal{Q} \)-interpretation and no permission values are involved). Each deterministic SL-graph carries a lot of structural properties about the \( \mathcal{Q} \)-memory states that satisfy it, which explains why this is a crucial structure to consider (see also the dependency graphs in [8]).

Let \( G_1 = (A_1,V_1,\rightarrow_1,\Rightarrow_1,\phi_1,L_1) \) and \( G_2 = (A_2,V_2,\rightarrow_2,\Rightarrow_2,\phi_2,L_2) \) 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 \( f : V_1 \rightarrow V_2 \) is a \emph{precise homomorphism} from \( G_1 \) to \( G_2 \) whenever the conditions below are satisfied:

\[(H_0) \text{ } f \text{ is surjective.} \]

\[(H_1) \text{ } A^* \vDash A_1 \text{ where } A^* = A_2 \land \bigwedge_{i,j} \text{defined}(p_i) \land \bigwedge_{j} \text{defined}(p'j) \text{, } \rightarrow_2 = \{ v_1 \xrightarrow{\phi_2} v_1', \ldots, v_n \xrightarrow{\phi_2} v_n' \} \text{ (multiset notation).} \]

\[(H_2) \text{ For all } v \in V_1, \text{ we have } \text{vars}(v) \subseteq \text{vars}(f(v)). \]

\[(H_3) \text{ For all } v, v' \in V_1, \text{ } v \xrightarrow{\phi_1} v' \text{ implies } f(v) \xrightarrow{\phi_2} f(v'). \]

\[(H_4) \text{ For all } v, v' \in V_1, \text{ } v \xrightarrow{\phi_1} v' \text{ implies there is some permission term } p' \text{ such that } f(v) \xrightarrow{\phi_2} f(v') \text{ contributes to the edge } v \xrightarrow{\phi_1} v'. \]

\[(H_5) \text{ For all } v, v' \in V_1, \text{ } v \xrightarrow{\phi_1} v' \text{ implies either } (f(v) = f(v')) \text{ and } A^* \vDash \text{defined}(p) \text{ or there is a path } v_0 \xrightarrow{\phi_2} v_1 \xrightarrow{\phi_2} v_2 \cdots \xrightarrow{\phi_2} v_n \text{ with } n \geq 1, v_0 = f(v), v_n = f(v') \text{ and for all} \]

\[ v_i \rightarrow_2 v_i' \text{ for all } i. \]
\[i < j \in [0,n], v_i \neq v_j \text{ (equivalent to } v_i \not\sim_2 v_j \text{ since } G_2 \text{ is deterministic). For each edge } v_k \not\sim_2 v_{k+1}, \text{ we say that it contributes to the edge } v \not\sim_1 v'. \text{ Since } G_2 \text{ is deterministic, there is a unique path satisfying the above condition (if any).} \]

(H6) Each edge \(v \not\sim_2 v'\) in \(G_2\) contributes to at least one edge of \(G_1\), and we have
\[A^* \models \bigoplus \{p \mid v \not\sim_2 v' \text{ contributes to } u \not\sim_1 u' \in G_1\} = p'.\]

Above, given a non-empty and finite multisets \(T = \{\{p_1, \ldots, p_k\}\}\) of permission terms, we write \(\bigoplus T\) 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 \(f\) implies that for all \(u \not\sim_1 u' \in G_1\), we have \(A^* \models \text{defined}(p)\), which is partly justified by \(\text{defined}(p_1 \oplus p_2) \equiv \text{defined}(p_1) \land \text{defined}(p_2)\).

The dotted arrows in Figure 3 partly materialize two precise homomorphisms from \(G_1\) to \(G_2\) or to \(G_2\) (assuming the permission formulae match).

A precise homomorphism \(f\) is strongly precise \(\iff\)

\[(H1') \quad A_2 = A_1 \land \bigwedge_{v \not\sim_2 v' \text{ in } G_1} \text{defined}(p) \text{ (which implies (H1))}, \]

\[(H5') \quad \text{is equal to (H5) except that in the case } f(v) = f(v'), \text{ we do not require that } A^* \models \text{defined}(p). \]

\[(H6'') \quad \text{each edge } v \not\sim_2 v' \text{ in } G_2 \text{ contributes to at least one edge of } G_1, \text{ and we have }\]
\[\bigoplus \{p \mid v \not\sim_2 v' \text{ contributes to } u \not\sim_1 u' \in G_1\} = p' \text{ modulo AC (implying (H6')).} \]

In Figure 3, there is a strongly precise homomorphism from \(G_2\) to \(G_2\) [resp. to \(G_2\)] with the adequate permission formula. Notably, checking whether \(f : V_1 \to V_2\) is a [resp. strongly] precise homomorphism can be checked in \(\text{coNP}\) [resp. \(\text{PTime}\)] when \(\text{ENT}(\Psi)\) is in \(\text{coNP}\), which is useful to establish the complexity upper bounds.

### 4.3 From symbolic heaps to SL-graphs

Let \((\Pi, \Sigma)\) be a symbolic heap, possibly with list predicates and, \(\Sigma\) is \(\text{emp-free}\) and \(T\)-free. Let us define the SL-graph \(\text{slg}(\Pi, \Sigma)\) as follows. If \((\Pi, \Sigma) \Rightarrow^* \perp\), then \(\text{slg}(\Pi, \Sigma)\) is defined as the inconsistent SL-graph \(\perp\) (see Section 3 for the definition of \(\Rightarrow^*\)).

Otherwise, assume that \((\Pi, \Sigma) \Rightarrow^* (\Pi', \Sigma')\) and \((\Pi', \Sigma')\) is in normal form (i.e., no rule can be further fired from \((\Pi', \Sigma')\)). Let us define \(\text{slg}(\Pi, \Sigma) \equiv (A, \Sigma, V, \Rightarrow, \Rightarrow, L)\) as follows:

- \(A \equiv \Pi_{\text{var}} \land \bigwedge \text{defined}(p)\).
- \(L(x_i) \equiv \min\{j \in V \mid \Pi' \models x_i = x_j\}\).
- If \(x \not\Rightarrow y\) occurs in \(\Sigma'\), then \(L(x) \not\Rightarrow L(y)\); if \(\text{lseg}_p(x, y)\) occurs in \(\Sigma'\), then \(L(x) \Rightarrow L(y)\).
- For all \(i \neq j \in [1, q]\), we have \((L(x_i), L(x_j)) \in \Rightarrow, \Rightarrow \Pi' \models x_i \neq x_j\).

In Figure 3, \(G_1 = \text{slg}(\perp, \text{lseg}, \text{emp}, (x_1, x_3) \not\Rightarrow (x_3, x_2) \not\Rightarrow (x_2, x_1))\). The soundness of the constructions between symbolic heaps and SL-graphs is best illustrated by Lemma 10.

**Lemma 10.** \((\Pi, \Sigma) \equiv \Psi (\text{pure}(\text{slg}(\Pi, \Sigma)), \text{spatial}(\text{slg}(\Pi, \Sigma)))\).

### 4.4 Relating memory states and deterministic SL-graphs

Given \(G_1\) and \(G_2\), we write \(f_{G_1, G_2} : V_1 \to V_2\) to denote the map s.t. \(f_{G_1, G_2}(v) \equiv L_2(\text{var}(v))\). Without any further assumption, note that \(f_{G_1, G_2}\) is not necessarily a precise homomorphism.
Given a $\mathfrak{P}$-memory state $(s,h,i)$ and a deterministic SL-graph $G = (A,V,\rightarrow,\Rightarrow,\mathfrak{F},L)$, we write $(s,h,i) \mathop{\approx}^{\text{lin}} G$ 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 $(s,h,i)$ from $s(x)$ to $s(x'')$,
2. there is no non-empty sequence of memory cells in $(s,h,i)$ from $s(x')$ to $s(x'')$.

Roughly speaking, $(s,h,i) \mathop{\approx}^{\text{lin}} G$ holds when $(s,h,i)$ 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 enterprise. An auxiliary definition is needed. Given a deterministic SL-graph $G'$ and a symbolic heap $(\Pi, \Sigma)$, we write $(\text{pure}(G'), \text{spatial}(G')) \models^{\text{lin}} \Pi, \Sigma$ iff for all $\mathfrak{P}$-memory states $(s,h,i)$ such that $(s,h,i) \mathop{\approx}^{\text{lin}} G$, if $(s,h,i) \models (\text{pure}(G'), \text{spatial}(G'))$, then $(s,h,i) \models \Pi, \Sigma$.

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

- $(\text{pure}(G'), \text{spatial}(G')) \models^{\text{lin}} \Pi, \Sigma$
- $(\text{pure}(G'), \text{spatial}(G')) \models (\text{pure}(G), \text{spatial}(G))$.

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

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

\textbf{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 $G_2 = \text{slg}(x_1 \neq x_3 \land x_1 \neq x_2 \land \alpha = \alpha', \text{lsseg}_v(x_1, x_2) + \text{lsseg}_v(x_1, x_3) + x_2 \mathop{\Rightarrow}^{\alpha'} x_1$). 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 $\models^{\text{lin}}$).

\textbf{Theorem 12.} Let $(\Pi, \Sigma)$, $(\Pi', \Sigma')$ be symbolic heaps s.t. neither slg$(\Pi, \Sigma)$ nor 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 \( v \xrightarrow{p} v' \) in \( G \), the permission term \( p \) is a sum of at most |\( \Sigma \)| terms from \( \text{slg}(\Pi, \Sigma) \). Moreover, the permission formula in \( G \) is precisely the permission part of \( \Pi \).

(SAT) \( (\text{pure}(G), \text{spatial}(G)) \) is satisfiable.

(PRE) \( \text{idg}(\Pi, \Sigma, G) \) is a strongly precise homomorphism.

(NOTPRE) \( \text{idg}(\Pi', \Sigma', G) \) is not a precise homomorphism.

Here is our characterisation for satisfiability checking.

\begin{itemize}
  \item \textbf{Theorem 13.} Let \( (\Pi, \Sigma) \) be a symbolic heap so that \( \Sigma \) is \( \top \)-free and \( \text{emp} \)-free and \( \text{slg}(\Pi, \Sigma) \) is not equal to \( \bot \). Then, \( (\Pi, \Sigma) \) is satisfiable iff there is a deterministic SL-graph \( G \) such that (SMALL), (SAT) and (PRE) hold.
\end{itemize}

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

\begin{itemize}
  \item \textbf{Theorem 14.} (I) \( \text{SAT}(\Psi_{\text{Boy}}) \) is \( \text{NP} \)-complete and \( \text{ENTSH}(\Psi_{\text{Boy}}) \) is \( \text{coNP} \)-complete.
  
  (II) For any permission model \( \Psi \) such that \( \text{width}(\Psi) = \omega \) and, \( \text{ENT}(\Psi) \) is in \( \text{coNP} \), \( \text{SAT}(\Psi) \) is \( \text{NP} \)-complete and \( \text{ENTSH}(\Psi) \) is \( \text{coNP} \)-complete.
\end{itemize}

Currently, the permission terms do not allow constants other than \( 1 \) but for many permission models, constants can be easily added. For instance, in \( \Psi_{\text{Boy}} \), one can deal with \( \frac{3}{4} \) by using the variable \( \alpha \) thanks to \((\alpha' + \alpha' \circ \alpha' \oplus \alpha' = 1) \land (\alpha = \alpha' \oplus \alpha' \oplus \alpha') \) (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 \( \Psi_{\text{Boy}} \).

\section{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 \( \Psi_1 \) and the fractional model \( \Psi_{\text{Boy}} \) that admit \( \text{PTIME} \) 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.

\begin{itemize}
  \item \textbf{Theorem 15.} Let \( \Psi \) and \( \Psi' \) be permission models. (I) \( \text{SAT}(\Psi) = \text{SAT}(\Psi') \) implies \( \text{SATSH}(\Psi) = \text{SATSH}(\Psi') \). (II) \( \text{ENT}(\Psi) = \text{ENT}(\Psi') \) implies \( \text{ENTSH}(\Psi) = \text{ENTSH}(\Psi') \).
\end{itemize}

We briefly review the permission models that have been introduced in the literature. The singleton model \( \Psi_1 \) 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 \( \Psi_{\text{Boy}} \) already presented in Section 2, is one of the most popular permission models and it enjoys nice complexity properties stated below.

\begin{itemize}
  \item \textbf{Theorem 16.} \( \text{SAT}(\Psi_{\text{Boy}}) \) and \( \text{ENT}(\Psi_{\text{Boy}}) \) are in \( \text{PTIME} \).
\end{itemize}

The \text{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 \( \Psi_{\text{Tok}} = (P_{\text{Tok}}, 1_{\text{Tok}}, \oplus_{\text{Tok}}) \), where a permission \( \pi \in P_{\text{Tok}} \) is either a finite or a co-finite, non-empty subset of \( \mathbb{N} \), \( 1_{\text{Tok}} \) is \( \mathbb{N} \), and \( \pi \oplus_{\text{Tok}} \pi' \) 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 \( \mathcal{P}_{\text{Bin}} = (\mathcal{T}(\mathcal{F}) / \equiv, \oplus, 1) \) where \( \mathcal{T}(\mathcal{F}) \) is the set of closed terms constructed over the function symbols \( \mathcal{F} = \{ f : 2, 0 : 0, 1 : 0 \} \), \( \equiv \) is the least congruence such that \( f(0, 0) \equiv 0 \) and \( f(1, 1) \equiv 1 \), \( \mathcal{T}(\mathcal{F}) / \equiv \) denotes the quotient, and \( \oplus \) is defined by \( 0 \oplus 1 \equiv 1 \not\equiv 0 \equiv 1, 0 \not\equiv 0, 1 \not\equiv 1 \) 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 defined(\( \alpha \oplus \alpha \)) is unsatisfiable and \( \lseg_{\alpha \oplus \alpha}(x, y) \models \psi_{\alpha} \iff x \neq y \).

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 \( \mathcal{B} = (B_\land, \lor, \top, \bot, \neg) \) be a Boolean algebra. The permission model \( \mathcal{P}_B \) associated to \( \mathcal{B} \) is \( \mathcal{P}_B \equiv (P_B, \oplus, \top_B, \bot_B, \neg_B) \) where \( P_B = B_\land \setminus \{ \bot \} \) and \( \pi \oplus_B \pi' \) is defined when \( \pi \land_B \pi' = \bot_B \), and in that case \( \pi \oplus_B \pi' \equiv \pi \lor_B \pi' \). A permission model is Boolean if it is isomorphic to \( \mathcal{P}_B \) for some Boolean algebra \( \mathcal{B} \). Both \( \mathcal{P}_{\text{Tok}} \) and \( \mathcal{P}_{\text{Bin}} \) 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 \( \mathcal{P} \) be a Boolean permission model with at least two elements such that \( A_L \models_{\mathcal{P}} A_R \), and width(\( \mathcal{P} \)) \( \geq \text{card} \langle \text{PVar}(A_L) \rangle \). Then for all Boolean permission models \( \mathcal{P}' \), we have \( A_L \models_{\mathcal{P}'} A_R \).

Lemma 17 entails \( \text{ENT}(\mathcal{P}_{\text{Tok}}) = \text{ENT}(\mathcal{P}_{\text{Bin}}) \). The case \( \text{card}(P_\mathcal{B}) = 1 \) cannot be added to Lemma 17 since \( \top \models_{\mathcal{P}_1} \alpha_1 = \alpha_2 \) but \( \top \not\models_{\mathcal{P}_{\alpha \oplus \alpha}} \alpha_1 \not\equiv \alpha_2 \) with \( \alpha_1 \) different from \( \alpha_2 \). Note also that if we could express the atomicity of a permission, \( \mathcal{P}_{\text{Tok}} \) and \( \mathcal{P}_{\text{Bin}} \) could be distinguished.

**Theorem 18.** \( \text{ENT}(\mathcal{P}_{\text{Tok}}) = \text{ENT}(\mathcal{P}_{\text{Bin}}) \) and \( \text{SAT}(\mathcal{P}_{\text{Tok}}) = \text{SAT}(\mathcal{P}_{\text{Bin}}) \).

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

**Theorem 19.** Let \( \mathcal{P}_B \) be a Boolean permission model such that \( \text{width}(\mathcal{P}_B) = \omega \). \( \text{SAT}(\mathcal{P}_B) \) is \( \text{NP}-\text{complete} \) and; \( \text{ENT}(\mathcal{P}_B) \) is \( \text{coNP}-\text{complete} \).

A reduction from the \( \text{NP}-\text{complete} \) problem 1-in-3 \( \text{SAT} \) [14] gives the hardness result.

We give below the proof idea for \( \text{SAT}(\mathcal{P}_B) \) is in \( \text{NP} \). Actually, we can consider only permission terms equal to \( 1 \) and of the form \( \bigoplus \alpha_i \) and atomic permission formulae of the form \( \bigoplus \alpha_i = \bigoplus \alpha_j, \bigoplus \alpha_i \leq \bigoplus \alpha_j, \bigoplus \alpha_i = 1 \) or defined(\( \bigoplus \alpha_i \)) and we can assume that each \( A \) contains a conjunct 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 \( \psi_A \) such that \( A \) is satisfiable if and only if \( \psi_A \) is satisfiable. The Boolean/arithmetic variables of \( \psi_A \) taking their values in \( \{0, 1\} \) are precisely \( x_1^n, x_2^n, \ldots, x_n^n \). The formula \( \psi_A \) is a conjunction of formulæ \( \psi_1 \land \cdots \land \psi_n \) where each \( \psi_i \) is built on the variables \( x_1^n, x_2^n, \ldots, x_i^n \). For each \( i \in [1, n] \), we define \( t^i(\alpha_j) = x_j^n \), and \( t^i(p) \) replaces each occurrence of \( \alpha_j \) by \( t^i(\alpha_j) \) and each occurrence of \( \oplus \) by \( + \), each occurrence of \( 1 \) by \( 1 \). Each \( \psi_i \) is a conjunct of constraints defined by: (a) for each \( p = p' \) [resp. \( p \leq p' \)] in \( A \), \( \psi_i \) contains \( t^i(p) \leq t^i(p') \) and \( t^i(p') \leq t^i(p) \) [resp. \( t^i(p') \leq t^i(p') \)] and (b) for each formula defined(p) in \( A \), \( \psi_i \) contains \( t^i(p) \leq 1 \) and \( x_i^n = 1 \) belongs to \( \psi_i \). The next lemma relates \( A \) and \( \psi_A \).

**Lemma 20.** \( A \) is satisfiable if and only if \( \psi_A \) is satisfiable.

This lemma entails that \( \text{SAT}(\mathcal{P}_B) \) is in \( \text{NP} \).
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.

<table>
<thead>
<tr>
<th>Boolean $\mathcal{P}$, width($\mathcal{P}$) = $\omega$ (Prop., $\mathcal{P}$ Bin)</th>
<th>$\mathcal{P}_1$</th>
<th>$\mathcal{P}_{\text{Boy}}$</th>
<th>$\mathcal{P}_{\text{SAT}}$</th>
<th>$\mathcal{P}_{\text{ENT}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAT($\mathcal{P}$)</td>
<td>in PTime</td>
<td>in PTime (Th. 16)</td>
<td>NP-C. (Th. 19)</td>
<td></td>
</tr>
<tr>
<td>ENT($\mathcal{P}$)</td>
<td>in PTime</td>
<td>in PTime (Th. 16)</td>
<td>coNP-C. (Th. 19)</td>
<td></td>
</tr>
<tr>
<td>SATSH($\mathcal{P}$)</td>
<td>in PTime [6]</td>
<td>NP-C. (Th. 9.14(I))</td>
<td>NP-C. (Th. 9.14(II))</td>
<td></td>
</tr>
<tr>
<td>ENTSH($\mathcal{P}$) \ list pred.</td>
<td>in PTime [6]</td>
<td>in PTime (Th. 7)</td>
<td>coNP-C. (Th. 9.14(II))</td>
<td></td>
</tr>
<tr>
<td>ENTSH($\mathcal{P}$)</td>
<td>in PTime [6]</td>
<td>coNP-C. (Th. 9.14(I))</td>
<td>coNP-C. (Th. 9.14(II))</td>
<td></td>
</tr>
</tbody>
</table>

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

References