On Symbolic Heaps modulo Permission Theories

Stéphane Demri, Etienne Lozes (LSV)
and Denis Lugiez (LIF)

LIF UMR 7279, Aix-Marseille Université-CNRS

Framework : Program Verification

- Issue 1 : Memory use via dynamic allocation (the heap)
  Pointers, Dynamic structures Lists, Trees, ...
- Issue 2 : shared memory (multithreading)
  Processes can read memory.
  Only owner can write on memory.

Separation Logic : syntax

$$\Pi ::= \quad \top \mid x = y \mid x \neq y \mid \Pi \wedge \Pi$$
(pure formula)

$$\Sigma ::= \quad \text{emp} \mid \top \mid x \mapsto y \mid \Sigma * \Sigma$$
(spatial formula)

where LVAR $= \{x, y, \ldots\}$ countably set of **location variables**.

Separation Logic : semantics

- Memory : set of locations Loc
- Interpretation (s, h)
  - Store : $s : \mathsf{LVAR} \to \mathsf{Loc}$
  - Heap : $h : \mathsf{Loc} \rightharpoonup_{\mathsf{fin}} \mathsf{Loc}$

Example

- $\mathsf{Loc} = \{l_1, l_2\}$
- $s(x_1) = l_1, s(x_2) = l_2$
- $h(l_1) = l_2$ (i.e. $l_2 \mapsto l_2$)

Composition of **disjoint** heaps heaps $h = h_1 \bullet h_2$

- Models $(s, h) \models (\Pi, \Sigma)$ defined by :
  - $h \models x \mapsto y$ iff $s(x) = l_1, s(y) = l_2$ and $h(l_1) = l_2$
  - $h \models \Sigma_1 * \Sigma_2$ iff $h = h_1 \bullet h_2$ and $h_i \models \Sigma_i$
  - ...
- Satisfiability (Π, Σ) has a model ?
- Entailment $(\Pi, \Sigma) \models (\Pi', \Sigma')$

Extension 1 : Permissions

Idea :

- ➡ full permission 1 = write authorized
- ➡ split permission : give read access to sons
- ➡ add permissions $\oplus$: gather all read permissions to recover write access

Permission model $\mathfrak{P} = (P_{\mathfrak{P}}, 1_{\mathfrak{P}}, \oplus_{\mathfrak{P}})$ with $\oplus_{\mathfrak{P}}$ partial function + some axioms (AC,...).

Modified syntax :

- Predicate $x \overset{p}{\mapsto} y$
- Permission Formula A

Modified semantics :

- A permission model $\mathfrak{P}$ and ι an interpretation of permission variables,
- $h : \mathsf{Loc} \rightharpoonup_{\mathsf{fin}} P_{\mathfrak{P}} \times \mathsf{Loc}$,
- Composition $h = h_1 \bullet h_2$ of non-disjoint heaps

$$h(l) = (\pi_1 \oplus_{\mathfrak{P}} \pi_2, l') \text{ if } \begin{cases} \mathit{defined}(\pi_1 \oplus_{\mathfrak{P}} \pi_2) \\ h_i(l) = (\pi_i, l') \\ l \in \mathit{dom}(h_1) \cap \mathit{dom}(h_2) \end{cases}$$

- Satisfaction relation $(s, h, \iota) \models (\Pi, \Sigma)$

Some permission models :

1. Trivial model (only 1).
2. Fractional model $\mathfrak{P}_{\mathsf{Boy}} = (]0, 1], 1, +)$ with $+$ defined if the sum is less or equal to 1,
3. tree share model (binary trees),
4. ...

Extension 2 : Lists

- Predicate : $\mathsf{lseg}(x, y)$
- Interpretation :
  $s(x) = s(y)$ or
  $s(x) = l_1, s(y) = l_n$ and $l_1 \mapsto l_2 \mapsto \ldots \mapsto l_n$

Separation Logic with Lists and Permissions

Syntax

$$\Pi ::= \top \mid x = y \mid x \neq y \mid A \mid \Pi \wedge \Pi$$
$$\Sigma ::= \mathsf{emp} \mid \top \mid x \overset{p}{\mapsto} y \mid x \mapsto y \mid \mathsf{lseg}_p(x, y) \mid \Sigma * \Sigma$$

where

$$A ::= \top \mid p = p \mid p \leq p \mid \mathsf{defined}(p) \mid A \wedge A$$
$$p ::= 1 \mid \alpha \mid p \oplus p$$

Separation Logic with lists and permissions

Semantics

- Permission model $\mathfrak{P}$
- Interpretation (s, h, ι) and satisfaction relation (merge previous definitions).

Some examples

Formulas

$$(\Pi_1, \Sigma_1) \stackrel{\text{def}}{=} (\bigwedge_{1 \leq i < j \leq 3} x_i \neq x_j, \mathsf{lseg}_\alpha(x_1, x_3))$$

$$(\Pi_2, \Sigma_2) \stackrel{\text{def}}{=} (\bigwedge_{1 \leq i < j \leq 3} x_i \neq x_j, \mathsf{lseg}_\alpha(x_1, x_2) * \mathsf{lseg}_\alpha(x_2, x_3))$$

$$(\Pi_3, \Sigma_3) \stackrel{\text{def}}{=} (\bigwedge_{1 \leq i < j \leq 3} x_i \neq x_j, \ \mathsf{lseg}_{\alpha \oplus \alpha}(x_1, x_3) * \mathsf{lseg}_\alpha(x_3, x_2) * \mathsf{lseg}_\alpha(x_2, x_1))$$

Interpretation

$\mathfrak{P} = \mathfrak{P}_{\mathsf{Boy}}$ and (s, h, ι) :

- $\iota(\alpha) = 0.25$
- $s(x_1) = 1, s(x_2) = 2, s(x_3) = 3,$
- $h = \{1 \mapsto (0.5, 3), 3 \mapsto (0.25, 2), 2 \mapsto (0.25, 1)\}$

Some examples

Formulas

$$(\Pi_1, \Sigma_1) \stackrel{\text{def}}{=} (\bigwedge_{1 \leq i < j \leq 3} x_i \neq x_j, \mathsf{lseg}_\alpha(x_1, x_3))$$

$$(\Pi_2, \Sigma_2) \stackrel{\text{def}}{=} (\bigwedge_{1 \leq i < j \leq 3} x_i \neq x_j, \mathsf{lseg}_\alpha(x_1, x_2) * \mathsf{lseg}_\alpha(x_2, x_3))$$

$$(\Pi_3, \Sigma_3) \stackrel{\text{def}}{=} (\bigwedge_{1 \leq i < j \leq 3} x_i \neq x_j, \mathsf{lseg}_{\alpha \oplus \alpha}(x_1, x_3) * \mathsf{lseg}_\alpha(x_3, x_2) * \mathsf{lseg}_\alpha(x_2, x_1))$$

Evaluation

$$s, h, \iota \not\models_{\mathfrak{P}_{\text{Boy}}} (\Pi_1, \Sigma_1)$$

$$s, h, \iota \models_{\mathfrak{P}_{\text{Boy}}} (\Pi_2, \Sigma_2)$$

$$s, h, \iota \models_{\mathfrak{P}_{\text{Boy}}}, (\Pi_3, \Sigma_3)$$

Our Results

A decision procedure for Satisfiability and Entailment for

- ➡ Separation Logic with Lists and Permission
- ➡ Parametrized by the permission model.

It works also for the intuitionistic case (not in this talk)

PART I : Decision procedure with permission but without lists

Normalization rules

$$(\Pi, \Sigma) \quad \Rightarrow \quad (\Pi, \Sigma[y/x]) \quad \text{if } \Pi \models x = y \text{ and } \{x, y\} \subseteq \mathsf{LVAR}(\Sigma)$$

$$(\Pi, \Sigma) \quad \Rightarrow \quad (\Pi, \Sigma[y/x]) \quad \text{if } \Pi \models x = y \text{ and } \{x, y\} \subseteq \mathsf{LVAR}(\Sigma)$$

$$(\Pi, \Sigma * x \overset{p}{\mapsto} y * x \overset{p'}{\mapsto} z) \quad \Rightarrow \quad (\Pi \wedge y = z, \Sigma * x \overset{p \oplus p'}{\mapsto} y)$$

$$(\Pi, \Sigma) \quad \Rightarrow \quad \bot \quad \text{if } \Pi_{pv} \models x \neq y,\ \Pi_{pv} \models x = y$$

$$(\Pi, \Sigma * \mathsf{emp}) \quad \Rightarrow \quad (\Pi, \Sigma) \quad \text{if non-empty } \Sigma$$

$$(\Pi, \Sigma * \top * \top) \quad \Rightarrow \quad (\Pi, \Sigma * \top)$$

$$(\Pi, \Sigma * \mathsf{lseg}_p(x, y) * \mathsf{lseg}_{p'}(x, y)) \quad \Rightarrow \quad (\Pi, \Sigma * \mathsf{lseg}_{p \oplus p'}(x, y))$$

Theorem

If satisfiability of permissions is in $\mathsf{C} \supseteq PTIME$ *then satisfiability for separation logic without lists is in* C.

1. Normalization terminates
2. (Π, Σ) satisfiable iff $\Pi_{pe} \wedge \mathit{defined}(\Sigma)$ satisfiable (Π_{pe} permission part of Π).

Theorem

Satisfiability of entailment $(\Pi, \Sigma) \models (\Pi', \Sigma')$ *is polynomial with an oracle for entailment of permissions.*

New rewrite rules for triples $(\Pi, \Sigma_l, \Sigma_r)$ representing formula $(\Pi, \Sigma_l) \models \Sigma_r$

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

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

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

PART II : a Decision procedure with permission and lists

Lower Bound

Theorem

If $\mathsf{width}(\mathfrak{P}) = \omega$ *then* $\mathrm{SATSH}(\mathfrak{P})$ *is* NP*-hard.*

1. 3-colorability instance $G = (V = \{x_1, \ldots, x_n\}, E)$.
2. Formula

$$(\bigwedge_{\{x_i,x_j\}\in E}(x_i \neq x_j)) \wedge y_1 \neq y_2 \wedge y_1 \neq y_3 \wedge y_2 \neq y_3$$
$$y_1 \overset{\alpha_0}{\mapsto} y_2 * y_2 \overset{\alpha'_0}{\mapsto} y_3 * y_3 \overset{1}{\mapsto} y_1 *_{x_i \in V} (\mathsf{lseg}_{\alpha_i}(y_1, x_i) * \mathsf{lseg}_{\alpha'_i}(x_i, y_3))$$

G 3-colorable iff (Π_G, Σ_G) satisfiable.

Corollary

Entailment is coNP*-hard*

Remark : Entailment for separation logic with list without permission is in PTIME(C. Haas et al.).

General Scheme for decidability

$$(\Pi, \Sigma) \models (\Pi', \Sigma')?$$

SL graph : graphical representation of a formula

SL-graph G $\leftrightarrow$ formula $(pure(G), spatial(G))$

Formula (Π, Σ) $\leftrightarrow$ SL graph

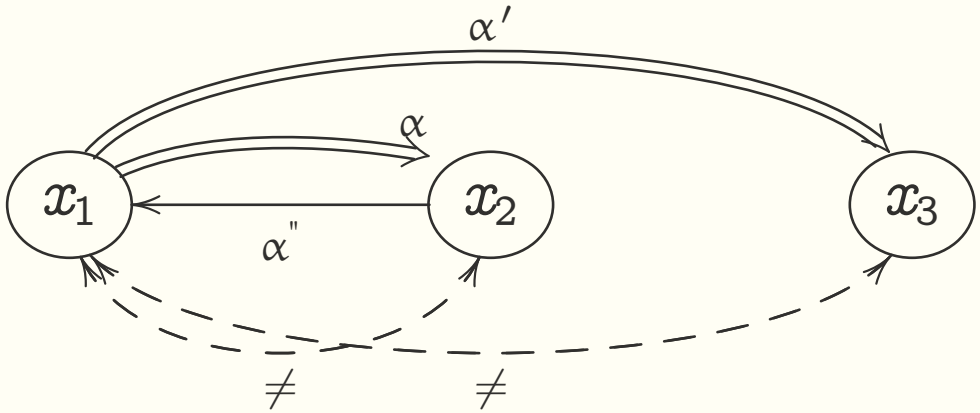
$(\Pi, \Sigma) \not\models (\Pi', \Sigma')$ iff

- $\exists G_D$ deterministic SL-graph
- G_D represents a satisfiable formula
- G_D is *small*
- $h : G \to G_D$ precise homomorphism
- $h' : G' \to G_D$ homomorphism

not strongly precised

SL-graphs

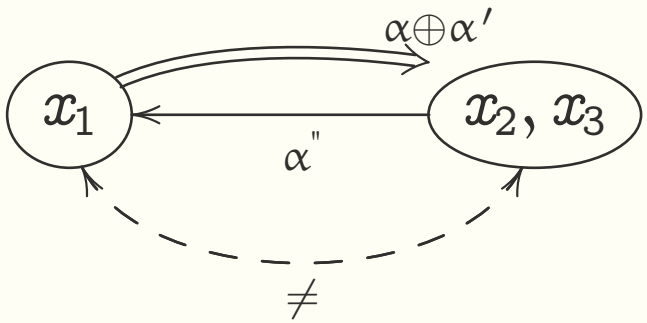
$$(A, V, \xrightarrow{p}, \overset{p}{\Rightarrow}, \overset{\neq}{\leftrightarrow}, L)$$



$(pure(G), spatial(G)))$ the formula associated to G.

SL-graphs

$(A, V, \xrightarrow{p}, \overset{p}{\Rightarrow}, \overset{\neq}{\leftrightarrow}, L)$



$(pure(G), spatial(G)))$ the formula associated to G.

SL-graphs

$(A, V, \xrightarrow{p}, \overset{p}{\Rightarrow}, \overset{\neq}{\leftrightarrow}, L)$

- V vertices, L labels vertices by disjoints sets of variables,
- $v \xrightarrow{p} v'$ corresponds to $x \overset{p}{\mapsto} y$ for x a label of v, y a label of v',
- $v \overset{p}{\Rightarrow} v'$ corresponds to $\mathsf{lseg}_p(x, y)$ for x a label of v, y a label of v',
- $v \overset{\neq}{\leftrightarrow} v'$ corresponds to $x \neq y$ for x a label of v, y a label of v',

$(\mathit{pure}(G), \mathit{spatial}(G)))$ the formula associated to G.

Formula to SL-graph

$slg(\Pi, \Sigma)$ SL-graph associated to (Π, Σ) :

1. Put (Π, Σ) in normal form (Π', Σ').
2. Derive graph $(A, V, \overset{p}{\rightarrow}, \overset{p}{\Rightarrow}, \overset{\neq}{\leftrightarrow}, L)$
   1. A permission part of Π',
   2. $\Pi' \models x = y$ then $L(x) = L(y)$
   3. $L(x) \overset{\neq}{\leftrightarrow} L(y)$ iff $\Pi \models L(x) \neq L(y)$
   4. $x \overset{p}{\mapsto} y \in \Pi$ then $L(x) \overset{p}{\rightarrow} L(y)$
   5. $\mathsf{lseg}_p(x, y) \in \Pi$ then $L(x) \overset{p}{\Rightarrow} L(y)$

Precise Graph homomorphism

G is deterministic iff there is at most one edge $\xrightarrow{p}$ or $\overset{p}{\Rightarrow}$ issued from a node.

Notation : $\overset{p}{\rightsquigarrow} = \xrightarrow{p} \cup \overset{p}{\Rightarrow}$

Precise Graph homomorphism

$G_1 = (A_1, V_1, \rightarrow_1, \Rightarrow_1, \overset{\neq}{\leftrightarrow}_1, L_1)$,
$G_2 = (A_2, V_2, \rightarrow_2, \Rightarrow_2, \overset{\neq}{\leftrightarrow}_2, L_2)$ deterministic
$h : G_1 \rightarrow G_2$ precise homomorphism iff :

1. $h : V_1 \rightarrow V_2$ surjective
2. $A_2' = A_2 \wedge \bigwedge_{p \in G_2} \mathit{defined}(p) \models A_1$
3. $V_1 \ni v \mathit{ar}(v) \subseteq \mathit{var}(h(v))$
4. $G_1 \ni v \xrightarrow{p} v'$ implies $\exists p'\ h(v) \xrightarrow{p'} h(v') \in G_2$,
5. $G_1 \ni v \overset{p}{\Rightarrow} v'$ implies $h(v) = h(v')$ and $A_2' \models \mathit{defined}(p)$ or there is a path $v_0 \overset{p_0}{\rightsquigarrow}_2 v_1 \overset{p_1}{\rightsquigarrow}_2 v_2 \cdots \overset{p_{n-1}}{\rightsquigarrow}_2 v_n = h(v')$
6. Each edge of G_2 contributes to at least one edge of G_1.

Strongly precise homomorphism

Modifications :

1' $A_2 = A_1 \wedge \bigwedge_{p \in G_2} \mathit{defined}(p)$

5' no condition $A_2' \models \mathit{defined}(p)$ when $h(v) = h(v')$

6' Each edge $\overset{p'}{\rightsquigarrow}$ of G_2 contributes to at least one edge $p' = \bigoplus\{\{p \mid v \overset{p'}{\rightsquigarrow} v' \text{ contributes to } u \overset{p}{\rightsquigarrow} u'\}\}$

Syntactic conditions replace semantics conditions.

Variable order

➡ Problem : determinism induces an "ordering" on variables.

Possible situation : $(s, h, \iota) \models (\Pi, \Sigma)$ and $(s, h, \iota) \not\models (\mathit{pure}(G_D), \mathit{spatial}(G_D))$ for G_D a deterministic SL-graph derived from the SL-graph associated to (Π, Σ).

➡ Solution : a suitable ordering exists, guess it !

Key Theorem

Theorem

$(\Pi, \Sigma) \not\models (\Pi', \Sigma')$ *iff exists* G_D *deterministic graph s.t.*

- ➡ *Any* p *of* $\overset{p}{\rightsquigarrow}$ *in* G_D *is a sum of at most* $|\Sigma|$ *permission terms of* G,
- ➡ $(pure(G_D), spatial(G_D))$ *satisfiable,*
- ➡ $h_{G,G_D} : G \to G_D$ *strongly precise,*
- ➡ $h_{G',G_D} : G \to G_D$ *not strongly precise.*

where G *SL-graph of* (Π, Σ), G *SL-graph of* (Π', Σ'),

Guess the right ordering of variables when guessing G_D.

Main Theorem

Theorem

- *Satisfiability is* NP*-complete and Entailment is* coNP*-complete for* $\mathfrak{P}_{\mathsf{Boy}}$
- *Satisfiability is* NP*-complete and Entailment is* coNP*-complete for any model* $\mathfrak{P}$ *of width* ω *with an entailment problem in* coNP*.*

Precise bound.

PART III : Complexity Results for Permission models

Fractional Model $\mathfrak{P}_{\mathsf{Boy}}$

Theorem

Satisfiability and entailment are polynomial for $\mathfrak{P}_{\mathsf{Boy}}$.

1. Satisfiability. Formula = conjunction of $p \# p'$ with $\# \in \{\leq, <\}$, p, p' sums of variables and constant 1. Decidable in polynomial time.
2. Entailment.
   $A \models B$ iff $A \models p \# p'$ for each conjunct of B.
   $A \not\models p \# p'$ iff $A \wedge \neg(p \# p')$ satisfiable
   decidable in polynomial time
   Polynomially number of polynomial tests.

Boolean Algebra Models

Boolean permission model = isomorphic to a Boolean algebra without minimal element.
$width(\mathfrak{P}) = \omega$ i.e. 1 is decomposable

Theorem

$SAT(\mathfrak{P})$ *is* NP*-hard and* $ENT(\mathfrak{P})$ *is* coNP*-hard*

Reduce 1-in-3 clause satisfiability.

Theorem

$SAT(\mathfrak{P})$ is NP*-hard and $ENT(\mathfrak{P})$ is* coNP*.*

Idea for Satisfiability.

$$
\begin{array}{lcl}
A \text{ formula on } \alpha_1, \ldots, \alpha_n & \leftrightarrow & \Psi_A \text{ formula on } X_1^1, \ldots, X_n^n \\
\alpha_i & \leftrightarrow & t_i^j(\alpha_i) = X_i^j \text{ for } j = 1, \ldots, n \\
\bigoplus_i \alpha_i & \leftrightarrow & \Sigma_i t_j(\alpha_i) \text{ for } j = 1, \ldots, n \\
\ldots & \leftrightarrow & \ldots \\
A & \leftrightarrow & \Psi_A^j \text{ for } j = 1, \ldots, n
\end{array}
$$

A satisfiable iff $\bigwedge_{j=1,\ldots,n} \Psi_A^j$ satisfiable.

CONCLUSION

- Optimal complexity results for separation logic with lists and permissions.
- Parametrized by the permission model
- Complexity results on permission models.

TODO list :

- A proof system ?
- Generalize the permission theory (existential theory ?)
- Recursive structures (trees,...)