Decidability of Entailments in Separation Logic with Arrays

Daisuke Kimura (Toho Univ.)

joint work with Makoto Tatsuta (NII)

Workshop on Mathematical Logic and its Application
JSPS Core-to-Core Program

16-17 Sep. 2016　Kyoto University

Introduction

Separation Logic

- Proposed by J.C.Reynolds in 2002
- Each formula states some state of memory
- Useful for verifying pointer-programs (like C)

On-going our project

- Separation-Logic-Based analyzer for C
- Fully automated system
- Checking memory errors (buffer-overflow, memory-leak)
- One of our main problem:
  decision procedure for entailment problem in SL
- Our current target system : separation logic with arrays

Syntax of SL$_{\text{ARRAY}}$

Terms $\quad t, u, n, m ::= x \mid 0 \mid 1 \mid \ldots \mid t + t \mid t - t$

Pure expressions

$$\Pi ::= t = t \mid t \neq t \mid t < t \mid \Pi \wedge \Pi$$

Spatial expressions

$$\begin{aligned} \Sigma ::= & \, \textit{emp} && \text{(Empty heap)} \\ & \mid t \mapsto (t, \ldots, t) && \text{(Points-to predicate)} \\ & \mid \textit{Array}(t, m) && \text{(Array predicate)} \\ & \mid \Sigma * \Sigma && \text{(Separating conjunction)} \end{aligned}$$

Symbolic Heaps $\quad \Pi \wedge \Sigma$

Heap model

Stores s : Vars $\to \mathbb{N}$

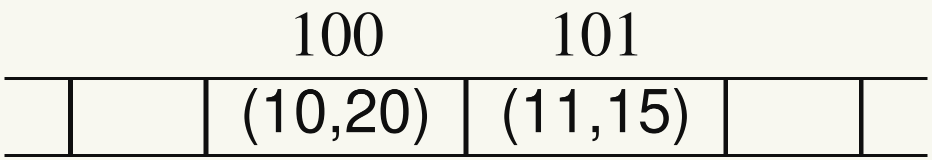
Heaps $h : \mathbb{N} \setminus \{0\} \longrightarrow_{fin} \mathbb{N}^n$ (n is the number of $t \mapsto (t_1, \ldots, t_n)$)

Heap model (s, h)

A heap model means a state of memory
For example, assume that

- $s(x) = 5$
- $\mathrm{Dom}(h) = \{100, 101\}$,
- $h(100) = (10, 20)$, $h(101) = (11, 15)$

This heap model (s, h) means the following memory state



The value of x is 5

Semantics of SL_{ARRAY}

$s \models \Pi$ and $s, h \models \Sigma$ are defined as follows

$$s \models t = u \quad \overset{\text{def}}{\Longleftrightarrow} \quad s(t) = s(u)$$

$$s \models t \neq u \quad \overset{\text{def}}{\Longleftrightarrow} \quad s(t) \neq s(u)$$

$$s \models t < u \quad \overset{\text{def}}{\Longleftrightarrow} \quad s(t) < s(u)$$

$$s \models \Pi_1 \wedge \Pi_2 \quad \overset{\text{def}}{\Longleftrightarrow} \quad s \models \Pi_1 \text{ and } s \models \Pi_2$$

$$s, h \models \textit{emp} \quad \overset{\text{def}}{\Longleftrightarrow} \quad \text{Dom}(h) = \emptyset$$

$$s, h \models t \mapsto (\vec{u}) \quad \overset{\text{def}}{\Longleftrightarrow} \quad h(s(t)) = (s(\vec{u})) \text{ and } \text{Dom}(h) = \{ s(t) \}$$

$$s, h \models \Sigma_1 * \Sigma_2 \quad \overset{\text{def}}{\Longleftrightarrow} \quad s, h_1 \models \Sigma_1 \text{ and } s, h_2 \models \Sigma_2 \text{ for some } h = h_1 + h_2$$

$$s, h \models \textit{Array}(t, m) \quad \overset{\text{def}}{\Longleftrightarrow} \quad \text{Dom}(h) = \{ s(t), \ldots, s(t + m) \}$$

Intuitively, $\textit{Array}(t, m)$ means there is an array starting from t of length $m + 1$

$$s, h \models \Pi \wedge \Sigma \quad \overset{\text{def}}{\Longleftrightarrow} \quad s \models \Pi \text{ and } s, h \models \Sigma$$

Entailments and main result

Entailments of **SL**ARRAY:

$$\Pi_1 \wedge \Sigma_1 \vdash \bigvee_i (\Pi_i \wedge \Sigma_i)$$

The above entailment is said to be valid if

$$s, h \models \Pi_1 \wedge \Sigma_1 \quad \text{implies} \quad s, h \models \Pi_i \wedge \Sigma_i \text{ for some } i$$

holds for any (s, h)

Our main result

Validity of entailments of **SL**ARRAY is decidable

Basic Idea

Approach
Translating entailments into Presburger formulas

Idea: Sorted separating conjunction $\circledast$

$$s, h \models \Sigma_1 \circledast \Sigma_2 \quad \overset{\text{def}}{\iff} \quad \begin{array}{l} s, h_1 \models \Sigma_1 \text{ and } s, h_2 \models \Sigma_2 \\ \text{and } h = h_1 + h_2 \\ \text{and } \max \mathrm{Dom}(h_1) < \min \mathrm{Dom}(h_2) \\ \text{for some } h_1, h_2 \end{array}$$

For example,

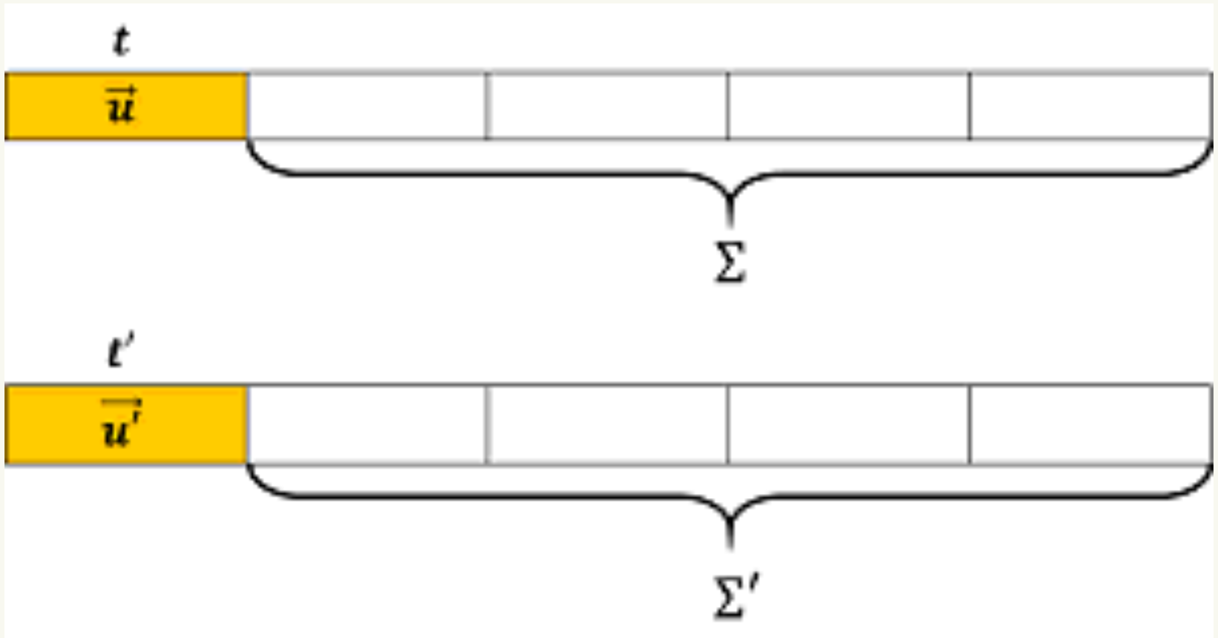
- $1 \mapsto (x) \circledast 2 \mapsto (y) \vdash 1 \mapsto (x) \circledast 2 \mapsto (y)$ is valid
- $1 \mapsto (x) \circledast 2 \mapsto (y) \vdash 2 \mapsto (y) \circledast 1 \mapsto (x)$ is invalid

Idea of translation

Observation1 ($\mapsto\mapsto$ case)

$$\Pi \wedge t \mapsto (\vec{u}) \circledast \Sigma \models \Pi' \wedge t' \mapsto (\vec{u'}) \circledast \Sigma'$$
$$\iff \quad \Pi \wedge t < \Sigma \wedge \Sigma \models \Pi' \wedge t = t' \wedge \vec{u} = \vec{u'} \wedge \Sigma'$$

where $t < \Sigma$ means that t is less than the first address of Σ



Idea of translation

Observation1 ($\mapsto\mapsto$ case)

$$\Pi \land t \mapsto (\vec{u}) \circledast \Sigma \models \Pi' \land t' \mapsto (\vec{u'}) \circledast \Sigma'$$
$$\iff \quad \Pi \land t < \Sigma \land \Sigma \models \Pi' \land t = t' \land \vec{u} = \vec{u'} \land \Sigma'$$

Example

$$1 \mapsto (x) \circledast 2 \mapsto (y) \models 1 \mapsto (x) \circledast 2 \mapsto (y)$$
$$\iff \quad 1 < 2 \land 2 \mapsto (y) \models 1 = 1 \land x = x \land 2 \mapsto (y)$$
$$\iff \quad 1 < 2 \models 1 = 1 \land x = x \land 2 = 2 \land y = y$$
$$\iff \quad \models_{PbA} 1 < 2 \Longrightarrow (1 = 1 \land x = x \land 2 = 2 \land y = y)$$

Idea of translation

Observation2 ($\mapsto$Array case)

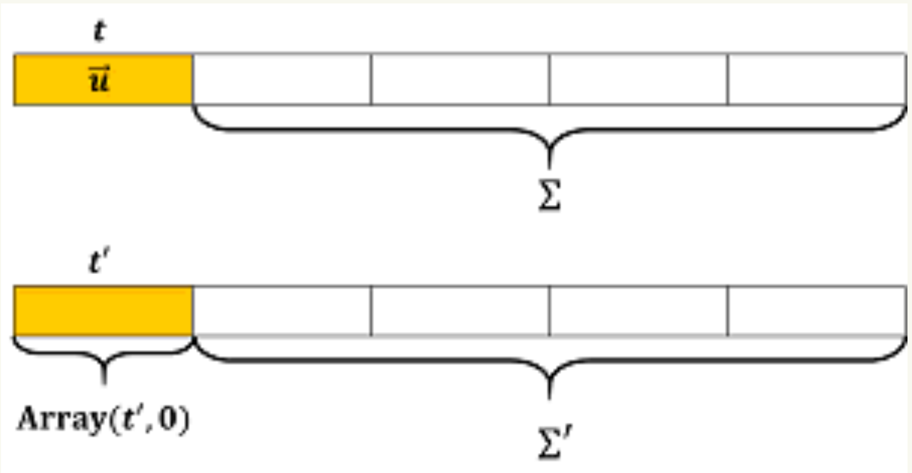
$$\Pi \wedge t \mapsto (\vec{u}) \circledast \Sigma \models \Pi' \wedge \mathit{Array}(t', m') \circledast \Sigma'$$

$$\Longleftrightarrow$$

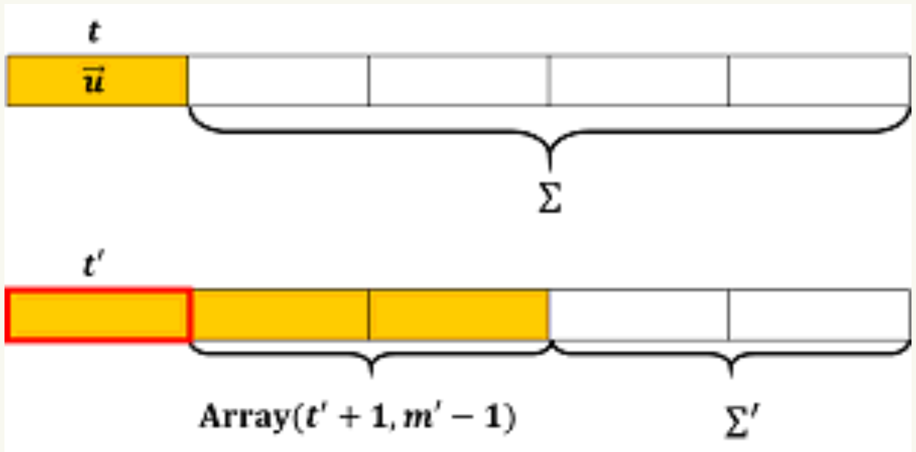
$$\Pi \wedge m' = 0 \wedge t \mapsto (\vec{u}) \circledast \Sigma \models \Pi' \wedge t = t' \wedge t' \mapsto (\vec{u}) \circledast \Sigma'$$

and

$$\Pi \wedge m' > 0 \wedge t \mapsto (\vec{u}) \circledast \Sigma \models \Pi' \wedge t = t' \wedge t' \mapsto (\vec{u}) \circledast \mathit{Array}(t' + 1, m' - 1) \circledast \Sigma'$$



Upper case ($m' = 0$)



Lower case ($m' > 0$)

Idea of translation

Observation3 (Array$\mapsto$ case)

$\Pi \wedge \mathit{Array}(t, m) \circledast \Sigma \models \Pi' \wedge t' \mapsto (\vec{u'}) \circledast \Sigma'$

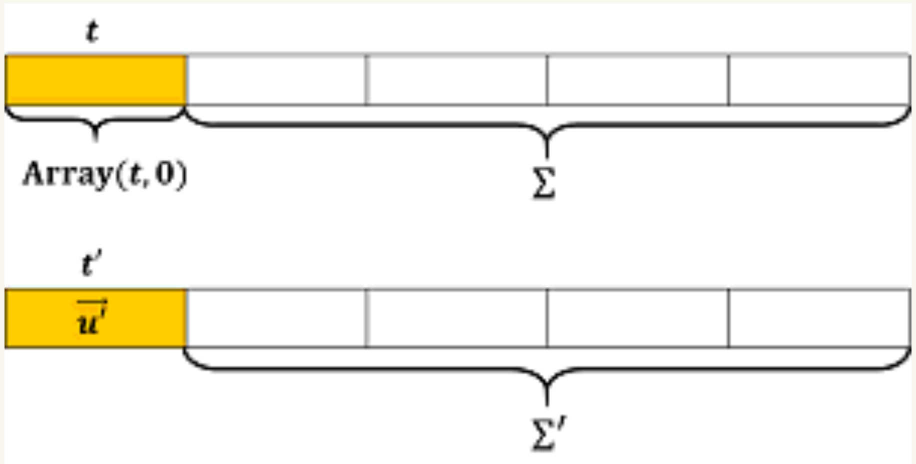
$\Longleftrightarrow$

$\Pi \wedge m = 0 \wedge t \mapsto (\vec{z}) \circledast \Sigma \models \Pi' \wedge t = t' \wedge t' \mapsto (\vec{u'}) \circledast \Sigma'$

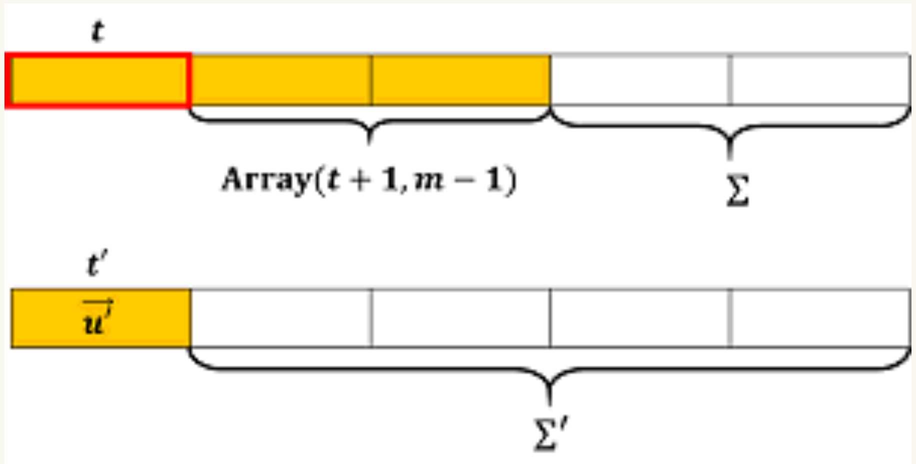
and

$\Pi \wedge m > 0 \wedge t \mapsto (\vec{z'}) \circledast \mathit{Array}(t + 1, m - 1) \circledast \Sigma \models \Pi' \wedge t = t' \wedge t' \mapsto (\vec{u'}) \circledast \Sigma'$

$\vec{z}, \vec{z'}$: fresh



Upper case ($m = 0$)



Lower case ($m > 0$)

Idea of translation

Observation4 (ArrayArray case)

$\Pi \wedge \mathit{Array}(t, m) \circledast \Sigma \models \Pi' \wedge \mathit{Array}(t', m') \circledast \Sigma'$

$\Longleftrightarrow$

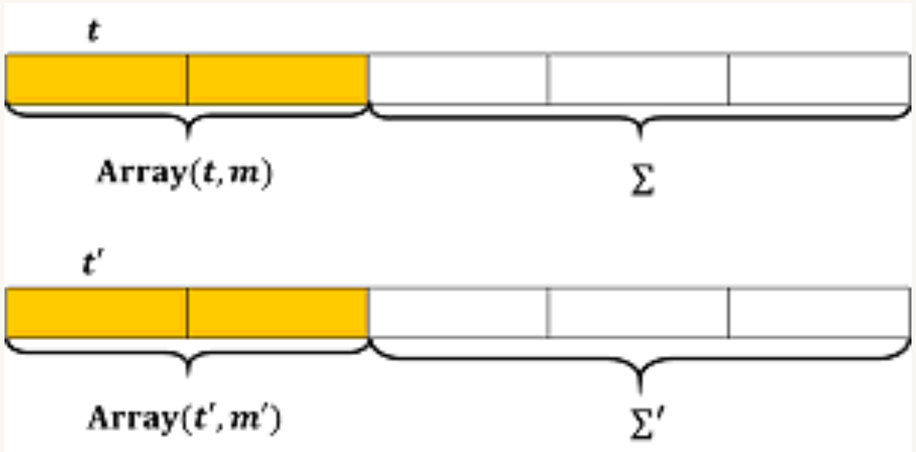
$\Pi \wedge m = m' \wedge \Sigma \models \Pi' \wedge t = t' \wedge \Sigma'$

and

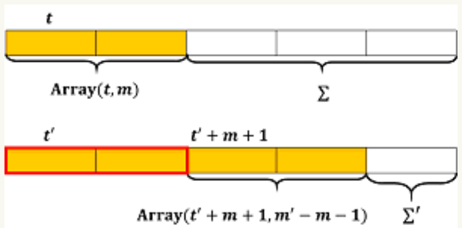
$\Pi \wedge m < m' \wedge \Sigma \models \Pi' \wedge t = t' \wedge \mathit{Array}(t + m + 1, m' - m - 1) \circledast \Sigma'$

and

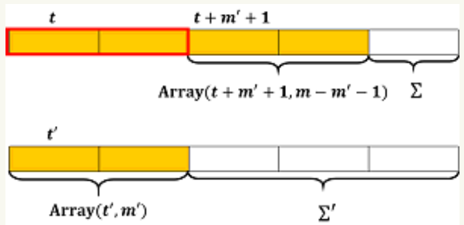
$\Pi \wedge m > m' \wedge \mathit{Array}(t + m' + 1, m - m' - 1) \circledast \Sigma \models \Pi' \wedge t = t' \wedge \Sigma'$



1st case $(m = m')$



2nd case $(m < m')$



3rd case $(m > m')$

Idea of translation

Observation4 (ArrayArray case)

$\Pi \wedge \mathit{Array}(t, m) \circledast \Sigma \models \Pi' \wedge \mathit{Array}(t', m') \circledast \Sigma'$

$\Longleftrightarrow$

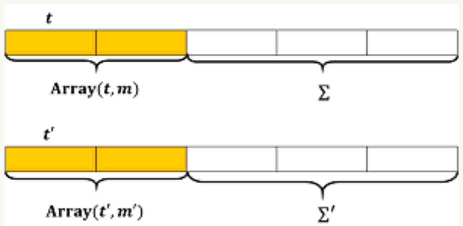
$\Pi \wedge m = m' \wedge \Sigma \models \Pi' \wedge t = t' \wedge \Sigma'$

and

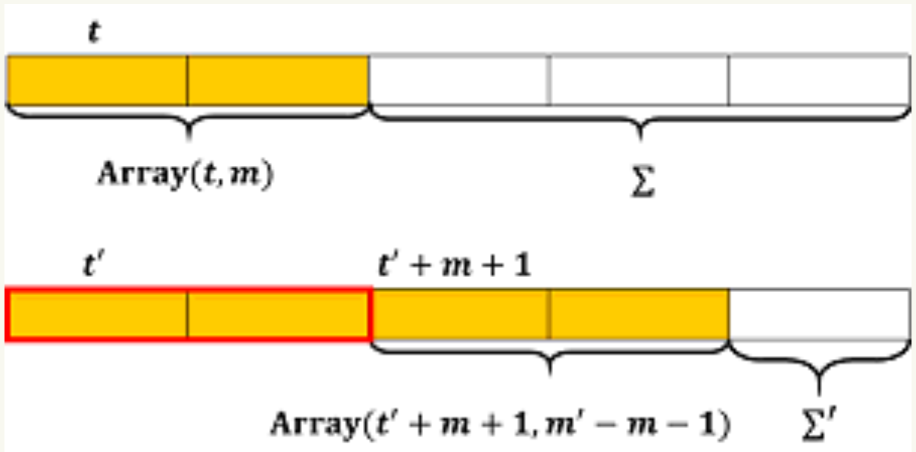
$\Pi \wedge m < m' \wedge \Sigma \models \Pi' \wedge t = t' \wedge \mathit{Array}(t + m + 1, m' - m - 1) \circledast \Sigma'$

and

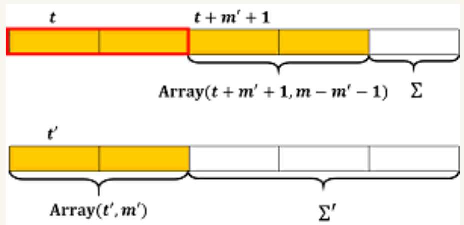
$\Pi \wedge m > m' \wedge \mathit{Array}(t + m' + 1, m - m' - 1) \circledast \Sigma \models \Pi' \wedge t = t' \wedge \Sigma'$



1st case $(m = m')$



2nd case $(m < m')$



3rd case $(m > m')$

Idea of translation

Observation4 (ArrayArray case)

$$\Pi \wedge \mathit{Array}(t, m) \circledast \Sigma \ \models \ \Pi' \wedge \mathit{Array}(t', m') \circledast \Sigma'$$

$$\Longleftrightarrow$$

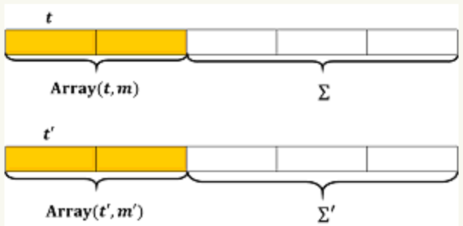
$$\Pi \wedge m = m' \wedge \Sigma \models \ \Pi' \wedge t = t' \wedge \Sigma'$$

and

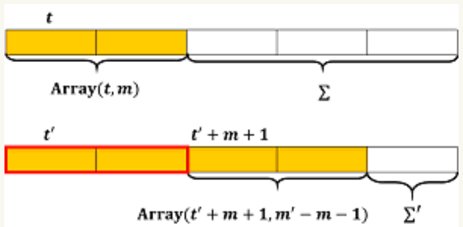
$$\Pi \wedge m < m' \wedge \Sigma \models \ \Pi' \wedge t = t' \wedge \mathit{Array}(t + m + 1, m' - m - 1) \circledast \Sigma'$$

and

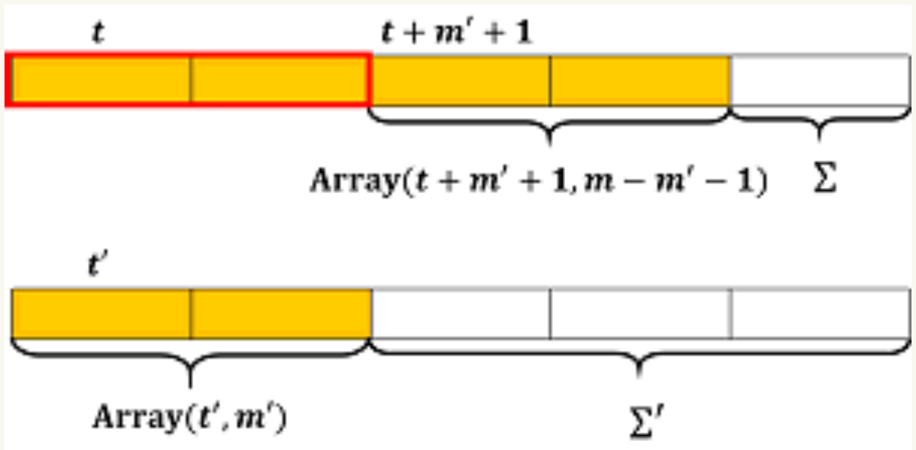
$$\Pi \wedge m > m' \wedge \mathit{Array}(t + m' + 1, m - m' - 1) \circledast \Sigma \models \ \Pi' \wedge t = t' \wedge \Sigma'$$



1st case ($m = m'$)



2nd case ($m < m'$)



3rd case ($m > m'$)

Translation

Our translation $\texttt{mkPb}(\Pi, \Sigma\,;\ \{(\Pi_i, \Sigma_i)\}_i)$ is defined by using the observations ($\texttt{mkPb}(\Pi, \Sigma\,;\ \{(\Pi_i, \Sigma_i)\}_i)$ is the result of translation of $\Pi \wedge \Sigma \vdash \bigvee_i (\Pi_i \wedge \Sigma_i)$)

$$\texttt{mkPb}(\Pi, \Sigma\,;\ S_1 \cup \{(\Pi', \mathit{emp} \circledast \Sigma')\} \cup S_2) \stackrel{\text{def}}{=} \texttt{mkPb}(\Pi, \Sigma\,;\ S_1 \cup \{(\Pi', \Sigma')\} \cup S_2)$$

$$\texttt{mkPb}(\Pi, \mathit{emp} \circledast \Sigma\,;\ S) \stackrel{\text{def}}{=} \texttt{mkPb}(\Pi, \Sigma\,;\ S)$$

$$\texttt{mkPb}(\Pi, \mathit{emp}\,;\ \{(\Pi_i, \mathit{emp})\}_i) \stackrel{\text{def}}{=} \Pi \Longrightarrow \bigvee_i \Pi_i$$

$$\texttt{mkPb}(\Pi, \mathit{emp}\,;\ S_1 \cup \{(\Pi', \Sigma')\} \cup S_2) \stackrel{\text{def}}{=} \texttt{mkPb}(\Pi, \mathit{emp}\,;\ S_1 \cup S_2)$$

(Σ' is not *emp*)

$$\texttt{mkPb}(\Pi, \mathit{emp}\,;\ \emptyset) \stackrel{\text{def}}{=} \neg\Pi$$

$$\texttt{mkPb}(\Pi, \Sigma\,;\ \emptyset) \stackrel{\text{def}}{=} \neg(\Pi \wedge \mathit{Sorted}(\Sigma))$$

(Σ is not *emp*)

($\mathit{Sorted}(\Sigma)$ means that the addresses in Σ is sorted)

Translation (cont.)

($\mapsto\mapsto$-case)

$$\mathtt{mkPb}(\Pi, t \mapsto (\vec{u}) \circledast \Sigma\,;\, \{(\Pi_i, t_i \mapsto (\vec{u_i}) \circledast \Sigma')\}_i)$$

$$\stackrel{\text{def}}{=} \quad \mathtt{mkPb}(\Pi \wedge t < \Sigma, \Sigma\,;\, \{(\Pi_i \wedge t = t' \wedge \vec{u} = \vec{u'}, \Sigma_i)\}_i)$$

($\mapsto$Array-case)

$$\mathtt{mkPb}(\Pi, t \mapsto (\vec{u}) \circledast \Sigma\,;\, S_1 \cup \{(\Pi', \mathit{Array}(t', m) \circledast \Sigma')\} \cup S_2)$$

$$\stackrel{\text{def}}{=} \quad \mathtt{mkPb}\begin{pmatrix} \Pi \wedge m = 0, t \mapsto (\vec{u}) \circledast \Sigma\,; \\ S_1 \cup \{(\Pi' \wedge t = t', t' \mapsto (\vec{u}) \circledast \Sigma')\} \cup S_2 \end{pmatrix}$$

$$\wedge$$

$$\mathtt{mkPb}\begin{pmatrix} \Pi \wedge m > 0, t \mapsto (\vec{u}) \circledast \Sigma\,; \\ S_1 \cup \{(\Pi' \wedge t = t', t' \mapsto (\vec{u}) \circledast \mathit{Array}(t' + 1, m - 1) \circledast \Sigma')\} \cup S_2 \end{pmatrix}$$

(Array$\mapsto$-case)

$$\mathtt{mkPb}(\Pi, \mathit{Array}(t, m) \circledast \Sigma\,;\, S) \qquad \text{(where } (\Pi', t' \mapsto (\vec{u'}) \circledast \Sigma') \in S\text{)}$$

$$\stackrel{\text{def}}{=} \quad \mathtt{mkPb}(\Pi \wedge m = 0, t \mapsto (\vec{z}) \circledast \Sigma\,;\, S)$$

$$\wedge$$

$$\mathtt{mkPb}(\Pi \wedge m > 0, t \mapsto (\vec{z'}) \circledast \mathit{Array}(t + 1, m - 1) \circledast \Sigma\,;\, S) \qquad (\vec{z}, \vec{z'} : \text{fresh})$$

Translation (cont.)

(ArrayArray-case)

$$\mathtt{mkPb}(\Pi, \mathit{Array}(t,n) \circledast \Sigma\,;\ \{(\Pi_i, \mathit{Array}(t_i,m_i) \circledast \Sigma')\}_{i\in I})$$

$$\stackrel{\text{def}}{=}$$

$$\bigwedge_{J\subseteq I} \mathtt{mkPb}\left(\begin{array}{l} \Pi \wedge n = \min(n,\{m_i\}_i) \wedge \bigwedge_{j\in J} n = m_j \wedge \bigwedge_{j\notin J} n < m_j \wedge t+n < \Sigma, \Sigma\,; \\ \{(\Pi_i \wedge t = t_i, \Sigma_i)\}_{i\in J} \\ \cup\ \{(\Pi_i \wedge t = t_i \wedge \mathit{Array}(t_i+n+1, m_i-n-1) \circledast \Sigma_i)\}_{i\notin J} \end{array}\right)$$

$$\wedge$$

$$\bigwedge_{\emptyset\neq J\subseteq I} \mathtt{mkPb}\left(\begin{array}{l} \Pi \wedge m' = \min(n,\{m_i\}_i) \wedge m' < n \wedge \bigwedge_{j\in J} m' = m_j \wedge \bigwedge_{j\notin J} m' < m_j \\ \qquad \wedge\, t+n < \Sigma, \mathit{Array}(t+m'+1, n-m'-1) \circledast \Sigma\,; \\ \{(\Pi_i \wedge t = t_i, \Sigma_i)\}_{i\in J} \\ \cup\ \{(\Pi_i \wedge t = t_i \wedge \mathit{Array}(t_i+m'+1, m_i-m'-1) \circledast \Sigma_i)\}_{i\notin J} \end{array}\right)$$

(where m' is $m_{\min(J)}$)

- The first clause: n is the least one and $J = \{i \in I \mid n = m_i\}$
- The second clause: $m'(\neq n)$ is the least one and $J = \{i \in I \mid m' = m_i\}$

Decidability of entailment problem

Proposition Suppose that Σ and Σ_i has the form $\sigma_1 \circledast \ldots \circledast \sigma_n$

$$\Pi \wedge \Sigma \models \bigvee_i \Pi_i \wedge \Sigma_i \quad \Leftrightarrow \quad \models_{PbA} \forall \vec{x}.\mathtt{mkPb}(\Pi, \Sigma; \{(\Pi_i, \Sigma_i)\}_i)$$

Decidability of entailment problem

Proposition Suppose that Σ and Σ_i has the form $\sigma_1 \circledast \ldots \circledast \sigma_n$

$$\Pi \wedge \Sigma \models \bigvee_i \Pi_i \wedge \Sigma_i \quad \Leftrightarrow \quad \models_{PbA} \forall \vec{x}.\mathrm{mkPb}(\Pi, \Sigma; \{(\Pi_i, \Sigma_i)\}_i)$$

We have the next theorem by applying the following fact:

$$\Pi \wedge \mathop{*}_{i \in I} \sigma_i \models \Pi' \wedge \mathop{*}_{j \in J} \sigma'_j$$

$$\Longleftrightarrow \quad \bigwedge_{p \in perm(I)} \left(\Pi \wedge \mathop{\circledast}_{i \in I} \sigma_{p(i)} \models \bigvee_{p' \in perm(J)} (\Pi' \wedge \mathop{\circledast}_{j \in J} \sigma'_{p'(j)}) \right)$$

Theorem Entailment problem of **SL**ARRAY is decidable

Implementation

- Prototype implementation of our decision procedure
- About 4000 lines of code written in OCaml
- SMT-solver **Z3 (Microsoft Research)** is internally called for checking presburger formula
- Quick response for "sorted" entailments
- Very slow for "unsorted" entailments (because of permutation)

Possible Improvement (next task)

- Eliminating hopeless conclusions
  $1 \mapsto (x) * 2 \mapsto (y) \ \vdash \ 3 \mapsto (z) \ \vee \ 1 \mapsto (x) * 2 \mapsto (y) \ \vee \ 2 \mapsto (y) * 1 \mapsto (x)$
  can be reduced to
  $1 \mapsto (x) * 2 \mapsto (y) \ \vdash$ ~~$3 \mapsto (z) \vee$~~ $1 \mapsto (x) * 2 \mapsto (y)$ ~~$\vee 2 \mapsto (y) * 1 \mapsto (x)$~~

Related work

Brotherston,Gorogiannis and Kanovich (2016,preprint)

- Symbolic heap SL with arrays
- Decidability of bi-abduction and entailment problems
- Only Array-predicate (no points-to predicate)
- Single-conclusion entailment

facebook INFER (managed by Peter O'Hearn)

- Best-known static analyzer of pointer-programs based on SL
- Arrays are not supported

Conclusion and Future work

Conclusion

- Symbolic-heap separation logic with array
- Decision procedure of entailment problem
- Idea
  - Sorted separating conjunction
  - Use decidability of Presburger arithmetic
- Implementation

Future work

- Optimization
- Adding existential quantifier
- Biabduction problem