Internal calculi for Separation Logics

Stéphane Demri[1] Étienne Lozes[2] **Alessio Mansutti**[1]

September 9, 2020

[1]LSV, CNRS, ENS Paris-Saclay

[2]I3S, Université Côte d'Azur

What we will see...

Elements of Separation Logic

- A modular Hoare-calculus for program with pointers;
- Basic results from the literature.

First axiomatisation of Separation Logic

- Full suite of operators: $*$, $-\!*$ and Boolean connectives;
- Internal: axioms/rules are made of formulae of the logic;
- Based on normalisation of non-classical connectives.

Program verification of 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"

Program verification of 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* are of the form

$$\{\ \varphi\ \}\ \mathtt{PROG}\ \{\ \psi\ \}; \quad \text{to be read as:}$$

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

Floyd-Hoare proof systems

Proofs are sequences of (instantiated) *inference rules*, e.g.

$$\frac{\{\ \varphi\ \}\ \texttt{PROG}_1\ \{\ \gamma\ \} \qquad \{\ \gamma\ \}\ \texttt{PROG}_2\ \{\ \psi\ \}}{\{\ \varphi\ \}\ \texttt{PROG}_1; \texttt{PROG}_2\ \{\ \psi\ \}}$$

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

$$\frac{\{\ \varphi\ \}\ \texttt{PROG}\ \{\ \psi\ \} \qquad \mathsf{modv}(\texttt{PROG}) \cap \mathsf{fv}(\boldsymbol{\gamma}) = \emptyset}{\{\ \varphi \wedge \boldsymbol{\gamma}\ \}\ \texttt{PROG}\ \{\ \psi \wedge \boldsymbol{\gamma}\ \}}$$

Floyd-Hoare proof systems

Proofs are sequences of (instantiated) *inference rules*, e.g.

$$\frac{\{\ \varphi\ \}\ \texttt{PROG}_1\ \{\ \gamma\ \} \qquad \{\ \gamma\ \}\ \texttt{PROG}_2\ \{\ \psi\ \}}{\{\ \varphi\ \}\ \texttt{PROG}_1 ; \texttt{PROG}_2\ \{\ \psi\ \}}$$

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

$$\boxed{\frac{\{\ \varphi\ \}\ \texttt{PROG}\ \{\ \psi\ \} \qquad \mathrm{modv}(\texttt{PROG}) \cap \mathrm{fv}(\boldsymbol{\gamma}) = \emptyset}{\{\ \varphi \wedge \boldsymbol{\gamma}\ \}\ \texttt{PROG}\ \{\ \psi \wedge \boldsymbol{\gamma}\ \}}}$$

Modular verification and pointers

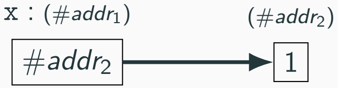
$$\frac{\{\ \varphi\ \}\ \texttt{PROG}\ \{\ \psi\ \} \quad \mathsf{modv}(\texttt{PROG}) \cap \mathsf{fv}(\boldsymbol{\gamma}) = \emptyset}{\{\ \varphi \wedge \boldsymbol{\gamma}\ \}\ \texttt{PROG}\ \{\ \psi \wedge \boldsymbol{\gamma}\ \}}$$

is not valid when modelling 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.

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



Separation Logic (J. Reynolds, P. O'Hearn, D. Pym)

- The memory is a resource that can be partitioned.
- The notion of **separation** ($*$) leads to a valid **frame rule**:

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

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

Separation Logic (J. Reynolds, P. O'Hearn, D. Pym)

- The memory is a resource that can be partitioned.
- The notion of **separation** ($*$) leads to a valid **frame rule**:

$$\frac{\{\varphi\}\ \mathtt{PROG}\ \{\psi\} \quad \mathsf{modv}(\mathtt{PROG}) \cap \mathsf{fv}(\gamma) = \emptyset}{\{\varphi * \gamma\}\ \mathtt{PROG}\ \{\psi * \gamma\}}$$

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

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

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

Separation Logic neighbourhood

'99 Logic of Bunched Implication (D. Pym, P. O'Hearn)

'02 Separation Logic

'05 Separation Logic with permissions (R. Bornat et al.)

'07 Concurrent Separation Logic (P. O'Hearn et al.)

'09 (Bi-abduction for Separation Logic) (C. Calcagno et al.)

'15 Separation Logic with modalities (S. Demri, M. Deters)

'18 Quantitative Separation Logic (J. Katoen et al.)

'19 Probabilistic Separation Logic (G. Barthe et al.)

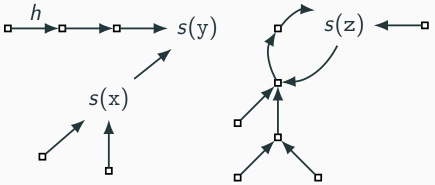
Common feature: model as a resource that can be decomposed.

Modelling the memory

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

- **store**, $s : \mathrm{VAR} \to \mathbb{N}$
- **heap**, $h : \mathbb{N} \to_{\mathsf{fin}} \mathbb{N}$

where $\mathrm{VAR} = \{\mathtt{x}, \mathtt{y}, \mathtt{z}, \ldots\}$ set of variables;
$\mathbb{N}$ represents the set of addresses.



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

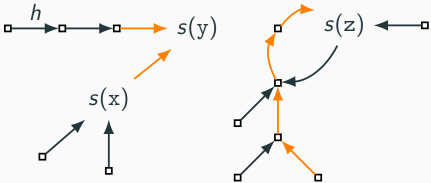
- Disjoint heaps ($h_1 \perp h_2$): $\mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) = \emptyset$
- Union of disjoint heaps ($h_1 + h_2$): union of partial functions.

Modelling the memory

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

- **store**, $s : \mathrm{VAR} \to \mathbb{N}$
- **heap**, $h : \mathbb{N} \to_{\mathsf{fin}} \mathbb{N}$

where $\mathrm{VAR} = \{\mathtt{x}, \mathtt{y}, \mathtt{z}, \dots\}$ set of variables;
$\mathbb{N}$ represents the set of addresses.

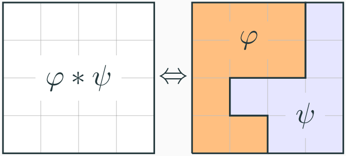


here, $h(s(\mathtt{x})) = s(\mathtt{y})$

- Disjoint heaps ($h_1 \perp h_2$): $\mathrm{dom}(h_1) \cap \mathrm{dom}(h_2) = \emptyset$
- Union of disjoint heaps ($h_1 + h_2$): union of partial functions.

The separating conjunction ($*$)

$$(s, h) \models \varphi * \psi$$



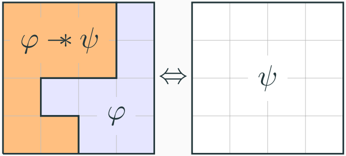
Semantics:

There are two heaps h_1 and h_2 s.t.

- $h_1 \perp h_2$ and $h = h_1 + h_2$;
- $(s, h_1) \models \varphi$;
- $(s, h_2) \models \psi$.

The separating implication ($-\!\!*$)

$$\frac{(s,h) \models \varphi \mathrel{-\!\!*} \psi}{}$$



Semantics:

For every heap h',

if $h' \perp h$ and $(s, h') \models \varphi$

then $(s, h + h') \models \psi$.

Note: $*$ and $-\!\!*$ are adjoint operators:

$$\varphi * \psi \models \gamma \quad \text{if and only if} \quad \varphi \models \psi \mathrel{-\!\!*} \gamma.$$

First-order Separation Logic

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

- $(s, h) \models \texttt{emp}$ iff $\mathrm{dom}(h) = \emptyset$
- $(s, h) \models \texttt{x} = \texttt{y}$ iff $s(\texttt{x}) = s(\texttt{y})$
- $(s, h) \models \texttt{x} \hookrightarrow \texttt{y}$ iff $s(\texttt{x}) \in \mathrm{dom}(h)$ and $h(s(\texttt{x})) = \texttt{y}$
- $(s, h) \models \forall \texttt{x}\, \varphi$ iff for every $n \in \mathbb{N}$, $(s[\texttt{x} \leftarrow n], h) \models \varphi$.

Satisfiability problem: some complexity results.

Tocl'15 $\mathtt{SL}$ with two quantified variables ($2\mathtt{SL}$) is undecidable.
(S. Demri, M. Deters)

Tcs'17 $1\mathtt{SL}$ is PSPACE-complete.
(S. Demri, D. Galmiche, D. Larchey-Wendling, D. Méry)

Fossacs'18 $0\mathtt{SL}$ + reachability predicates is undecidable.
Without $-\!\!*$ it is PSPACE-complete.
(S. Demri, E. Lozes, A. M.)

Fsttcs'18 $1\mathtt{SL}$ + restricted reachability predicate is PSPACE-c.
Weakening restrictions makes it TOWER-hard.

Satisfiability $\approx$ Validity $\approx$ Entailment $\approx$ Model checking

Let $\varphi \mathbin{-\circledast} \psi \stackrel{\text{def}}{=} \neg(\varphi \mathbin{-\!*} \neg\psi)$.

$(s, h) \models \varphi \mathbin{-\circledast} \psi$ iff there is a heap h' s.t. $h' \perp h$

$$(s, h') \models \varphi \text{ and } (s, h+h') \models \psi.$$

In order to check if φ is satisfiable we can

- Consider the set $\mathbb{X}$ of equivalence relations on $\mathsf{fv}(\varphi)$.
- Check if there is $X \in \mathbb{X}$ s.t. the following formula is valid:

$$\big(\mathtt{emp} \wedge \bigwedge_{(\mathtt{x},\mathtt{y}) \in X} \mathtt{x} = \mathtt{y} \wedge \bigwedge_{(\mathtt{x},\mathtt{y}) \notin X} \mathtt{x} \neq \mathtt{y}\big) \Rightarrow (\varphi \mathbin{-\circledast} \top)$$

Undecidability implies non-axiomatisability

Validity R.E. $\rightarrow$ Satisfiability R.E. $\rightarrow$ Unvalidity R.E.
$\rightarrow$ Validity decidable.

Tocl'15 ~~$\mathtt{SL}$ with two quantified variables ($\mathtt{2SL}$) is undecidable.~~

Fossacs'18 ~~$\mathtt{0SL}$ + reachability predicates is undecidable.~~

CSL'20 (S. Demri, E. Lozes, A. M.) internal proof systems for

- Quantifier-free Separation Logic;
- $\mathtt{SL}$ without $\mathrel{-\!\!*}$ and with a guarded form of quantification that can express reachability predicates.

Undecidability implies non-axiomatisability

Validity R.E. $\rightarrow$ Satisfiability R.E. $\rightarrow$ Unvalidity R.E.
$\rightarrow$ Validity decidable.

Tocl'15 ~~$\mathtt{SL}$ with two quantified variables ($\mathtt{2SL}$) is undecidable.~~

Fossacs'18 ~~$\mathtt{0SL}$ + reachability predicates is undecidable.~~

CSL'20 (S. Demri, E. Lozes, A. M.) internal proof systems for

- Quantifier-free Separation Logic;
- $\mathtt{SL}$ without $\mathrel{-\!\!*}$ and with a guarded form of quantification that can express reachability predicates.

On axiomatising 0SL, internally

Methodology:

- Model theoretical analysis of 0SL (Lozes'04);
- We define a "normal form" for formulae of 0SL
  ($*$ and $-\!\!*$ restricted to specific patterns);
- Axiomatisation specific to the formulae in this normal form;
- Add axioms & rules to put every formula in normal form.

What can $\mathtt{OSL}$ express?

- the heap has size at least β:

$$\mathtt{size} \geq 0 \stackrel{\text{def}}{=} \top \qquad\qquad \mathtt{size} \geq \beta{+}1 \stackrel{\text{def}}{=} \mathtt{size} \geq \beta * \neg\mathtt{emp}$$

- $\mathtt{x}$ corresponds to a location in the domain of the heap:

$$\mathtt{alloc}(\mathtt{x}) \stackrel{\text{def}}{=} \neg\big(\mathtt{x} \hookrightarrow \mathtt{x} \mathrel{-\!\!\circledast} \top\big)$$

Let $\mathtt{X} \subseteq_{\mathsf{fin}} \mathsf{VAR}$ and $\alpha \in \mathbb{N}$. We define the set of *core formulae*:

$$\mathtt{Core}(\mathtt{X}, \alpha) \stackrel{\text{def}}{=} \{\mathtt{x} = \mathtt{y},\ \mathtt{x} \hookrightarrow \mathtt{y},\ \mathtt{alloc}(\mathtt{x}),\ \mathtt{size} \geq \beta \mid \mathtt{x}, \mathtt{y} \in \mathtt{X}, \beta \in [0, \alpha]\}.$$

An indistinguishability relation for $\mathtt{OSL}$

$$(s,h) \approx^{\mathtt{X}}_{\alpha} (s',h') \ \text{ iff } \ \forall \varphi \in \mathtt{Core}(\mathtt{X},\alpha), (s,h) \models \varphi \Leftrightarrow (s',h') \models \varphi.$$

An indistinguishability relation for $\mathtt{OSL}$

$$(s,h) \approx^{\mathtt{X}}_{\alpha} (s',h') \text{ iff } \forall \varphi \in \mathtt{Core}(\mathtt{X},\alpha),\ (s,h) \models \varphi \Leftrightarrow (s',h') \models \varphi.$$

$*$ Normalisation Lemma

Let $(s,h) \approx^{\mathtt{X}}_{\alpha} (s',h')$.
$\forall \alpha_1, \alpha_2$ satisfying $\alpha_1 + \alpha_2 = \alpha$, $\forall h_1, h_2$ satisfying $h_1 + h_2 = h$,
$\exists h'_1, h'_2$ such that $(s,h_1) \approx^{\mathtt{X}}_{\alpha_1} (s',h'_1)$ and $(s,h_2) \approx^{\mathtt{X}}_{\alpha_2} (s',h'_2)$.

Similar lemma for $-\!\!*$.

An indistinguishability relation for $\mathtt{OSL}$

$$(s, h) \approx^{\mathtt{X}}_{\alpha} (s', h') \;\text{ iff }\; \forall \varphi \in \mathtt{Core}(\mathtt{X}, \alpha),\ (s, h) \models \varphi \Leftrightarrow (s', h') \models \varphi.$$

$*$ Normalisation Lemma

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

$\forall \alpha_1, \alpha_2$ satisfying $\alpha_1 + \alpha_2 = \alpha$, $\forall h_1, h_2$ satisfying $h_1 + h_2 = h$,

$\exists h'_1, h'_2$ such that $(s, h_1) \approx^{\mathtt{X}}_{\alpha_1} (s', h'_1)$ and $(s, h_2) \approx^{\mathtt{X}}_{\alpha_2} (s', h'_2)$.

This lemma hides a Spoiler/Duplicator EF-games for $\mathtt{OSL}$, and shows the existence of a winning strategy for Duplicator.

For every move of Spoiler, the Duplicator has a winning answer.

An indistinguishability relation for $\mathtt{OSL}$

$$(s, h) \approx_{\alpha}^{\mathtt{X}} (s', h') \;\text{ iff }\; \forall \varphi \in \mathtt{Core}(\mathtt{X}, \alpha),\ (s, h) \models \varphi \Leftrightarrow (s', h') \models \varphi.$$

$*$ Normalisation Lemma

Let $(s, h) \approx_{\alpha}^{\mathtt{X}} (s', h')$.
$\forall \alpha_1, \alpha_2$ satisfying $\alpha_1 + \alpha_2 = \alpha$, $\forall h_1, h_2$ satisfying $h_1 + h_2 = h$,
$\exists h_1', h_2'$ such that $(s, h_1) \approx_{\alpha_1}^{\mathtt{X}} (s', h_1')$ and $(s, h_2) \approx_{\alpha_2}^{\mathtt{X}} (s', h_2')$.

Similar lemma for $-\!\!*$.

A "Gaifman locality theorem" for $\mathtt{OSL}$

Every formula φ in $\mathtt{OSL}$ is logically equivalent to a Boolean combination of core formulae from $\mathtt{Core}(\mathrm{vars}(\varphi), \mathtt{size}(\varphi))$.

Normalising connectives & reasoning on core formulae

Normalisation of $*$ and $\mathrel{-\!\!*}$

$\vdash \psi_1 * \psi_2 \Leftrightarrow \psi_3$

$\vdash \psi_4 \mathrel{-\!\!*} \psi_5 \Leftrightarrow \psi_6$

Completeness for core formulae

$$\frac{\vdash \varphi \Leftrightarrow \psi \qquad \vdash \psi}{\vdash \varphi}$$

where φ in $\mathtt{SL}$, and ψ_i, ψ are in $\bigcup_{\mathtt{X},\alpha} \textbf{Bool}(\mathtt{Core}(\mathtt{X}, \alpha))$.

From a simple calculus for Core formulae...

(PC) propositional calculus;

(R) $\mathtt{x} = \mathtt{x}$

(S) $\varphi \wedge \mathtt{x} = \mathtt{y} \Rightarrow \varphi[\mathtt{y} \leftarrow \mathtt{x}]$

(A) $\mathtt{x} \hookrightarrow \mathtt{y} \Rightarrow \mathtt{alloc}(\mathtt{x})$

(F) $\mathtt{x} \hookrightarrow \mathtt{y} \wedge \mathtt{x} \hookrightarrow \mathtt{z} \Rightarrow \mathtt{y} = \mathtt{z}$

(H1) $\mathtt{size} \geq \beta{+}1 \Rightarrow \mathtt{size} \geq \beta$

(H2) $\bigwedge_{\mathtt{x} \in \mathtt{X}} (\mathtt{alloc}(\mathtt{x}) \wedge \bigwedge_{\mathtt{y} \in \mathtt{X} \setminus \{\mathtt{x}\}} \mathtt{x} \neq \mathtt{y}) \Rightarrow \mathtt{size} \geq \mathrm{card}(\mathtt{X})$, where $\mathtt{X} \subseteq_{\mathsf{fin}}$ VAR.

$\mathtt{CoreTypes}(\mathtt{X}, \alpha)$: set of *complete*[1] conjunctions of formulae in $\mathtt{Core}(\mathtt{X}, \mathrm{card}(\mathtt{X}) + \alpha)$.

Lemma
Let $\varphi \in \mathtt{CoreTypes}(\mathtt{X}, \alpha)$. We have $\neg\varphi$ is valid iff $\vdash \neg\varphi$.

[1] Every $\varphi \in \mathtt{Core}(\mathtt{X}, \mathrm{card}(\mathtt{X}) + \alpha)$ appears in a literal of the conjunction.

From a simple calculus for Core formulae...

(PC) propositional calculus;

(R) $\mathtt{x} = \mathtt{x}$

(S) $\varphi \wedge \mathtt{x} = \mathtt{y} \Rightarrow \varphi[\mathtt{y} \leftarrow \mathtt{x}]$

(A) $\mathtt{x} \hookrightarrow \mathtt{y} \Rightarrow \mathtt{alloc}(\mathtt{x})$

(F) $\mathtt{x} \hookrightarrow \mathtt{y} \wedge \mathtt{x} \hookrightarrow \mathtt{z} \Rightarrow \mathtt{y} = \mathtt{z}$

(H1) $\mathtt{size} \geq \beta{+}1 \Rightarrow \mathtt{size} \geq \beta$

(H2) $\bigwedge_{\mathtt{x} \in \mathtt{X}} (\mathtt{alloc}(\mathtt{x}) \wedge \bigwedge_{\mathtt{y} \in \mathtt{X} \setminus \{\mathtt{x}\}} \mathtt{x} \neq \mathtt{y}) \Rightarrow \mathtt{size} \geq \mathrm{card}(\mathtt{X})$, where $\mathtt{X} \subseteq_{\mathsf{fin}} \mathsf{VAR}$.

$\mathtt{CoreTypes}(\mathtt{X}, \alpha)$: set of *complete*[1] conjunctions of formulae in $\mathtt{Core}(\mathtt{X}, \mathrm{card}(\mathtt{X}) + \alpha)$.

Lemma

A Boolean combination of core formulae φ is valid iff $\vdash \varphi$.

[1] Every $\varphi \in \mathtt{Core}(\mathtt{X}, \mathrm{card}(\mathtt{X}) + \alpha)$ appears in a literal of the conjunction.

...to the proof system of $\mathtt{OSL}$

(M) $\mathtt{alloc}(\mathtt{x}) * \top \Rightarrow \mathtt{alloc}(\mathtt{x})$

(N) $\neg\mathtt{alloc}(\mathtt{x}) * \neg\mathtt{alloc}(\mathtt{x}) \Rightarrow \neg\mathtt{alloc}(\mathtt{x})$

(I) $\mathtt{alloc}(\mathtt{x}) \Rightarrow (\mathtt{alloc}(\mathtt{x}) \wedge \mathtt{size} = 1) * \top$

$$\frac{\varphi \Rightarrow \gamma}{\varphi * \psi \Rightarrow \gamma * \psi}$$

Lemma
$\forall \varphi, \psi {\in} \mathtt{CoreTypes}(\mathtt{X}, \alpha)\ \exists \gamma {\in} \mathsf{Bool}(\mathtt{Core}(\mathtt{X}, 2\alpha))$ s.t. $\vdash \varphi * \psi \Leftrightarrow \gamma$.

(P) $\neg\mathtt{alloc}(\mathtt{x}) \Rightarrow ((\mathtt{x} \hookrightarrow \mathtt{y} \wedge \mathtt{size} = 1) \mathrel{-\!\circledast} \top)$

$$\frac{\varphi * \psi \Rightarrow \gamma}{\varphi \Rightarrow (\psi \mathrel{-\!\!*} \gamma)}$$

Lemma
$\forall \varphi, \psi {\in} \mathtt{CoreTypes}(\mathtt{X}, \alpha)\ \exists \gamma {\in} \mathsf{Bool}(\mathtt{Core}(\mathtt{X}, \alpha))$ s.t. $\vdash (\varphi \mathrel{-\!\circledast} \psi) \Leftrightarrow \gamma$.

A sound and complete axiomatisation of $\mathtt{OSL}$

Normalisation of $*$ and $\mathrel{-\!\!*}$

$\vdash \psi_1 * \psi_2 \Leftrightarrow \psi_3$

$\vdash \psi_4 \mathrel{-\!\!*} \psi_5 \Leftrightarrow \psi_6$

Completeness for core formulae

$$\frac{\vdash \varphi \Leftrightarrow \psi \qquad\qquad \vdash \psi}{\vdash \varphi}$$

where φ in $\mathtt{SL}$, and ψ_i, ψ are in **Bool**$(\bigcup_{\mathtt{X},\alpha} \mathtt{Core}(\mathtt{X}, \alpha))$.

Recap of the method

1. Study the expressivity of the logic from a semantical perspective;
2. Define the corresponding set of *core formulae*;
3. Axiomatise the logic of Boolean combinations of core formulae;
4. Add axioms & rules to transform every formula into a Boolean combination of core formulae.

The same methodology has been used to axiomatise a Separation Logic featuring a guarded form of quantification (CSL'20), as well as two Modal Separation Logics (Jelia'19 – See Stéphane's talk).