On the Complexity of Pointer Arithmetic in Separation Logic

James Brotherston[1] and Max Kanovich[1,2]

[1] University College London, UK
[2] National Research University Higher School of Economics, Russian Federation

Abstract. We investigate the complexity consequences of adding pointer arithmetic to separation logic. Specifically, we study an extension of the points-to fragment of symbolic-heap separation logic with sets of simple "difference constraints" of the form $x \leq y + k$, where x and y are pointer variables and k is an integer offset. This extension can be considered a practically minimal language for separation logic with pointer arithmetic.
Most significantly, we find that, even for this minimal language, polynomial-time decidability is already impossible: satisfiability becomes NP-complete, while quantifier-free entailment becomes coNP-complete and quantified entailment becomes Π_2^P-complete (where Π_2^P is the second class in the polynomial-time hierarchy).
However, the language does satisfy the small model property, meaning that any satisfiable formula A has a model of size polynomial in A, whereas this property fails when richer forms of arithmetical constraints are permitted.

Keywords: Separation logic, pointer arithmetic, complexity.

1 Introduction

Separation logic (SL) [23] is a well-known and popular Hoare-style framework for verifying the memory safety of heap-manipulating programs. Its power stems from the use of *separating conjunction* in its assertion language, where $A * B$ denotes a portion of memory that can be split into two disjoint fragments satisfying A and B respectively. Using separating conjunction, the *frame rule* becomes sound [27], capturing the fact that any valid Hoare triple can be extended with the same separate memory in its pre- and postconditions and remain valid, which empowers the framework to scale to large programs (see e.g. [26]). Indeed, separation logic now forms the basis for verification tools used in industrial practice, notably Facebook's Infer [8] and Microsoft's SLAyer [3].

Most separation logic analyses and tools restrict the form of assertions to a simple propositional structure known as *symbolic heaps* [2]. Symbolic heaps are (possibly existentially quantified) pairs of so-called "pure" and "spatial" assertions, where pure assertions mention only equalities and disequalities between variables and spatial formulas are $*$-conjoined lists of pointer formulas $x \mapsto y$ and data structure formulas typically describing segments of *linked lists* ($\mathsf{ls}\, x\, y$) or sometimes binary trees. This fragment of the logic enjoys decidability in polynomial time [11] and is therefore highly suitable for use in large-scale analysers. However, in recent years, various authors have investigated the computational complexity of (and/or developed prototype analysers for) many other fragments employing various different assertion constructs, including user-defined inductive predicates [18, 5, 7, 1, 10], pointers with *fractional permissions* [22, 13], arrays [6, 19], separating *implication* ($-\!\!*$) [9, 4], reachability predicates [14] and arithmetic [20, 21].

It is with this last feature, arithmetic, and more specifically *pointer arithmetic*, with which we are concerned in this paper. Although most programming languages do not allow the explicit use of pointer arithmetic (with the execption of C, where it is nevertheless discouraged), it nevertheless occurs *implicitly* in many programming situations, of which the most common are array indexing and structure / union member selection. For example, a C expression like `ptr[i]` implicitly generates an address expression of the form `ptr+(sizeof(*ptr)*i)`. Thus a program analysis performing bounds checking for C arrays or strings, say, must account for such implicit pointer arithmetic. We therefore set out by asking the following question: *How much pointer arithmetic can one add to separation logic and remain within polynomial time?*

Unfortunately, and perhaps surprisingly, the answer turns out to be: essentially none at all.

We study the complexity of symbolic-heap separation logic with pointers, but no other data structures, when pure formulas are extended by a minimal form of pointer arithmetic. Specifically, we permit only conjunctions of "difference constraints" $x \leq y + k$, where x and y are pointer variables and k is an integer. We certainly do *not* claim that this fragment is appropriate for practical program verification; clearly, lacking constructs for lists or other data structures, and using only a very weak form of arithmetic, it will be insufficiently expressive for most purposes (although it might *possibly* be practical e.g. for some concurrent programs that deal only with shared memory buffers of a small fixed size). The point is that any practical fragment of separation logic employing pointer arithmetic will almost inevitably include our minimal language and thus inherit its computational lower bounds.

We establish tight complexity bounds for the satisfiability and entailment problems, in both quantified and quantifier-free forms, for our minimal SL pointer arithmetic. Perhaps our most striking result is that the satisfiability problem is already NP-complete; however, the language does at least enjoy the *small model property*, meaning that any satisfiable symbolic heap A has a model of size polynomial in A (a property that fails when richer forms of arithmetical constraints are permitted in the language). In the case of the entailment problem, the problem becomes coNP-complete for quantifier-free entailments and Π_2^P-complete for entailments with existential quantifiers in the consequent (where Π_2^P is the second class in the *polynomial-time hierarchy* [25]). In all cases, the lower bounds follow by reduction from the 3-colourability problem or its 2-round variant [15]. The upper bounds are by straightforward encodings into Presburger arithmetic, but the Π_2^P upper bound for quantified entailments is *not* trivial, as it requires us to show that all quantified variables in the resulting Presburger formula can be polynomially bounded; again, a small model property.

The remainder of this paper is structured as follows. In Section 2 we define symbolic-heap separation logic with minimal pointer arithmetic. Sections 3 and 4 study the satisfiability and quantifier-free entailment problems, respectively, for this language, and Sections 5 and 6 establish the lower and upper complexity bounds, respectively, for the general entailment problem. Section 7 concludes.

This is a workshop version of the paper, representing almost-finished work. We apologise in advance for any remaining presentational inconsistencies.

2 Separation logic with minimal pointer arithmetic

Here, we introduce a minimal language for *separation logic with pointer arithmetic* ($\mathsf{SL}_{\mathsf{MPA}}$ for short), a simple variant of the well-known "symbolic heap" fragment over pointers [2].

Our choice of language is influenced primarily by the need to 'balance' the arithmetical part of the language against the spatial part. To show lower complexity bounds, we have to challenge the fact that Σ_1^0 Presburger arithmetic is already NP-hard by itself; thus, to reveal the true memory-related nature of the problem, we restrict the language to a minimal form of pointer arithmetic, which is simple enough that it can be processed in polynomial time. This leads us to consider only conjunctions of "difference constraints" of the form $x = y + k$, and $x \leq y + k$ where x and y are variables and k is an integer (even disequality, $x' \neq x$ is not permitted).

Definition 2.1 (Syntax). *A* **symbolic heap** *is given by*

$$\exists \mathbf{z}.\ \Pi : F$$

where $\mathbf{z}$ *is a tuple of variables from an infinite set* Var*, and* Π *and* F *are respectively* pure *and* spatial *formulas, defined along with* terms t *by:*

$$\begin{aligned} t &::= x \mid x + k \\ \Pi &::= x = t \mid x \leq t \mid \Pi \wedge \Pi \\ F &::= \mathsf{emp} \mid t \mapsto t \mid t \mapsto \mathsf{nil} \mid F * F \end{aligned}$$

where x *ranges over* Var *and* k *over integers* $\mathbb{Z}$*. If* Π *is empty in a symbolic heap* $\exists \mathbf{z}.\ \Pi : F$*, we omit the colon. We sometimes abbreviate* $*$*-conjunctions of spatial formulas using "big star" notation:*

$$\mathop{\Huge *}_{i=1}^{n} A_i \ =_{def}\ A_1 * \ldots * A_n\ ,$$

which is interpreted as emp *if* $n < 1$*.*

In our $\mathsf{SL}_{\mathsf{MPA}}$, the pure part of a symbolic heap is a conjunction of *'difference constraints'* of the form $x = y + k$ or $x \leq y + k$, where x and y are variables, and k is a fixed offset in $\mathbb{Z}$. Thus $x < y + k$ can be encoded as $x \leq y + (k-1)$, $x \leq y - k$ as $x \leq y + (-k)$ and $x + k \leq y$ as $x \leq y - k$; however, note that unlike the conventional symbolic heap fragment in [2], we *cannot* express disequality $x \neq y$. The satisfiability of such formulas can be decided in polynomial time; see [12]. The crucial observation for polynomial-time decidability is:

Proposition 2.2. *A 'circular' system of difference constraints* $x_1 \leq x_2 + k_{12}$, $\ldots, x_{m-1} \leq x_m + k_{m-1,m}$, $x_m \leq x_1 + k_{m,m+1}$ *implies that* $x_1 - x_1 \leq \sum_{i=1}^{m} k_{i,i+1}$, *which is a contradiction iff the latter sum is negative.*

Semantics. As usual, we interpret symbolic heaps in a stack-and-heap model; for convenience we consider locations to be natural numbers, and values to be either natural numbers or a non-addressable null value *nil*. Thus a *stack* is a function $s\colon \mathsf{Var} \to \mathbb{N} \cup \{nil\}$. We extend stacks to terms by $s(\mathsf{nil}) = nil$ and, insisting that any pointer-offset sum should always be non-negative: $s(x+k) = s(x) + k$ if $s(x) + k \geq 0$, and undefined otherwise. If s is a stack, $z \in \mathsf{Var}$ and v is a value, we write $s[z \mapsto c]$ for the stack defined as s except that $s[z \mapsto v](z) = v$. We extend stacks pointwise over term tuples.

A *heap* is a finite partial function $h\colon \mathbb{N} \rightharpoonup_{\mathrm{fin}} \mathbb{N} \cup \{nil\}$ mapping finitely many locations to values; we write $\mathrm{dom}\,(h)$ for the domain of h, and e for the empty heap that is undefined on all locations. We write $\circ$ for *composition* of domain-disjoint heaps: if h_1 and h_2 are heaps, then $h_1 \circ h_2$ is the union of h_1 and h_2 when $\mathrm{dom}\,(h_1)$ and $\mathrm{dom}\,(h_2)$ are disjoint, and undefined otherwise.

Definition 2.3. *The* satisfaction relation $s, h \models A$, *where s is a stack, h a heap and A a symbolic heap, is defined by structural induction on A.*

$$
\begin{array}{lcl}
s,h \models x \sim y + k & \Leftrightarrow & s(x) \sim s(y) + k \quad \textit{where} \ \sim \ \textit{is} = \textit{or} \leq \\
s,h \models \Pi_1 \wedge \Pi_2 & \Leftrightarrow & s,h \models \Pi_1 \ \textit{and} \ s,h \models \Pi_2 \\
s,h \models \mathsf{emp} & \Leftrightarrow & h = e \\
s,h \models t_1 \mapsto t_2 & \Leftrightarrow & \mathrm{dom}\,(h) = \{s(t_1)\} \ \textit{and} \ h(s(t_1)) = s(t_2) \\
s,h \models F_1 * F_2 & \Leftrightarrow & \exists h_1, h_2.\ h = h_1 \circ h_2 \ \textit{and} \ s,h_1 \models F_1 \ \textit{and} \ s,h_2 \models F_2 \\
s,h \models \exists \mathbf{z}.\ \Pi : F & \Leftrightarrow & \exists \mathbf{m} \in \mathbb{N}^{|\mathbf{z}|}.\ s[\mathbf{z} \mapsto \mathbf{m}], h \models \Pi \ \textit{and} \ s[\mathbf{z} \mapsto \mathbf{m}], h \models F
\end{array}
$$

We remark that the satisfaction of pure formulas Π does not depend on the heap, which justifies writing $s \models \Pi$ rather than $s, h \models \Pi$.

3 Satisfiability and the small model property

In this section we investigate the *satisfiability* problem for our $\mathsf{SL}_{\mathsf{MPA}}$, defined formally as follows:

Satisfiability problem for $\mathsf{SL}_{\mathsf{MPA}}$. *Given a symbolic heap A, decide whether there is a stack s and heap h with $s, h \models A$. (Without loss of generality, we may consider A to be quantifier-free.)*

We establish three main results about this problem: (a) an NP upper bound; (b) an NP lower bound; and (c) the small model property, meaning that any satisfiable formula has a model of polynomial size.

In fact, the NP upper bound is fairly trivial; there is a simple encoding of the satisfiability problem into Σ^0_1 *Presburger arithmetic* (as is also done for a more complicated *array separation logic* in [6]). Nevertheless, we include the details here, since they will be useful in setting up later results.

Definition 3.1. Presburger arithmetic *(*PbA*) is defined as the first-order theory of the natural numbers* $\mathbb{N}$ *over the signature* $\langle 0, s, +, = \rangle$*, where* s *is the successor function, and* $0, +, =$ *have their usual interpretations. The relations* $\neq$*,* $\leq$ *and* $<$ *can be straightforwardly encoded (possibly introducing an existential quantifier).*

Note that a stack is just a first-order valuation, and a pure formula in $\mathsf{SL}_{\mathsf{MPA}}$ is also a formula of PbA, with exactly the same interpretation. Thus we overload $\models$ to include the standard first-order satisfaction relation of PbA.

Definition 3.2. *Let* A *be a quantifier-free symbolic heap, of the general form*

$$\Pi : \mathop{\Huge\ast}_{i=1}^{m} t_i \mapsto u_i \ .$$

We define a corresponding PbA *formula* γ_A *by enriching the pure part* Π *with the constraints that the allocated addresses* t_i *must be distinct:*

$$\gamma_A =_{def} \Pi \wedge \bigwedge\nolimits_{1 \leq i < j \leq m} \ t_i \neq t_j \ .$$

The above γ_A can be easily rewritten as a Boolean combination of elementary formulas of the form $x \leq y + k$ where the 'offset' k is an integer.

Lemma 3.3. *For any symbolic heap* A *in* $\mathsf{SL}_{\mathsf{MPA}}$*, we have*

$$s, h \models A \ \Leftrightarrow \ s \models \gamma_A \ .$$

Proof. We assume A of the general form given by Definition 3.2.

($\Rightarrow$) By assumption, we have $s \models \Pi$ and $\mathrm{dom}\,(h) = \{s(t_1), \ldots, s(t_m)\}$, which implies that all the t_i are distinct. Hence $s \models \gamma_A$ as required.

($\Leftarrow$) By assumption, we have $s \models \Pi$ and all of $s(t_1), \ldots, s(t_m)$ are distinct. Hence, defining a heap h by $\mathrm{dom}\,(h) = \{s(t_1), \ldots, s(t_m)\}$ and $h(s(t_i)) = u_i$ for each i, we have $s, h \models A$ as required. □

Proposition 3.4. *Satisfiability for* $\mathsf{SL}_{\mathsf{MPA}}$ *is in* NP*.*

Proof. Follows from Lemma 3.3 and the fact that satisfiability for quantifier-free Presburger arithmetic belongs to NP [24]. □

Next, we tackle the lower bound. Satisfiability is shown NP-hard by reduction from the 3-*colourability problem [15].*

3-colourability problem. *Given an undirected graph with $n \geq 4$ vertices, decide whether there is a "perfect" 3-colouring of the vertices, such that no two adjacent vertices share the same colour.*

Definition 3.5. *Let $G = (V, E)$ be a graph with n vertices $v_1, \ldots, v_n$. We encode a perfect 3-colouring of G with the following symbolic heap A_G.*

First, we introduce n variables $c_1, \ldots, c_n$ to represent the colour (1, 2, or 3) assigned to each vertex. The fact that no two adjacent vertices v_i and v_j share the same colour will be encoded by allocating two cells with base address $e_{ij} \in \mathbb{N}$ and offsets c_i and c_j respectively in A_G. To ensure that all *such pairs of cells are disjoint, the base addresses e_{ij} are defined by:*

$$e_{ij} = i \cdot n^2 + j \cdot n \quad (1 \leq i < j \leq n) \tag{1}$$

We then define A_G to be the following quantifier-free symbolic heap:

$$\bigwedge_{i=1}^{n}(a + 1 \leq c_i \wedge c_i \leq a + 3) : \; \mathop{\ast}_{(v_i,v_j)\in E} (c_i + e_{ij} \mapsto \mathsf{nil} * c_j + e_{ij} \mapsto \mathsf{nil})$$

where a is a "dummy" variable (ensuring that A_G adheres to the strict formatting of pure assertions in $\mathsf{SL}_{\mathsf{MPA}}$).

The relevant fact concerning our definition of the base addresses e_{ij} in Definition 3.5 is the following.

Proposition 3.6. *For distinct pairs of numbers (i, j) and (i', j'), with $1 \leq i, i', j, j' \leq n$, we have $|e_{i',j'} - e_{ij}| \geq n$.*

Although for the present purposes we *could* have used a simpler definition of the e_{ij}, such that they are all spaced 4 cells apart, the definition by equation (1) is convenient as it will be re-used later on; see Definition 5.1.)

Lemma 3.7. *Let G be an instance of the 3-colouring problem. Then A_G from Definition 3.5 is satisfiable iff there is a perfect 3-colouring of G.*

Proof. Let $G = (V, E)$ have vertices $v_1, \ldots, v_n$, where $n \geq 4$.

($\Leftarrow$) Suppose G has a perfect 3-colouring given by assigning a colour b_i to each vertex v_i, with each $b_i \in \{1, 2, 3\}$. We define a stack s by $s(a) = 0$ and $s(c_i) = b_i$ for each $1 \leq i \leq n$. Note that since $b_i \in \{1, 2, 3\}$ we have $s(a + 1) \leq s(c_i) \leq s(a + 3)$ for each i, and so s satisfies the pure part of A_G. Now define heap h by

$$\mathrm{dom}\,(h) =_{\mathrm{def}} \bigcup_{(v_i,v_j)\in E} (\{s(c_i) + e_{ij}\} \cup \{s(c_j) + e_{ij}\})$$

and $h(\ell) = nil$ for all $\ell \in \mathrm{dom}\,(h)$. Clearly, by construction, $s, h \models A_G$ provided that none of the singleton sets involved in the definition of $\mathrm{dom}\,(h)$ are overlapping.

Since we have a perfect 3-colouring of G, for any edge $(v_i, v_j) \in E$ we have $s(c_i) \neq s(c_j)$, so the subsets $\{s(c_i) + e_{ij}\}$ and $\{s(c_j) + e_{ij}\}$ of $\mathrm{dom}\,(h)$ do not overlap. Furthermore, by Proposition 3.6, for any two distinct edges (v_i, v_j) and

$(v_{i'}, v_{j'})$ in E, the base addresses e_{ij} and $e_{i'j'}$ are at least 4 cells apart (because $n \geq 4$). Since $1 \leq s(c_i) \leq 3$ for any i, we cannot have $s(c_i) + e_{ij} = s(c_{i'}) + ei'j'$ either. Thus all involved singleton sets are non-overlapping as required.

($\Rightarrow$) Supposing that $s, h \models A_G$, we define a 3-colouring of G by $b_i = s(c_i) - s(a)$ for each $1 \leq i \leq n$. Since $s \models a + 1 \leq c_i \wedge c_i \leq a + 3$ by assumption, we have $b_i \in \{1, 2, 3\}$ for each i, so this is indeed a 3-colouring. To see that it is a *perfect* 3-colouring, let $(v_i, v_j) \in E$. By construction, we have that $s, h' \models c_i + e_{ij} \mapsto \mathsf{nil} * c_j + e_{ij} \mapsto \mathsf{nil}$ for some subheap h' of h. Using the definition of $*$, this means that $s(c_i) + e_{ij} \neq s(c_j) + e_{ij}$, i.e. $s(c_i) \neq s(c_j)$, and so $b_i \neq b_j$ as required. □

Theorem 3.8. *Satisfiability for* $\mathsf{SL_{MPA}}$ *is* NP*-hard.*

Proof. From Lemma 3.7 and the fact that 3-colourability is NP-hard [15]. □

Corollary 3.9. *Satisfiability in* $\mathsf{SL_{MPA}}$ *is* NP*-complete.*

Proof. From Proposition 3.4 and Theorem 3.8. □

Finally, we tackle the *small model property* for $\mathsf{SL_{MPA}}$; that is, any satisfiable formula A has a model of size polynomial w.r.t. A (see e.g. [1]). But, before we do, we point out that this property breaks if we increase the expressivity of our system only slightly.

Remark 3.10. The small model property fails if we allow our symbolic heaps to contain constraints of the form $x \leq y \pm z$ where x, y and z are *all* variables. In that case, we could define, e.g.,

$$A_n =_{\text{def}} \bigwedge_{i=0}^{n-1} x_{i+1} > x_i + x_i \colon \mathop{\scalebox{1.5}{$\ast$}}_{i=1}^{n} x_i \mapsto \mathsf{nil}$$

(Note that the constraint $x_{i+1} > x_i + x_i$ can be expressed in our syntax, e.g., as $x_i \leq x_{i+1} - y_i \wedge y_i = x_i + 1$.) Then, for any model (s, h) of A_n, and for any $i < n$, we have that $s(x_{i+1}) > 2s(x_i)$, which implies $s(x_{i+1}) > 2^{i+1}$. Thus, (the distances between) at least half the addresses in h must be of exponential size.

In order to prove the small model property for our $\mathsf{SL_{MPA}}$, we need a more workable specification of γ_A:

Definition 3.11. *Given a symbolic heap A , we rewrite the Presburger formula γ_A by replacing every formula $x = y + k$ by $x \leq y + k \wedge y \leq x - k$, and every formula $t_i \neq t_j$ by $t_i \leq t_j - 1 \vee t_j \leq t_i - 1$. Then γ_A can be viewed as*

$$\gamma_A \equiv f_A(Z_1, Z_2, \ldots, Z_m) \tag{2}$$

where $f_A(z_1, z_2, .., z_m)$ is a Boolean function, and within (2) the Boolean variable z_i is substituted with a difference constraint Z_i of the form $x_i \leq y_i + k_i$ (where k_i is an integer).

Proposition 3.12. *Any model (s, h) for a symbolic heap A can be conceived of as a non-negative integer solution to the system $\gamma_{A,\bar{\zeta}}$ given by*

$$\begin{cases} Z_1 & \equiv \zeta_1, \\ \ldots & \ldots \ldots, \\ Z_m & \equiv \zeta_m \ . \end{cases} \tag{3}$$

with an appropriate Boolean vector $\bar{\zeta} = \zeta_1, .., \zeta_m$ such that $f_A(\zeta_1, .., \zeta_m) = \top$.

Proof. Given a model (s, h) of A, we can evaluate each of the Z_i, and then calculate the appropriate $\bar{\zeta} = \zeta_1, \zeta_2, .., \zeta_m$ using the equations in (3). $\square$

Definition 3.13. *In its turn, the system $\gamma_{A,\bar{\zeta}}$, see (3), is encoded by a* constraint graph, *$\widetilde{G}_{A,\bar{\zeta}}$, constructed as follows.*

With each variable x_i, we will associate the node labelled by $\widehat{x_i}$.

In the case of $Z_i \equiv \zeta_i \equiv \top$, we depict the arrow from the node $\widehat{x_i}$ to the node $\widehat{x'_i}$ and label it with k_i, with getting the edge: $\widehat{x_1} \xrightarrow{k_i} \widehat{x'_i}$.

In the case of $Z_i \equiv \zeta_i \equiv \bot$, which means that "$x_i \leq x'_i - k_i - 1$", we depict the opposite arrow from the node $\widehat{x'_i}$ to the node $\widehat{x_i}$ and label it with the number $-k_i - 1$, with getting the edge: $\widehat{x_1} \xleftarrow{-k_i-1} \widehat{x'_i}$.

To provide the connectivity we need for models, we always add, if necessary, a "maximum node" $\widehat{x_0}$, with the constraint "$x_i \leq x_0$" for all x_i. Cf. Figure 1.

Example 3.14. Let A be a symbolic heap of the form: $(y \leq x) \colon x \mapsto \mathsf{nil} \ * \ y \mapsto \mathsf{nil}$, with its γ_A being of the form: $(y \leq x) \wedge ((x \leq y - 1) \vee (y \leq x - 1))$.
Following Proposition 3.12, we rewrite γ_A as: $\gamma_A(x, y) \equiv f_A(Z_0, Z_1, Z_2)$, where $f_A(z_0, z_1, z_2) = (z_0 \wedge (z_1 \vee z_2))$, and Z_0 stands for "$y \leq x$", and Z_1 stands for "$x \leq y - 1$", and Z_2 means "$y \leq x - 1$".
Since Z_1 and Z_2 are mutually exclusive, it suffices to consider the following two Boolean vectors $\bar{\zeta} = \zeta_0, \zeta_1, \zeta_2$:

(a) $\bar{\zeta} = \top, \top, \bot$. We denote the corresponding system of difference constraints' $\gamma_{A,\bar{\zeta}}(x, y)$ by $\gamma_1(x, y)$:

$$\gamma_1(x, y) =_{\mathrm{def}} \gamma_{A,\bar{\zeta}}(x, y) \equiv (y \leq x) \wedge (x \leq y - 1).$$

(b) $\bar{\zeta} = \top, \bot, \top$. We denote the corresponding system $\gamma_{A,\bar{\zeta}}(x, y)$ by $\gamma_2(x, y)$:

$$\gamma_2(x, y) =_{\mathrm{def}} \gamma_{A,\bar{\zeta}}(x, y) \equiv (y \leq x) \wedge (y \leq x - 1).$$

In Figure 1 we show the constraint graphs for γ_1 and γ_2, resp. Notice that, because of $y \leq x$, the node $\widehat{x}$ is a "maximum node" in both cases.

In the case of (a), we have no solution. Namely, there is a negative cycle of the form $\widehat{x} \xrightarrow{0} \widehat{y} \xrightarrow{-1} \widehat{x}$, which provides a contradictory $x \leq x - 1$.

In the case of (b), the minimal weighted path from $\widehat{x}$ to $\widehat{y}$ is of the weight -1, which guarantees that $y = x - 1$ is a model for γ_A and thereby for A. $\square$

(a) $\gamma_1 = (y \le x) \wedge (x \le y-1)$ $\qquad$ (b) $\gamma_2 = (y \le x) \wedge (y \le x-1)$

Fig. 1. The constraint graphs for γ_1 and γ_2 from Example 3.14.

Theorem 3.15 (Small model property). *Let A be a satisfiable symbolic heap in minimal pointer arithmetic. Then we can find a model (s,h) for A in which all values are bounded by M, here $M = \sum_i (|k_i|+1)$, where k_i ranges over all occurrences of numbers occurred in A.*

Proof. According to Proposition 3.12, there is a Boolean vector $\bar{\zeta} = \zeta_1, \zeta_2, .., \zeta_m$ such that the corresponding system, $\gamma_{A,\bar{\zeta}}$, has a solution. Hence, the associated constraint graph, $\widetilde{G}_{A,\bar{\zeta}}$, has no negative cycles, see Proposition 2.2.

We define our small model with the following mapping s with providing an evaluation $(s(x_1), .., s(x_n))$ which makes γ_A true. First we define: $s(x_0) = M$, for the "maximum node" $\widehat{x_0}$. Then $s(x_i)$ is defined as: $M + d_i$, where d_i is the minimal weighted path leading from $\widehat{x_0}$ to $\widehat{x}_i$. (d_i never happens to be positive) E.g., in Example 3.14 the small model is: $s(x) = M$, and $s(y) = M-1$. $\square$

Remark 3.16. In addition, the corresponding polytime sub-procedures are running as the shortest paths procedures with negative weights allowed (e.g., Bellman-Ford algorithm), which provides polynomials of low degrees.

4 Quantifier-free Entailment

We now turn to the *entailment* problem for our $\mathsf{SL_{MPA}}$, given as follows:

Entailment in $\mathsf{SL_{MPA}}$. *Given symbolic heaps A and B, decide whether $s, h \models A$ implies $s, h \models B$ for all stacks s and heaps h (we say $A \models B$ is* valid*).*

Without loss of generality, A may be assumed quantifier-free, and any quantified variables in B assumed disjoint from the free variables in A and B.

In this section, we focus on the case of quantifier-free entailments, for which we establish both an upper and a lower bound of coNP.

Definition 4.1. *Let $A \models B$ be an $\mathsf{SL_{MPA}}$ entailment, where A and B are symbolic heaps of the form*

$$A \;=\; \Pi_A \colon \mathop{\ast}\nolimits_{i=1}^{\ell}\; t_i \mapsto t'_i \quad \text{and} \quad B \;=\; \exists \bar{y}.\, \Pi_B \colon \mathop{\ast}\nolimits_{j=1}^{\ell'}\; u_j \mapsto u'_j$$

We express validity of $A \models B$, by means of the following PbA formula $\varepsilon_{A,B}$:

$$\forall \bar{x}\,(\gamma_A \to \exists \bar{y}\,\big(\gamma_B \wedge \textstyle\bigwedge_i \bigvee_j (t_i = u_j \wedge t'_i = u'_j) \wedge \bigwedge_j \bigvee_i (u_j = t_i \wedge u'_j = t'_i)\big) \qquad (4)$$

where $\gamma_{\_}$ is given by Defn. 3.2, and $\bar{x}$ is the set of all free variables in A and B.

Lemma 4.2. *$A \models B$ is valid (in $\mathsf{SL_{MPA}}$) if and only if $\varepsilon(A, B)$ is valid (in PbA).*

Proof. Similar to Lemma 3.3. $\square$

As an immediate consequence of Lemma 4.2, the general entailment problem for $\mathsf{SL_{MPA}}$ is in Π^0_2 Presburger arithmetic, which corresponds to Π^{EXP}_1 in the *exponential-time hierarchy* [17]. However, as it turns out, this bound is exponentially overstated; as we show in Theorem 6.1, the problem also belongs to the much smaller class Π^P_2, the second class in the polynomial time hierarchy [25]. The crucial difference between Presburger Π^0_2 and polynomial Π^P_2 is that, in the latter, all variables must be *polynomially bounded*.

However, the construction above does yield an optimal upper bound for the quantifier-free version of the problem.

Theorem 4.3. *The quantifier-free entailment problem for $\mathsf{SL_{MPA}}$ is in coNP.*

Proof. According to Lemma 4.2, deciding whether $A \models B$ is valid is equivalent to deciding whether the PbA formula $\varepsilon(A, B)$ is valid. Although $\varepsilon(A, B)$ is in general a Π^0_2 formula, it becomes a Π^0_1 formula when B is quantifier-free; the validity of such formulas can be decided in coNP time. $\square$

We now turn to the small model property. We note that this property is sensitive to the exact form of our arithmetical constraints, and, similar to Remark 3.10, it fails when we allow the addition of two pointer variables.

Theorem 4.4 (Small model property). *Suppose that the quantifier-free entailment $A \models B$ is not valid. Then we can find a counter-model (s, h) such that $(s, h) \models A$ but $(s, h) \not\models B$, in which all values are bounded by $M = \sum_i(|k_i| + 1)$, where k_i ranges over all occurrences of numbers in A and B.*

Proof. (Sketch) The proof follows the structure of the small model property for satisfiability (Theorem 3.15), noting first that we can rewrite the PbA formula $\varepsilon(A, B)$ as a Π^0_2 Boolean combination of difference constraints $x \leq y + k$, similar to Defn. 3.11. $\square$

As for the coNP *lower* bound, we use a construction similar to Definition 3.5, based on the complement of 3-colourability.

Definition 4.5. *Given a graph G with n vertices, and reusing notation from Definition 3.5, we introduce a satisfiable symbolic heap A'_G by:*

$$\bigwedge_{i=1}^{n}(a + 1 \leq c_i \wedge c_i \leq d) : \; \mathop{\ast}_{(v_i,v_j)\in E} c_i + e_{ij} \mapsto \mathsf{nil} * c_j + e_{ij} \mapsto \mathsf{nil}$$

and a satisfiable symbolic heap B'_G by $d \geq a + 4 \wedge \; A'_G$.

Lemma 4.6. *Let G be an instance of the 3-colouring problem, and let A'_G and B'_G be given by Defn. 4.5 above. Then $A'_G \models B'_G$ is not valid iff there is a perfect 3-colouring of G.*

Proof. Let $G = (V, E)$ have n vertices $v_1, \ldots, v_n$, where $n \geq 4$.

($\Leftarrow$) Suppose G has a perfect 3-colouring given by assigning colours $b_i \in \{1, 2, 3\}$ to vertices v_i. By the argument in the ($\Leftarrow$) case of the proof of Lemma 3.7, if we define $s(a) = 0$, $s(c_i) = b_i$ and (new here) $s(d) = 3$ then there is a heap h such that $s, h \models A'_G$. However, we do not have $s, h \models B'_G$ because $s \not\models d \geq a + 4$. Thus $A'_G \models B'_G$ is not valid, as required.

($\Rightarrow$) Conversely, suppose $s, h \models A'_G$ but $s, h \not\models B'_G$ for some (s, h). By construction of B'_G, this implies that $s \not\models a \leq d - 4$, which implies $s(d) \leq s(a) + 3$. We can then use this fact together with the fact that $s, h \models A'_G$ to obtain a 3-colouring of G exactly as in the ($\Rightarrow$) case of the proof of Lemma 3.7. $\Box$

Theorem 4.7. *The quantifier-free entailment problem for* $\mathsf{SL}_{\mathsf{MPA}}$ *is* coNP*-hard, even when both symbolic heaps are satisfiable.*

Proof. Lemma 4.6 gives a reduction from the complement of the 3-colourability problem, which is coNP-hard, using only satisfiable symbolic heaps. $\Box$

Corollary 4.8. *The quantifier-free entailment problem for* $\mathsf{SL}_{\mathsf{MPA}}$ *is* coNP*-complete (even when both symbolic heaps are satisfiable).*

Proof. Theorems 4.3 and 4.7 give the upper and lower bounds respectively. $\Box$

5 Quantified entailment: $\boldsymbol{\Pi_2^P}$ lower bound

In this section, and the following one, we investigate the general form of the entailment problem $A \models B$ for our $\mathsf{SL}_{\mathsf{MPA}}$, where B may contain existential quantifiers. Here, we establish a lower bound for this problem of Π_2^P in the *polynomial-time hierarchy* (see [25]); in the next section we shall establish an identical upper bound.

To prove Π_2^P-hardness, we build a reduction from the so-called *2-round* version of the 3-colourability problem, defined as follows.

2-round 3-colourability problem. *Let $G = (V, E)$ be an undirected graph with $n \geq 4$ vertices and k leaves (vertices of degree 1). The problem is to decide whether or not every 3-colouring of the leaves can be extended to a perfect 3-colouring of the entire graph, such that no two adjacent vertices share the same colour.*

Definition 5.1. *Let $G = (V, E)$ be an instance graph with n vertices and k leaves. In addition to the variables c_i and a and the numbers e_{ij} which we reuse from Definition 3.5, to each edge (v_i, v_j) we also associate a new variable $\widetilde{c_{ij}}$, representing the colour "complementary" to c_i and c_j.*

To encode the fact that no two adjacent vertices v_i and v_j share the same colour, we shall use c_i, c_j, and $\widetilde{c_{ij}}$ as the addresses, relative to the base-offset e_{ij}, for three consecutive cells within a memory chunk of length 3, which forces c_i, c_j, and $\widetilde{c_{ij}}$ to form a permutation *of $(1, 2, 3)$.*

Formally, we define A''_G to be the following quantifier-free symbolic heap:

$$\bigwedge_{i=1}^{k}(a+1 \leq c_i \wedge c_i \leq a+3) \colon \mathop{\ast}\nolimits_{(v_i,v_j)\in E}^{\ell\in\{1,2,3\}} \; a+(e_{ij}+\ell) \mapsto \mathsf{nil}$$

and B''_G to be the following quantified symbolic heap:

$$\begin{array}{l}\exists \bar{z}.\ \bigwedge_{i=1}^{n}(a+1 \leq c_i \leq a+3) \wedge \bigwedge_{(v_i,v_j)\in E}(a+1 \leq \widetilde{c_{ij}} \leq a+3) : \\ \mathop{\ast}\nolimits_{(v_i,v_j)\in E} c_i + e_{ij} \mapsto \mathsf{nil} \ \ast\ c_j + e_{ij} \mapsto \mathsf{nil} \ \ast\ \widetilde{c_{ij}} + e_{ij} \mapsto \mathsf{nil}\end{array} \tag{5}$$

where the existentially quantified variables $\mathbf{z}$ *are all variables occurring in B''_G that are not mentioned explicitly in A''_G. Note both A''_G and B''_G are satisfiable.*

Lemma 5.2. *Let G be an instance of the 2-round 3-colouring problem, and let A''_G and B''_G be given by Defn. 5.1 above. Then $A''_G \models B''_G$ is valid iff there is a perfect 3-colouring of G given any 3-colouring of its leaves.*

Proof. Let $G = (V, E)$ have $n \geq 4$ vertices $v_1, \ldots, v_n$ of which the first k are leaves.

($\Leftarrow$) Suppose that there is a winning strategy such that every 3-colouring of the leaves can be extended to a perfect 3-colouring of the whole G. We will prove that $A''_G \models B''_G$.

Let s, h be a stack-heap pair satisfying $s, h \models A''_G$.

The spatial part of A''_G yields a decomposition of h as the disjoint collection of the cells (we recall that $s(e_{ij}) = e_{ij}$ and $s(\ell) = \ell$):

$$h = \mathop{\ast}_{(v_i,v_j)\in E,\ \ell=1,2,3} s(a) + e_{ij} + \ell \mapsto \mathsf{nil} \tag{6}$$

and $\bigwedge_{i=1}^{k}(s(a)+1 \leq s(c_i) \leq (s(a)+3)$. Take the 3-colouring of the leaves obtained by assigning the colours b_i to the leaves v_1, v_2,..., v_k resp.. where $b_i = s(c_i) - s(a)$. According to the winning strategy, we can assign colours, denote them by b_i, $i > k$, to the rest of vertices v_{k+1}, ..., v_n, resp., obtaining a 3-colouring of the whole G such that no adjacent vertices share the same colour. In addition, we mark edges (v_i, v_j) by $\widetilde{b_{ij}}$ complementary to b_i and b_j.

We extend the stack s for quantified variables in B''_G so that for all $i \leq k$,

$$s(c_i) = s(a) + b_i,$$

and, for each $(v_i, v_j) \in E$, we have $s(\widetilde{c_{ij}}) = s(a) + 6 - b_i - b_j$. The fact that no adjacent vertices v_i and v_j share the same colour means that

$$(s(c_i),\ s(c_j),\ s(\widetilde{c_{ij}}))$$

is a *permutation* of

$$(s(a)+1,\ s(a)+2,\ s(a)+3),$$

and, as a result, (s, h) is also a model for B''_G:

$$h = \mathop{\ast}_{(v_i,v_j)\in E} s(c_i) + e_{ij} \mapsto \mathsf{nil} \ \ast\ s(c_j) + e_{ij} \mapsto \mathsf{nil} \ \ast\ s(\widetilde{c_{ij}}) + e_{ij} \mapsto \mathsf{nil} \tag{7}$$

($\Rightarrow$) As for the opposite direction, let $A''_G \models B''_G$. Since A''_G is satisfiable, there is a model (s,h) for A''_G so that, in particular, h satisfies (6).

We will construct the required winning strategy in the following way. Assume a 3-colouring of the leaves be given by assigning colours, say b_i, to the leaves v_1, v_2,..., v_k respectively. We modify our original s to a stack s' by defining, for each $1 \le i \le k$,

$$s'(c_i) = s(a) + b_i.$$

which does not change the heap h, but provides

$$\bigwedge_{i=1}^{k}(s(a) + 1 \le s'(c_i) \le (s(a) + 3).$$

It is clear that the modified (s',h) is still a model for A''_G, and, hence, a model for B''_G. Then for some stack s_B, which is extension of s' to the existentially quantified variables in B, we get $(s_B,h) \models B''_G$.

For each $1 \le i \le k$, $s_B(c_i) = s'(c_i) = s_B(a) + b_i$, which means that, for $1 \le i \le k$, these $s_B(c_i)$ represent correctly the original 3-colouring of the leaves.

By assigning the colours $b_i = s_B(c_i) - s_B(a)$ to the rest of vertices v_{k+1}, v_{k+2}, ..., v_n resp. we obtain a 3-colouring of the whole G.

The spatial part of the form (7) provides that $s_B(c_i) \ne s_B(c_j)$, which results in that no adjacent vertices v_i and v_j share the same colours b_i and b_j, providing a perfect 3-colouring of G. □

Theorem 5.3. *The general entailment problem for* $\mathsf{SL}_{\mathsf{MPA}}$ *is* Π_2^P*-hard, even when both symbolic heaps are satisfiable.*

Proof. Lemma 5.2 gives a reduction from the 2-round 3-colourability problem, which is Π_2^P-hard [15]. □

6 Quantified entailments: The Π_2^P upper bound

The Π_2^P lower bound is given in Theorem 5.3. For the case of quantified entailments in $\mathsf{SL}_{\mathsf{MPA}}$, we establish here, Theorem 6.1, an upper bound also of Π_2^P, as well as the small model property.

Theorem 6.1. *The entailment problem in minimal pointer arithmetic belongs to* Π_2^P*. Moreover, given A and B, for some conjunction of difference constraints* $R(x_1, x_2, .., x_n, y_1, y_2, .., y_m)$, $A \models B$ *is equivalent to*

$$\forall x_1 \forall x_2 .. \forall x_n \exists y_1 \exists y_2 .. \exists y_m \, R(x_1, x_2, .., x_n, y_1, y_2, .., y_m) \ . \quad (8)$$

where all x_i *are bounded by* $(n+1) \cdot M$ *and all* y_j *by* $(n+m+2) \cdot M$*, where* M *is defined as:* $M = \sum_i (|k_i| + 1)$*, with* k_i *ranging over all occurrences of 'offsets' numbers occurred in A and B.*

Proof. This follows from the small model property provided by Theorem 6.2 □

Theorem 6.2 (Small model property). *Given A and B, quantified symbolic heaps, suppose that $A \models B$, encoded with (4), is not valid.*

Then we can find a counter-model (s, h) such that $(s, h) \models A$ but $(s, h) \not\models B$, in which all x-values and y-values are bounded in accordance with Theorem 6.1.

Proof Sketch. Taking x_1 as a "zero" node, and y_m as a "maximum node", we assume that $x_1 < x_2 < \cdots < x_n$, and $x_n \leq y_m$, and for all y_j, $x_1 \leq y_j \leq y_m$.

Let (s, h) be a concrete counter-model for $A \models B$, such that $s(x_1) = 0$, and, being a model for A, (s, h) be fully determined by the system:

$$\gamma_{A,s}(x_1, x_2, \ldots, x_n) = \bigwedge_{i=1}^{n-1} (x_{i+1} = x_i + d_{i,i+1}) \tag{9}$$

where for all $1 \leq i < j \leq n$, the d_{ij} is defined as: $d_{ij} = s(x_j) - s(x_i)$.

Following Proposition 3.12, the fact that $(s, h) \not\models B$ means that for a certain Boolean function $f_{A,B}$, whatever a Boolean vector $\bar{\zeta} = \zeta_1, .., \zeta_\ell$ such that $f_{A,B}(\zeta_1, .., \zeta_\ell) = \top$ we take, the following system, $G_{A,B,s,\bar{\zeta}}$, has no integer solution for $s(x_1)$, .., $s(x_n)$ fixed by $\gamma_{A,s}$ from (9):

$$\gamma_{A,s}(x_1, x_2, \ldots, x_n) \wedge ((Z_1 \equiv \zeta_1) \wedge \cdots \wedge (Z_\ell \equiv \zeta_\ell)) \tag{10}$$

Given a smaller M, we introduce a smaller counter-model (s', h') by simply replacing all large gaps $d_{i,i+1}$ with *one and the same* smaller M as follows:

$$\gamma_{A,s'}(x_1, x_2, \ldots, x_n) = \bigwedge_{i=1}^{n-1} (x_{i+1} = x_i + d'_{i,i+1}) \tag{11}$$

where $d'_{ij} = s'(x_j) - s'(x_i)$, for $1 \leq i < j \leq n$, and a smaller s' is defined by:

$$s'(x_{i+1}) := \begin{cases} s'(x_i) + d_{i,i+1}, & \text{if } d_{i,i+1} \leq M \\ s'(x_i) + M, & \text{otherwise} \end{cases} \tag{12}$$

Example 6.3. **(On the edge of disaster)** Here we show that it is a real challenge to prove that our (s', h') is a small counter-example we look for.
Assuming $x_1 < x_2 < x_3 < x_4$, let A be of the form

$$(x_1 < x_2) \wedge (x_2 < x_3) \wedge (x_3 < x_4) \wedge (x_3 \leq x_2 + 3) \colon x_1 \mapsto \mathsf{nil} \; * \; x_3 \mapsto \mathsf{nil} \tag{13}$$

and B be of the form

$$\exists y_1 \exists y_2 \exists y_3 \exists y_4 \, (y_2 = x_2) \wedge (y_4 = x_4) \wedge (y_2 \leq y_4 - 5) \wedge (y_3 = y_1 + 7) \colon y_1 \mapsto \mathsf{nil} \; * \; y_3 \mapsto \mathsf{nil} \tag{14}$$

Then the validity of $A \models B$ can be reduced to $\varepsilon_{A,B}$ of the form:

$$\varepsilon_{A,B} \;\equiv\; \forall \bar{x} \, (\gamma_A(\bar{x}) \to \exists \bar{y} \, G^1_B(\bar{x}, \bar{y})) \tag{15}$$

where $G^1_B(\bar{x}, \bar{y})$ is the following conjunction

$$(y_1 = x_1) \wedge (y_3 = x_3) \wedge (y_2 = x_2) \wedge (y_4 = x_4) \wedge (y_2 \leq y_4 - 5) \wedge (y_3 = y_1 + 7) \tag{16}$$

Let (s, h) be a 'large' counter-model for $A \models B$, defined by the following s (here D is a very large number, say 2^{1024}):

$$\begin{cases} s(x_2) = s(x_1) + 2D, \\ s(x_3) = s(x_2) + 2, \\ s(x_4) = s(x_3) + D, \end{cases} \tag{17}$$

Our (s, h), being a model for A, is fully determined by the system:

$$\gamma_{A,s}(x_1, x_2, x_3, x_4) \;=\; (x_2 = x_1 + 2D) \wedge (x_3 = x_2 + 2) \wedge (x_4 = x_3 + D)\ . \tag{18}$$

The constraint graph, $\widetilde{G}_{A,s}$, consists of the following pairs of labelled edges

$$\widehat{x_1} \xrightarrow{2D} \widehat{x_2},\ \ \widehat{x_2} \xrightarrow{-2D} \widehat{x_1},\ \ \widehat{x_2} \xrightarrow{2} \widehat{x_3},\ \ \widehat{x_3} \xrightarrow{-2} \widehat{x_2},\ \ \widehat{x_3} \xrightarrow{D} \widehat{x_4},\ \ \widehat{x_4} \xrightarrow{-D} \widehat{x_3},$$

According to (15), $(s, h) \not\models B$ means that the following system has no solution:

$$\gamma_{A,s}(s(x_1), s(x_2), s(x_3), s(x_4)) \wedge G_B^1(s(x_1), s(x_2), s(x_3), s(x_4), y_1, y_2, y_3, y_4) \tag{19}$$

which is the case because of the cycle with the negative weight, $-D + 2$:

$$\widehat{x_4} \xrightarrow{0} \widehat{y_4} \xrightarrow{-5} \widehat{y_2} \xrightarrow{0} \widehat{x_2} \xrightarrow{-2D} \widehat{x_1} \xrightarrow{0} \widehat{y_1} \xrightarrow{7} \widehat{y_3} \xrightarrow{0} \widehat{x_3} \xrightarrow{D} \widehat{x_4} \tag{20}$$

By our construction, a small counter-model (s', h') is defined with the following s' by replacing the large D and $2D$ with *one and the same* M:

$$\begin{cases} s'(x_2) = s'(x_1) + M, \\ s'(x_3) = s'(x_2) + 2, \\ s'(x_4) = s'(x_3) + M. \end{cases} \tag{21}$$

$(s', h') \not\models B$ means that the following system has no solution, cf. (19):

$$\gamma_{A,s'}(s'(x_1), s'(x_2), s'(x_3), s'(x_4)) \wedge G_B^1(s'(x_1), s'(x_2), s'(x_3), s'(x_4), y_1, y_2, y_3, y_4)$$

A natural idea to detect a cycle with the negative weight for (s', h'), is to take (20) defined for (s, h), and transform it into a hopefully negative cycle in terms of (s', h') by replacing its large D and $2D$ with the modest M, resulting in:

$$\widehat{x_4} \xrightarrow{0} \widehat{y_4} \xrightarrow{-5} \widehat{y_2} \xrightarrow{0} \widehat{x_2} \xrightarrow{-M} \widehat{x_1} \xrightarrow{0} \widehat{y_1} \xrightarrow{7} \widehat{y_3} \xrightarrow{0} \widehat{x_3} \xrightarrow{M} \widehat{x_4} \tag{22}$$

But the weight of this cycle happens to be **positive**. $\square$

The challenge to our construction can be resolved by the following lemma.

Lemma 6.4. *Having got a cycle $\mathcal{C}$ with the negative weight for (10), we can extract a smaller cycle with the negative weight for (10), which is good for (s', h'), as well.*

Proof. We introduce the following reductions for $i < j$:

(a1) Let $\widehat{x_j} \longrightarrow \widehat{y} \overset{\sigma}{\Longrightarrow} \widehat{y'} \longrightarrow \widehat{x_i}$ be a part of $\mathcal{C}$, which does not use edges from $\gamma_{A,s}$, see (9). Here σ is the sum of all integers the edges invoked in this part are labelled by.

In the case where $d_{ij} + \sigma \geq 0$, we replace the above part with the part: $\widehat{x_j} \xrightarrow{-d_{ij}} \widehat{x_i}$.

Since $-d_{ij} \leq \sigma$, the weight of the whole updated $\mathcal{C}$ remains negative.

(a2) For $d_{ij} + \sigma < 0$, we can identify the following cycle with a negative weight:

$$\widehat{x_j} \longrightarrow \widehat{y} \overset{\sigma}{\Longrightarrow} \widehat{y'} \longrightarrow \widehat{x_i} \xrightarrow{d_{ij}} \widehat{x_j} \tag{23}$$

Since $d_{ij} < -\sigma \leq M$, we have $d'_{ij} = d_{ij}$, and hence this smaller cycle with the negative weight is good for (s', h'), as well.

(b1) Let $\widehat{x_i} \longrightarrow \widehat{y} \overset{\sigma}{\Longrightarrow} \widehat{y'} \longrightarrow \widehat{x_j}$ be a part of $\mathcal{C}$, which does not use edges from $\gamma_{A,s}$, see (9).

For $d_{ij} \leq \sigma$, we replace the above part with the part: $\widehat{x_i} \xrightarrow{d_{ij}} \widehat{x_j}$.

Since $d_{ij} \leq \sigma$, the weight of the whole updated $\mathcal{C}$ remains negative.

(b2) For $d_{ij} > \sigma$, we can identify the following cycle with a negative weight:

$$\widehat{x_i} \longrightarrow \widehat{y} \overset{\sigma}{\Longrightarrow} \widehat{y'} \longrightarrow \widehat{x_j} \xrightarrow{-d_{ij}} \widehat{x_i}$$

Suppose that for all k such that $i \leq k < j$, $d_{k,k+1} \leq M$. Then $d'_{ij} = d_{ij}$, and hence this smaller cycle with the negative weight is good for (s', h'), as well. Otherwise, for some k such that $i \leq k < j$, $d_{k,k+1} > M$, and thereby by construction $d'_{k,k+1} = M$, and, hence, $d'_{ij} \geq M$.

Then the following cycle defined in terms of (s', h'),

$$\widehat{x_i} \longrightarrow \widehat{y} \overset{\sigma}{\Longrightarrow} \widehat{y'} \longrightarrow \widehat{x_j} \xrightarrow{-d'_{ij}} \widehat{x_i}$$

is of negative weight, since $\sigma - d'_{ij} \leq \sigma - M < 0$.

E.g., in Example 6.3 the following part of its negative cycle (20):

$$\widehat{x_1} \xrightarrow{0} \widehat{y_1} \xrightarrow{7} \widehat{y_3} \xrightarrow{0} \widehat{x_3}$$

provides the following negative cycle in terms of (s', h'):

$$\widehat{x_1} \xrightarrow{0} \widehat{y_1} \xrightarrow{7} \widehat{y_3} \xrightarrow{0} \widehat{x_3} \xrightarrow{-2} \widehat{x_2} \xrightarrow{-M} \widehat{x_1}$$

We can prove that *any chain of reductions must terminate in (a2) or in (b2)*. This concludes the proof of Lemma 6.4 and thereby of Theorem 6.2. $\square$

Remark 6.5. The proof of Theorem 6.2 provides quite efficient procedures for the entailment problem in Theorem 6.1, in which the corresponding polytime subprocedures are running as the shortest paths procedures with negative weights allowed with providing polynomials of low degrees.

In fact we prove that the entailment problem is Π_2^P-complete, and enjoys the small model property, even if we allow any Boolean combinations of elementary formulas $(x' \leq x + k_0)$, and, in addition to the points-to formulas, we allow spatial formulas of the arrays the length of which is bounded by k_0 and lists which length is bounded by a fixed integer k_0. $\square$

7 Conclusions

In this paper, we study the points-to fragment of symbolic-heap separation logic extended with pointer arithmetic, in a minimal form allowing only conjunctions of difference constraints $x \leq y + k$ for $k \in \mathbb{Z}$.

Perhaps surprisingly, we find that polynomial time algorithms are out of reach even for minimal SL pointer arithmetic: for example, satisfiability is already NP-complete and quantifier-free entailment is coNP-complete.

We point out that, for the case of quantified entailments in minimal pointer arithmetic, we establish here an *exact* upper bound of Π_2^P, as well as the small model property.

We note that some of our upper bound complexity results can be seen as following already from our earlier results for *array separation logic*, where we allow array predicates $\mathrm{array}(x, y)$ as well as pointers and arithmetic constraints. Of course, pointer arithmetic is often an essential feature in reasoning about array-manipulating programs. The main value of our findings, we believe, is in our *lower* bound complexity results, which show that NP-hardness or worse is an inevitable consequence of admitting pointer arithmetic of almost any kind.

We remark that our lower-bound results do however rely on the presence of *pointer* arithmetic, as opposed to arithmetic per se. If pointers and data values are strictly distinguished and arithmetic permitted only over data, as is done e.g. in [16], then polynomial-time algorithms may still be achievable in that case.

References

1. Antonopoulos, T., Gorogiannis, N., Haase, C., Kanovich, M., Ouaknine, J.: Foundations for decision problems in separation logic with general inductive predicates. In: Proc. FoSSaCS-17. pp. 411–425. Springer (2014)
2. Berdine, J., Calcagno, C., O'Hearn, P.: A decidable fragment of separation logic. In: Proc. FSTTCS-24. LNCS, vol. 3328, pp. 97–109. Springer (2004)
3. Berdine, J., Cook, B., Ishtiaq, S.: SLAyer: memory safety for systems-level code. In: Proc. CAV-23. pp. 178–183. Springer (2011)
4. Brochenin, R., Demri, S., Lozes, E.: On the almighty wand. Information and Computation 211, 106–137 (2012)
5. Brotherston, J., Fuhs, C., Gorogiannis, N., Navarro Pérez, J.: A decision procedure for satisfiability in separation logic with inductive predicates. In: Proc. CSL-LICS. pp. 25:1–25:10. ACM (2014)
6. Brotherston, J., Gorogiannis, N., Kanovich, M.: Biabduction (and related problems) in array separation logic. In: Proc. CADE-26. LNAI, vol. 10395, pp. 472–490. Springer (2017)
7. Brotherston, J., Gorogiannis, N., Kanovich, M., Rowe, R.: Model checking for symbolic-heap separation logic with inductive predicates. In: Proc. POPL-43. pp. 84–96. ACM (2016)
8. Calcagno, C., Distefano, D., Dubreil, J., Gabi, D., Hooimeijer, P., Luca, M., O'Hearn, P., Papakonstantinou, I., Purbrick, J., Rodriguez, D.: Moving fast with software verification. In: Proc. NFM-7. LNCS, vol. 9058, pp. 3–11. Springer (2015)

9. Calcagno, C., Yang, H., O'Hearn, P.W.: Computability and complexity results for a spatial assertion language for data structures. In: Proc. FSTTCS-21. pp. 108–119. Springer (2001)
10. Chen, T., Song, F., Wu, Z.: Tractability of separation logic with inductive definitions: Beyond lists. In: Proc. CONCUR-28. pp. 33:1–33:16. Dagstuhl (2017)
11. Cook, B., Haase, C., Ouaknine, J., Parkinson, M., Worrell, J.: Tractable reasoning in a fragment of separation logic. In: Proc. CONCUR-22. LNCS, vol. 6901, pp. 235–249. Springer (2011)
12. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, 3rd edn. (2009)
13. Demri, S., Lozes, E., Lugiez, D.: On symbolic heaps modulo permission theories. In: Proc. FSTTCS-37. pp. 25:1–25:13. Dagstuhl (2017)
14. Demri, S., Lozes, É., Mansutti, A.: The effects of adding reachability predicates in propositional separation logic. In: Proc. FoSSaCS-21. LNCS, Springer (2018), to appear
15. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
16. Gu, X., Chen, T., Wu, Z.: A complete decision procedure for linearly compositional separation logic with data constraints. In: Proc. IJCAR. LNAI, vol. 9706, pp. 532–549. Springer (2016)
17. Haase, C.: Subclasses of Presburger arithmetic and the weak EXP hierarchy. In: Proceedings of CSL-LICS. pp. 47:1–47:10. ACM (2014)
18. Iosif, R., Rogalewicz, A., Simacek, J.: The tree width of separation logic with recursive definitions. In: Proc. CADE-24. LNAI, vol. 7898, pp. 21–38. Springer (2013)
19. Kimura, D., Tatsuta, M.: Decision procedure for entailment of symbolic heaps with arrays. In: Proc. APLAS-15. LNCS, vol. 10695, pp. 169–189. Springer (2017)
20. Le, Q.L., Sun, J., Chin, W.N.: Satisfiability modulo heap-based programs. In: Proc. CAV-28. LNCS, vol. 9779, pp. 382–404. Springer (2016)
21. Le, Q.L., Tatsuta, M., Sun, J., Chin, W.N.: A decidable fragment in separation logic withinductive predicates and arithmetic. In: Proc. CAV-29. LNCS, vol. 10427, pp. 495–517. Springer (2017)
22. Le, X.B., Gherghina, C., Hobor, A.: Decision procedures over sophisticated fractional permissions. In: Proc. APLAS-10. LNCS, vol. 7705, pp. 368–385. Springer (2012)
23. Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In: Proc. LICS-17. pp. 55–74. IEEE (2002)
24. Scarpellini, B.: Complexity of subcases of Presburger arithmetic. Trans. American Mathematical Society 284(1), 203–218 (1984)
25. Stockmeyer, L.J.: The polynomial-time hierarchy. Theoretical Computer Science 3, 1–22 (1977)
26. Yang, H., Lee, O., Berdine, J., Calcagno, C., Cook, B., Distefano, D., O'Hearn, P.: Scalable shape analysis for systems code. In: Proc. CAV-20. LNCS, vol. 5123, pp. 385–398. Springer (2008)
27. Yang, H., O'Hearn, P.: A semantic basis for local reasoning. In: Proc. FOSSACS-5. pp. 402–416. Springer (2002)