Extending propositional separation logic for robustness properties

Alessio Mansutti

INFINI - November 2018

Program verification with Hoare calculus

Hoare calculus is based on proof rules manipulating Hoare triples.

$$\{\varphi\}\ C\ \{\varphi'\}$$

where

- C is a program
- φ and φ' are assertions in some logical language.

Any (memory) state that satisfies φ will satisfy φ' after being modified by C.

Programming languages with pointers

The so-called **rule of constancy**

$$\frac{\{\varphi\}\ C\ \{\varphi'\}}{\{\varphi \wedge \psi\}\ C\ \{\varphi' \wedge \psi\}}$$

"C does not mess with ψ"

is generally not valid: it is unsound if C manipulates pointers.

Programming languages with pointers

The so-called **rule of constancy**

$$\frac{\{\varphi\}\ C\ \{\varphi'\}}{\{\varphi \wedge \psi\}\ C\ \{\varphi' \wedge \psi\}} \qquad \text{“}C \text{ does not mess with } \psi\text{”}$$

is generally not valid: it is unsound if C manipulates pointers.

Example:

$$\frac{\{\exists u.[\mathrm{x}] = u\}\ [\mathrm{x}] \leftarrow 4\ \{[\mathrm{x}] = 4\}}{\{[\mathrm{y}] = 3\ \wedge\ \exists u.[\mathrm{x}] = u\}\ [\mathrm{x}] \leftarrow 4\ \{[\mathrm{y}] = 3\ \wedge\ [\mathrm{x}] = 4\}}$$

not true if `x` and `y` are in aliasing.

Reynolds'02: Separation logic

Separation logic add the notion of **separation** ($*$) of a state, so that the **frame rule**

$$\frac{\{\varphi\}\ C\ \{\varphi'\} \quad \mathsf{modv}(C) \cap \mathsf{fv}(\psi) = \emptyset}{\{\varphi * \psi\}\ C\ \{\varphi' * \psi\}}$$

is valid.
Morally, separation means $([\mathtt{x}] = n\ * [\mathtt{y}] = m) \implies \mathtt{x} \neq \mathtt{y}$

Reynolds'02: Separation logic

Separation logic add the notion of **separation** ($*$) of a state, so that the **frame rule**

$$\frac{\{\varphi\}\ C\ \{\varphi'\} \quad \mathsf{modv}(C) \cap \mathsf{fv}(\psi) = \emptyset}{\{\varphi * \psi\}\ C\ \{\varphi' * \psi\}}$$

is valid.
Morally, separation means $([\mathtt{x}] = n\ * [\mathtt{y}] = m) \implies \mathtt{x} \neq \mathtt{y}$

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

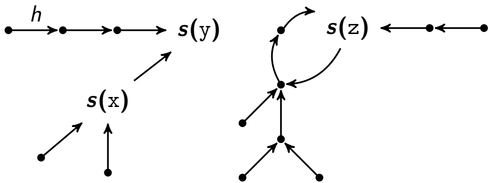
Also, see "Why Separation Logic Works" (Pym et al. '18)

Memory states

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

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

where $\mathtt{VAR} = \{\mathrm{x}, \mathrm{y}, \mathrm{z}, \dots\}$ set of (program) variables,
$\mathtt{LOC}$ set of locations (typically $\mathtt{LOC} \cong \mathbb{N} \cong \mathtt{VAR}$).



- Disjointed heaps: $\mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) = \emptyset$
- Sum of disjointed heaps $(h_1 + h_2)$ = sum of partial functions

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

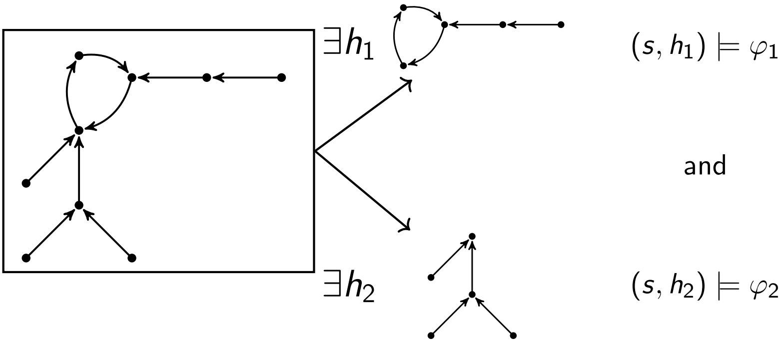
$$\varphi := \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathtt{emp} \mid \mathtt{x} = \mathtt{y} \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \varphi_1 * \varphi_2 \mid \varphi_1 -\!\!* \varphi_2$$

Semantics

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

Separating conjunction ($*$)

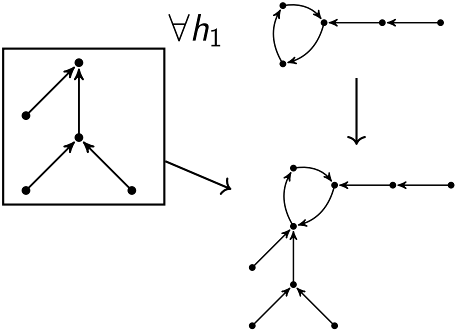
$(s, h) \models \varphi_1 * \varphi_2$ if and only if



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
  - weakest precondition/strongest postcondition

$$\frac{P \implies P' \qquad \{P'\}\ C\ \{Q'\} \qquad Q' \implies Q}{\{P\}\ C\ \{Q\}} \text{ consequence rule}$$

- satisfiability is $\mathrm{PS{\scriptsize PACE}}$-complete for $\mathtt{SL}(*, -\!*)$

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

Robusness properties

- **Acyclicity** holds for φ iff every model of φ is acyclic
- **Garbage freedom** holds for φ iff in every model of φ, each memory cell is reachable from a program variable of φ

C. Jansen et al., ESOP'17

Checking for robustness properties is $\textsc{ExpTime}$-complete for Symbolic Heaps with Inductive Predicates.

- Symbolic Heaps $\Rightarrow$ no negation, no $-\!\!*$
- Inductive Predicates: akin of Horn clauses where $*$ replaces $\wedge$

$$P \Leftarrow Q_1 \overset{*}{\times} \dots \overset{*}{\times} Q_n$$

Our Goal

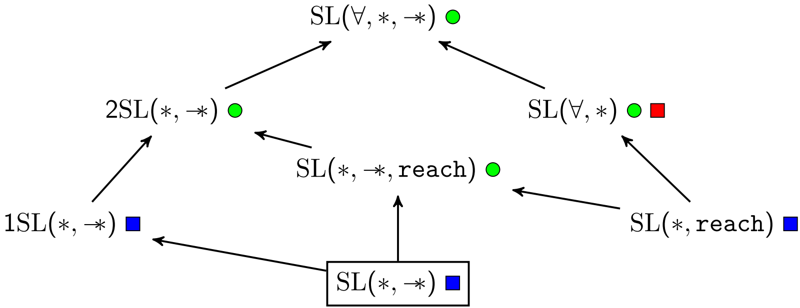
Provide similar results, but for **propositional** separation logic.

Desiderata

We aim to an extension of propositional separation logic where

- ■ satisfiability, validity and entailment are decidable
- ■ in PSPACE (as propositional separation logic)
- ● robustness properties reduce to one of these problems

Known extensions



$\mathrm{SL}(*, -\!\!*)$ + reachability and one quantified variable

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

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

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

Restrictions

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

- $\mathtt{R1}$ it is not on the right side of its first $\mathrel{-\!\!*}$ ancestor (seeing the formula as a tree)
- $\mathtt{R2}$ if $\mathtt{x} = \mathtt{u}$ then $\mathtt{y} = \mathtt{u}$

For example, given φ, ψ satisfying these conditions,

- $\mathtt{reach}^+(\mathtt{u}, \mathtt{x}) * (\varphi \mathrel{-\!\!*} \psi)$ only satisfies $\mathtt{R1}$
- $\varphi \mathrel{-\!\!*} (\mathtt{reach}^+(\mathtt{x}, \mathtt{u}) \mathrel{-\!\!*} \psi)$ satisfies both $\mathtt{R1}$ and $\mathtt{R2}$
- $\varphi \mathrel{-\!\!*} (\psi * \mathtt{reach}^+(\mathtt{u}, \mathtt{u}))$ only satisfies $\mathtt{R2}$

Note: robustness properties are expressible in this fragment.

Results

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

Proof Techniques

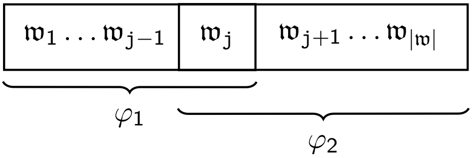
(1) reduce *Propositional interval temporal logic under locality principle* (PITL) to a logic captured by $1\mathrm{SL}_{\mathtt{R1}}(*, -\!\!*, \mathtt{reach}^+)$

(2) use the *test formulae technique* already used for $\mathrm{SL}(*, \mathtt{reach})$

PITL (Moszkowski'83)

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

- interpreted on finite non-empty words over a finite alphabet Σ
- $\mathfrak{w} \models \mathtt{pt} \iff |\mathfrak{w}| = 1$
- $\mathfrak{w} \models \mathtt{a} \iff \mathfrak{w}$ headed by $\mathtt{a}$
- $\mathfrak{w} \models \varphi_1 \mathbin{\boldsymbol{|}} \varphi_2 \iff \mathfrak{w}[1:\mathtt{j}] \models \varphi_1$ and $\mathfrak{w}[\mathtt{j}:|\mathfrak{w}|] \models \varphi_2$
  for some $\mathtt{j} \in [1, |\mathfrak{w}|]$



Note: Satisfiability is decidable, but NON-ELEMENTARY

Auxiliary Logic on Trees (ALT)

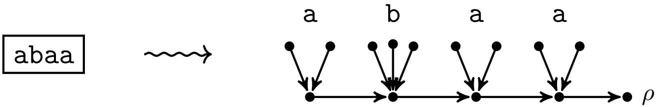
$$\varphi \coloneqq \varphi_1 \wedge \varphi_2 \ \mid\ \neg\varphi \ \mid\ \varphi_1 * \varphi_2 \ \mid\ \exists \mathtt{u}\ \varphi \ \mid\ \mathsf{T}(\mathtt{u}) \ \mid\ \mathsf{G}(\mathtt{u})$$

- interpreted on acyclic memory states
- one *special* location: the root ρ of a tree
- $(s, h) \models \mathsf{T}(\mathtt{u})$ iff $s(\mathtt{u}) \in \mathrm{dom}(h)$ and it does reach ρ
- $(s, h) \models \mathsf{G}(\mathtt{u})$ iff $s(\mathtt{u}) \in \mathrm{dom}(h)$ and it does not reach ρ
- $\exists \mathtt{u}\ \varphi$ and $\varphi_1 * \varphi_2$ as before

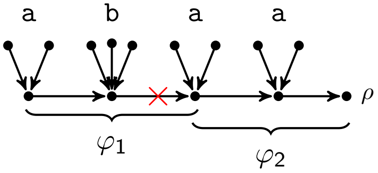
Note: ALT is captured by $1\mathrm{SL}_{\mathtt{R1}}(*, \mathrel{-\!\!*}, \mathtt{reach}^+)$.

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 $\varphi_1 | \varphi_2$:
  ALT cannot express properties about the set of locations that do not reach ρ, apart from its size



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

Reducing PITL to ALT

Solution: we introduce an alternative semantics for PITL, based on marking letters.

- $\boxed{\mathtt{a}}$ marked representation of $\mathtt{a}$
- $\mathfrak{w} \models \varphi_1 \mid \varphi_2$ if and only if $\mathfrak{w}[\mathtt{j} : |\mathfrak{w}|] \models \varphi_2$ and

$$\mathfrak{w}[1 : \mathtt{j}{-}1] \cdot \boxed{\mathfrak{w}_{\mathtt{j}}} \cdot \mathfrak{w}[\mathtt{j}{+}1 : |\mathfrak{w}|] \models \varphi_1$$

for some $\mathtt{j} \in [1, |\mathfrak{w}|]$ s.t. $\mathfrak{w}[1 : \mathtt{j}{-}1]$ has no marked symbols

1. marking a symbol can be simulated in ALT
2. ALT and $1\mathrm{SL}_{\mathtt{R1}}(*, \mathrel{-\!\!*}, \mathtt{reach}^+)$ are NON-ELEMENTARY
3. 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

$$\begin{aligned}
\pi &:= \mathtt{x} = \mathtt{y} \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \mathtt{emp} \mid \underline{\mathcal{A} \mathrel{-\!\!*} \mathcal{C}\ (\mathtt{R1})} \\
\mathcal{C} &:= \pi \mid \mathcal{C} \wedge \mathcal{C} \mid \neg \mathcal{C} \mid \exists \mathtt{u}\ \mathcal{C} \mid \mathcal{C} * \mathcal{C} \\
\mathcal{A} &:= \pi \mid \underline{\mathtt{reach}^+(v_1, v_2)} \mid \mathcal{A} \wedge \mathcal{A} \mid \neg \mathcal{A} \mid \exists \mathtt{u}\ \mathcal{A} \mid \mathcal{A} * \mathcal{A}
\end{aligned}$$

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

Test Formulae

1. Design an equivalence relation on models, based on the satisfaction of atomic predicates (test formulae $\mathrm{Test}(\mathtt{X}, \alpha)$)
2. **Show that any formula of our logic is equivalent to a Boolean combination of test formulae** $\mathrm{Bool}(\mathrm{Test}(\mathtt{X}, \alpha))$
3. Prove small-model property for $\mathrm{Bool}(\mathrm{Test}(\mathtt{X}, \alpha))$

Designing Test Formulae

- Fix $\mathtt{X} \subseteq_{\mathsf{fin}} \mathtt{VAR}$ and let $\alpha \in \mathbb{N}^+$
- $\mathsf{Test}(\mathtt{X}, \alpha)$ finite set of predicates, e.g.
  - $(s, h) \models \#\mathtt{loops}(2) \geq \beta$ iff h has at least β loops of size 2
  - $(s, h) \models \mathtt{size} \geq \gamma$ iff $\mathrm{dom}(h)$ has at least γ locations not inside loops of size 2

  where $\beta \in [1, \mathcal{L}(\alpha)]$ and $\gamma \in [1, \mathcal{G}(\alpha)]$
- $\mathcal{L}$ and $\mathcal{G}$ are functions in $[\mathbb{N} \to \mathbb{N}]$

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

for every $T \in \mathsf{Test}(\mathtt{X}, \alpha)$, $(s, h) \models T$ iff $(s', h') \models T$

Note: φ of SL will be translated into a formula of $\mathsf{Bool}(\mathsf{Test}(\mathtt{X}, \alpha))$ with $\mathtt{X} = \mathsf{fv}(\varphi)$ and α roughly the number of $*$ and $\mathrel{-\!\!*}$ in a φ.

$*$ elimination Lemma

We want to design $\mathrm{Test}(\mathtt{X}, \alpha)$ so that the following result holds

Hypothesis:

- $(s, h) \approx^{\mathtt{X}}_{\alpha} (s', h')$
- $\alpha_1, \alpha_2 \in \mathbb{N}^+$ s.t. $\alpha_1 + \alpha_2 = \alpha$
- $h_1 + h_2 = h$

Thesis: there are h'_1, h'_2 s.t.

- $h'_1 + h'_2 = h'$
- $(s, h_1) \approx^{\mathtt{X}}_{\alpha_1} (s', h'_1)$
- $(s, h_2) \approx^{\mathtt{X}}_{\alpha_2} (s', h'_2)$

Note: it can be restated as an EF-style game. Spoiler split α and h, Duplicator has to mimic the split on h' so that $\approx$ still holds.

$*$ elimination Lemma

We want to design $\mathrm{Test}(\mathtt{X}, \alpha)$ so that the following result holds

Hypothesis:

- $(s, h) \approx^{\mathtt{X}}_{\alpha} (s', h')$
- α
- h

Thesis

- h'
- $(s$
- $(s$

$$\mathrm{Test}(\mathtt{X}, \alpha) \supseteq \left\{ \begin{array}{r|l} \#\mathtt{loops}(2) \geq \beta, & \beta \in [1, \mathcal{L}(\alpha)] \\ \mathtt{size} \geq \gamma & \gamma \in [1, \mathcal{G}(\alpha)] \end{array} \right\}$$

find $\mathcal{L}(\alpha)$ and $\mathcal{G}(\alpha)$ so that lemma holds.

Note: it can be restated as an EF-style game. Spoiler split α and h, Duplicator has to mimic the split on h' so that $\approx$ still holds.

Finding $\mathcal{G}$ for $\texttt{size} \geq \gamma$ formulae

Given $h = h_1 + h_2$, every location not in a loop of size 2 of h cannot be in a loop of size 2 of h_1 or h_2. Then $\mathcal{G}$ must satisfy

$$\mathcal{G}(\alpha) \geq \max_{\substack{\alpha_1,\alpha_2 \in \mathbb{N}^+ \\ \alpha_1+\alpha_2=\alpha}} (\mathcal{G}(\alpha_1) + \mathcal{G}(\alpha_2))$$

Finding $\mathcal{L}$ for $\#\texttt{loops}(2) \geq \beta$ formulae

Take $h = h_1 + h_2$. Given a loop of size 2 of h, two cases:

- both locations of the loop are in the same heap (h_1 or h_2);
- one location of the loop is in h_1 and the other is in h_2.

$$\mathcal{L}(\alpha) \geq \max_{\substack{\alpha_1,\alpha_2 \in \mathbb{N}^+ \\ \alpha_1+\alpha_2=\alpha}} (\mathcal{L}(\alpha_1) + \mathcal{L}(\alpha_2) + \mathcal{G}(\alpha_1) + \mathcal{G}(\alpha_2))$$

Finding $\mathcal{L}$ and $\mathcal{G}$

We have the inequalities

$$\mathcal{G}(1) \geq 1 \qquad \mathcal{G}(\alpha) \geq \max_{\substack{\alpha_1,\alpha_2 \in \mathbb{N}^+ \\ \alpha_1+\alpha_2=\alpha}} (\mathcal{G}(\alpha_1) + \mathcal{G}(\alpha_2))$$

$$\mathcal{L}(1) \geq 1 \qquad \mathcal{L}(\alpha) \geq \max_{\substack{\alpha_1,\alpha_2 \in \mathbb{N}^+ \\ \alpha_1+\alpha_2=\alpha}} (\mathcal{L}(\alpha_1) + \mathcal{L}(\alpha_2) + \mathcal{G}(\alpha_1) + \mathcal{G}(\alpha_2))$$

Which admit $\mathcal{G}(\alpha) = \alpha$ and $\mathcal{L}(\alpha) = \frac{1}{2}\alpha(\alpha + 3) - 1$ as a solution.

An indistinguishability relation built on the set

$$\left\{ \begin{matrix} \#\texttt{loops}(2) \geq \beta, \\ \texttt{size} \geq \gamma \end{matrix} \;\middle|\; \begin{matrix} \beta \in \left[1, \dfrac{1}{2}\alpha(n+3) - 1\right] \\ \gamma \in [1, \alpha] \end{matrix} \right\}$$

satisfy the $*$ elimination Lemma.

$1\mathrm{SL}^{\mathtt{R2}}_{\mathtt{R1}}(*, -\!\!*, \mathtt{reach}^+)$, not so easy

- Find the right set of test formulae that capture the logic
- Asymmetric $\mathcal{A} -\!\!* \mathcal{C}$.
  - two indistinguishability relation, two sets of test formulae
  - $-\!\!*$ elimination Lemma that works on these relations
- $\exists$ elimination Lemma

If you like bounds: test formulae for $\mathcal{A}$

$$\left\{\begin{array}{l|r}
v_1 = v_2,\ \mathtt{sees}_{\mathtt{X}}(v_1, v_2) \geq \beta^{\lightning} & \\
\#\mathtt{loop}_{\mathtt{X}}(\beta) \geq \beta^{\circlearrowleft},\ \#\mathtt{loop}^{\Uparrow}_{\mathtt{X}} \geq \beta^{\circlearrowleft} & \beta^{\lightning} \in \left[1, \tfrac{1}{6}(\alpha+1)(\alpha+2)(\alpha+3)\right] \\
\#\mathtt{pred}^{\mathcal{A}}_{\mathtt{X}}(\mathtt{x}) \geq \beta,\ \mathtt{size}^{\mathcal{A}}_{\mathtt{X}} \geq \beta & \beta^{\circlearrowleft} \in \left[1, \tfrac{1}{2}\alpha(\alpha+3)-1\right],\ \beta \in [1, \alpha] \\
\mathtt{u} \in \mathtt{sees}_{\mathtt{X}}(v_1, v_2) \geq (\overleftarrow{\beta}, \overrightarrow{\beta}) & \overleftarrow{\beta} \in \left[1, \tfrac{1}{6}\alpha(\alpha+1)(\alpha+2)+1\right] \\
\mathtt{u} = v_1,\ \mathtt{u} \in \mathtt{loop}_{\mathtt{X}}(\beta),\ \mathtt{u} \in \mathtt{loop}^{\Uparrow}_{\mathtt{X}} & \overrightarrow{\beta} \in \left[1, \tfrac{1}{2}\alpha(\alpha+3)\right] \\
\mathtt{u} \in \mathtt{pred}^{\mathcal{A}}_{\mathtt{X}}(\mathtt{x}),\ \mathtt{u} \in \mathtt{size}^{\mathcal{A}}_{\mathtt{X}} & \mathtt{x} \in \mathtt{X},\ v_1, v_2 \in \mathcal{A}\mathsf{TERM}_{\mathtt{X}}
\end{array}\right\}$$

Test formulae, after $*, \multimap, \exists$ elimination

Suppose we have two family of test formulae such that

- captures the atomic predicates of $1\mathrm{SL}^{\mathtt{R2}}_{\mathtt{R1}}(*, \multimap, \mathtt{reach}^+)$
- satisfies the $*$, $\multimap$ and $\exists$ elimination Lemmata

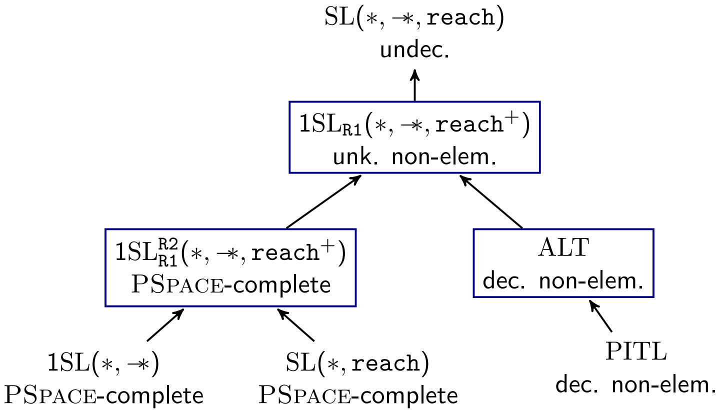
Then, let $\alpha \geq |\varphi|$ and $\mathtt{X} \supseteq \mathsf{fv}(\varphi)$

- If $(s, h) \approx^{\mathtt{X}}_{\alpha} (s, h')$ then we have $(s, h) \models \varphi$ iff $(s, h') \models \varphi$.
- φ is equivalent to a boolean combination of test formulae.

Small-model property

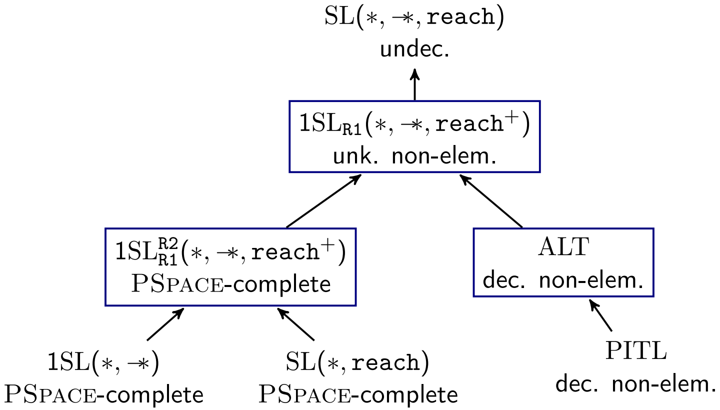
1. Small-model property for boolean combination of test formulae carries over to $1\mathrm{SL}^{\mathtt{R2}}_{\mathtt{R1}}(*, \multimap, \mathtt{reach}^+)$.
2. All bounds are polynomial $\implies$ test formulae in $\mathrm{PS}\textsc{pace}$
3. $1\mathrm{SL}^{\mathtt{R2}}_{\mathtt{R1}}(*, \multimap, \mathtt{reach}^+)$ is $\mathrm{PS}\textsc{pace}$-complete

Recap



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

Recap



- 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. $\mathrm{QCTL}(\mathtt{U})$, $M\mathrm{SL}(\Diamond, \langle \mathrm{U} \rangle, *)$, $2\mathrm{SL}(*)$