On the Complexity of Pointer Arithmetic in Separation Logic

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

[1]UCL

[2]National Research University Higher School of Economics, Russian Federation

ADSL workshop (FLoC), University of Oxford, July 2018

Overview

- Industrial SL analysis is usually based on symbolic heaps over pointers and list segments, which is PTIME-decidable.

Overview

- Industrial SL analysis is usually based on symbolic heaps over pointers and list segments, which is PTIME-decidable.

- Many other features have been studied…
  - user-defined inductive predicates;
  - fractional permissions;
  - separating implication ($-\!\!*$);
  - arrays;
  - reachability predicates;
  - arithmetic;

…but they typically come with a complexity cost.

Overview

- Industrial SL analysis is usually based on symbolic heaps over pointers and list segments, which is PTIME-decidable.
- Many other features have been studied…
  - user-defined inductive predicates;
  - fractional permissions;
  - separating implication ($-\!\!*$);
  - arrays;
  - reachability predicates;
  - arithmetic;

  …but they typically come with a complexity cost.
- Our focus is on pointer arithmetic in SL.

Pointer arithmetic in program analysis

- Pointer arithmetic is usually disallowed or at least discouraged in modern programming practice.

Pointer arithmetic in program analysis

- Pointer arithmetic is usually disallowed or at least discouraged in modern programming practice.
- However, it still arises implicitly, e.g., in array indexing and structure / union member selection.

$$\texttt{ptr[i]} \quad \Rightarrow \quad \texttt{ptr + (sizeof(*ptr)*i)}$$

Pointer arithmetic in program analysis

- Pointer arithmetic is usually disallowed or at least discouraged in modern programming practice.
- However, it still arises implicitly, e.g., in array indexing and structure / union member selection.

$$\texttt{ptr[i]} \quad \Rightarrow \quad \texttt{ptr + (sizeof(*ptr)*i)}$$

- Thus program analyses must deal with pointer arithmetic even when programmers don't!

Pointer arithmetic in program analysis

- Pointer arithmetic is usually disallowed or at least discouraged in modern programming practice.
- However, it still arises implicitly, e.g., in array indexing and structure / union member selection.

$$\texttt{ptr[i]} \quad \Rightarrow \quad \texttt{ptr + (sizeof(*ptr)*i)}$$

- Thus program analyses must deal with pointer arithmetic even when programmers don't!

Question: *How much pointer arithmetic can one add to separation logic and remain within polynomial time?*

Minimal fragment, $\mathsf{SL}_{\mathsf{MPA}}$

- Terms t, pure formulas Π and spatial formulas F given by:

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

  where $x \in \mathsf{Var}$, $k \in \mathbb{Z}$.

Minimal fragment, $\mathsf{SL}_{\mathsf{MPA}}$

- Terms t, pure formulas Π and spatial formulas F given by:

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

  where $x \in \mathsf{Var}$, $k \in \mathbb{Z}$.

- Symbolic heaps given by $\exists \mathbf{x}.\ \Pi : F$.

Minimal fragment, $\mathsf{SL}_{\mathsf{MPA}}$

- Terms t, pure formulas Π and spatial formulas F given by:

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

  where $x \in \mathsf{Var}$, $k \in \mathbb{Z}$.

- Symbolic heaps given by $\exists \mathbf{x}.\ \Pi : F$.

- Semantics given as usual by $s, h \models A$ in a stack-and-heap model over locations $\mathbb{N}$ and values $\mathbb{N} \cup \{nil\}$.

Difference constraints

Pure formulas are conjunctions of difference constraints

$$x \le y + k \ ,$$

where x, y are pointer variables and $k \in \mathbb{Z}$ is an integer offset.

Difference constraints

Pure formulas are conjunctions of difference constraints

$$x \leq y + k \ ,$$

where x, y are pointer variables and $k \in \mathbb{Z}$ is an integer offset.

Note: the satisfiability of these formulas can be decided in polynomial time.

Difference constraints

Pure formulas are conjunctions of difference constraints

$$x \leq y + k \ ,$$

where x, y are pointer variables and $k \in \mathbb{Z}$ is an integer offset.

Note: the satisfiability of these formulas can be decided in polynomial time.

$$\left.\begin{array}{l} x_1 \leq x_2 + k_1, \\ \dots \\ x_{m-1} \leq x_m + k_{m-1}, \\ x_m \leq x_1 + k_m \end{array}\right\} \Rightarrow \ x_1 - x_1 \leq \sum_{i=1}^{m} k_i$$

Problems of interest

Satisfiability problem. *Given symbolic heap A, decide if there is a stack-heap pair (s, h) with $s, h \models A$.*

Problems of interest

Satisfiability problem. *Given symbolic heap A, decide if there is a stack-heap pair (s, h) with $s, h \models A$.*

Entailment problem. *Given symbolic heaps A and B, decide whether $A \models B$.*

Problems of interest

Satisfiability problem. *Given symbolic heap A, decide if there is a stack-heap pair (s, h) with $s, h \models A$.*

Entailment problem. *Given symbolic heaps A and B, decide whether $A \models B$.*

Small model property. *Given a satisfiable symbolic heap A, does A have a model using only addresses and values of size polynomial in A?*

Problems of interest

Satisfiability problem. *Given symbolic heap A, decide if there is a stack-heap pair (s, h) with $s, h \models A$.*

Entailment problem. *Given symbolic heaps A and B, decide whether $A \models B$.*

Small model property. *Given a satisfiable symbolic heap A, does A have a model using only addresses and values of size polynomial in A?*

(The latter fails if we allow pointer sums, $x \leq y + z$.)

Some known upper bounds

$\mathsf{SL}_{\mathsf{MPA}}$ is subsumed by the array separation logic in

J. Brotherston, N. Gorogiannis, and M. Kanovich. Biabduction (and related problems) in array separation logic. In Proc. *CADE* 2017.

Some known upper bounds

$\mathsf{SL}_{\mathsf{MPA}}$ is subsumed by the array separation logic in

> J. Brotherston, N. Gorogiannis, and M. Kanovich. Biabduction (and related problems) in array separation logic. In Proc. *CADE* 2017.

This gives some immediate upper bounds by encoding into Presburger arithmetic (PbA):

- Satisfiability is in NP.

Some known upper bounds

$\mathsf{SL}_{\mathsf{MPA}}$ is subsumed by the array separation logic in

> J. Brotherston, N. Gorogiannis, and M. Kanovich. Biabduction (and related problems) in array separation logic. In Proc. *CADE* 2017.

This gives some immediate upper bounds by encoding into Presburger arithmetic (PbA):

- Satisfiability is in NP.
- Quantifier-free entailment is in coNP.

Some known upper bounds

$\mathsf{SL}_{\mathsf{MPA}}$ is subsumed by the array separation logic in

> J. Brotherston, N. Gorogiannis, and M. Kanovich. Biabduction (and related problems) in array separation logic. In Proc. *CADE* 2017.

This gives some immediate upper bounds by encoding into Presburger arithmetic (PbA):

- Satisfiability is in NP.
- Quantifier-free entailment is in coNP.
- Quantified entailment is in Π_1^{EXP}.

Satisfiability, lower bound

In fact, the lower bound for satisfiability is also NP.

3-colourability problem (NP-hard)

Given an undirected graph, decide whether there is a "perfect" 3-colouring of the vertices, such that no two adjacent vertices share the same colour.

Satisfiability, lower bound

In fact, the lower bound for satisfiability is also NP.

3-colourability problem (NP-hard)

Given an undirected graph, decide whether there is a "perfect" 3-colouring of the vertices, such that no two adjacent vertices share the same colour.

First, choose numbers e_{ij} for each edge (v_i, v_j) such that $|e_{i'j'} - e_{ij}| \geq 4$ for any two distinct edges.

Satisfiability, lower bound

In fact, the lower bound for satisfiability is also NP.

3-colourability problem (NP-hard)

Given an undirected graph, decide whether there is a "perfect" 3-colouring of the vertices, such that no two adjacent vertices share the same colour.

First, choose numbers e_{ij} for each edge (v_i, v_j) such that $|e_{i'j'} - e_{ij}| \geq 4$ for any two distinct edges.

Next take a variable c_i for each vertex v_i.

Satisfiability, lower bound

In fact, the lower bound for satisfiability is also NP.

3-colourability problem (NP-hard)

Given an undirected graph, decide whether there is a "perfect" 3-colouring of the vertices, such that no two adjacent vertices share the same colour.

First, choose numbers e_{ij} for each edge (v_i, v_j) such that $|e_{i'j'} - e_{ij}| \geq 4$ for any two distinct edges.

Next take a variable c_i for each vertex v_i.

Then encode in $\mathsf{SL}_{\mathsf{MPA}}$ as (slightly simplified)

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

Small model property

Suppose A is satisfiable by (s, h).

Small model property

Suppose A is satisfiable by (s, h).

There is an equisatisfiable PbA formula γ_A with $s \models \gamma_A$.

Small model property

Suppose A is satisfiable by (s, h).

There is an equisatisfiable PbA formula γ_A with $s \models \gamma_A$.

The formula γ_A can be written as a Boolean combination of difference constraints $x \leq y + k$.

Small model property

Suppose A is satisfiable by (s, h).

There is an equisatisfiable PbA formula γ_A with $s \models \gamma_A$.

The formula γ_A can be written as a Boolean combination of difference constraints $x \leq y + k$.

Thus s can be viewed as a solution to the equation system

$$x_1 \leq y_1 + k_1 \equiv \zeta_1, \ldots, x_m \leq y_m + k_m \equiv \zeta_m$$

where each $\zeta_i \in \{\top, \bot\}$.

Small model property

Suppose A is satisfiable by (s, h).

There is an equisatisfiable PbA formula γ_A with $s \models \gamma_A$.

The formula γ_A can be written as a Boolean combination of difference constraints $x \leq y + k$.

Thus s can be viewed as a solution to the equation system

$$x_1 \leq y_1 + k_1 \equiv \zeta_1, \ldots, x_m \leq y_m + k_m \equiv \zeta_m$$

where each $\zeta_i \in \{\top, \bot\}$.

Note that $x \leq y + k \equiv \bot$ means $y \leq x - k - 1$.

Small model property (2)

View the difference equations as a constraint graph, as follows:

Small model property (2)

View the difference equations as a constraint graph, as follows:

$$(x_i \leq y_i + k_i) \equiv \top \quad \sim \quad y_i \xrightarrow{k} x_i$$

Small model property (2)

View the difference equations as a constraint graph, as follows:

$$
\begin{array}{lcl}
(x_i \leq y_i + k_i) \equiv \top & \sim & y_i \xrightarrow{k} x_i \\
(x_i \leq y_i + k_i) \equiv \bot & \sim & x_i \xrightarrow{-k-1} y_i
\end{array}
$$

Small model property (2)

View the difference equations as a constraint graph, as follows:

$$\begin{array}{rcl} (x_i \leq y_i + k_i) \equiv \top & \sim & y_i \xrightarrow{k} x_i \\ (x_i \leq y_i + k_i) \equiv \bot & \sim & x_i \xrightarrow{-k-1} y_i \\ \text{all } x_i: & & x_0 \xrightarrow{0} x_i \end{array}$$

where x_0 is a new "maximum node".

Small model property (2)

View the difference equations as a constraint graph, as follows:

$$
\begin{array}{rcl}
(x_i \leq y_i + k_i) \equiv \top & \sim & y_i \xrightarrow{k} x_i \\
(x_i \leq y_i + k_i) \equiv \bot & \sim & x_i \xrightarrow{-k-1} y_i \\
\text{all } x_i\text{:} & & x_0 \xrightarrow{0} x_i
\end{array}
$$

where x_0 is a new "maximum node".

FACT: This graph cannot have a negative-weight cycle (else the equation system would have no solutions).

Small model property (2)

View the difference equations as a constraint graph, as follows:

$$
\begin{array}{rcl}
(x_i \leq y_i + k_i) \equiv \top & \sim & y_i \xrightarrow{k} x_i \\
(x_i \leq y_i + k_i) \equiv \bot & \sim & x_i \xrightarrow{-k-1} y_i \\
\text{all } x_i\text{:} & & x_0 \xrightarrow{0} x_i
\end{array}
$$

where x_0 is a new "maximum node".

FACT: This graph cannot have a negative-weight cycle (else the equation system would have no solutions).

We will construct a model s, h of A in which all values are bounded by

$$M = \Sigma_{i=1}^{m} |k_i| + 1$$

Small model property (3)

Define a new, small model s' of γ_A as follows:

Small model property (3)

Define a new, small model s' of γ_A as follows:

$$d_i \quad = \quad \text{minimal path weight from } x_0 \text{ to } x_i$$

Small model property (3)

Define a new, small model s' of γ_A as follows:

$$\begin{array}{rcl} d_i & = & \text{minimal path weight from } x_0 \text{ to } x_i \\ s'(x_i) & = & M + d_i \end{array}$$

Small model property (3)

Define a new, small model s' of γ_A as follows:

$$\begin{array}{rcl} d_i & = & \text{minimal path weight from } x_0 \text{ to } x_i \\ s'(x_i) & = & M + d_i \end{array}$$

Note d_i is always well defined and $d_i \leq 0$ (a 0-weight path exists by construction, and no negative-weight cycles). So s' is small.

Why is it a model?

Small model property (3)

Define a new, small model s' of γ_A as follows:

$$\begin{array}{rcl} d_i & = & \text{minimal path weight from } x_0 \text{ to } x_i \\ s'(x_i) & = & M + d_i \end{array}$$

Note d_i is always well defined and $d_i \leq 0$ (a 0-weight path exists by construction, and no negative-weight cycles). So s' is small.

Why is it a model?

Consider constraint $x \leq y + k \equiv \top$.

Small model property (3)

Define a new, small model s' of γ_A as follows:

$$\begin{array}{rcl} d_i & = & \text{minimal path weight from } x_0 \text{ to } x_i \\ s'(x_i) & = & M + d_i \end{array}$$

Note d_i is always well defined and $d_i \leq 0$ (a 0-weight path exists by construction, and no negative-weight cycles). So s' is small.

Why is it a model?

Consider constraint $x \leq y + k \equiv \top$. There is an edge $y \xrightarrow{k} x$.

Small model property (3)

Define a new, small model s' of γ_A as follows:

$$\begin{aligned} d_i &= \text{minimal path weight from } x_0 \text{ to } x_i \\ s'(x_i) &= M + d_i \end{aligned}$$

Note d_i is always well defined and $d_i \leq 0$ (a 0-weight path exists by construction, and no negative-weight cycles). So s' is small.

Why is it a model?

Consider constraint $x \leq y + k \equiv \top$. There is an edge $y \xrightarrow{k} x$. Thus $d_x \leq d_y + k$ and so $s'(x) \leq s'(y) + k$.

Small model property (3)

Define a new, small model s' of γ_A as follows:

$$\begin{array}{rcl} d_i & = & \text{minimal path weight from } x_0 \text{ to } x_i \\ s'(x_i) & = & M + d_i \end{array}$$

Note d_i is always well defined and $d_i \leq 0$ (a 0-weight path exists by construction, and no negative-weight cycles). So s' is small.

Why is it a model?

Consider constraint $x \leq y + k \equiv \top$. There is an edge $y \xrightarrow{k} x$. Thus $d_x \leq d_y + k$ and so $s'(x) \leq s'(y) + k$.

So s' satisfies our difference equation system, and thus γ_A.

Small model property (3)

Define a new, small model s' of γ_A as follows:

$$\begin{array}{rcl} d_i & = & \text{minimal path weight from } x_0 \text{ to } x_i \\ s'(x_i) & = & M + d_i \end{array}$$

Note d_i is always well defined and $d_i \leq 0$ (a 0-weight path exists by construction, and no negative-weight cycles). So s' is small.

Why is it a model?

Consider constraint $x \leq y + k \equiv \top$. There is an edge $y \xrightarrow{k} x$. Thus $d_x \leq d_y + k$ and so $s'(x) \leq s'(y) + k$.

So s' satisfies our difference equation system, and thus γ_A. Then we can create a suitable h' with $s', h' \models A$.

Quantifier-free entailment

By adapting the corresponding tricks for satisfiability, we have the following for the quantifier-free entailment problem:

1. a lower bound of coNP;
2. the small model property (any invalid entailment has a small countermodel).

Quantified entailment, lower bound

Lower bound is Π_2^P in the polynomial-time hierarchy.

2-round 3-colourability problem (Π_2^P-hard)

Given an undirected graph, decide whether every 3-colouring of the leaves can be extended to a perfect 3-colouring of the graph.

Quantified entailment, lower bound

Lower bound is Π_2^P in the polynomial-time hierarchy.

2-round 3-colourability problem (Π_2^P-hard)

Given an undirected graph, decide whether every 3-colouring of the leaves can be extended to a perfect 3-colouring of the graph.

Reuse the variables c_i and numbers e_{ij} from previous reduction. Also use $\widetilde{c_{ij}}$ for the colour "complementary" to c_i and c_j.

Quantified entailment, lower bound

Lower bound is Π_2^P in the polynomial-time hierarchy.

2-round 3-colourability problem (Π_2^P-hard)

Given an undirected graph, decide whether every 3-colouring of the leaves can be extended to a perfect 3-colouring of the graph.

Reuse the variables c_i and numbers e_{ij} from previous reduction.
Also use $\widetilde{c_{ij}}$ for the colour "complementary" to c_i and c_j.
Simplified encoding:

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

Quantified entailment, lower bound

Lower bound is Π_2^P in the polynomial-time hierarchy.

2-round 3-colourability problem (Π_2^P-hard)

Given an undirected graph, decide whether every 3-colouring of the leaves can be extended to a perfect 3-colouring of the graph.

Reuse the variables c_i and numbers e_{ij} from previous reduction.
Also use $\widetilde{c_{ij}}$ for the colour "complementary" to c_i and c_j.
Simplified encoding:

LHS: $\bigwedge_{i=1}^{k}(1 \leq c_i \leq 3)\colon \ \mathop{\huge\ast}_{(v_i,v_j)\in E}^{\ell\in\{1,2,3\}} (e_{ij} + \ell) \mapsto \mathsf{nil}$

RHS: $\exists \mathbf{z}.\ \bigwedge_{(v_i,v_j)\in E}^{1\leq k\leq n} (1 \leq c_k, \widetilde{c_{ij}} \leq 3) :$
$\mathop{\huge\ast}_{(v_i,v_j)\in E} c_i + e_{ij} \mapsto \mathsf{nil} \ * \ c_j + e_{ij} \mapsto \mathsf{nil} \ * \ \widetilde{c_{ij}} + e_{ij} \mapsto \mathsf{nil}$

Quantified entailment, upper bound

We have an encoding into Π^0_2 PbA, an upper bound of Π^{EXP}_1 (in the exponential-time hierarchy).

Quantified entailment, upper bound

We have an encoding into Π^0_2 PbA, an upper bound of Π_1^{EXP} (in the exponential-time hierarchy).

In fact this upper bound is exponentially overstated! The upper bound is also Π_2^P.

Quantified entailment, upper bound

We have an encoding into Π_2^0 PbA, an upper bound of Π_1^{EXP} (in the exponential-time hierarchy).

In fact this upper bound is exponentially overstated! The upper bound is also Π_2^P.

The key difference between Π_2^0 PbA and Π_2^P is that in the latter all variables must be polynomially bounded.

Quantified entailment, upper bound

We have an encoding into Π^0_2 PbA, an upper bound of Π_1^{EXP} (in the exponential-time hierarchy).

In fact this upper bound is exponentially overstated! The upper bound is also Π_2^P.

The key difference between Π^0_2 PbA and Π_2^P is that in the latter all variables must be polynomially bounded.

This follows from the small model property for quantified entailment. Construction uses similar ideas to satisfiability case, but is (quite a bit) more complex.

Conclusions

- NP-hardness or worse is an inevitable consequence of adding pointer arithmetic to SL,

Conclusions

- NP-hardness or worse is an inevitable consequence of adding pointer arithmetic to SL,
  - even for pointer data only, and

Conclusions

- NP-hardness or worse is an inevitable consequence of adding pointer arithmetic to SL,
  - even for pointer data only, and
  - even for pointer-offset comparisons, $x \leq y + k$.

Conclusions

- NP-hardness or worse is an inevitable consequence of adding pointer arithmetic to SL,
  - even for pointer data only, and
  - even for pointer-offset comparisons, $x \leq y + k$.
- Satisfiability is NP-complete.

Conclusions

- NP-hardness or worse is an inevitable consequence of adding pointer arithmetic to SL,
  - even for pointer data only, and
  - even for pointer-offset comparisons, $x \leq y + k$.
- Satisfiability is NP-complete.
- Quantifier-free entailment is coNP-complete.

Conclusions

- NP-hardness or worse is an inevitable consequence of adding pointer arithmetic to SL,
  - even for pointer data only, and
  - even for pointer-offset comparisons, $x \leq y + k$.
- Satisfiability is NP-complete.
- Quantifier-free entailment is coNP-complete.
- Quantified entailment is Π_2^P-complete.

Conclusions

- NP-hardness or worse is an inevitable consequence of adding pointer arithmetic to SL,
  - even for pointer data only, and
  - even for pointer-offset comparisons, $x \leq y + k$.
- Satisfiability is NP-complete.
- Quantifier-free entailment is coNP-complete.
- Quantified entailment is Π_2^P-complete.
- The small model property holds.

Thanks for listening!

Revised paper available (my webpage)