Extending propositional separation logic for robustness properties

FSTTCS'18

Alessio Mansutti

LSV, CNRS, ENS Paris-Saclay

Ahmedabad - December 2018

Separation logic and program verification

- Separation Logic (Reynolds'02) is used in Hoare proof systems for program verification of languages with pointers.

- Hoare calculus: axioms and rules reason about triples:

$$\{\varphi\}\ \mathtt{P}\ \{\varphi'\}$$

  Any (memory) model that satisfies φ will satisfy φ' after being modified by the program $\mathtt{P}$.

Separation logic and program verification

- Separation Logic (Reynolds'02) is used in Hoare proof systems for program verification of languages with pointers.

- Hoare calculus: axioms and rules reason about triples:

$$\{\varphi\}\ \mathtt{P}\ \{\varphi'\}$$

  Any (memory) model that satisfies φ will satisfy φ' after being modified by the program $\mathtt{P}$.

- Tools: Infer (Facebook), SLAyer (Microsoft)...

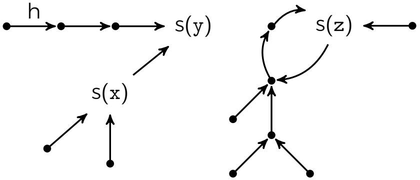
Also, see "Why Separation Logic Works" (Pym et al. '18)

Memory states

Separation Logic is interpreted over **memory states (s, h)** where:

- store, $\mathsf{s} : \mathtt{VAR} \to \mathtt{LOC}$
- heap, $\mathsf{h} : \mathtt{LOC} \to_{\mathsf{fin}} \mathtt{LOC}$

where $\mathtt{VAR} = \{\mathtt{x}, \mathtt{y}, \mathtt{z}, \dots\}$ set of (program) variables,
$\mathtt{LOC}$ set of locations. $\mathtt{VAR}$ and $\mathtt{LOC}$ are countably infinite sets.



here, $\mathsf{h}(\mathsf{s}(\mathtt{x})) = \mathsf{s}(\mathtt{y})$

- Disjoint heaps: $\mathrm{dom}(\mathsf{h}_1) \cap \mathrm{dom}(\mathsf{h}_2) = \emptyset$
- Sum of disjoint heaps $(\mathsf{h}_1 + \mathsf{h}_2)$
  is defined as the sum of partial functions.

Propositional Separation Logic $\mathtt{SL}(*, \mathrel{-\!\!*})$

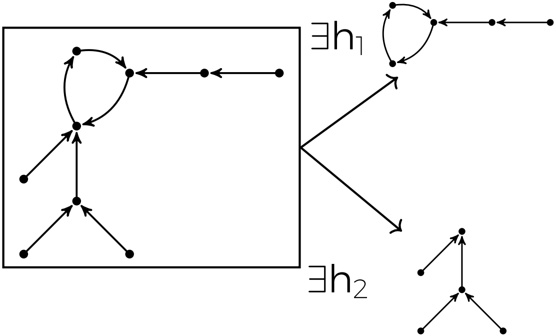
$$\varphi := \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathtt{emp} \mid \mathtt{x} = \mathtt{y} \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \varphi_1 * \varphi_2 \mid \varphi_1 \mathrel{-\!\!*} \varphi_2$$

Semantics

- standard for $\wedge$ and $\neg$;
- $(s, h) \models \mathtt{emp} \iff \mathrm{dom}(h) = \emptyset$
- $(s, h) \models \mathtt{x} = \mathtt{y} \iff s(\mathtt{x}) = s(\mathtt{y})$
- $(s, h) \models \mathtt{x} \hookrightarrow \mathtt{y} \iff h(s(\mathtt{x})) = s(\mathtt{y})$

Separating conjunction ($*$)

$(s, h) \models \varphi_1 * \varphi_2$ if and only if



$(s, h_1) \models \varphi_1$

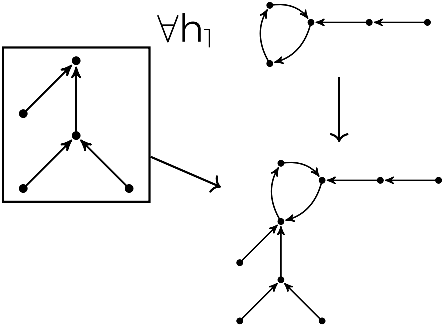
and

$(s, h_2) \models \varphi_2$

There is a way to split the heap into two so that, together with the store, one part satisfies φ_1 and the other satisfies φ_2.

Separating implication ($-\!\!*$)

$(s, h) \models \varphi_1 -\!\!* \varphi_2$ if and only if



$$\mathrm{dom}(h) \cap \mathrm{dom}(h_1) = \emptyset$$
$$(s, h_1) \models \varphi_1$$

$$\Downarrow$$

$$(s, h + h_1) \models \varphi_2$$

Whenever a (disjoint) heap that, together with the store, satisfies φ_1 is added, the resulting memory state satisfies φ_2.

Decision Problems

- Hoare proof-system requires to solve classical problems: satisfiability/validity/entailment

$$\frac{\varphi \Rightarrow \psi \quad \{\psi\}\ \mathtt{P}\ \{\psi'\} \quad \psi' \Rightarrow \varphi'}{\{\varphi\}\ \mathtt{P}\ \{\varphi'\}} \text{ consequence rule}$$

- satisfiability is PSPACE-complete for $\mathtt{SL}(*, -\!\!*)$

Note: entailment and validity reduce to satisfiability for $\mathtt{SL}(*, -\!\!*)$.

Robustness properties

- **Acyclicity** holds for φ iff every model of φ is acyclic
- **Garbage freedom** holds for φ iff in every model of φ, each $\ell \in \mathrm{dom}(\mathsf{h})$ is reachable from a program variable of φ

C. Jansen et al., ESOP'17

Checking for robustness properties is EXPTIME-complete for Symbolic Heaps with Inductive Predicates.

- Symbolic Heaps $\implies$ no $-\!\!*$, no $\wedge$ and $\neg$ inside $*$
- Inductive Predicates $\sim$ Horn clauses where $*$ replaces $\wedge$

$$P(\vec{\mathbf{x}}) \Leftarrow \exists \vec{\mathbf{z}}\; Q_1 \overset{*}{\cancel{\wedge}} \ldots \overset{*}{\cancel{\wedge}} Q_n \qquad \mathrm{fv}(Q_i) \subseteq \vec{\mathbf{x}}, \vec{\mathbf{z}}$$

Our Goal

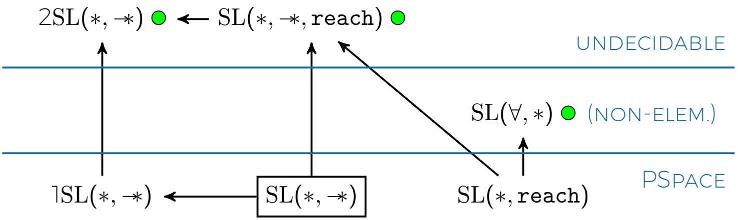
Provide similar results for **propositional** separation logic.

Desiderata

We aim to an extension of propositional separation logic where

- satisfiability is decidable in PSPACE (as $\mathrm{SL}(*, \mathrel{-\!\!*})$)
- robustness properties reduce to one of these problems

Known extensions



$\mathrm{SL}(*, \twoheadrightarrow)$ + reachability and 1 quantified variable

- $(\mathsf{s}, \mathsf{h}) \models \mathtt{reach}^{+}(\mathtt{x}, \mathtt{y}) \iff \mathsf{h}^{\mathsf{L}}(\mathsf{s}(\mathtt{x})) = \mathsf{s}(\mathtt{y})$ for some $\mathsf{L} \geq 1$
- $(\mathsf{s}, \mathsf{h}) \models \exists \mathtt{u}\, \varphi \iff$ there is $\ell \in \mathtt{LOC}$ s.t. $(\mathsf{s}[\mathtt{u} \leftarrow \ell], \mathsf{h}) \models \varphi$

It is only possible to quantify over the variable name $\mathtt{u}$.

Robustness properties reduce to entailment

- Acyclicity: $\varphi \models \neg \exists \mathtt{u}\, \mathtt{reach}^{+}(\mathtt{u}, \mathtt{u})$
- Garbage freedom: $\varphi \models \forall \mathtt{u}\, (\mathtt{alloc}(\mathtt{u}) \Rightarrow \bigvee_{\mathtt{x} \in \mathsf{fv}(\varphi)} \mathtt{reach}(\mathtt{x}, \mathtt{u}))$

where $\mathtt{u} \notin \mathsf{fv}(\varphi)$ and

- $\mathtt{alloc}(\mathtt{x}) \stackrel{\text{def}}{=} (\mathtt{x} \hookrightarrow \mathtt{x}) \twoheadrightarrow \perp$
- $\mathtt{reach}(\mathtt{x}, \mathtt{y}) \stackrel{\text{def}}{=} \mathtt{x} = \mathtt{y} \vee \mathtt{reach}^{+}(\mathtt{x}, \mathtt{y})$

Restrictions

The logic $1\mathrm{SL}(*, \mathrel{-\!\!*}, \mathtt{reach}^+)$ is undecidable. We syntactically restrict the logic so that for each occurrence of $\mathtt{reach}^+(\mathtt{x}, \mathtt{y})$:

R1 it is not on the right side of its first $\mathrel{-\!\!*}$ ancestor (seeing the formula as a tree)

- $\varphi \mathrel{-\!\!*} (\psi * \mathtt{reach}^+(\mathtt{u}, \mathtt{u}))$ violates **R1**

R2 if $\mathtt{x} = \mathtt{u}$ then $\mathtt{y} = \mathtt{u}$ (syntactically)

- $\mathtt{reach}^+(\mathtt{u}, \mathtt{x})$ violates **R2**

Note: robustness properties formulae are still expressible.

Results

1. $1\mathrm{SL}_{\mathtt{R1}}(*, -\!\!*, \mathbf{reach}^+)$: satisfiability is NON-ELEMENTARY (more precisely, TOWER-hard)
2. $1\mathrm{SL}_{\mathtt{R1}}^{\mathtt{R2}}(*, -\!\!*, \mathbf{reach}^+)$: satisfiability is PSPACE-complete

Proof Techniques

(1) reduce *Propositional interval temporal logic under locality principle* (PITL) to a logic captured by $1\mathrm{SL}_{\mathtt{R1}}(*, -\!\!*, \mathbf{reach}^+)$

(2) extend the *test formulae technique* used for $\mathrm{SL}(*, -\!\!*)$

PITL (Moszkowski'83)

$$\varphi := \mathtt{pt} \;\mid\; \mathtt{a} \;\mid\; \varphi_1 \mathbin{\boldsymbol{|}} \varphi_2 \;\mid\; \neg\varphi \;\mid\; \varphi_1 \wedge \varphi_2$$

- interpreted on finite non-empty words
  over a finite alphabet Σ
- $\mathfrak{w} \models \mathtt{pt} \iff |\mathfrak{w}| = 1$
- $\mathfrak{w} \models \mathtt{a} \iff$ first letter of $\mathfrak{w}$ is $\mathtt{a} \in \Sigma$ (locality principle)
- $\mathfrak{w} \models \varphi_1 \mathbin{\boldsymbol{|}} \varphi_2 \iff \mathfrak{w}[1:\mathtt{j}] \models \varphi_1$ and $\mathfrak{w}[\mathtt{j} : |\mathfrak{w}|] \models \varphi_2$
  for some $\mathtt{j} \in [1, |\mathfrak{w}|]$

$$\boxed{\mathfrak{w}_1 \dots \mathfrak{w}_{\mathtt{j}-1} \quad \mathfrak{w}_{\mathtt{j}} \quad \mathfrak{w}_{\mathtt{j}+1} \dots \mathfrak{w}_{|\mathfrak{w}|}}$$

$$\varphi_1 \qquad\qquad \varphi_2$$

- Satisfiability is decidable, but NON-ELEMENTARY

Auxiliary Logic on Trees (ALT)

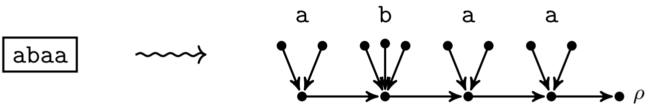
$$\varphi := \varphi_1 \land \varphi_2 \mid \neg\varphi \mid \varphi_1 * \varphi_2 \mid \exists \mathtt{u}\ \varphi \mid \mathtt{T}(\mathtt{u}) \mid \mathtt{G}(\mathtt{u})$$

- interpreted on acyclic memory states (set of rooted trees)
- one special tree, rooted in $\rho \in \mathtt{LOC}$
- $\exists \mathtt{u}\ \varphi$ and $\varphi_1 * \varphi_2$ as before
- $(\mathsf{s}, \mathsf{h}) \models_\rho \mathtt{T}(\mathtt{u})$ iff $\mathsf{s}(\mathtt{u}) \in \mathrm{dom}(\mathsf{h})$ and it does reach ρ
- $(\mathsf{s}, \mathsf{h}) \models_\rho \mathtt{G}(\mathtt{u})$ iff $\mathsf{s}(\mathtt{u}) \in \mathrm{dom}(\mathsf{h})$ and it does not reach ρ

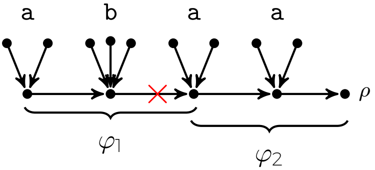
Note: ALT is captured by $1\mathrm{SL}_{\mathtt{R1}}(*, \mathrel{-\!\!*}, \mathtt{reach}^+)$.

Reducing PITL to ALT

- Easy to encode words as acyclic memory states



- Set of models encoding words can be characterised in ALT
- However, difficult to translate $\varphi_1 \mid \varphi_2$:
  cannot express properties about the trees not rooted in ρ, apart from their size



After the cut, left side does not reach ρ anymore.

PITL to ALT: alternative semantics for PITL

- $\boxed{\mathtt{a}}$ marked representation of $\mathtt{a} \in \Sigma$

$$\boxed{\mathfrak{w}_1 \ldots \mathfrak{w}_{j-1}\ \mathfrak{w}_j\ \mathfrak{w}_{j+1} \ldots \boxed{\mathfrak{w}_{|\mathfrak{w}|}}}$$

- $\varphi \mid \psi$ on standard semantics:

$$\underbrace{\boxed{\mathfrak{w}_1 \ldots \mathfrak{w}_{j-1}\ \mathfrak{w}_j}}_{\varphi_1} \qquad \underbrace{\boxed{\mathfrak{w}_j\ \mathfrak{w}_{j+1} \ldots \mathfrak{w}_{|\mathfrak{w}|}}}_{\varphi_2}$$

- $\varphi \mid \psi$ on marked semantics (can be simulated in ALT)

$$\boxed{\underbrace{\mathfrak{w}_1 \ldots \mathfrak{w}_{j-1}\ \boxed{\mathfrak{w}_j}}_{\varphi_1}\ \mathfrak{w}_{j+1} \ldots \boxed{\mathfrak{w}_{|\mathfrak{w}|}}} \qquad \underbrace{\boxed{\mathfrak{w}_j \mathfrak{w}_{j+1} \ldots \boxed{\mathfrak{w}_{|\mathfrak{w}|}}}}_{\varphi_2}$$

1. ALT and $1\mathrm{SL}_{\mathtt{R1}}(*, \mathrel{-\!\!*}, \mathtt{reach}^+)$ are NON-ELEMENTARY
2. ALT is decidable in TOWER, as it is captured by $\mathrm{SL}(\forall, *)$

$1\mathrm{SL}^{\mathtt{R2}}_{\mathtt{R1}}(*, \mathrel{-\!\!*}, \mathtt{reach}^{+})$ is in PSPACE

$\cancel{1\mathrm{SL}^{\mathtt{R2}}_{\mathtt{R1}}(*, \ *, \mathbf{reach}^{+})}$ ~~is in PSPACE~~

Test Formulae “technique”

Test Formulae example on $\mathrm{SL}(*, -\!\!*)$

$$\varphi := \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathtt{emp} \mid \mathtt{x} = \mathtt{y} \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \varphi_1 * \varphi_2 \mid \varphi_1 -\!\!* \varphi_2$$

1. Design an equivalence relation on models, based on the satisfaction of atomic predicates (test formulae), e.g.

$$\mathtt{x} = \mathtt{y} \qquad \mathtt{x} \hookrightarrow \mathtt{y} \qquad \mathtt{alloc}(\mathtt{x}) \qquad \mathtt{size} \geq \beta$$

2. Show that any formula of our logic is equivalent to a Boolean combination of test formulae, e.g.

$$(\mathtt{x} \hookrightarrow \mathtt{y}) * \neg\mathtt{emp} \iff \mathtt{x} \hookrightarrow \mathtt{y} \wedge \mathtt{size} \geq 2$$

3. Prove small-model property for the logic of test formulae.

1: Designing Test Formulae

- Fix $\alpha \in \mathbb{N}^+$, $\mathtt{X} \subseteq_{\mathsf{fin}} \mathtt{VAR}$
- Let us define $\mathsf{Test}(\mathtt{X}, \alpha)$ as the **finite** set of predicates:

$$\{\mathtt{x} = \mathtt{y},\ \mathtt{x} \hookrightarrow \mathtt{y},\ \mathtt{alloc}(\mathtt{x}),\ \mathtt{size} \geq \beta \ \mid\ \beta \in [1, \alpha],\ \mathtt{x}, \mathtt{y} \in \mathtt{X}\}$$

Indistinguishability relation $(s, h) \approx_{\alpha}^{\mathtt{X}} (s', h')$

$$\text{for every } \varphi \in \mathsf{Test}(\mathtt{X}, \alpha),\ (\mathsf{s}, \mathsf{h}) \models \varphi \text{ iff } (\mathsf{s}', \mathsf{h}') \models \varphi$$

Note: α is related to the number of occurrences of $*$ and $-\!\!*$ in a formula of separation logic.

2: $*$ elimination Lemma

We want to design $\mathsf{Test}(\mathtt{X}, \alpha)$ so that the following results hold

- For every $\varphi \in \mathsf{Bool}(\mathsf{Test}(\mathtt{X}, \alpha_1))$, $\psi \in \mathsf{Bool}(\mathsf{Test}(\mathtt{X}, \alpha_2))$ there is $\gamma \in \mathsf{Bool}(\mathsf{Test}(\mathtt{X}, \alpha_1 + \alpha_2))$ such that

$$\varphi * \psi \iff \gamma$$

- Similar elimination result for $-\!\!*$.

Lemmata holds for

$$\mathsf{Test}(\mathtt{X}, \alpha) = \left\{ \begin{matrix} \mathtt{x} = \mathtt{y},\ \mathtt{x} \hookrightarrow \mathtt{y} \\ \mathtt{alloc}(\mathtt{x}),\ \mathtt{size} \geq \beta \end{matrix} \;\middle|\; \begin{matrix} \beta \in [1, \alpha] \\ \mathtt{x}, \mathtt{y} \in \mathtt{X} \end{matrix} \right\}$$

3: Test formulae, after $*$ and $\twoheadrightarrow$ elimination

Hypothesis: A family of test formulae, such that

- captures the atomic predicates of $\mathrm{SL}(*, \twoheadrightarrow)$
- satisfies the $*$ and $\twoheadrightarrow$ elimination Lemmata

Thesis: for every formulae φ of $\mathrm{SL}(*, \twoheadrightarrow)$,

- φ is equivalent to a Boolean combination of test formulae.
- If $\alpha \geq |\varphi|$, $\mathtt{X} \supseteq \mathrm{v}(\varphi)$ and $(\mathsf{s}, \mathsf{h}) \approx_{\alpha}^{\mathtt{X}} (\mathsf{s}', \mathsf{h}')$ then

$$(\mathsf{s}, \mathsf{h}) \models \varphi \text{ iff } (\mathsf{s}', \mathsf{h}') \models \varphi.$$

Small-model property derived from $\approx_{\alpha}^{\mathtt{X}}$

- Small-model property for Boolean combination of test formulae carries over to $\mathrm{SL}(*, \twoheadrightarrow)$.
- test formulae in PSPACE $\implies$ $\mathrm{SL}(*, \twoheadrightarrow)$ is in PSPACE.

$1\mathrm{SL}^{\mathtt{R2}}_{\mathtt{R1}}(*, \mathrel{-\!\!*}, \mathtt{reach}^+)$ is in PSPACE

$$\pi := \mathtt{x} = \mathtt{y} \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \mathtt{emp} \mid \underline{\mathcal{A} \mathrel{-\!\!*} \mathcal{C}\ (\mathtt{R1})}$$

$$\mathcal{C} := \pi \mid \mathcal{C} \wedge \mathcal{C} \mid \neg\mathcal{C} \mid \exists \mathtt{u}\ \mathcal{C} \mid \mathcal{C} * \mathcal{C}$$

$$\mathcal{A} := \pi \mid \underline{\mathtt{reach}^+(v_1, v_2)} \mid \mathcal{A} \wedge \mathcal{A} \mid \neg\mathcal{A} \mid \exists \mathtt{u}\ \mathcal{A} \mid \mathcal{A} * \mathcal{A}$$

where $(\mathtt{R2})$ if $v_1 = \mathtt{u}$ then $v_2 = \mathtt{u}$

Not so easy...

- Asymmetric $\mathcal{A} \mathrel{-\!\!*} \mathcal{C}$.
  - two sets of test formulae: two $*/\exists$ elimination Lemmata
  - $\mathrel{-\!\!*}$ elimination Lemma that glues the two set of test formulae
- instead of "$\mathtt{size} \geq \beta$ s.t. $\beta \in [1, \alpha]$", the βs of new test formulae are bounded by functions on α, e.g.

  $$\#\mathtt{loop}_{\mathtt{x}}(\beta) \geq \gamma \qquad \gamma \in [1, \tfrac{1}{2}\alpha(\alpha + 3) - 1]$$

  bounds are found by solving a set of recurrence equations!

Recap

$\mathrm{SL}(*, \mathbin{-\!\!*}, \mathtt{reach})$
undec.

$\boxed{\mathrm{1SL}_{\mathtt{R1}}(*, \mathbin{-\!\!*}, \mathtt{reach}^+)}$
unk. non-elem.

$\boxed{\mathrm{1SL}^{\mathtt{R2}}_{\mathtt{R1}}(*, \mathbin{-\!\!*}, \mathtt{reach}^+)}$
PSPACE-complete

$\boxed{\mathrm{ALT}}$
dec. non-elem.

$\mathrm{1SL}(*, \mathbin{-\!\!*})$
PSPACE-complete

$\mathrm{SL}(*, \mathtt{reach})$
PSPACE-complete

PITL
dec. non-elem.

- $\mathrm{1SL}^{\mathtt{R2}}_{\mathtt{R1}}(*, \mathbin{-\!\!*}, \mathtt{reach}^+)$ strictly generalise other PSPACE-compl. extensions of propositional separation logic
- Can be used to check for robustness properties.

.

Recap

$$
\begin{tikzcd}
 & & \begin{array}{c}\mathrm{SL}(*, \mathbin{-\!\!*}, \texttt{reach}) \\ \text{undec.}\end{array} & & \\
 & & \boxed{\begin{array}{c}\mathrm{1SL}_{\texttt{R1}}(*, \mathbin{-\!\!*}, \texttt{reach}^{+}) \\ \text{unk. non-elem.}\end{array}} \arrow[u] & & \\
 & \boxed{\begin{array}{c}\mathrm{1SL}^{\texttt{R2}}_{\texttt{R1}}(*, \mathbin{-\!\!*}, \texttt{reach}^{+}) \\ \text{PSPACE-complete}\end{array}} \arrow[ur] & & \boxed{\begin{array}{c}\mathrm{ALT} \\ \text{dec. non-elem.}\end{array}} \arrow[ul] & \\
\begin{array}{c}\mathrm{1SL}(*, \mathbin{-\!\!*}) \\ \text{PSPACE-complete}\end{array} \arrow[ur] & & \begin{array}{c}\mathrm{SL}(*, \texttt{reach}) \\ \text{PSPACE-complete}\end{array} \arrow[ul] & & \begin{array}{c}\mathrm{PITL} \\ \text{dec. non-elem.}\end{array} \arrow[ul]
\end{tikzcd}
$$

- ALT seems to be an interesting tool for reductions, as it is a fragment or it is easily captured by many logics in TOWER e.g. $\mathrm{QCTL}(\mathtt{U})$, $\mathrm{MSL}(\Diamond, \langle \mathrm{U} \rangle, *)$, $\mathrm{2SL}(*)$

.