Fictional Separation Logic: Examples and Intuition

Jonas B. Jensen
joint work with Lars Birkedal

IT University of Copenhagen

December 10, 2013

Separation logic on a slide

- Traditional Hoare logic struggles with aliasing.

$$\{x \mapsto 1 \wedge y \mapsto 2\}\ [x] := 3\ \{x \mapsto 3 \wedge (x \neq y \Rightarrow y \mapsto 2)\}$$

- Separation logic makes non-aliasing of pointers the default.

$$\{x \mapsto 1 * y \mapsto 2\}\ [x] := 3\ \{x \mapsto 3 * y \mapsto 2\}$$

Derived using the *frame rule*

$$\frac{\{P\}\ c\ \{Q\}}{\{P * R\}\ c\ \{Q * R\}}$$

Motivation

Separation logic is about framing out as much as possible

$$\frac{\{P\}\ c\ \{Q\}}{\{P * R\}\ c\ \{Q * R\}}$$

Abstraction is necessary for modularity in large developments

$$\{Stack(\mathsf{s}, \alpha)\}\ \textbf{push}(\mathsf{s}, \mathsf{v})\ \{Stack(\mathsf{s}, \mathsf{v} :: \alpha)\}$$

Abstract predicates make assertions high-level, except for $(*)$.

Copy-on-write collection

Pretty but restrictive spec:

$$\{emp\}\ \textbf{new}()\ \{Coll(\mathsf{ret}, \emptyset)\}$$
$$\{Coll(\mathsf{s}, V)\}\ \textbf{free}(\mathsf{s})\ \{emp\}$$
$$\{Coll(\mathsf{s}, V)\}\ \textbf{contains}(\mathsf{s}, \mathsf{v})\ \{Coll(\mathsf{s}, V) \wedge \mathsf{ret} = (\mathsf{v} \in V)\}$$
$$\{Coll(\mathsf{s}, V)\}\ \textbf{add}(\mathsf{s}, \mathsf{v})\ \{Coll(\mathsf{s}, V \cup \{\mathsf{v}\})\}$$
$$\{Coll(\mathsf{s}, V)\}\ \textbf{remove}(\mathsf{s}, \mathsf{v})\ \{Coll(\mathsf{s}, V \setminus \{\mathsf{v}\})\}$$
$$\{Coll(\mathsf{s}, V)\}\ \textbf{clone}(\mathsf{s})\ \{Coll(\mathsf{s}, V) * Coll(\mathsf{ret}, V)\}$$

Flexible but ugly spec:

$$\{I_{\mathrm{cow}}(\phi)\}\ \textbf{new}()\ \{I_{\mathrm{cow}}(\{(\mathsf{ret}, \emptyset)\} \uplus \phi)\}$$
$$\{I_{\mathrm{cow}}(\{(\mathsf{s}, V)\} \uplus \phi)\}\ \textbf{free}(\mathsf{s})\ \{I_{\mathrm{cow}}(\phi)\}$$
$$\{I_{\mathrm{cow}}(\{(\mathsf{s}, V)\} \uplus \phi)\}\ \textbf{clone}(\mathsf{s})\ \{I_{\mathrm{cow}}(\{(\mathsf{s}, V)\} \uplus \{(\mathsf{ret}, V)\} \uplus \phi)\} \ldots$$

FSL spec: $I_{\mathrm{cow}}.\ \{Coll(\mathsf{s}, V)\}\ \textbf{clone}(\mathsf{s})\ \{Coll(\mathsf{s}, V) * Coll(\mathsf{ret}, V)\}\ \ldots$

Fine-grained collection

Recall the **remove** function:

$$\{\, Coll(\mathsf{s}, V)\} \ \textbf{remove}(\mathsf{s}, \mathsf{v}) \ \{\, Coll(\mathsf{s}, V \setminus \{\mathsf{v}\})\}$$

Alternative but equivalent specification:

$$\{\, Coll(\mathsf{s}, \{\mathsf{v}\} \uplus V_{\in}, V_{\notin})\} \ \textbf{remove}(\mathsf{s}, \mathsf{v}) \ \{\, Coll(\mathsf{s}, V_{\in}, \{\mathsf{v}\} \uplus V_{\notin})\}$$
$$\{\, Coll(\mathsf{s}, V_{\in}, \{\mathsf{v}\} \uplus V_{\notin})\} \ \textbf{remove}(\mathsf{s}, \mathsf{v}) \ \{\, Coll(\mathsf{s}, V_{\in}, \{\mathsf{v}\} \uplus V_{\notin})\}$$

What if $Coll(s, V_{\in}, V_{\notin}) * Coll(s, V'_{\in}, V'_{\notin}) \dashv\vdash Coll(s, V_{\in} \uplus V'_{\in}, V_{\notin} \uplus V'_{\notin})$?

$$I_{\text{fine}}.\ \{\, Coll(\mathsf{s}, \{\mathsf{v}\}, \emptyset)\} \ \textbf{remove}(\mathsf{s}, \mathsf{v}) \ \{\, Coll(\mathsf{s}, \emptyset, \{\mathsf{v}\})\}$$
$$I_{\text{fine}}.\ \{\, Coll(\mathsf{s}, \emptyset, \{\mathsf{v}\})\} \ \textbf{remove}(\mathsf{s}, \mathsf{v}) \ \{\, Coll(\mathsf{s}, \emptyset, \{\mathsf{v}\})\}$$

Or by the disjunction rule,

$$I_{\text{fine}}.\ \{\, Coll(\mathsf{s}, \{\mathsf{v}\}, \emptyset) \vee Coll(\mathsf{s}, \emptyset, \{\mathsf{v}\})\} \ \textbf{remove}(\mathsf{s}, \mathsf{v}) \ \{\, Coll(\mathsf{s}, \emptyset, \{\mathsf{v}\})\}$$

Fine-grained collection

Now define

$$In(s, v) \triangleq Coll(s, \{v\}, \emptyset) \qquad Out(s, v) \triangleq Coll(s, \emptyset, \{v\})$$

and specify

$$I_{\text{fine}}.\ \{emp\}\ \textbf{new}()\ \{\forall_* v : val.\ Out(\mathsf{ret}, v)\}$$
$$I_{\text{fine}}.\ \{\forall_* v : val.\ In(\mathsf{s}, v) \vee Out(\mathsf{s}, v)\}\ \textbf{free}(\mathsf{s})\ \{emp\}$$
$$I_{\text{fine}}.\ \{In(\mathsf{s}, \mathsf{v})\}\ \textbf{contains}(\mathsf{s}, \mathsf{v})\ \{In(\mathsf{s}, \mathsf{v}) \wedge \mathsf{ret} = true\}$$
$$I_{\text{fine}}.\ \{Out(\mathsf{s}, \mathsf{v})\}\ \textbf{contains}(\mathsf{s}, \mathsf{v})\ \{Out(\mathsf{s}, \mathsf{v}) \wedge \mathsf{ret} = false\}$$
$$I_{\text{fine}}.\ \{In(\mathsf{s}, \mathsf{v}) \vee Out(\mathsf{s}, \mathsf{v})\}\ \textbf{add}(\mathsf{s}, \mathsf{v})\ \{In(\mathsf{s}, \mathsf{v})\}$$
$$I_{\text{fine}}.\ \{In(\mathsf{s}, \mathsf{v}) \vee Out(\mathsf{s}, \mathsf{v})\}\ \textbf{remove}(\mathsf{s}, \mathsf{v})\ \{Out(\mathsf{s}, \mathsf{v})\}$$

Fractional permissions: splitting atoms

How should we split the points-to assertion, $p \mapsto v$?
Fractional permissions: $p \overset{z}{\mapsto} v$, where $0 < z \leq 1$!

$$\frac{}{p \overset{z_1}{\mapsto} v * p \overset{z_2}{\mapsto} v \dashv\vdash p \xmapsto{z_1+z_2} v} \qquad \frac{}{p \overset{z_1}{\mapsto} v_1 * p \overset{z_2}{\mapsto} v_2 \vdash v_1 = v_2}$$

$$I_{\text{frac}}.\ \{e \overset{z}{\mapsto} e'\}\ x := [e]\ \{e \overset{z}{\mapsto} e' \wedge x = e'\} \text{ if } x \notin fv(e, e')$$

$$I_{\text{frac}}.\ \{e \overset{1}{\mapsto} \_\}\ [e] := e'\ \{e \overset{1}{\mapsto} e'\}$$

$$I_{\text{frac}}.\ \{emp\}\ x := \mathsf{alloc}\ 1\ \{x \overset{1}{\mapsto} \_\}$$

Free feature: predicates other than points-to can be fractional.

$$I_{\text{fracColl}}.\ \{Coll^z(\mathsf{s}, V)\}\ \textbf{contains}(\mathsf{s}, \mathsf{v})\ \{Coll^z(\mathsf{s}, V) \wedge \mathsf{ret} = (\mathsf{v} \in V)\}$$

Clients and separating products

$$\begin{array}{c} 1.\ \{P\}\ c\ \{Q\} \\ \hline\hline \{P\}\ c\ \{Q\} \end{array} \qquad \text{and} \qquad \begin{array}{c} I * J.\ \{P \times \mathit{emp}\}\ c\ \{Q \times \mathit{emp}\} \\ \hline\hline I.\ \{P\}\ c\ \{Q\} \end{array}$$

Used to bootstrap FSL:

$$\frac{1 * I_1 * \cdots * I_n.\ \{P \times \mathit{emp}^n\}\ c\ \{Q \times \mathit{emp}^n\}}{\{P\}\ c\ \{Q\}}$$

Used to frame out interpretations:

$$\frac{I_i.\ \{P_i\}\ c\ \{Q_i\} \qquad \forall j \neq i.\ P_j = Q_j}{I_1 * \cdots * I_n.\ \{P_1 \times \cdots \times P_n\}\ c\ \{Q_1 \times \cdots \times Q_n\}}$$

Composing abstractions

$$(I'_{\text{frac}}\,;I_{\text{fine}}).\ \{In^z(\mathsf{s},\mathsf{v})\}\ \textbf{contains}(\mathsf{s},\mathsf{v})\ \{In^z(\mathsf{s},\mathsf{v}) \wedge \mathsf{ret} = true\}$$
$$(I'_{\text{frac}}\,;I_{\text{fine}}).\ \{In^z(\mathsf{s},\mathsf{v})\}\ \textbf{add}(\mathsf{s},\mathsf{v})\ \{In^z(\mathsf{s},\mathsf{v})\}$$
$$(I'_{\text{frac}}\,;I_{\text{fine}}).\ \{Out^1(\mathsf{s},\mathsf{v})\}\ \textbf{add}(\mathsf{s},\mathsf{v})\ \{In^1(\mathsf{s},\mathsf{v})\}$$

$$\begin{aligned}(I''_{\text{frac}}\,;I'_{\text{fine}}\,;I_{\text{cow}}).\,&\{Coll^z(\mathsf{s},V_\in,V_\notin)\}\ \textbf{clone}(\mathsf{s})\\ &\{Coll^z(\mathsf{s},V_\in,V_\notin) * Coll^1(\mathsf{ret},V_\in,V_\notin)\}\end{aligned}$$

Results

Examples we can encode

- Copy-on-write data (Mehnert, Sieczkowski, Birkedal & Sestoft)
- Fine-grained data structures (Dinsdale-Young, Dodds, Gardner, Parkinson & Vafeiadis)
- Permission accounting (Bornat, Calcagno, O'Hearn & Parkinson)
- Monotonic counters (Pilkiewicz & Pottier)
- Weak-update type system (Tan, Shao, Feng & Cai)

Attractive properties of fictional separation logic

- Simple and general metatheory
- Defined on top of standard separation logic
- Interpretations composable from smaller primitives

Technical details

Given separation algebras $(\Sigma, \circ_\Sigma, 0_\Sigma)$ and $(\Sigma', \circ_{\Sigma'}, 0_{\Sigma'})$, define

$$\Sigma \searrow \Sigma' \triangleq \{I : \Sigma \to \mathcal{P}(\Sigma') \mid I(0_\Sigma) = \{0_{\Sigma'}\}\}$$

Given $I : \Sigma \searrow \mathit{heap}$ and $P, Q : \mathit{stack} \to \mathcal{P}(\Sigma)$, define

$$I.\ \{P\}\ c\ \{Q\} \triangleq \forall \phi : \Sigma.\ \{\exists \sigma \in P.\ I(\sigma \circ \phi)\}\ c\ \{\exists \sigma \in Q.\ I(\sigma \circ \phi)\}$$
$$P \models_I Q \triangleq \forall \phi.\ \big([\exists \sigma \in P.\ I(\sigma \circ \phi)] \vdash [\exists \sigma \in Q.\ I(\sigma \circ \phi)]\big)$$

$$(*) : \Sigma_1 \searrow \Sigma \quad \to \quad \Sigma_2 \searrow \Sigma \quad \to \quad \Sigma_1 \times \Sigma_2 \searrow \Sigma$$
$$(;) : \Sigma_1 \searrow \Sigma_2 \quad \to \quad \Sigma_2 \searrow \Sigma_3 \quad \to \quad \Sigma_1 \searrow \Sigma_3$$

$$I_1 * I_2 \triangleq \lambda(\sigma_1, \sigma_2).\ I_1(\sigma_1) * I_2(\sigma_2)$$
$$I\ ;\ J \triangleq \lambda \sigma_1.\ \exists \sigma_2' \in I(\sigma_1).\ J(\sigma_2')$$