Propositional Separation Logic in PSPACE
A classical result

Stéphane Demri

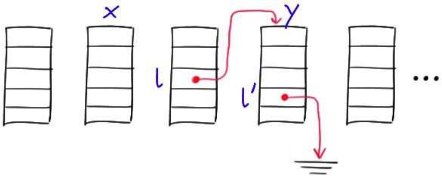
LSV, ENS Cachan, CNRS, INRIA

DYNRES Kick-off Meeting, February 2012

Separation logic

- Introduced by Reynolds, Pym and O'Hearn.
- Reasoning about the heap with a strong form of locality built-in.
- $\mathcal{A} * \mathcal{B}$ is true whenever the heap can be divided into two disjoint parts, one satisfies $\mathcal{A}$, the other one $\mathcal{B}$.
- $\mathcal{A} {-\!\!*} \mathcal{B}$ is true whenever $\mathcal{A}$ is true for a (fresh) disjoint heap, $\mathcal{B}$ is true for the combined heap.

Modelling memory states



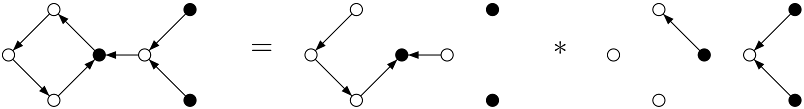
- Set of variables $\mathtt{Var} = \{\mathtt{x}, \mathtt{y}, \mathtt{z}, \ldots\}$.
- Set of selectors/labels $\mathtt{Lab}$.
- Set of values $\mathtt{Val} = \mathbb{N} \uplus \{nil\}$.

- Set of stores: $\mathcal{S} \stackrel{\text{def}}{=} \mathtt{Var} \to \mathtt{Val}$.
- Set of heaps: $\mathcal{H} \stackrel{\text{def}}{=} \mathbb{N} \rightharpoonup_{fin} (\mathtt{Lab} \rightharpoonup_{fin+} \mathtt{Val})$.

Memory state (s, h)

Disjoint heaps

- h_1 and h_2 are disjoint whenever $\mathtt{dom}(h_1) \cap \mathtt{dom}(h_2) = \emptyset$.
  Notation: $h_1 \perp h_2$.
- Disjointness does not concern records.
- Disjoint union $h_1 * h_2$ whenever $h_1 \perp h_2$.
- Disjoint heap graphs (with a unique selector and $\mathtt{Val} = \mathbb{N}$):



Syntax

- Expressions

$$e ::= \mathtt{x} \mid \mathtt{null}$$

- Atomic formulae

$$\pi ::= \ e = e' \mid \mathtt{x} \stackrel{l}{\hookrightarrow} e \mid \mathtt{emp}$$

- $\mathtt{x} \hookrightarrow e_1, e_2$ can be encoded with $\mathtt{x} \stackrel{1}{\hookrightarrow} e_1 \land \mathtt{x} \stackrel{2}{\hookrightarrow} e_2$.

- Formulae:

$$\mathcal{A} ::= \pi \mid \mathcal{A} \land \mathcal{B} \mid \neg\mathcal{A} \mid \mathcal{A} * \mathcal{B} \mid \mathcal{A} {-\!\!*} \mathcal{B}$$

Semantics

- $(s,h) \models \texttt{emp}$ iff $\texttt{dom}(h) = \emptyset$.

- $(s,h) \models e = e'$ iff $[\![\ e\ ]\!]_s = [\![\ e'\ ]\!]_s$, with $[\![\ \mathrm{x}\ ]\!]_s = s(\mathrm{x})$ and $[\![\ \texttt{null}\ ]\!]_s = \mathit{nil}$.

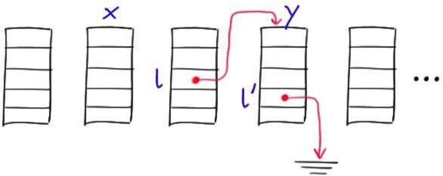
- $(s,h) \models \mathrm{x} \overset{l}{\hookrightarrow} e'$ iff $[\![\ \mathrm{x}\ ]\!]_s \in \mathbb{N}$ and $[\![\ \mathrm{x}\ ]\!] \in \texttt{dom}(h)$ and $h(s(\mathrm{x}))(l) = [\![\ e'\ ]\!]_s$.

- $(s,h) \models \mathcal{A}_1 * \mathcal{A}_2$ iff $\exists\ h_1, h_2$ such that $h = h_1 * h_2$, $(s,h_1) \models \mathcal{A}_1$ and $(s,h_2) \models \mathcal{A}_2$.

- $(s,h) \models \mathcal{A}_1 {-\!\!*} \mathcal{A}_2$ iff for all h', if $h \perp h'$ and $(s,h') \models \mathcal{A}_1$ then $(s,h*h') \models \mathcal{A}_2$.

- + clauses for Boolean operators.

Memory states with arithmetic and records



$x+1 \overset{l}{\hookrightarrow} y \quad h(s(x)+1)(l) = s(y)$

$y \overset{l'}{\hookrightarrow} \texttt{null} \quad h(s(y))(l') = nil$

Simple properties on memory states

- The memory heap has at least two cells ($\texttt{size} \geq 2$):

$$\neg\texttt{emp} * \neg\texttt{emp}$$

- The memory heap has exactly one cell at address $\texttt{x}$ ($\texttt{x} \overset{l}{\mapsto} e$):

$$\texttt{x} \overset{l}{\hookrightarrow} e \land \neg(\texttt{size} \geq 2)$$

- The variable $\texttt{x}$ is allocated in the heap ($\texttt{alloc}(\texttt{x})$):

$$(\texttt{x} \overset{l}{\hookrightarrow} \texttt{null}) {-\!\!*} \bot$$

Model-checking and satisfiability problems

- Satisfiability problem:

  input: A formula $\mathcal{A}$ in SL.
  question: Is there a memory state (s, h) such that $(s, h) \models \mathcal{A}$?

- Model-checking problem:

  input: A formula $\mathcal{A}$ in SL, a memory state (s, h).
  question: $(s, h) \models \mathcal{A}$?

- Standard property: $\mathcal{A}$ is satisfiable iff there is a store s such that $(s, \emptyset) \models \neg(\mathcal{A} -\!\!* \perp)$.

- $\mathcal{A}$ is satisfiable iff there is a s such that $(s, \emptyset) \models \neg(\mathcal{A} -\!\!* \perp)$, and for each $\mathsf{x} \in Y$, $s(\mathsf{x}) \leq (|Y| + 1)$ where Y is the set of variables occuring in $\mathcal{A}$.

On the complexity of SL

- Model-checking, satisfiability and validity for SL are PSPACE-complete problems.
  [Calcagno & Yang & O'Hearn, FSTTCS'01]

- PSPACE upper bound is obtained thanks to a "small memory state property".

- $\mathrm{SL} + \exists$ is undecidable [C. & Y. & O'H., FSTTCS 01]
  with a unique label [Brochenin & Demri & Lozes, I&C 12]

Bounding the syntactic resources

- Test formulae

$$e ::= \mathtt{x} \mid \mathtt{null}$$
$$\mathcal{B} ::= \mathtt{x} \stackrel{l}{\hookrightarrow} e \mid \mathtt{alloc}(\mathtt{x}) \mid e = e' \mid \mathtt{size} \geq k$$

where $k \in \mathbb{N}$, x is a variable and l is a label.

- Measure μ restricts the test formulae

$$\mu = (w_\mu, \mathtt{Lab}_\mu, \mathtt{Var}_\mu) \in \mathbb{N} \times \mathcal{P}_f(\mathtt{Lab}) \times \mathcal{P}_f(\mathtt{Var})$$

- $\mathcal{T}_\mu$: set of test formulae restricted to the resources from the measure ($k < w_\mu$, $l \in \mathtt{Lab}_\mu$, $\mathtt{x} \in \mathtt{Var}_\mu$).

Measure from a formula $\mathcal{A}$

- $\mu_{\mathcal{A}} = (w_{\mathcal{A}}, \mathtt{Lab}_{\mathcal{A}}, \mathtt{Var}_{\mathcal{A}})$
- $\mathtt{Lab}_{\mathcal{A}}$: set of labels in $\mathcal{A}$ (analogous for defining $\mathtt{Var}_{\mathcal{A}}$).
- Definition for $w_{\mathcal{A}}$:
  - $w_{\mathcal{B}} \stackrel{\text{def}}{=} 1$ if $\mathcal{B}$ is atomic.
  - $w_{\mathcal{A}_1 \oplus \mathcal{A}_2} \stackrel{\text{def}}{=} \mathtt{Max}(w_{\mathcal{A}_1}, w_{\mathcal{A}_2})$ for $\oplus \in \{\wedge, -\!\!*, \Rightarrow\}$.
  - $w_{\mathcal{A}_1 * \mathcal{A}_2} \stackrel{\text{def}}{=} w_{\mathcal{A}_1} + w_{\mathcal{A}_2}$
- Cardinal of $\mathcal{T}_{\mu_{\mathcal{A}}}$ is polynomial in the size of $\mathcal{A}$.

Equivalence relation $\simeq_\mu$

- $Abs_\mu(s,h) \stackrel{\text{def}}{=} \{\mathcal{A} \in \mathcal{T}_\mu : (s,h) \models \mathcal{A}\}$.

- $(s,h) \simeq_\mu (s',h') \stackrel{\text{def}}{\Leftrightarrow} Abs_\mu(s,h) = Abs_\mu(s',h')$.
  i.e. formulae in $\mathcal{T}_\mu$ cannot distinguish the two memory states.

- If $(s,h) \simeq_\mu (s',h')$ then for every formula $\mathcal{A}$ with $\mu_\mathcal{A} \leqslant \mu$, we have $(s,h) \models \mathcal{A}$ iff $(s',h') \models \mathcal{A}$.

- As a corollary, every $\mathcal{A}$ is logically equivalent to a Boolean combination of test formulae from $\mathcal{T}_{\mu_\mathcal{A}}$.

$$\mathcal{A} \Leftrightarrow \bigvee_{(s,h)\models\mathcal{A}} \Big(\bigwedge_{\mathcal{B}\in Abs_{\mu_\mathcal{A}}(s,h)} \mathcal{B}\Big) \wedge \Big(\bigwedge_{\mathcal{B}\in \mathcal{T}_{\mu_\mathcal{A}} \setminus Abs_{\mu_\mathcal{A}}(s,h)} \neg\mathcal{B}\Big)$$

[Lozes, PhD 04]

Distributivity Lemma

- Set of measures has a natural lattice structure for the pointwise order.

- Suppose $\mu = \mu_1 + \mu_2$, $(s, h) \simeq_\mu (s', h')$ and $h = h_1 * h_2$.

- Then, there are h'_1 and h'_2 such that
  1. $h' = h'_1 * h'_2$,
  2. $(s, h_1) \simeq_{\mu_1} (s', h'_1)$,
  3. $(s, h_2) \simeq_{\mu_2} (s', h'_2)$.

- Another useful property: if $(s, h) \simeq_\mu (s', h')$, then for all $h_0 \perp h$, there is $h'_0 \perp h'$ s.t. $(s, h_0) \simeq_\mu (s', h'_0)$.

Congruence Lemma

- (s, h_0), (s', h'_0), (s, h_1), (s', h'_1) with $h_0 \bot h_1$, $h'_0 \bot h'_1$.
- Assume $(s, h_0) \simeq_\mu (s', h'_0)$ and $(s, h_1) \simeq_\mu (s', h'_1)$.
- Then, $(s, h_0 * h_1) \simeq_\mu (s', h'_0 * h'_1)$.

Soundness of Abstraction (bis)

- If $(s, h) \simeq_\mu (s', h')$ then for every $\mathcal{A}$ with $\mu_\mathcal{A} \leqslant \mu$, we have $(s, h) \models \mathcal{A}$ iff $(s', h') \models \mathcal{A}$.
- Proof by structural induction. By way of example, we treat the case $\mathcal{A} = \mathcal{A}_1 * \mathcal{A}_2$ and suppose that $(s, h) \models \mathcal{A}$.
- There are h_1 and h_2 s.t. $h = h_1 * h_2$, $(s_1, h_1) \models \mathcal{A}_1$ and $(s_2, h_2) \models \mathcal{A}_2$.
- As $\mu \geqslant \mu_\mathcal{A}$ and $\mu_\mathcal{A} \geqslant \mu_{\mathcal{A}_1} + \mu_{\mathcal{A}_2}$, there are μ_1 and μ_2 such that $\mu_1 \geqslant \mu_{\mathcal{A}_1}$, $\mu_2 \geqslant \mu_{\mathcal{A}_2}$ and $\mu_1 + \mu_2 = \mu$.
- By Distributivity Lemma, there are h'_1 and h'_2 such that
  1. $h' = h'_1 * h'_2$,
  2. $(s, h_1) \simeq_{\mu_1} (s', h'_1)$,
  3. $(s, h_2) \simeq_{\mu_2} (s', h'_2)$.
- By (IH), $(s', h'_1) \models \mathcal{A}_1$ and $(s', h'_2) \models \mathcal{A}_2$, whence $(s', h') \models \mathcal{A}$.

Building small disjoint heaps

- Measure $\mu = (w, \texttt{Lab}_\mu, \texttt{Var}_\mu)$ and $l_0 \notin \texttt{Lab}_\mu$.
- Assume that $(s, h) \simeq_\mu (s', h')$ and $h_0 \perp h$.
- Then, there is h'_0 such that
  - $h'_0 \perp h'$ and $(s, h_0) \simeq_\mu (s', h'_0)$,
  - $\mathrm{card}(\texttt{dom}(h'_0)) \leq \max(w, \mathrm{card}(\texttt{Var}_\mu))$,
  - $\max(\texttt{dom}(h'_0) \cup \mathrm{Im}^2(h'_0)) \leq \max((s'(\texttt{Var}_\mu) \cap \mathbb{N}) \cup \texttt{dom}(h')) + w$,
  - for all $n \in \texttt{dom}(h'_0)$, $\{l : h'_0(n)(l) \text{ is defined}\} \subseteq \texttt{Lab}_\mu \uplus \{l_0\}$.
- h'_0: *small disjoint heap* w.r.t. μ and (s', h').
- h'_0 can be represented in polynomial space in $\mathrm{size}(\mu) + \mathrm{size}_{\texttt{Lab}_\mu}(h_0) + \mathrm{size}_{\texttt{Var}_\mu}(s') + \mathrm{size}_{\texttt{Lab}_\mu}(h')$.

Model-checking problem in PSPACE

$\mathrm{MC}((s,h),\mathcal{A},\mu)$

(base-cases) If $\mathcal{A}$ is atomic, then return $(s,h) \models \mathcal{A}$;

(Boolean-cases) If $\mathcal{A} = \mathcal{A}_1 \wedge \mathcal{A}_2$, then return (MC($(s,h),\mathcal{A}_1,\mu$) and MC($(s,h),\mathcal{A}_2,\mu$));
Other Boolean operators are treated analogously.

($*$ case) If $\mathcal{A} = \mathcal{A}_1 * \mathcal{A}_2$, then return $\perp$ if there are no h_1, h_2 such that $h = h_1 * h_2$ and MC($(s,h_1),\mathcal{A}_1,\mu$) and MC($(s,h_2),\mathcal{A}_2,\mu$));

($-\!\!*$ case) If $\mathcal{A} = \mathcal{A}_1 -\!\!* \mathcal{A}_2$, then return $\perp$ if for some small disjoint heap h' with respect to μ and (s,h) verifying MC($(s,h'),\mathcal{A}_1,\mu$), we have not MC($(s,h*h'),\mathcal{A}_2,\mu$);

Return $\top$;

Ingredients for the PSPACE upper bound

- Recursion depth is linear in $|\mathcal{A}|$.
- Quantifications are over sets of exponential size in $|\mathcal{A}| + \mathrm{size}_{\mathtt{Var}_\mu, \mathtt{Lab}_\mu}((s,h))$ where $\mu_{\mathcal{A}} = (w_{\mathcal{A}}, \mathtt{Lab}_\mu, \mathtt{Var}_\mu)$.
- So, all the heaps considered in the algorithm are of polynomial-size in $|\mathcal{A}| + \mathrm{size}_{\mathtt{Var}_\mu, \mathtt{Lab}_\mu}((s,h))$.

Correctness

- Given $\mathcal{A}$ with $\mu_{\mathcal{A}} \leq \mu$, we show $(s, h) \models \mathcal{A}$ iff MC($(s, h), \mathcal{A}, \mu$) returns $\top$.

- Whenever $(s, h) \not\models \mathcal{A}_1 \twoheadrightarrow \mathcal{A}_2$, there is $h_0 \perp h$ such that $(s, h_0) \models \mathcal{A}_1$ and $(s, h * h_0) \not\models \mathcal{A}_2$.

- We have seen that there is a small disjoint heap h_0' with respect to μ and (s, h) such that $(s, h_0') \simeq_\mu (s, h_0)$.

- Since the measure of $\mathcal{A}_1$ is less than μ, Soundness Lemma implies $(s, h_0') \models \mathcal{A}_1$.

- By Congruence Lemma, $(s, h * h_0') \not\models \mathcal{A}_2$.

- Hence, $(s, h) \not\models \mathcal{A}_1 \twoheadrightarrow \mathcal{A}_2$ iff there is a *small* heap h_0' such that $(s, h_0') \models \mathcal{A}_1$ and $(s, h * h_0') \not\models \mathcal{A}_2$.

Summary

- Model-checking problem is in PSPACE.
- Satisfiability can be reduced to model-checking in logspace.
- $\mathcal{A}$ is satisfiable iff there is (s, h) such that $(s, h) \models \mathcal{A}$, $\mathrm{card}(\mathtt{dom}(h)) \leq \mathrm{size}(\mathcal{A})$ and $\mathrm{ran}(s) \subseteq \{0, \ldots, \mathrm{size}(\mathcal{A})\}$.
- Satisfiability problem is in PSPACE.
- PSPACE-hardness is by reducing QBF.
  [Calcagno & Yang & O'Hearn, FSTTCS 01]

Decidability status of first-order SL with a unique individual variable?

- Formulae:

  $$\mathcal{A} := \neg\mathcal{A} \mid \mathcal{A} \wedge \mathcal{A} \mid \exists \mathsf{x}_{\mathsf{Unique}}\, \mathcal{A} \mid \mathsf{x} \hookrightarrow \mathsf{y} \mid \mathsf{x} = \mathsf{y} \mid \mathcal{A} * \mathcal{A} \mid \mathcal{A} \mathrel{-\!\!*} \mathcal{A}$$

- $(s,h) \models \exists \mathsf{x}_{\mathsf{Unique}}\, \mathcal{A}$ iff there is $l \in \mathbb{N}$ such that $(s[\mathsf{x}_{\mathsf{Unique}} \mapsto l], h) \models \mathcal{A}$.

- What is the right set of test formulae ?

Work in progress

- Suggestions for test formulae (apart from those of SL):
  - $\mathtt{alloc}^{-1}(\mathrm{x}_i)$: $\exists \mathrm{x}_U \; \mathrm{x}_U \hookrightarrow \mathrm{x}_i$
  - $\exists\mathtt{selfloop}$: $\exists \mathrm{x}_U \; \mathrm{x}_U \hookrightarrow \mathrm{x}_U$

  - $\mathtt{toloop}(\mathrm{x}_i)$: $\exists \mathrm{x}_U \; \mathrm{x}_i \hookrightarrow \mathrm{x}_U \wedge \mathrm{x}_U \hookrightarrow \mathrm{x}_U$
  - $\sharp\mathtt{selfloops} \geq k$: $\exists\mathtt{selfloop} * \cdots * \exists\mathtt{selfloop}$ (k times)
  - $\mathtt{inbetween}_1(\mathrm{x}_i, \mathrm{x}_j)$: $\exists \mathrm{x}_U \; \mathrm{x}_i \hookrightarrow \mathrm{x}_U \wedge \mathrm{x}_U \hookrightarrow \mathrm{x}_j$
  - $\mathtt{inbetween}_2(\mathrm{x}_i, \mathrm{x}_j)$: $\exists \mathrm{x}_U \; \mathrm{x}_i \hookrightarrow \mathrm{x}_U \wedge \mathrm{x}_j \hookrightarrow \mathrm{x}_U$
  - $\sharp\mathrm{x}_i \geq k$: $\mathtt{alloc}^{-1}(\mathrm{x}_i) * \cdots * \mathtt{alloc}^{-1}(\mathrm{x}_i)$ (k times)
  - *Le petit dernier* $\mathtt{toalloc}(\mathrm{x}_i)$:

$$\exists \mathrm{x}_U \; \mathrm{x}_i \hookrightarrow \mathrm{x}_U \wedge (\mathrm{x}_U \hookrightarrow \mathtt{null}) \mathrel{-\!\!*} \bot$$

- More atomic formulae? complexity if this works?

- Is it possible to eliminate quantifiers syntactically ?

- Generalization to other types of memory cells ?

- Which other macros defined from $\exists$ can we add while preserving decidability ?

Tasks in DYNRES

- Is first-order SL restricted to one variable decidable?
  (see Task 2.3 “Decidable fragments”)

- Tableaux calculus for SL restricted to one variable, if decidable?
  (see Task 3 “Proof Systems for Separation and Update Logics”)

- Automata-based decision procedures for known decidable fragments of SL?
  (see Task 3.1 “Structures, calculi and automata”)