Reasoning with Separation Logic

Complexity, Expressive Power, Proof Systems

Alessio Mansutti

10 December 2020

CNRS, LSV, ENS Paris-Saclay, Université Paris-Saclay

Verifying industrial software

- millions of lines of code
- written by hundreds of programmer
- changes daily

Verifying industrial software

- millions of lines of code
- written by hundreds of programmer
- changes daily

Verification should be automatic and modular

"A single instruction should have a local effect in the syntactical proof"

Verifying industrial software

- millions of lines of code
- written by hundreds of programmer
- changes daily

Verification should be automatic and modular

Separation Logic [O'Hearn, Pym, Reynolds et al. – '01]

Modular reasoning for **pointer programs**.

Modularity in the absence of side effects

$$\{\mathrm{x} = 0 \wedge \mathrm{y} = 0\} \qquad \mathrm{x} \leftarrow 1 \qquad \{\mathrm{x} = 1 \wedge \mathrm{y} = 0\}$$

Modularity in the absence of side effects

$$\underbrace{\{\mathrm{x} = 0 \wedge \mathrm{y} = 0\}}_{\text{precondition}} \qquad \mathrm{x} \leftarrow 1 \qquad \{\mathrm{x} = 1 \wedge \mathrm{y} = 0\}$$

	$\cdots$
x:	0
y:	0
	$\cdots$

Modularity in the absence of side effects

$$\{x = 0 \wedge y = 0\} \qquad \underbrace{x \leftarrow 1}_{\text{program}} \qquad \{x = 1 \wedge y = 0\}$$

$$\begin{array}{r|c|}
\cline{2-2} & \cdots \\
\cline{2-2} x\text{:} & 1 \\
\cline{2-2} y\text{:} & 0 \\
\cline{2-2} & \cdots \\
\cline{2-2}
\end{array}$$

Modularity in the absence of side effects

$$\{x = 0 \land y = 0\} \qquad x \leftarrow 1 \qquad \underbrace{\{x = 1 \land y = 0\}}_{\text{postcondition}}$$

	$\cdots$
x:	1
y:	0
	$\cdots$

Modularity in the absence of side effects

$$\{\mathrm{x} = 0 \land \mathrm{y} = 0\} \qquad \mathrm{x} \leftarrow 1 \qquad \{\mathrm{x} = 1 \land \mathrm{y} = 0\}$$

Modularity

$$\frac{\{\varphi\}\ P\ \{\gamma\} \quad \mathsf{fv}(\psi) \cap \mathsf{mv}(P) = \emptyset}{\{\varphi \land \psi\}\ P\ \{\gamma \land \psi\}}$$

- $\mathsf{fv}(\psi)$: free variables of ψ
- $\mathsf{mv}(P)$: variables modified by P

Modularity in the absence of side effects

$$\frac{\{\mathtt{x} = 0\} \quad \mathtt{x} \leftarrow 1 \quad \{\mathtt{x} = 1\}}{\{\mathtt{x} = 0 \land \mathtt{y} = 0\} \quad \mathtt{x} \leftarrow 1 \quad \{\mathtt{x} = 1 \land \mathtt{y} = 0\}}$$

Modularity

$$\frac{\{\varphi\}\ P\ \{\gamma\} \quad \mathsf{fv}(\psi) \cap \mathsf{mv}(P) = \emptyset}{\{\varphi \land \psi\}\ P\ \{\gamma \land \psi\}}$$

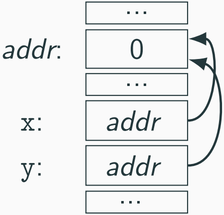
- $\mathsf{fv}(\psi)$: free variables of ψ
- $\mathsf{mv}(P)$: variables modified by P

Modularity rule fails with pointers

$$\frac{\overline{\{\mathtt{x} \hookrightarrow 0\} \quad {}^*\mathtt{x} \leftarrow 1 \quad \{\mathtt{x} \hookrightarrow 1\}}}{\{\mathtt{x} \hookrightarrow 0 \land \mathtt{y} \hookrightarrow 0\} \quad {}^*\mathtt{x} \leftarrow 1 \quad \{\mathtt{x} \hookrightarrow 1 \land \mathtt{y} \hookrightarrow 0\}}$$

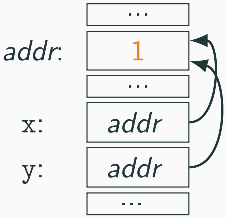
Modularity rule fails with pointers

$$\dfrac{\dfrac{}{\{\mathtt{x} \hookrightarrow 0\} \quad {}^{*}\mathtt{x} \leftarrow 1 \quad \{\mathtt{x} \hookrightarrow 1\}}}{\underbrace{\{\mathtt{x} \hookrightarrow 0 \land \mathtt{y} \hookrightarrow 0\}}_{\text{precondition}} \quad {}^{*}\mathtt{x} \leftarrow 1 \quad \{\mathtt{x} \hookrightarrow 1 \land \mathtt{y} \hookrightarrow 0\}}$$



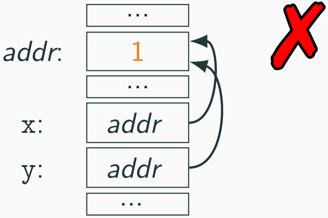
Modularity rule fails with pointers

$$\dfrac{\dfrac{}{\{x \hookrightarrow 0\} \quad {}^*x \leftarrow 1 \quad \{x \hookrightarrow 1\}}}{\{x \hookrightarrow 0 \land y \hookrightarrow 0\} \quad \underbrace{{}^*x \leftarrow 1}_{\text{program}} \quad \{x \hookrightarrow 1 \land y \hookrightarrow 0\}}$$



Modularity rule fails with pointers

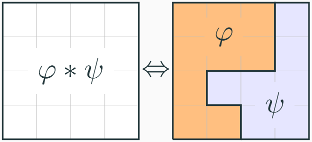
$$\dfrac{\dfrac{}{\{x \hookrightarrow 0\} \qquad {}^{*}x \leftarrow 1 \qquad \{x \hookrightarrow 1\}}}{\{x \hookrightarrow 0 \land y \hookrightarrow 0\} \qquad {}^{*}x \leftarrow 1 \qquad \underbrace{\{x \hookrightarrow 1 \land y \hookrightarrow 0\}}_{\text{postcondition}}}$$



Modular analysis of the memory

Assertion language

‘99 Bunched Logics
[Pym, O’Hearn]

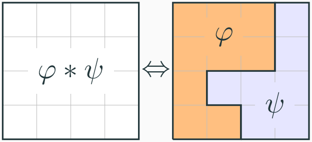


Hoare calculus

Modular analysis of the memory

Assertion language

'99 Bunched Logics
[Pym, O'Hearn]



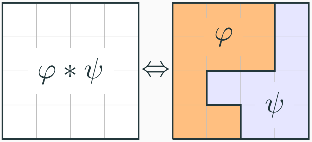
$$(\mathrm{x} \hookrightarrow 0 * \mathrm{y} \hookrightarrow 0) \Rightarrow \mathrm{x} \neq \mathrm{y}$$

Hoare calculus

Modular analysis of the memory

Assertion language

'99 Bunched Logics
[Pym, O'Hearn]



$$(\mathrm{x} \hookrightarrow 0 * \mathrm{y} \hookrightarrow 0) \Rightarrow \mathrm{x} \neq \mathrm{y}$$

Hoare calculus

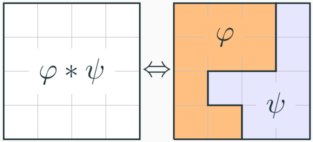
Frame rule:

$$\frac{\{\varphi\}\, P\, \{\gamma\} \quad \mathsf{fv}(\psi) \cap \mathsf{mv}(P) = \emptyset}{\{\varphi * \psi\}\, P\, \{\gamma * \psi\}}$$

Modular analysis of the memory

Assertion language

‘99 Bunched Logics
[Pym, O’Hearn]



$$(\mathrm{x} \hookrightarrow 0 * \mathrm{y} \hookrightarrow 0) \Rightarrow \mathrm{x} \neq \mathrm{y}$$

Hoare calculus

Frame rule:

$$\frac{\{\varphi\}\, P\, \{\gamma\} \quad \mathsf{fv}(\psi) \cap \mathsf{mv}(P) = \emptyset}{\{\varphi * \psi\}\, P\, \{\gamma * \psi\}}$$

$$\frac{\overline{\{\mathrm{x} \hookrightarrow 0\} \quad {}^*\mathrm{x} \leftarrow 1 \quad \{\mathrm{x} \hookrightarrow 1\}}}{\{\mathrm{x} \hookrightarrow 0 * \mathrm{y} \hookrightarrow 0\} \quad {}^*\mathrm{x} \leftarrow 1 \quad \{\mathrm{x} \hookrightarrow 1 * \mathrm{y} \hookrightarrow 0\}}$$

This Thesis

Reachability in separation logic (Chapters 3, 4 & 5)

Goal: A separation logic for acyclicity and garbage freedom

This Thesis

Reachability in separation logic (Chapters 3, 4 & 5)

Goal: A separation logic for acyclicity and garbage freedom

Internal calculi for spatial logics (Chapters 6 & 7)

Goal: First Hilbert-style proof system for separation logic

This Thesis

Reachability in separation logic (Chapters 3, 4 & 5)

Goal: A separation logic for acyclicity and garbage freedom

Internal calculi for spatial logics (Chapters 6 & 7)

Goal: First Hilbert-style proof system for separation logic

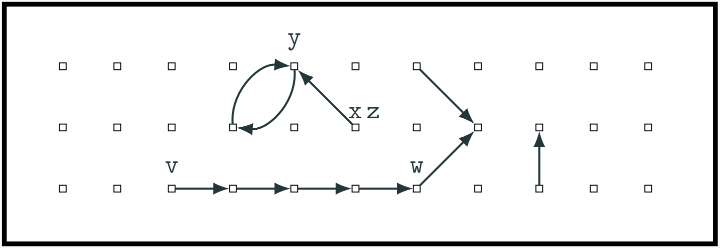
Comparing composition operators (Chapters 8 & 9)

Goal: Differences between $*$ of SL and $|$ of ambient logic

Memory states

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

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

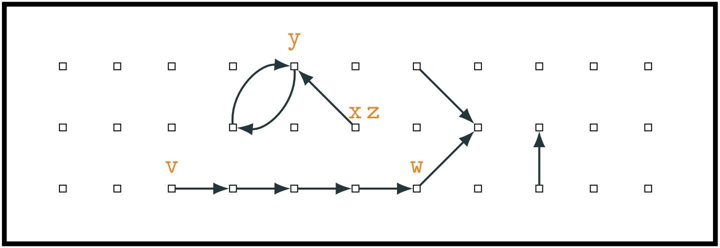


Memory states

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

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

$\{\mathtt{x}, \mathtt{y}, \mathtt{z}, \ldots\}$

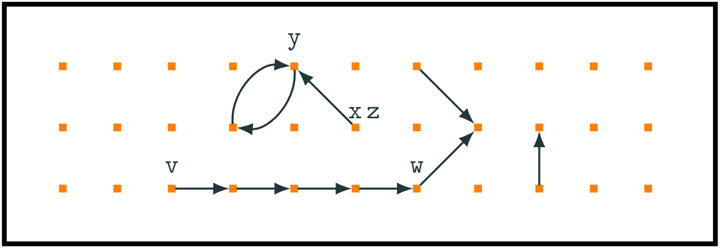


Memory states

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

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

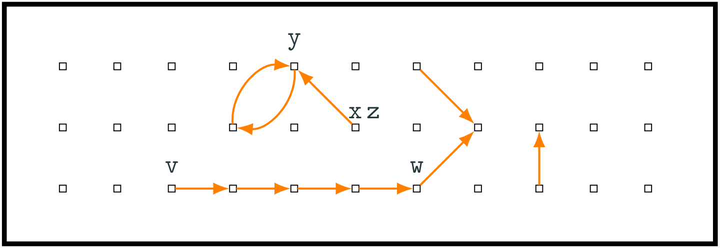
$\{\ell, \ell_1, \ell_2, \ldots\}$



Memory states

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

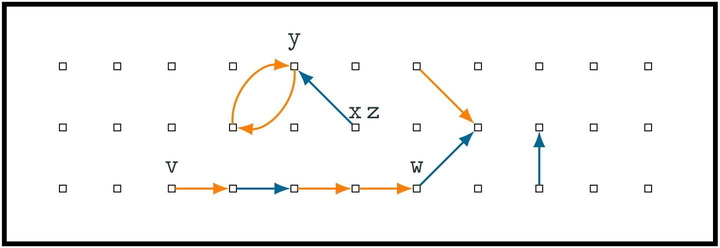
- **store**, $s : \mathrm{VAR} \to \mathrm{LOC}$
- **heap**, $h : \mathrm{LOC} \to_{\mathrm{fin}} \mathrm{LOC}$



Memory states

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

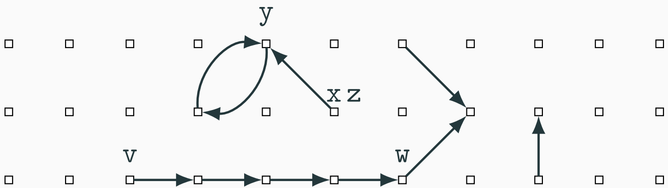
- **store**, $s : \mathrm{VAR} \rightarrow \mathrm{LOC}$
- **heap**, $h : \mathrm{LOC} \rightarrow_{\mathrm{fin}} \mathrm{LOC}$



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

SL($*, -\!\!*$): Quantifier-free separation logic

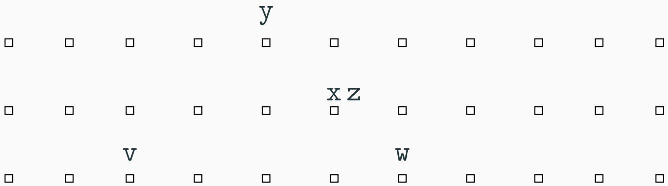
$$\varphi := \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathtt{emp} \mid \mathtt{x} = \mathtt{y} \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \varphi_1 * \varphi_2 \mid \varphi_1 -\!\!* \varphi_2$$



SL($*$, $-\!\!*$): Quantifier-free separation logic

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

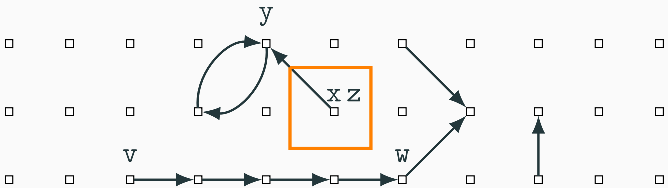
$\mathrm{dom}(h) = \emptyset$



SL($*$, $-\!\!*$): Quantifier-free separation logic

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

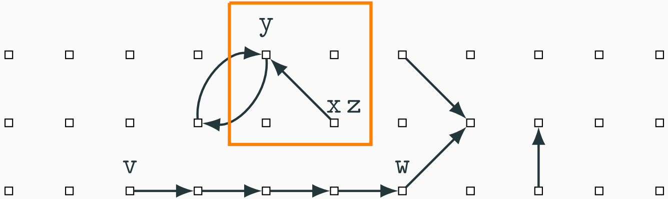
$s(\mathtt{x}) = s(\mathtt{y})$



SL($*, -\!\!*$): Quantifier-free separation logic

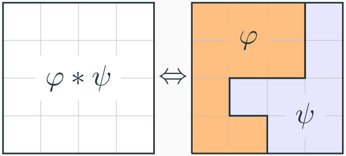
$$\varphi := \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathtt{emp} \mid \mathtt{x} = \mathtt{y} \mid \mathtt{x} \hookrightarrow \mathtt{y} \mid \varphi_1 * \varphi_2 \mid \varphi_1 \mathrel{-\!\!*} \varphi_2$$

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



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

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



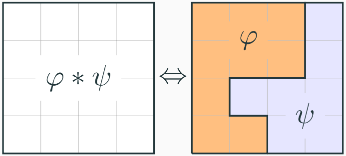
There are two heaps h_1, h_2 s.t.

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

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

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

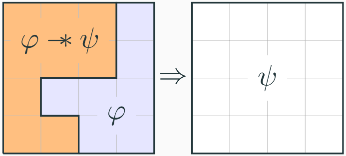
$$(s, h) \models \varphi * \psi$$



There are two heaps h_1, h_2 s.t.

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

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



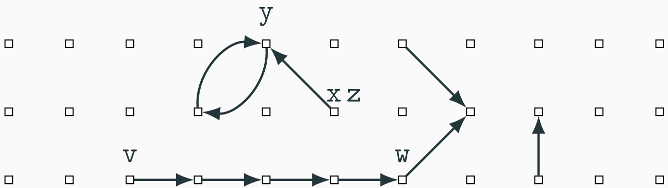
For every heap h',

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

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

First-order quantification and reachability

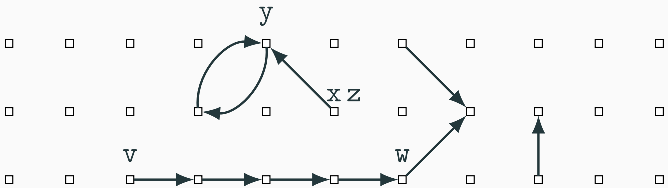
- $(s, h) \models \exists \mathtt{x}\, \varphi \quad \Leftrightarrow \quad (s[\mathtt{x} \leftarrow \ell], h) \models \varphi$ for some $\ell \in \mathsf{LOC}$
- $(s, h) \models \mathtt{reach}^{+}(\mathtt{x}, \mathtt{y}) \Leftrightarrow \quad h^{n}(s(\mathtt{x})) = s(\mathtt{y})$ for some $n \geq 1$



$$\exists \mathtt{x}\ \mathtt{reach}^{+}(\mathtt{x}, \mathtt{x})$$

First-order quantification and reachability

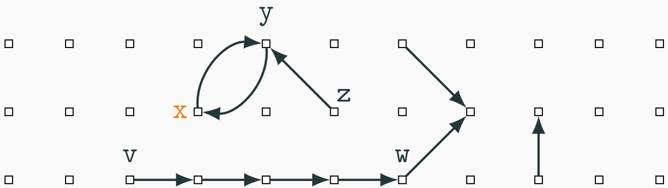
- $(s, h) \models \exists \mathtt{x}\, \varphi \quad \Leftrightarrow \quad (s[\mathtt{x} \leftarrow \ell], h) \models \varphi$ for some $\ell \in \mathrm{LOC}$
- $(s, h) \models \mathtt{reach}^+(\mathtt{x}, \mathtt{y}) \Leftrightarrow \; h^n(s(\mathtt{x})) = s(\mathtt{y})$ for some $n \geq 1$



$$\exists \mathtt{x}\ \mathtt{reach}^+(\mathtt{x}, \mathtt{x})$$

First-order quantification and reachability

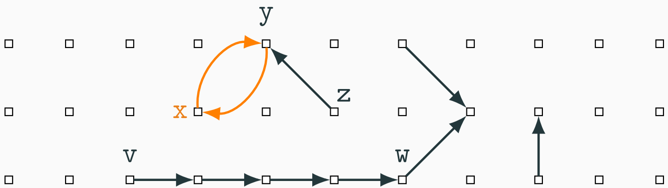
- $(s, h) \models \exists \mathtt{x}\, \varphi \iff (s[\mathtt{x} \leftarrow \ell], h) \models \varphi$ for some $\ell \in \mathrm{LOC}$
- $(s, h) \models \mathtt{reach}^+(\mathtt{x}, \mathtt{y}) \iff h^n(s(\mathtt{x})) = s(\mathtt{y})$ for some $n \geq 1$



$$\exists \mathtt{x}\ \mathtt{reach}^+(\mathtt{x}, \mathtt{x})$$

First-order quantification and reachability

- $(s, h) \models \exists \mathtt{x}\, \varphi \iff (s[\mathtt{x} \leftarrow \ell], h) \models \varphi$ for some $\ell \in \mathrm{LOC}$
- $(s, h) \models \mathtt{reach}^{+}(\mathtt{x}, \mathtt{y}) \iff h^{n}(s(\mathtt{x})) = s(\mathtt{y})$ for some $n \geq 1$



$$\exists \mathtt{x}\ \mathtt{reach}^{+}(\mathtt{x}, \mathtt{x})$$

Reachability in separation logic (Chapters 3, 4 & 5)

Goal: A separation logic for acyclicity and garbage freedom

Internal calculi for spatial logics (Chapters 6 & 7)

Goal: First Hilbert-style proof system for separation logic

Comparing composition operators (Chapters 8 & 9)

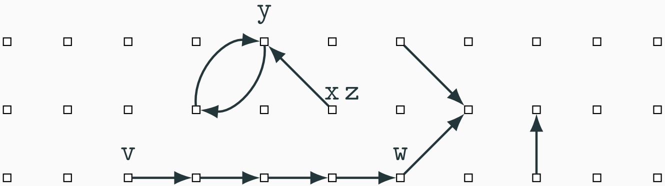
Goal: Differences between $*$ of SL and $|$ of ambient logic

Robustness Properties [Jansen et al. – ESOP'17]

Set of properties that are important for a wide range of reasoning tasks in automated program analysis

Acyclicity:

input:	an assertion φ
question:	is every model of φ acyclic?

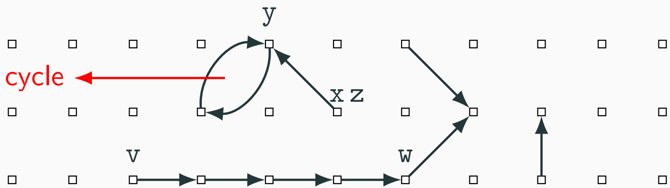


Set of properties that are important for a wide range of reasoning tasks in automated program analysis

Acyclicity:

input: an assertion φ

question: is every model of φ acyclic?



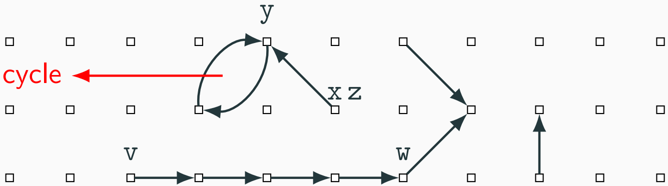
Robustness Properties [Jansen et al. – ESOP'17]

Set of properties that are important for a wide range of reasoning tasks in automated program analysis

Garbage freedom:

input: an assertion φ

question: in every (s, h) satisfying φ, is every location in $\mathrm{dom}(h)$ reached by a variable in $\mathsf{fv}(\varphi)$?

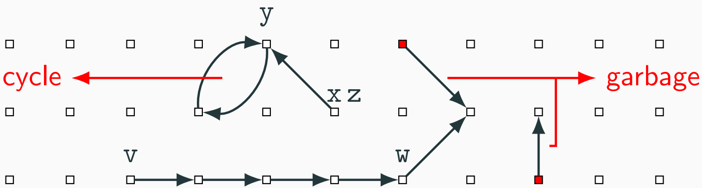


Set of properties that are important for a wide range of reasoning tasks in automated program analysis

Garbage freedom:

input: an assertion φ

question: in every (s, h) satisfying φ, is every location in $\mathrm{dom}(h)$ reached by a variable in $\mathsf{fv}(\varphi)$?



Extending separation logic for robustness properties

Goal (Chapters 3, 4 & 5)

Find an extension of quantifier-free separation logic

- with good decidability status
- where robustness properties reduce to **entailment**

Extending separation logic for robustness properties

Goal (Chapters 3, 4 & 5)

Find an extension of quantifier-free separation logic

- with good decidability status
- where robustness properties reduce to **entailment**

Acyclicity:

input:	an assertion φ
question:	does $\varphi \models \neg\exists \mathtt{u}\, \mathtt{reach}^{+}(\mathtt{u},\mathtt{u})$ hold?

Extending separation logic for robustness properties

Goal (Chapters 3, 4 & 5)

Find an extension of quantifier-free separation logic

- with good decidability status
- where robustness properties reduce to **entailment**

Garbage freedom:

input: an assertion φ (with $\mathtt{u} \notin \mathsf{fv}(\varphi)$)

question: does $\varphi \models \forall \mathtt{u} \left(\mathtt{alloc}(\mathtt{u}) \Rightarrow \bigvee_{\mathtt{x} \in \mathsf{fv}(\varphi)} \mathtt{reach}(\mathtt{x}, \mathtt{u})\right)$ hold?

Goal (Chapters 3, 4 & 5)

Find an extension of quantifier-free separation logic

- with good decidability status
- where robustness properties reduce to **entailment**

Garbage freedom:

input: an assertion φ (with $\mathtt{u} \notin \mathsf{fv}(\varphi)$)

question: does $\varphi \models \forall \mathtt{u} \left(\underbrace{\mathtt{alloc}(\mathtt{u})}_{(\mathtt{u} \hookrightarrow \mathtt{u}) \mathrel{-\!\!*} \bot} \Rightarrow \bigvee_{\mathtt{x} \in \mathsf{fv}(\varphi)} \mathtt{reach}(\mathtt{x}, \mathtt{u})\right)$ hold?

Extending separation logic for robustness properties

Goal (Chapters 3, 4 & 5)

Find an extension of quantifier-free separation logic

- with good decidability status
- where robustness properties reduce to **entailment**

Garbage freedom:

input: an assertion φ (with $\mathtt{u} \notin \mathsf{fv}(\varphi)$)

question: does $\varphi \models \forall \mathtt{u} \left(\mathtt{alloc}(\mathtt{u}) \Rightarrow \bigvee_{\mathtt{x} \in \mathsf{fv}(\varphi)} \mathtt{reach}(\mathtt{x}, \mathtt{u})\right)$ hold?

$\mathtt{x} = \mathtt{u} \vee \mathtt{reach}^{+}(\mathtt{x}, \mathtt{u})$

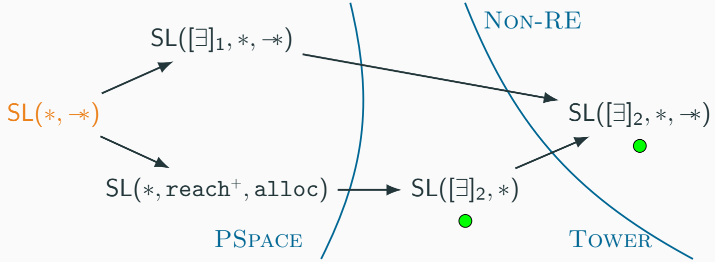
Extending separation logic for robustness properties

Goal (Chapters 3, 4 & 5)

Find an extension of quantifier-free separation logic

- with good decidability status
- where robustness properties reduce to entailment

Known extensions:



First try: $\mathsf{SL}(*, \mathrel{-\!\!*})$ + $\mathtt{reach}$

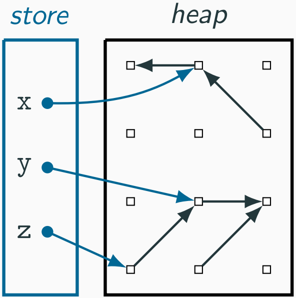
Theorem [Fossacs'18]

The satisfiability (+ model checking, entailment, validity) problem for $\mathsf{SL}(*, \mathrel{-\!\!*}, \mathtt{reach})$ is $\textsc{Non-RE}$.

Insight: $\mathtt{reach}$, $*$ and $\mathrel{-\!\!*}$ simulate first-order quantification.

Theorem [Fossacs'18]
The satisfiability (+ model checking, entailment, validity) problem for SL($*, -\!\!*, \mathtt{reach}$) is $\textsc{Non-RE}$.

Insight: `reach`, $*$ and $-\!\!*$ simulate first-order quantification.

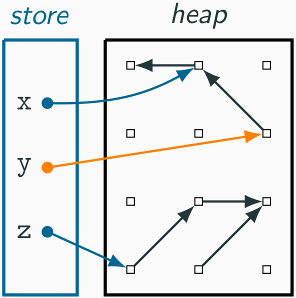


$\exists y\; y \hookrightarrow x$

First try: SL($*, -\!*$) + $\texttt{reach}$

Theorem [Fossacs'18]
The satisfiability (+ model checking, entailment, validity) problem for SL($*, -\!*, \texttt{reach}$) is $\textsc{Non-RE}$.

Insight: $\texttt{reach}$, $*$ and $-\!*$ simulate first-order quantification.

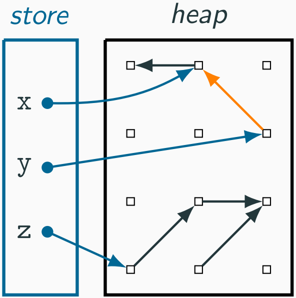


$\exists y\; y \hookrightarrow x$

First try: SL($*, \mathrel{-\!\!*}$) + $\mathtt{reach}$

Theorem [Fossacs'18]
The satisfiability (+ model checking, entailment, validity) problem for SL($*, \mathrel{-\!\!*}, \mathtt{reach}$) is $\textsc{Non-RE}$.

Insight: $\mathtt{reach}$, $*$ and $\mathrel{-\!\!*}$ simulate first-order quantification.

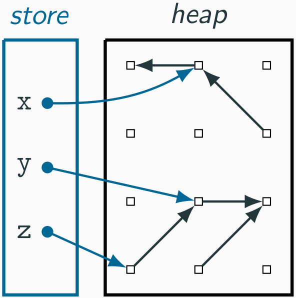


$\exists \mathtt{y}\ \mathtt{y} \hookrightarrow \mathtt{x}$

First try: SL($*$, $-\!\!*$) + $\mathtt{reach}$

Theorem [Fossacs'18]
The satisfiability (+ model checking, entailment, validity) problem for SL($*, -\!\!*, \mathtt{reach}$) is $\textsc{Non-RE}$.

Insight: $\mathtt{reach}$, $*$ and $-\!\!*$ simulate first-order quantification.

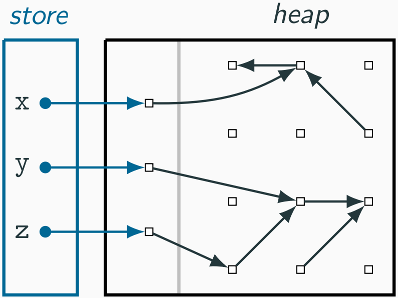


$\exists \mathrm{y}\; \mathrm{y} \hookrightarrow \mathrm{x}$

First try: SL($*, \mathrel{-\!\!*}$) + $\texttt{reach}$

Theorem [Fossacs'18]
The satisfiability (+ model checking, entailment, validity) problem for SL($*, \mathrel{-\!\!*}, \texttt{reach}$) is $\textsc{Non-RE}$.

Insight: $\texttt{reach}$, $*$ and $\mathrel{-\!\!*}$ simulate first-order quantification.

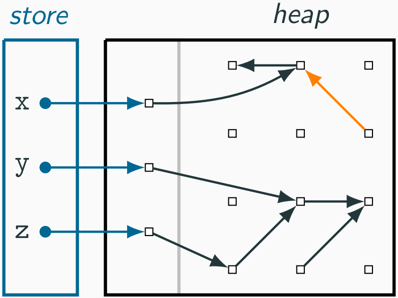


$$\exists \mathtt{y}\ \mathtt{y} \hookrightarrow \mathtt{x}$$

First try: SL($*, -\!*$) + $\mathtt{reach}$

Theorem [Fossacs'18]
The satisfiability (+ model checking, entailment, validity) problem for SL($*, -\!*, \mathtt{reach}$) is $\textsc{Non-RE}$.

Insight: $\mathtt{reach}$, $*$ and $-\!*$ simulate first-order quantification.



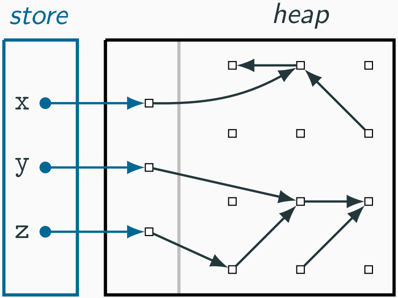
def. with reach

$\exists y\ h(y) \hookrightarrow h(x)$

First try: SL($*, \mathrel{-\!\!*}$) + `reach`

Theorem [Fossacs'18]
The satisfiability (+ model checking, entailment, validity) problem for SL($*, \mathrel{-\!\!*}$, `reach`) is NON-RE.

Insight: `reach`, $*$ and $\mathrel{-\!\!*}$ simulate first-order quantification.

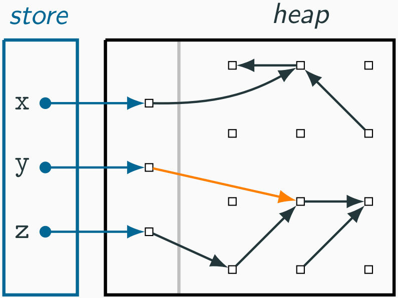


$$(\mathtt{y} \mapsto \_) * \neg\Big((\mathtt{y} \mapsto \_) \mathrel{-\!\!*} \neg h(\mathtt{y}) \hookrightarrow h(\mathtt{x})\Big)$$

First try: $\mathsf{SL}(*, \mathrel{-\!\!*})$ + `reach`

Theorem [Fossacs'18]
The satisfiability (+ model checking, entailment, validity) problem for $\mathsf{SL}(*, \mathrel{-\!\!*}, \texttt{reach})$ is $\textsc{Non-RE}$.

Insight: `reach`, $*$ and $\mathrel{-\!\!*}$ simulate first-order quantification.



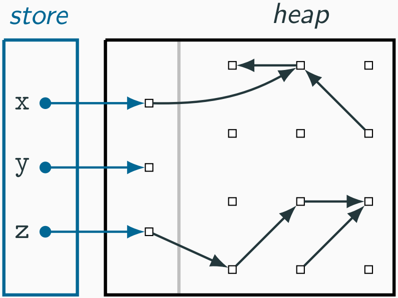
$\mathrm{dom}(h') = \{s(\mathrm{y})\}$

$$(\mathrm{y} \mapsto \_)* \neg\Big((\mathrm{y} \mapsto \_) \mathrel{-\!\!*} \neg h(\mathrm{y}) \hookrightarrow h(\mathrm{x})\Big)$$

First try: $\mathsf{SL}(*, -\!\!*) + \mathtt{reach}$

Theorem [Fossacs'18]
The satisfiability (+ model checking, entailment, validity) problem for $\mathsf{SL}(*, -\!\!*, \mathtt{reach})$ is $\textsc{Non-RE}$.

Insight: $\mathtt{reach}$, $*$ and $-\!\!*$ simulate first-order quantification.

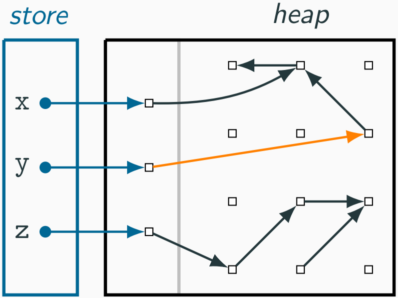


$$(\mathrm{y} \mapsto \_)*$$
$$\neg\Big((\mathrm{y} \mapsto \_) -\!\!*$$
$$\neg h(\mathrm{y}) \hookrightarrow h(\mathrm{x})\Big)$$

First try: $\mathsf{SL}(*, \mathrel{-\!\!*}) + \mathtt{reach}$

Theorem [Fossacs'18]
The satisfiability (+ model checking, entailment, validity) problem for $\mathsf{SL}(*, \mathrel{-\!\!*}, \mathtt{reach})$ is $\textsc{Non-RE}$.

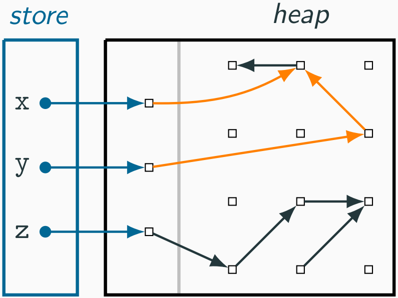
Insight: $\mathtt{reach}$, $*$ and $\mathrel{-\!\!*}$ simulate first-order quantification.



$$
\begin{aligned}
&(\mathrm{y} \mapsto \_)\,* \\
&\quad \neg\Big((\mathrm{y} \mapsto \_) \mathrel{-\!\!*} \\
&\qquad \neg h(\mathrm{y}) \hookrightarrow h(\mathrm{x})\Big)
\end{aligned}
$$

Theorem [Fossacs'18]
The satisfiability (+ model checking, entailment, validity) problem for $\mathsf{SL}(*, \mathrel{-\!\!*}, \mathtt{reach})$ is $\textsc{Non-RE}$.

Insight: $\mathtt{reach}$, $*$ and $\mathrel{-\!\!*}$ simulate first-order quantification.



$$(\mathrm{y} \mapsto \_) * \neg\Big((\mathrm{y} \mapsto \_) \mathrel{-\!\!*} \neg h(\mathrm{y}) \hookrightarrow h(\mathrm{x})\Big)$$

Second try: $\mathsf{SL}([\exists]_1, *, \mathtt{alloc}) + \mathtt{reach}^+$

Theorem [Fossacs'20]
The satisfiability problem for $\mathsf{SL}([\exists]_1, *, \mathtt{alloc}, \mathtt{reach}^+)$ is $\textsc{Tower}$-complete.

Second try: $\mathsf{SL}([\exists]_1, *, \mathtt{alloc})$ + $\mathtt{reach}^+$

Theorem [Fossacs'20]
The satisfiability problem for $\mathsf{SL}([\exists]_1, *, \mathtt{alloc}, \mathtt{reach}^+)$ is $\textsc{Tower}$-complete.

- We introduce and study `ALT` (*Auxiliary Logic on Trees*)

  Reachability predicates + sabotage modalities (e.g. $\blacklozenge\,\varphi$)

Second try: $\mathsf{SL}([\exists]_1, *, \mathtt{alloc}) + \mathtt{reach}^+$

Theorem [Fossacs'20]
The satisfiability problem for $\mathsf{SL}([\exists]_1, *, \mathtt{alloc}, \mathtt{reach}^+)$ is $\textsc{Tower}$-complete.

- We introduce and study `ALT` (*Auxiliary Logic on Trees*)

  Reachability predicates + sabotage modalities (e.g. $\blacklozenge\,\varphi$)

- The satisfiability problem of `ALT` is $\textsc{Tower}$-complete

  $\textsc{Tower}$-hard $\sim$ non-emptiness $*$-free reg. expr.

Second try: $\mathsf{SL}([\exists]_1, *, \mathtt{alloc}) + \mathtt{reach}^+$

Theorem [Fossacs'20]
The satisfiability problem for $\mathsf{SL}([\exists]_1, *, \mathtt{alloc}, \mathtt{reach}^+)$ is $\textsc{Tower}$-complete.

- We introduce and study ALT (*Auxiliary Logic on Trees*)

  Reachability predicates + sabotage modalities (e.g. $\blacklozenge\, \varphi$)

- The satisfiability problem of ALT is $\textsc{Tower}$-complete

  $\textsc{Tower}$-hard $\sim$ non-emptiness $*$-free reg. expr.

- Several logics capture ALT

  $\mathsf{SL}([\exists]_1, *, \mathtt{alloc}, \mathtt{reach}^+)$ QCTL ...

Third try: SL($[\exists]_1, *, \mathrel{-\!\!*}, \mathtt{reach}^+$)

Theorem [Fsttcs'18]

Under syntactical restrictions, SL($[\exists]_1, *, \mathrel{-\!\!*}, \mathtt{reach}^+$) admits a $\mathrm{PS}\small{\mathrm{PACE}}$-complete satisfiability problem.

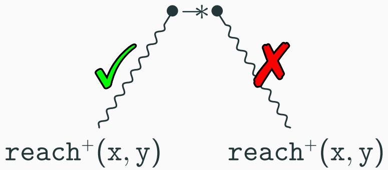
Third try: $\mathsf{SL}([\exists]_1, *, \mathrel{-\!\!*}, \mathtt{reach}^+)$

Theorem [Fsttcs'18]

Under syntactical restrictions, $\mathsf{SL}([\exists]_1, *, \mathrel{-\!\!*}, \mathtt{reach}^+)$ admits a $\mathrm{PS{\small PACE}}$-complete satisfiability problem.

Syntactical restrictions:

No $\mathtt{reach}^+(\mathtt{u}, \mathtt{x})$
($\mathtt{u}$ quantified, $\mathtt{x}$ free)



Third try: SL($[\exists]_1, *, -\!\!*, \texttt{reach}^+$)

Theorem [Fsttcs'18]
Under syntactical restrictions, SL($[\exists]_1, *, -\!\!*, \texttt{reach}^+$) admits a $\mathrm{PS}\textsc{pace}$-complete satisfiability problem.

- Captures the robustness properties
- Extends SL($[\exists]_1, *, -\!\!*$) and SL($*, \texttt{reach}^+, \texttt{alloc}$)

Third try: SL($[\exists]_1, *, -\!\!*, \mathtt{reach}^+$)

Theorem [Fsttcs'18]
Under syntactical restrictions, SL($[\exists]_1, *, -\!\!*, \mathtt{reach}^+$) admits a $\mathrm{PS{\small PACE}}$-complete satisfiability problem.

- Captures the robustness properties
- Extends SL($[\exists]_1, *, -\!\!*$) and SL($*, \mathtt{reach}^+, \mathtt{alloc}$)

Proof

We establish a polynomial **small-heap property** based on the "core formulae technique".

Core formulae

Theorem [M. – Fsttcs'18]
Every $\mathsf{SL}([\exists]_1, *, \mathrel{-\!\!*}, \mathtt{reach}^+)$ formula* is logically equivalent to a Boolean combination of core formulae.

- Bound on the minimal model satisfying the equivalent Boolean combination of core formulae yields $\textsc{PSpace}$

$\star$: satisfying the syntactical conditions

Theorem [Gaifman – 1981]
Every first-order sentence is logically equivalent to a Boolean combination of local formulae.

- proof via Ehrenfeucht-Fraïssé games

Expressiveness		EF-games		Relation between models
$\mathfrak{M} \approx_n \mathfrak{M}'$	$\rightleftarrows$	Duplicator has a winning strategy	$\rightleftarrows$	$\mathfrak{M} \leftrightarrow_n \mathfrak{M}'$
(n nested quantifiers)		(n round game)		(partial iso. up to n)

~~Core~~ Local formulae

Theorem [Gaifman – 1981]
Every first-order sentence is logically equivalent to a Boolean combination of local formulae.

- proof via Ehrenfeucht-Fraïssé games

Expressiveness

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

(n nested quantifiers)

$\rightleftarrows$

EF-games

Duplicator has a winning strategy

(n round game)

$\rightleftarrows$

Relation between models

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

eq.sat. local formulae

Simulation lemmata

Lemma ($*$ simulation)
Let $(s, h) \approx_n^{core} (s', h')$.
For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)
there are $h_1' + h_2' = h'$ such that (Duplicator)

$$(s, h_1) \approx_{n_1}^{core} (s', h_1') \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h_2').$$

Simulation lemmata

Lemma ($*$ simulation)

Let $(s, h) \approx_{n}^{core} (s', h')$.

For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)

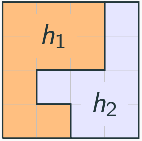
there are $h_1' + h_2' = h'$ such that (Duplicator)

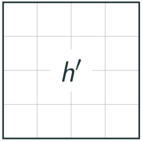
$$(s, h_1) \approx_{n_1}^{core} (s', h_1') \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h_2').$$

Simulation lemmata

Lemma ($*$ simulation)
Let $(s, h) \approx_n^{core} (s', h')$.
For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)
there are $h_1' + h_2' = h'$ such that (Duplicator)

$$(s, h_1) \approx_{n_1}^{core} (s', h_1') \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h_2').$$

$\forall$ 



Simulation lemmata

Lemma ($*$ simulation)

Let $(s, h) \approx_n^{core} (s', h')$.

For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)

there are $h'_1 + h'_2 = h'$ such that (Duplicator)

$$(s, h_1) \approx_{n_1}^{core} (s', h'_1) \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h'_2).$$

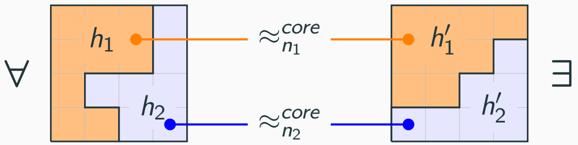
Lemma ($*$ simulation)

Let $(s, h) \approx_n^{core} (s', h')$.

For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)

there are $h'_1 + h'_2 = h'$ such that (Duplicator)

$$(s, h_1) \approx_{n_1}^{core} (s', h'_1) \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h'_2).$$



Lemma ($*$ simulation)
Let $(s, h) \approx_n^{core} (s', h')$.
For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)
there are $h'_1 + h'_2 = h'$ such that (Duplicator)

$$(s, h_1) \approx_{n_1}^{core} (s', h'_1) \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h'_2).$$

Game Hopping

(s, h) (s', h')

Simulation lemmata

Lemma ($*$ simulation)
Let $(s, h) \approx_n^{core} (s', h')$.
For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)
there are $h_1' + h_2' = h'$ such that (Duplicator)

$$(s, h_1) \approx_{n_1}^{core} (s', h_1') \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h_2').$$

Game Hopping

$$(s, h) \;\; \approx_n^{core} (s, h_1) \;\; \approx_n^{core} (s, h_2) \;\; \approx_n^{core} (s, h_3) \;\; \approx_n^{core} \;\; (s', h')$$

Simulation lemmata

Lemma ($*$ simulation)
Let $(s, h) \approx_n^{core} (s', h')$.
For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)
there are $h'_1 + h'_2 = h'$ such that (Duplicator)

$$(s, h_1) \approx_{n_1}^{core} (s', h'_1) \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h'_2).$$

Game Hopping

$$\underbrace{(s, h) \approx_n^{core} (s, h_1)}_{\checkmark} \approx_n^{core} (s, h_2) \approx_n^{core} (s, h_3) \approx_n^{core} (s', h')$$

Simulation lemmata

Lemma ($*$ simulation)
Let $(s, h) \approx_n^{core} (s', h')$.
For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)
there are $h_1' + h_2' = h'$ such that (Duplicator)

$$(s, h_1) \approx_{n_1}^{core} (s', h_1') \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h_2').$$

Game Hopping

$$(s, h) \underbrace{\approx_n^{core} (s, h_1)}_{\checkmark} \underbrace{\approx_n^{core} (s, h_2)}_{\checkmark} \approx_n^{core} (s, h_3) \approx_n^{core} (s', h')$$

Simulation lemmata

Lemma ($*$ simulation)
Let $(s, h) \approx_n^{core} (s', h')$.
For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)
there are $h_1' + h_2' = h'$ such that (Duplicator)

$$(s, h_1) \approx_{n_1}^{core} (s', h_1') \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h_2').$$

Game Hopping

$$(s, h) \approx_n^{core} (s, h_1) \approx_n^{core} (s, h_2) \approx_n^{core} (s, h_3) \approx_n^{core} (s', h')$$

Simulation lemmata

Lemma ($*$ simulation)
Let $(s, h) \approx_n^{core} (s', h')$.
For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)
there are $h_1' + h_2' = h'$ such that (Duplicator)

$$(s, h_1) \approx_{n_1}^{core} (s', h_1') \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h_2').$$

Game Hopping

$$(s, h) \approx_n^{core} (s, h_1) \approx_n^{core} (s, h_2) \approx_n^{core} (s, h_3) \approx_n^{core} (s', h')$$

Simulation lemmata

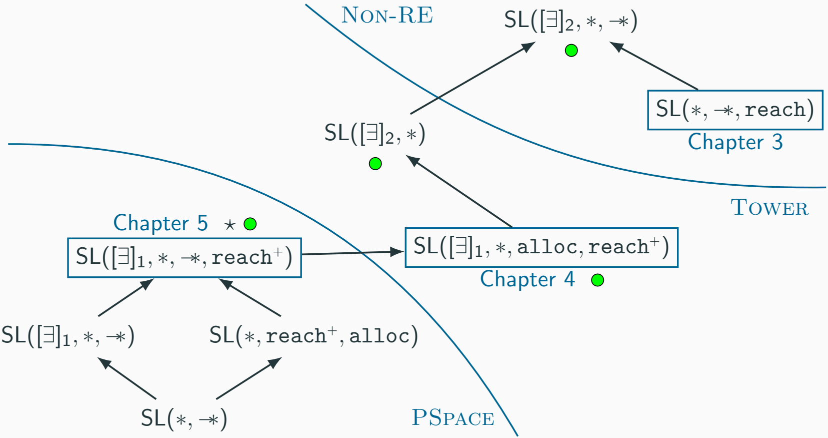
Lemma ($*$ simulation)
Let $(s, h) \approx_n^{core} (s', h')$.
For every $n_1 + n_2 = n$ and every $h_1 + h_2 = h$, (Spoiler)
there are $h'_1 + h'_2 = h'$ such that (Duplicator)

$$(s, h_1) \approx_{n_1}^{core} (s', h'_1) \text{ and } (s, h_2) \approx_{n_2}^{core} (s', h'_2).$$

Game Hopping

$$(s, h) \ \approx_n^{core} (s, h_1) \ \approx_n^{core} (s, h_2) \ \approx_n^{core} (s, h_3) \ \approx_n^{core} \ (s', h')$$

Extending separation logic for robustness properties



⋆: under syntactical constraints ●: expresses robustness properties

Reachability in separation logic (Chapters 3, 4 & 5)

Goal: A separation logic for acyclicity and garbage freedom

Internal calculi for spatial logics (Chapters 6 & 7)

Goal: First Hilbert-style proof system for separation logic

Comparing composition operators (Chapters 8 & 9)

Goal: Differences between $*$ of SL and $|$ of ambient logic

Internal proof systems

- sound and complete $\quad \vdash \varphi \qquad\qquad\qquad \models \varphi$
- All **axioms** and **rules** are made of formulae from the logic

Internal proof systems

- sound and complete $\vdash \varphi \xrightarrow{\text{sound}} \models \varphi$

- All **axioms** and **rules** are made of formulae from the logic

Internal proof systems

- sound and complete $\qquad \vdash \varphi \ \underset{\text{complete}}{\overset{\text{sound}}{\rightleftarrows}} \ \models \varphi$
- All **axioms** and **rules** are made of formulae from the logic

Internal proof systems

- sound and complete $\qquad \vdash \varphi \underset{\text{complete}}{\overset{\text{sound}}{\rightleftarrows}} \models \varphi$

- All **axioms** and **rules** are made of formulae from the logic

Goal (Chapters 6 & 7)

Design Hilbert-style proof systems for spatial logics.

- quantifier-free separation logic SL($*$, $-\!\!*$)
- modal logic with composition operators (ambient logic)

Axiomatising SL($*$, $-\!\!*$) with the core formulae

Theorem [Lozes – 2004]
Every formula of SL($*$, $-\!\!*$) is logically equivalent to a Boolean combination of formulae (in SL($*$, $-\!\!*$)) of the form:

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

$\mathrm{card}(\mathrm{dom}(h)) \geq n$

Can we use these types of theorems to design Hilbert-style proof systems for separation logics?

Axiomatising SL($*, -\!*$) with the core formulae

Theorem [Lozes – 2004]
Every formula of SL($*, -\!*$) is logically equivalent to a Boolean combination of formulae (in SL($*, -\!*$)) of the form:

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

$$\frac{}{\vdash \varphi}$$

Axiomatising SL($*$, $-\!*$) with the core formulae

Theorem [Lozes – 2004]
Every formula of SL($*$, $-\!*$) is logically equivalent to a Boolean combination of formulae (in SL($*$, $-\!*$)) of the form:

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

Normalisation

$\vdash C_1 * C_2 \Leftrightarrow C'$

$\vdash C_1 -\!* C_2 \Leftrightarrow C'$

$$\frac{\vdash \varphi \Leftrightarrow C}{\vdash \varphi}$$

Axiomatising SL($*$, $-\!\!*$) with the core formulae

Theorem [Lozes – 2004]
Every formula of SL($*$, $-\!\!*$) is logically equivalent to a Boolean combination of formulae (in SL($*$, $-\!\!*$)) of the form:

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

Normalisation

$\vdash C_1 * C_2 \Leftrightarrow C'$

$\vdash C_1 -\!\!* C_2 \Leftrightarrow C'$

Completeness for core formulae

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

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

(M) $\mathtt{size} \geq n{+}1 \Rightarrow \mathtt{size} \geq n$

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

Lemma

Given φ Boolean combination of core formulae, $\models \varphi$ iff $\vdash \varphi$.

Proof. Standard countermodel construction.

...to $\mathcal{H}(*, -\!*)$, an adequate proof system for $\mathsf{SL}(*, -\!*)$

(E$_1$) $\mathtt{alloc(x)} * \top \Rightarrow \mathtt{alloc(x)}$

(E$_2$) $\neg\mathtt{alloc(x)} * \neg\mathtt{alloc(x)} \Rightarrow \neg\mathtt{alloc(x)}$

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

(I$_1$) $\mathtt{alloc(x)} \Rightarrow (\mathtt{alloc(x)} \wedge \mathtt{size} = 1) * \top$

(I$_2$) $\neg\mathtt{emp} \Rightarrow \mathtt{size} = 1 * \top$

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

Lemma

For all Boolean combinations of core formulae φ, ψ, there is a Boolean combination of core formulae γ s.t. $\vdash \varphi * \psi \Leftrightarrow \gamma$.

...to $\mathcal{H}(*, -\!\!*)$, an adequate proof system for $\mathsf{SL}(*, -\!\!*)$

(E$_1$) $\mathtt{alloc}(\mathtt{x}) * \top \Rightarrow \mathtt{alloc}(\mathtt{x})$

(E$_2$) $\neg\mathtt{alloc}(\mathtt{x}) * \neg\mathtt{alloc}(\mathtt{x}) \Rightarrow \neg\mathtt{alloc}(\mathtt{x})$

(I$_1$) $\mathtt{alloc}(\mathtt{x}) \Rightarrow (\mathtt{alloc}(\mathtt{x}) \land \mathtt{size} = 1) * \top$

(I$_2$) $\neg\mathtt{emp} \Rightarrow \mathtt{size} = 1 * \top$

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

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

Lemma

For all Boolean combinations of core formulae φ, ψ, there is a Boolean combination of core formulae γ s.t. $\vdash \varphi -\!\!* \psi \Leftrightarrow \gamma$.

...to $\mathcal{H}(*, \mathrel{-\!\!*})$, an adequate proof system for $\mathsf{SL}(*, \mathrel{-\!\!*})$

(E$_1$) $\mathtt{alloc}(\mathtt{x}) * \top \Rightarrow \mathtt{alloc}(\mathtt{x})$

(E$_2$) $\neg\mathtt{alloc}(\mathtt{x}) * \neg\mathtt{alloc}(\mathtt{x}) \Rightarrow \neg\mathtt{alloc}(\mathtt{x})$

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

(I$_1$) $\mathtt{alloc}(\mathtt{x}) \Rightarrow (\mathtt{alloc}(\mathtt{x}) \wedge \mathtt{size} = 1) * \top$

(I$_2$) $\neg\mathtt{emp} \Rightarrow \mathtt{size} = 1 * \top$

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

Lemma

For all Boolean combinations of core formulae φ, ψ, there is a Boolean combination of core formulae γ s.t. $\vdash \varphi \mathrel{-\!\!*} \psi \Leftrightarrow \gamma$.

Theorem [CSL'20]

$\mathcal{H}(*, \mathrel{-\!\!*})$ is sound and complete for $\mathsf{SL}(*, \mathrel{-\!\!*})$.

(E$_1$) $\mathtt{alloc(x)} * \top \Rightarrow \mathtt{alloc(x)}$

(E$_2$) $\neg\mathtt{alloc(x)} * \neg\mathtt{alloc(x)} \Rightarrow \neg\mathtt{alloc(x)}$

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

(I$_1$) $\mathtt{alloc(x)} \Rightarrow (\mathtt{alloc(x)} \land \mathtt{size} = 1) * \top$

(I$_2$) $\neg\mathtt{emp} \Rightarrow \mathtt{size} = 1 * \top$

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

Lemma

For all Boolean combinations of core formulae φ, ψ, there is a

B

Question:

Can we do the same for **ambient logics**?

Theorem [CSL 20]

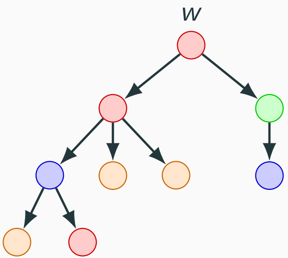
$\mathcal{H}(*, \mathrel{-\!\!*})$ is sound and complete for $\mathsf{SL}(*, \mathrel{-\!\!*})$.

Axiomatising a modal logic with ambient-like composition

Standard ML **+ composition operator** $\varphi|\psi$**:**

$$\varphi \ := \ p \mid \top \mid \varphi \wedge \psi \mid \neg\varphi \mid \Diamond\varphi \mid \varphi|\psi$$

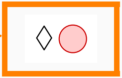
Interpretation on Kripke-style finite forests $(\mathfrak{M}, w)$**:**



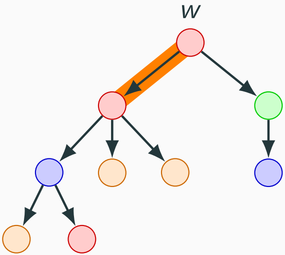
Axiomatising a modal logic with ambient-like composition

Standard ML **+ composition operator** $\varphi \,\boldsymbol{|}\, \psi$**:**

$$\varphi \;:=\; p \mid \top \mid \varphi \wedge \psi \mid \neg\varphi \mid \Diamond\varphi \mid \varphi \,\boldsymbol{|}\, \psi$$



Interpretation on Kripke-style finite forests $(\mathfrak{M}, w)$**:**

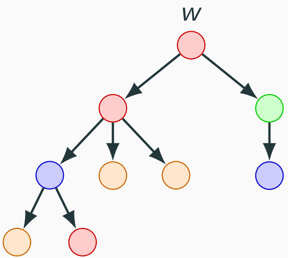


Axiomatising a modal logic with ambient-like composition

Standard ML **+ composition operator** $\varphi | \psi$**:**

$$\varphi \ := \ p \mid \top \mid \varphi \wedge \psi \mid \neg\varphi \mid \Diamond\varphi \mid \varphi | \psi$$

Interpretation on Kripke-style finite forests $(\mathfrak{M}, w)$**:**



$$\frac{(\mathfrak{M}, w) \models \varphi | \psi}{\text{there are } \mathfrak{M}_1,\ \mathfrak{M}_2 \text{ s.t.}}$$

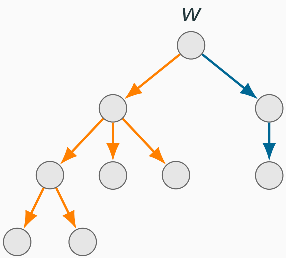
- $\mathfrak{M} = \mathfrak{M}_1 \mid_w \mathfrak{M}_2$
- $\mathfrak{M}_1, w \models \varphi$
- $\mathfrak{M}_2, w \models \psi$

Axiomatising a modal logic with ambient-like composition

Standard ML **+ composition operator** $\varphi|\psi$**:**

$$\varphi \;:=\; p \mid \top \mid \varphi \wedge \psi \mid \neg\varphi \mid \Diamond\varphi \mid \varphi|\psi$$

Interpretation on Kripke-style finite forests $(\mathfrak{M}, w)$**:**



$$\frac{(\mathfrak{M}, w) \models \varphi|\psi}{\text{there are } \mathfrak{M}_1,\ \mathfrak{M}_2 \text{ s.t.}}$$

- $\mathfrak{M} = \mathfrak{M}_1 \mid_w \mathfrak{M}_2$
- $\mathfrak{M}_1, w \models \varphi$
- $\mathfrak{M}_2, w \models \psi$

Axiomatising a modal logic with ambient-like composition

A proof system for $\mathsf{ML}(\mid)$**:**

- $\mathsf{GML} = \mathsf{ML} + \Diamond_{\geq k}\varphi$

$$\mathfrak{M}, w \models \Diamond_{\geq k}\varphi \ \text{ iff } \ w \text{ has} \geq k \text{ children satisfying } \varphi$$

Theorem
For every φ, ψ in GML there is γ in GML such that $\varphi \mid \psi \equiv \gamma$.

- Use GML as a family of core formulae
- Extend proof system of GML to prove $\varphi \mid \psi \equiv \gamma$ syntactically

Two ways to chop a tree

Reachability in separation logic (Chapters 3, 4 & 5)

Goal: A separation logic for acyclicity and garbage freedom

Internal calculi for spatial logics (Chapters 6 & 7)

Goal: First Hilbert-style proof system for separation logic

Comparing composition operators (Chapters 8 & 9)

Goal: Differences between $*$ of SL and $|$ of ambient logic

Separation logic and ambient logic

Separation logic and ambient logic are cousins:

- both instantiate the Bunched Logic BBI
- the first decidability result in ambient logic is based on decidability results for SL [Calcagno et al. – 2003]
- proof systems are surprisingly close (Chapters 6 & 7)

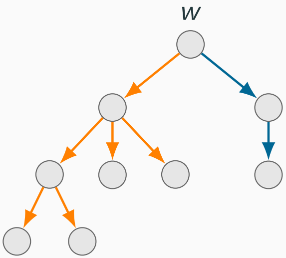
Goal (Chapters 8 & 9)
Build a common framework to compare the composition operators $*$ and $|$ in terms of **expressive power** and **complexity**.

Modal logics with composition operators

Standard ML **+** $\varphi \boldsymbol{|} \psi$ **from ambient logic:**

$$\varphi \ := \ p \mid \top \mid \varphi \wedge \psi \mid \neg\varphi \mid \Diamond\varphi \mid \varphi \boldsymbol{|} \psi$$

Interpretation on Kripke-style finite forests $(\mathfrak{M}, w)$**:**



$$\frac{(\mathfrak{M}, w) \models \varphi \boldsymbol{|} \psi}{\text{there are } \mathfrak{M}_1, \mathfrak{M}_2 \text{ s.t.}}$$

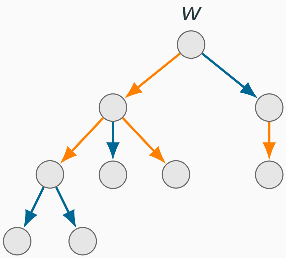
- $\mathfrak{M} = \mathfrak{M}_1 \boldsymbol{|}_w \mathfrak{M}_2$
- $\mathfrak{M}_1, w \models \varphi$
- $\mathfrak{M}_2, w \models \psi$

Modal logics with composition operators

Standard ML + $\varphi * \psi$ **from separation logic**

$$\varphi \ := \ p \mid \top \mid \varphi \wedge \psi \mid \neg\varphi \mid \Diamond\varphi \mid \varphi * \psi$$

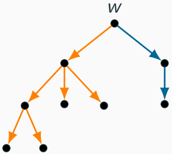
Interpretation on Kripke-style finite forests $(\mathfrak{M}, w)$**:**

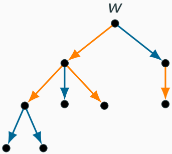


$$\frac{(\mathfrak{M}, w) \models \varphi * \psi}{\text{there are } \mathfrak{M}_1, \mathfrak{M}_2 \text{ s.t.}}$$

- $\mathfrak{M} = \mathfrak{M}_1 + \mathfrak{M}_2$
- $\mathfrak{M}_1, w \models \varphi$
- $\mathfrak{M}_2, w \models \psi$

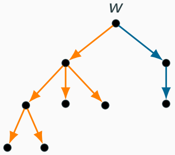
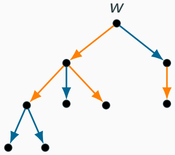
ML($\mid$) vs ML($*$) [Lics'20]



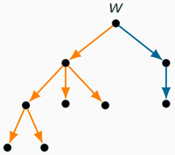
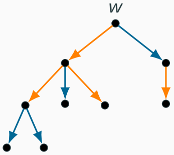


	ML($\mid$)	ML($*$)
Expressive Power	GML	$<$ GML
Complexity (SAT)	$\textsc{AExp}_{\textsc{Poly}}$-complete	$\textsc{Tower}$-complete

ML($|$) **vs** ML($*$) **[Lics'20]**

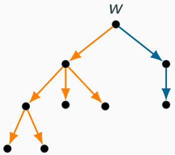
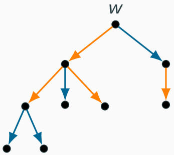
	 ML($|$)	 ML($*$)
Expressive Power	GML	$<$ GML
Complexity (SAT)	$\textsc{AExp}_{\textsc{Poly}}$-complete	$\textsc{Tower}$-complete

ML($|$) vs ML($*$) [Lics'20]

	 ML($|$)	 ML($*$)
Expressive Power	GML	$<$ GML
Complexity (SAT)	$\textsc{AExp}_{\textsc{Poly}}$-complete	$\textsc{Tower}$-complete

$\Diamond_{=2}\Diamond_{=1}\top$ cannot be expressed in ML($*$) (via EF-games)

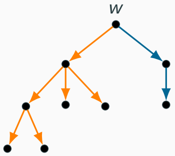
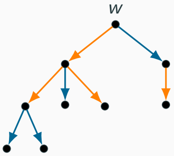
ML(|) vs ML($*$) **[Lics'20]**

	 ML(|)	 ML($*$)
Expressive Power	GML	$<$ GML
Complexity (SAT)	$\textsc{AExp}_{\textsc{Poly}}$-complete	$\textsc{Tower}$-complete

Upper: exponential-size small model property

Lower: from SAT of propositional logic under team semantics

ML($\mid$) **vs** ML($*$) **[Lics'20]**

	 ML($\mid$)	 ML($*$)
Expressive Power	GML	$<$ GML
Complexity (SAT)	$\textsc{AExp}_{\textsc{Poly}}$-complete	$\textsc{Tower}$-complete

Tower-hardness: uniform reduction from k-$\textsc{NExpTime}$ version of the tiling problem, for every $k \geq 2$

Conclusion & Perspectives

Reachability in separation logic (Chapters 3, 4 & 5)

Result: A PSPACE SL for acyclicity and garbage freedom

Internal calculi for spatial logics (Chapters 6 & 7)

Result: First Hilbert-style proof system for SL($\ast, \mathrel{-\!\!\ast}$)

Comparing composition operators (Chapters 8 & 9)

Result: Expressiveness and complexity of ML($|$) and ML($\ast$)

Conclusion & Perspectives

Alternative solutions

- Path quantifiers [CSL'20]
- "strong" $\mathsf{SL}(*, \mathrel{-\!\!*}, \mathtt{ls})$ is in PSPACE [Pagel, Zuleger – '20]

Internal calculi for spatial logics (Chapters 6 & 7)

Result: First Hilbert-style proof system for $\mathsf{SL}(*, \mathrel{-\!\!*})$

Comparing composition operators (Chapters 8 & 9)

Result: Expressiveness and complexity of $\mathsf{ML}(|)$ and $\mathsf{ML}(*)$

Conclusion & Perspectives

Alternative solutions

- Path quantifiers [CSL'20]
- "strong" SL($*, \mathrel{-\!\!*}, \mathtt{ls}$) is in $\textrm{PS}\textsc{pace}$ [Pagel, Zuleger – '20]

Applications

- Improve calculi that are more geared for automation
- Preprocessing of formulae via axiom-based rewriting

Comparing composition operators (Chapters 8 & 9)

Result: Expressiveness and complexity of ML($|$) and ML($*$)

Conclusion & Perspectives

Alternative solutions

- Path quantifiers [CSL'20]
- "strong" $\mathsf{SL}(*, \mathrel{-\!\!*}, \mathtt{ls})$ is in $\textsc{PSpace}$ [Pagel, Zuleger – '20]

Applications

- Improve calculi that are more geared for automation
- Preprocessing of formulae via axiom-based rewriting

Close the gap

$$\begin{array}{ccccccccc} \mathsf{ML}(|) & \Bigg) & & \Bigg) & & \Bigg) & \cdots & \Bigg) & \mathsf{ML}(*) \\ \textsc{AExp}_{\textsc{Poly}} & & 2\textsc{AExp}_{\textsc{Poly}} & & 3\textsc{AExp}_{\textsc{Poly}} & & & & \textsc{Tower} \end{array}$$