Hoare calculus, Separation Logic and robustness properties of imperative programs.

Alessio Mansutti

LSV, CNRS, ENS Paris-Saclay

Sophia Antipolis - May 2019

An introduction to separation logic

- Floyd-Hoare proof systems for program verification;
- dealing with pointers with separation logic;
- revisit some classical results for propositional separation logic.

- An extension of propositional separation logic that can express interesting properties for program verification.

Some ingredients of program verification in stateful systems

- (memory) states of the system; $\{\mathtt{x} = 3\ ,\ \mathtt{y} = 5\ ,\ \ldots\}$
- programs (state transformers); $\mathtt{x} \leftarrow \mathtt{y};\ \mathtt{x} \leftarrow \mathtt{x} + 1;$
- logical assertions/properties. "$\mathtt{x} > \mathtt{y}$ holds"

Some ingredients of program verification in stateful systems

- (memory) states of the system; $\{\mathtt{x} = 3\ ,\ \mathtt{y} = 5\ ,\ \dots\}$
- programs (state transformers); $\mathtt{x} \leftarrow \mathtt{y};\ \mathtt{x} \leftarrow \mathtt{x} + 1;$
- logical assertions/properties. "$\mathtt{x} > \mathtt{y}$ holds"

'69: Floyd-Hoare proof systems

A logical system where **judgements** (i.e. **Hoare triples**) are of the form

$\{\ \varphi\ \}\ \mathtt{Prog}\ \{\ \psi\ \};$ read as:

*"Every state $\mathfrak{M}$ satisfying the **precondition** φ, will satisfy the **postcondition** ψ after being modified by the program* `Prog`*".*

Floyd-Hoare proof systems

$$\texttt{Prog} \ := \ \texttt{x} \leftarrow \mathsf{Expr} \mid \texttt{Prog}_1;\texttt{Prog}_2 \mid \textbf{while } \mathsf{B} \textbf{ do } \texttt{Prog} \mid \ \ldots$$

Proofs of are done by instantiating and chaining **inference rules**, e.g.

$$\frac{}{\{\ \varphi[\texttt{x}/\mathit{Expr}]\ \}\ \texttt{x} \leftarrow \mathit{Expr}\ \{\ \varphi\ \}}$$

$$\frac{\{\ \varphi\ \}\ \texttt{Prog}_1\ \{\ \chi\ \} \qquad \{\ \chi\ \}\ \texttt{Prog}_2\ \{\ \psi\ \}}{\{\ \varphi\ \}\ \texttt{Prog}_1;\texttt{Prog}_2\ \{\ \psi\ \}}$$

$$\frac{\{\ \varphi_{\mathit{Inv}} \wedge B\ \}\ \texttt{Prog}\ \{\ \varphi_{\mathit{Inv}}\ \}}{\{\ \varphi_{\mathit{Inv}}\ \}\ \textbf{while } \mathsf{B} \textbf{ do } \texttt{Prog}\ \{\ \varphi_{\mathit{Inv}} \wedge \neg B\ \}}$$

$$\frac{\varphi_2 \models \varphi_1 \qquad \{\ \varphi_1\ \}\ \texttt{Prog}\ \{\ \psi_1\ \} \qquad \psi_1 \models \psi_1}{\{\ \varphi_2\ \}\ \texttt{Prog}\ \{\ \psi_2\ \}}$$

Note: φ_{Inv} is a loop invariant, $\models$ is the logical entailment.

Floyd-Hoare proof systems

$$\texttt{Prog} \;:=\; \texttt{x} \leftarrow \mathsf{Expr} \mid \texttt{Prog}_1;\texttt{Prog}_2 \mid \textbf{while } \mathsf{B} \textbf{ do } \texttt{Prog} \mid \ldots$$

Proofs of are done by instantiating and chaining **inference rules**, e.g.

$$\frac{}{\{\ \varphi[\texttt{x}/\mathit{Expr}]\ \}\ \texttt{x} \leftarrow \mathit{Expr}\ \{\ \varphi\ \}}$$

$$\frac{\{\ \varphi\ \}\ \texttt{Prog}_1\ \{\ \chi\ \} \qquad \{\ \chi\ \}\ \texttt{Prog}_2\ \{\ \psi\ \}}{\{\ \varphi\ \}\ \texttt{Prog}_1;\texttt{Prog}_2\ \{\ \psi\ \}}$$

$$\frac{\{\ \varphi_{\mathit{Inv}} \wedge B\ \}\ \texttt{Prog}\ \{\ \varphi_{\mathit{Inv}}\ \}}{\{\ \varphi_{\mathit{Inv}}\ \}\ \textbf{while } \mathsf{B} \textbf{ do } \texttt{Prog}\ \{\ \varphi_{\mathit{Inv}} \wedge \neg B\ \}}$$

$$\frac{\varphi_2 \models \varphi_1 \qquad \{\ \varphi_1\ \}\ \texttt{Prog}\ \{\ \psi_1\ \} \qquad \psi_1 \models \psi_1}{\{\ \varphi_2\ \}\ \texttt{Prog}\ \{\ \psi_2\ \}}$$

For the Type–Theorists:

$$\frac{\sigma_2 \;\leq\; \sigma_1 \qquad \tau_1 \;\leq\; \tau_2}{\sigma_1 \rightarrow \tau_1 \;\leq\; \sigma_2 \rightarrow \tau_2}$$

entailment $\sim$ subtyping

Note: φ_{Inv} is a loop invariant, $\models$ is the logical entailment.

Floyd-Hoare proof systems

$$\texttt{Prog} \;:=\; \texttt{x} \leftarrow \mathsf{Expr} \mid \texttt{Prog}_1;\texttt{Prog}_2 \mid \textbf{while}\ \mathsf{B}\ \textbf{do}\ \texttt{Prog} \mid \ldots$$

Proofs of are done by instantiating and chaining **inference rules**, e.g.

$$\frac{}{\{\ \varphi[\texttt{x}/\mathit{Expr}]\ \}\ \texttt{x} \leftarrow \mathit{Expr}\ \{\ \varphi\ \}}$$

$$\frac{\{\ \varphi\ \}\ \texttt{Prog}_1\ \{\ \chi\ \} \qquad \{\ \chi\ \}\ \texttt{Prog}_2\ \{\ \psi\ \}}{\{\ \varphi\ \}\ \texttt{Prog}_1;\texttt{Prog}_2\ \{\ \psi\ \}}$$

$$\frac{\{\ \varphi_{\mathit{Inv}} \wedge B\ \}\ \texttt{Prog}\ \{\ \varphi_{\mathit{Inv}}\ \}}{\{\ \varphi_{\mathit{Inv}}\ \}\ \textbf{while}\ \mathsf{B}\ \textbf{do}\ \texttt{Prog}\ \{\ \varphi_{\mathit{Inv}} \wedge \neg B\ \}}$$

$$\frac{\varphi_2 \models \varphi_1 \qquad \{\ \varphi_1\ \}\ \texttt{Prog}\ \{\ \psi_1\ \} \qquad \psi_1 \models \psi_1}{\{\ \varphi_2\ \}\ \texttt{Prog}\ \{\ \psi_2\ \}}$$

For the Type–Theorists:

$$\frac{\mathbb{N} \leq \mathbb{Z} \qquad \mathbb{Z} \leq \mathbb{Q}}{\mathbb{Z} \to \mathbb{Z}\ \leq\ \mathbb{N} \to \mathbb{Q}}$$

entailment $\sim$ subtyping

Note: φ_{Inv} is a loop invariant, $\models$ is the logical entailment.

Soundness and completeness

- **Soundness:** if $\{ \varphi \}$ `Prog` $\{ \psi \}$ can be proved, then executing `Prog` from a state satisfying φ will only terminate in states satisfying ψ.
- **Completeness:** the converse of soundness.
- Hoare calculus is sound and complete, provided that $\varphi, \psi, \ldots$ come from a sound and complete logic.

Modular verification and pointers

To analyse large programs it is vital to reason locally. We would like:

$$\frac{\{\ \varphi\ \}\ \texttt{Prog}\ \{\ \psi\ \}\qquad \mathsf{modv}(\texttt{Prog})\cap\mathsf{fv}(\chi)=\emptyset}{\{\ \varphi\wedge\chi\ \}\ \texttt{Prog}\ \{\ \psi\wedge\chi\ \}}$$

but this rule is not valid when considering as a state the standard heap/RAM memory, containing pointers:

$$\frac{\{\ \mathtt{x}\hookrightarrow 1\ \}\ {}^{*}\mathtt{x}\leftarrow 0\ \{\ \mathtt{x}\hookrightarrow 0\ \}}{\{\ \mathtt{x}\hookrightarrow 1\wedge\mathtt{y}\hookrightarrow 1\ \}\ {}^{*}\mathtt{x}\leftarrow 0\ \{\ \mathtt{x}\hookrightarrow 0\wedge\mathtt{y}\hookrightarrow 1\ \}}$$

does not hold whenever $\mathtt{x}$ and $\mathtt{y}$ are in aliasing.

Here, $\mathtt{x}\hookrightarrow 1$ holds in memory models such that:

Separation logic (Reynolds'02)

SL adds the notion of **separation** ($*$) of a state and a valid **frame rule**:

$$\frac{\{\varphi\}\ \texttt{Prog}\ \{\psi\} \quad \mathsf{modv}(\texttt{Prog}) \cap \mathsf{fv}(\chi) = \emptyset}{\{\varphi * \chi\}\ \texttt{Prog}\ \{\psi * \chi\}}$$

Intuitively, separation means $(\texttt{x} \hookrightarrow n \ * \texttt{y} \hookrightarrow m) \implies \texttt{x} \neq \texttt{y}$.

Separation logic (Reynolds'02)

SL adds the notion of **separation** ($*$) of a state and a valid **frame rule**:

$$\frac{\{\varphi\}\ \texttt{Prog}\ \{\psi\} \quad \mathsf{modv}(\texttt{Prog}) \cap \mathsf{fv}(\chi) = \emptyset}{\{\varphi * \chi\}\ \texttt{Prog}\ \{\psi * \chi\}}$$

Intuitively, separation means $(\mathtt{x} \hookrightarrow n\ * \mathtt{y} \hookrightarrow m) \implies \mathtt{x} \neq \mathtt{y}$.

- **Automatic Verifiers**: Infer, SLAyer, Predator
- **Semi-automatic Verifiers**: Smallfoot, Verifast

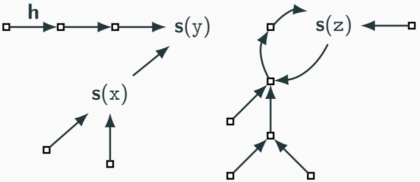
Also, see "Why Separation Logic Works" (Pym et al. '18).

Separation logic: Memory states

Separation Logic is interpreted over **memory states** $(\mathbf{s}, \mathbf{h})$ where:

- **store**, $\mathbf{s} : \mathtt{VAR} \rightarrow \mathtt{LOC}$
- **heap**, $\mathbf{h} : \mathtt{LOC} \rightarrow_{\mathbf{fin}} \mathtt{LOC}$

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



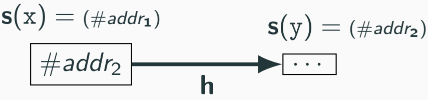
here, $\mathbf{h}(\mathbf{s}(\mathrm{x})) = \mathbf{s}(\mathrm{y})$

- Disjoint heaps $(\mathbf{h}_1 \perp \mathbf{h}_2)$: $\mathrm{dom}(\mathbf{h}_1) \cap \mathrm{dom}(\mathbf{h}_2) = \emptyset$
- Union of disjoint heaps $(\mathbf{h}_1 + \mathbf{h}_2)$: union of partial functions.

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

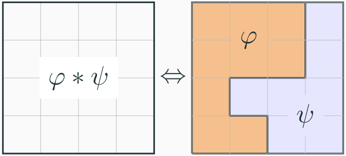
$$\varphi \coloneqq \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathtt{emp} \mid \mathtt{x} = \mathtt{y} \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \varphi_1 * \varphi_2 \mid \varphi_1 -\!\!* \varphi_2$$

- $(\mathbf{s}, \mathbf{h}) \models \mathtt{emp} \iff \mathrm{dom}(\mathbf{h}) = \emptyset$
- $(\mathbf{s}, \mathbf{h}) \models \mathtt{x} = \mathtt{y} \iff \mathbf{s}(\mathtt{x}) = \mathbf{s}(\mathtt{y})$
- $(\mathbf{s}, \mathbf{h}) \models \mathtt{x} \hookrightarrow \mathtt{y} \iff \mathbf{s}(\mathtt{x}) \in \mathrm{dom}(\mathbf{h})$ and $\mathbf{h}(\mathbf{s}(\mathtt{x})) = \mathtt{y}$



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

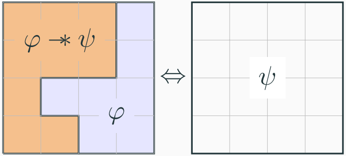
$$(\mathsf{s}, \mathbf{h}) \models \varphi * \psi$$



there are $\mathbf{h}_1, \mathbf{h}_2$ s.t.

- $\mathbf{h}_1 \perp \mathbf{h}_2$ and $\mathbf{h} = \mathbf{h}_1 + \mathbf{h}_2$,
- $(\mathsf{s}, \mathbf{h}_1) \models \varphi$ and $(\mathsf{s}, \mathbf{h}_2) \models \psi$

$$(\mathsf{s}, \mathbf{h}) \models \varphi -\!\!* \psi$$



for every $\mathbf{h}'$

if $\mathbf{h}' \perp \mathbf{h}$ and $(\mathsf{s}, \mathbf{h}') \models \varphi$,

then $(\mathsf{s}, \mathbf{h} + \mathbf{h}') \models \psi$

Decision Problems

- Hoare proof-system requires to solve classical problems:
  - satisfiability/validity/entailment;
  - weakest precondition/strongest postcondition;

$$\frac{\varphi \models \varphi' \qquad \{\ \varphi'\ \}\ \texttt{Prog}\ \{\ \psi'\ \} \qquad \psi' \models \psi}{\{\ \varphi\ \}\ \texttt{Prog}\ \{\ \psi\ \}}$$

- sat. is $\texttt{PSpace}$-complete for $\texttt{SL}(*, \mathrel{-\!\!*})$

  [Calcagno et al. – FSTTCS'03] **[Lozes – Space'04]**.

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

How to: decide satisfiability

- **Model checking:** given φ and $(\mathbf{s}, \mathbf{h})$, does $(\mathbf{s}, \mathbf{h})$ satisfies φ?
- **Satisfiability:** Is φ satisfied by some memory state $(\mathbf{s}, \mathbf{h})$?

Usually, to prove that satisfiability is decidable...

1. prove decidability of model checking;
2. find a small-model property (SMP) for satisfiability;
3. enumerate the finite set of models bounded by the SMP;
4. apply model checking on these models.

In separation logic, we can express satisfiability with model checking!

$-\!\!*$ and how to go from satisfiability to model checking

Let $\varphi \mathrel{-\!\!\circledast} \psi$ defined as $\neg(\varphi \mathrel{-\!\!*} \neg\psi)$.

$(\mathbf{s}, \mathbf{h}) \models \varphi \mathrel{-\!\!\circledast} \psi$ iff there is $\mathbf{h}' \perp \mathbf{h}$ s.t. $(\mathbf{s}, \mathbf{h}') \models \varphi$ and $(\mathbf{s}, \mathbf{h} + \mathbf{h}') \models \psi$.

Given φ,

$$\exists \mathbf{s}\ \exists \mathbf{h} \text{ s.t. } (\mathbf{s}, \mathbf{h}) \models \varphi \qquad \text{(i.e. satisfiability)}$$

is equivalent to

$$\exists \mathbf{s}\ (\mathbf{s}, \emptyset) \models \varphi \mathrel{-\!\!\circledast} \top.$$

$-\!\!*$ and how to go from satisfiability to model checking

Let $\varphi \mathbin{-\!\!\circledast} \psi$ defined as $\neg(\varphi -\!\!* \neg\psi)$.

$(\mathbf{s}, \mathbf{h}) \models \varphi \mathbin{-\!\!\circledast} \psi$ iff there is $\mathbf{h}' \perp \mathbf{h}$ s.t. $(\mathbf{s}, \mathbf{h}') \models \varphi$ and $(\mathbf{s}, \mathbf{h} + \mathbf{h}') \models \psi$.

Given φ,

$$\exists \mathbf{s}\ \exists \mathbf{h} \text{ s.t. } (\mathbf{s}, \mathbf{h}) \models \varphi \qquad \text{(i.e. satisfiability)}$$

is equivalent to

$$\exists \mathbf{s}\ (\mathbf{s}, \emptyset) \models \varphi \mathbin{-\!\!\circledast} \top.$$

- Let X the (finite) set of variables in φ.
- Let $eq(X)$ be the set of all eq. relations on X.
- For every $E \in eq(X)$, consider one $\mathbf{s}$ s.t. $\forall \mathtt{x} \in X\ \mathbf{s}(\mathtt{x}) = \mathbf{s}(\mathtt{y})$ iff $\mathtt{x} E \mathtt{y}$.

  Check $(\mathbf{s}, \emptyset) \models \varphi \mathbin{-\!\!\circledast} \top$.

"Locality theorem" for $\mathtt{SL}(*, \mathrel{-\!\!*})$

Theorem (Lozes, 2004 – Space)

Every formula of $\mathtt{SL}(*, \mathrel{-\!\!*})$ *is logically equivalent to a Boolean combination of* ***core formulae****.*

From this theorem we can get:

- expressive power results
- complexity result (small model property)
- axiomatisation

Note: When considering extensions of the logic, we need to derive new core formulae and reprove the theorem.

First order theories: Gaifman Locality Theorem

Theorem (Gaifman – 1982, Herbrand Symposium)

*Every FO sentence is logically equivalent to a Boolean combination of **local formulae**.*

- application of Ehrenfeucht-Fraïssé games

Relation between models

$\mathfrak{M} \leftrightarrow_n \mathfrak{M}'$

(partial iso. up to n)

$\rightleftarrows$

EF-games

Duplicator has a winning strategy

(n round game)

$\rightleftarrows$

Semantics of logic

$\mathfrak{M} \approx_n \mathfrak{M}'$

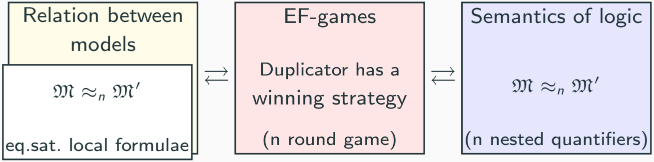
(n nested quantifiers)

First order theories: Gaifman Locality Theorem

Theorem (Gaifman – 1982, Herbrand Symposium)

*Every FO sentence is logically equivalent to a Boolean combination of **local formulae**.*

- application of Ehrenfeucht-Fraïssé games



[Libkin'12]

Core formulae for $\mathtt{SL}(*, -\!*)$

Fix $\mathtt{X} \subseteq \mathtt{VAR}$ and $\alpha \in \mathbb{N}^+$

$$\textbf{Core}(\mathtt{X}, \alpha) \stackrel{\text{def}}{=} \left\{ \begin{array}{cc|c} \mathtt{x} = \mathtt{y}, & \mathtt{x} \hookrightarrow \mathtt{y}, & \beta \in [0, \alpha], \\ \mathtt{alloc}(\mathtt{x}), & \mathtt{size} \geq \beta & \mathtt{x}, \mathtt{y} \in \mathtt{X} \end{array} \right\}$$

where

$$(\mathbf{s}, \mathbf{h}) \models \mathtt{size} \geq \beta \;\text{ iff }\; \mathrm{card}(\mathrm{dom}(\mathbf{h})) \geq \beta;$$
$$(\mathbf{s}, \mathbf{h}) \models \mathtt{alloc}(\mathtt{x}) \;\text{ iff }\; \mathbf{s}(\mathtt{x}) \in \mathrm{dom}(\mathbf{h}).$$

- indistinguishability $\boxed{\text{Relation}}$:

  $$(\mathbf{s}, \mathbf{h}) \leftrightarrow^{\mathtt{X}}_{\alpha} (\mathbf{s}', \mathbf{h}') \;\text{ iff }\; \forall \varphi \in \textbf{Core}(\mathtt{X}, \alpha),\; (\mathbf{s}, \mathbf{h}) \models \varphi \text{ iff } (\mathbf{s}', \mathbf{h}') \models \varphi$$

- Both EF-game and winning strategy for Duplicator are hidden inside two (technical) elimination lemmas.

Core formulae: $*$ elimination lemma

Lemma

Suppose $(\boldsymbol{s}, \boldsymbol{h}) \leftrightarrow^{\mathtt{X}}_{\alpha} (\boldsymbol{s}', \boldsymbol{h}')$. *Then,*

for every $\alpha_1 + \alpha_2 = \alpha$ $(\alpha_1, \alpha_2 \in \mathbb{N}^+)$, *and every* $\boldsymbol{h}_1 + \boldsymbol{h}_2 = \boldsymbol{h}$, *(Spoiler)*

there are $\boldsymbol{h}_1' + \boldsymbol{h}_2' = \boldsymbol{h}'$ *such that* *(Duplicator)*

$$(\boldsymbol{s}, \boldsymbol{h}_1) \leftrightarrow^{\mathtt{X}}_{\alpha_1} (\boldsymbol{s}', \boldsymbol{h}_1') \text{ and } (\boldsymbol{s}, \boldsymbol{h}_2) \leftrightarrow^{\mathtt{X}}_{\alpha_2} (\boldsymbol{s}', \boldsymbol{h}_2').$$

- necessary to obtain a winning strategy for Duplicator

Core formulae: $*$ elimination lemma

Lemma

Suppose $(\boldsymbol{s},\boldsymbol{h}) \leftrightarrow^{\mathtt{X}}_{\alpha} (\boldsymbol{s}',\boldsymbol{h}')$. *Then,*

for every $\alpha_1+\alpha_2=\alpha$ $(\alpha_1,\alpha_2\in\mathbb{N}^+)$, *and every* $\boldsymbol{h}_1+\boldsymbol{h}_2=\boldsymbol{h}$, *(Spoiler)*

there are $\boldsymbol{h}_1'+\boldsymbol{h}_2'=\boldsymbol{h}'$ *such that* *(Duplicator)*

$$(\boldsymbol{s},\boldsymbol{h}_1) \leftrightarrow^{\mathtt{X}}_{\alpha_1} (\boldsymbol{s}',\boldsymbol{h}_1') \text{ and } (\boldsymbol{s},\boldsymbol{h}_2) \leftrightarrow^{\mathtt{X}}_{\alpha_2} (\boldsymbol{s}',\boldsymbol{h}_2').$$

- necessary to obtain a winning strategy for Duplicator

By $\boxed{\text{Relation}}$ $\leftrightarrows$ $\boxed{\text{EF-games}}$ $\leftrightarrows$ $\boxed{\text{Semantics}}$ it leads to:

For every $\varphi \in \mathbf{Bool}(\mathbf{Core}(\mathtt{X},\alpha_1))$ and $\psi \in \mathbf{Bool}(\mathbf{Core}(\mathtt{X},\alpha_2))$ there is $\chi \in \mathbf{Bool}(\mathbf{Core}(\mathtt{X},\alpha_1+\alpha_2))$ such that

$$\varphi * \psi \iff \chi$$

Note: similar elimination lemma for $-\!\!*$.

Core formulae: after $*$ and $\mathrel{-\!\!*}$ elimination

Theorem

For every φ in $\mathtt{SL}(*, \mathrel{-\!\!*})$:

1. *there is en equivalent Boolean combination of core formulae.*
2. *for every $\alpha \geq |\varphi|$, $\mathtt{X} \supseteq \mathsf{v}(\varphi)$ and $(\boldsymbol{s}, \boldsymbol{h}) \leftrightarrow^{\mathtt{X}}_{\alpha} (\boldsymbol{s}', \boldsymbol{h}')$,*

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

[2] give us a bound on the smallest model satisfying a formula.
Then, we have a small-model property.
It leads to a proof that $\mathrm{SAT}(\mathtt{SL}(*, \mathrel{-\!\!*}))$ is in $\mathtt{PSpace}$.

Extending propositional separation logic for robustness properties.

Robustness Properties (Jansen, et al. – ESOP'17)

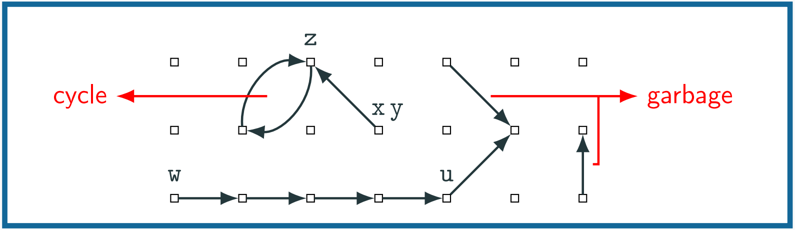
- φ comply with the **acyclicity** property iff every model of φ is acyclic.
- φ comply with the **garbage freedom** property iff in every model $(\mathbf{s}, \mathbf{h}) \models \varphi$, for each $\ell \in \mathrm{dom}(\mathbf{h})$ there is $\mathtt{x} \in \mathsf{v}(\varphi)$ s.t. $\mathbf{s}(\mathtt{x})$ reaches ℓ.

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

Our Goal

Provide a similar result for **propositional** separation logic.

Robustness Properties (Jansen, et al. – ESOP'17)



ic.

ℓ.

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

Our Goal

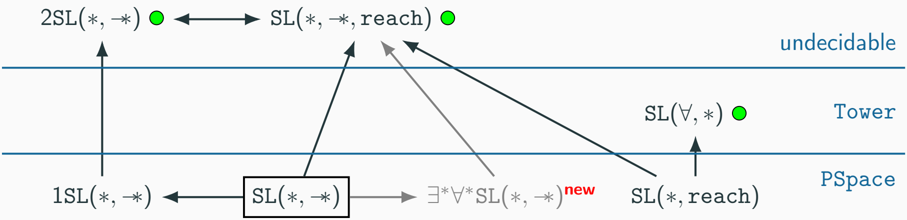
Provide a similar result for **propositional** separation logic.

Desiderata

We aim to an extension of propositional separation logic where

- satisfiability/entailment are decidable in PSpace (as $\mathtt{SL}(*, -\!\!*)$)
- robustness properties reduce to one of these classical problems

Known extensions



Let's start with reachability + 1 quantified variable

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

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**: $\varphi \models \forall \mathtt{u}\ (\mathtt{alloc}(\mathtt{u}) \Rightarrow \bigvee_{\mathtt{x} \in \mathbf{fv}(\varphi)} \mathtt{reach}(\mathtt{x}, \mathtt{u}))$

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

- $\mathtt{reach}(\mathtt{x}, \mathtt{y}) \stackrel{\mathsf{def}}{=} \mathtt{x} = \mathtt{y} \vee \mathtt{reach}^+(\mathtt{x}, \mathtt{y})$

Undecidability and Restrictions

Theorem (Demri, Lozes, M. – 2018, Fossacs)

$\mathtt{SL}(*, -\!\!*)$ *enriched with* $\mathtt{reach}(\mathtt{x}, \mathtt{y}) = 2$ *and* $\mathtt{reach}(\mathtt{x}, \mathtt{y}) = 3$ *is undecidable.*

$\Longrightarrow$ $\mathsf{SAT}(1\mathtt{SL}(*, -\!\!*, \mathtt{reach}^+))$ is undecidable.

We syntactically restrict the logic so that $\mathtt{reach}^+(\mathtt{x}, \mathtt{y})$ is s.t.

R1: it does not appear on the right side of its first $-\!\!*$ ancestor (seeing the formula as a tree)

- $\varphi -\!\!* (\psi * \mathtt{reach}^+(\mathtt{u}, \mathtt{u}))$ 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 are still expressible (formulae as before)!

Results (FSTTCS'18)

1. SAT($\mathtt{1SL}_{\mathtt{R1}}^{\mathtt{R2}}(*, -\!\!*, \mathtt{reach}^+)$) is $\mathtt{PSpace}$-complete
   - strictly subsumes $\mathtt{1SL}(*, -\!\!*)$ and $\mathtt{SL}(*, \mathtt{reach}^+)$.

2. SAT($\mathtt{1SL}_{\mathtt{R1}}(*, -\!\!*, \mathtt{reach}^+)$) is $\mathtt{Tower}$-hard.

Proof Techniques

(1) extend the **core formulae technique** used for $\mathtt{SL}(*, -\!\!*)$.

(2) from the non-emptyness problem of star-free regular expression.

$\texttt{1SL}^{\texttt{R2}}_{\texttt{R1}}(*, \mathrel{-\!\!*}, \texttt{reach}^+)$ is in $\texttt{PSpace}$...

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

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

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

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

- Asymmetric $\mathcal{A} \mathrel{-\!\!*} \mathcal{C}$: design two sets of core formulae against
  - two $*$ and two $\exists$ elimination lemmas;
  - one $\mathrel{-\!\!*}$ elimination lemma that glues the two set of core formulae.

- instead of "$\texttt{size} \geq \beta$ s.t. $\beta \in [1, \alpha]$", the βs of new core formulae are bounded by functions on α, e.g.

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

  bounds are found by solving a set of recurrence equations.

Recap

- Floyd-Hoare proof system for program verification.
- We want modular proof. Problematic if the language has pointers.
- Separation logic works.

- In $\mathtt{SL}(*, -\!\!*)$, satisfiability $\rightsquigarrow$ model checking.
- Core formulae to prove that satisfiability is $\mathtt{PSpace}$-complete.

- We want to express robustness properties for program verification.
- $\mathtt{1SL}^{\mathtt{R2}}_{\mathtt{R1}}(*, -\!\!*, \mathtt{reach}^+)$ can express robustness properties, in $\mathtt{PSpace}$.