Local Reasoning about the Presence of Bugs:Incorrectness Separation Logic

September 3, 2020

Overview

- A brief review of the incorrectness logic.
- Introduction to separation logic.
- ISL (incorrectness separation logic).
- Investigation of the tool INFER.

Incorrectness Logic

Let ϵ range over a collection of exit conditions, to include at lease "ok" and "er". An *under-approximate triple* is of the form:

$$[p]C[\epsilon : q]$$

meaning q under-approximates the states when C exits via ϵ starting form states in p.
Sometimes, we write $[p]C[ok : q][er : r]$ as shorthand for $[p]C[ok : q]$ and $[p]C[er : r]$
An important point: the triple $[p]C[\epsilon : q]$ express the reachability property that involves termination. **Every state in the result is reachable from some states in the presumption**

Separation Logic

Separation logic is an extension of Hoare logic, which employs novel logical operators and able to handle heap-manipulating programs with aliasing.
More types of variables: $x = 1$ and $y \mapsto 1$

Example

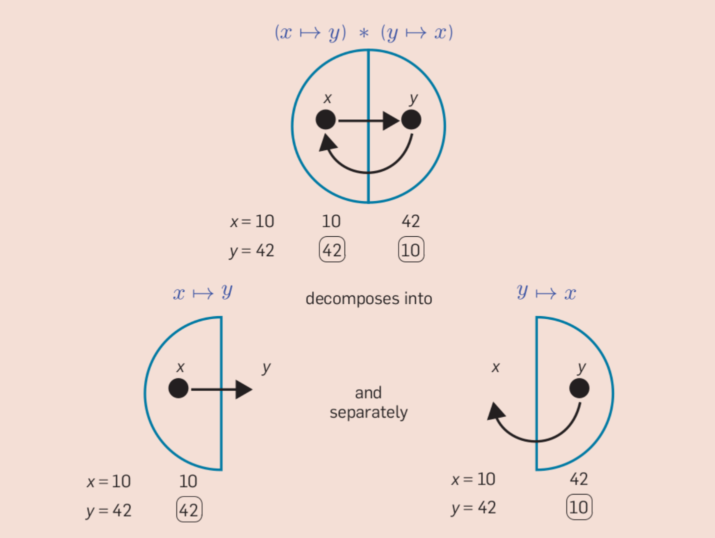
$\{x \neq y, y \neq z, x \neq z\}$
$[x] = 1;$
$[y] = 2;$
$[z] = 3;$
$\{x \mapsto 1, y \mapsto 2, x \mapsto 3\}$

Example

$\{x1 \neq x2, x1 \neq x3, \cdots\}$
$[x1] = 1;$
$[x2] = 2;$
$\cdots$
$[xn] = n;$
$\{x1 \mapsto 1, x2 \mapsto 2, \cdots, xn \mapsto n\}$

Therefore we introduce a new operator $*$ read as "and seperately".
$x1 \mapsto - * \cdots * xn \mapsto -$
$\forall x, v, v'. x \mapsto v * x \mapsto v' \Rightarrow \textbf{false}$
We also use *emp* to represent the empty heap.

SL: Semantic


$(x \mapsto y) * (y \mapsto x)$
x
y
x = 10
y = 42
10
42
42
10
$x \mapsto y$
decomposes into
$y \mapsto x$
x
y
and
separately
x
y
x = 10
y = 42
10
42
x = 10
y = 42
42
10

SL: Semantic

Let $s : \mathit{Vars} \to \mathit{Ints}$ and $h : \mathit{Nats} \to \mathit{Ints}$ be the "store" and "heap".

s is similar to the evaluation.

$$
\begin{array}{lll}
s,h \models E \mapsto F & \text{iff} & \{[\![E]\!]\,s\} = \mathit{dom}(h) \ \text{and}\ h([\![E]\!]\,s) = [\![F]\!]\,s \\
s,h \models \mathtt{emp} & \text{iff} & h = [\,] \text{ is the empty heap} \\
s,h \models P * Q & \text{iff} & \exists h_0, h_1.\ \ h_0 \bullet h_1 = h,\ s,h_0 \models P \ \text{and}\ s,h_1 \models Q \\
s,h \models P \mathrel{-\!\!*} Q & \text{iff} & \forall h'.\,\text{if } h' \# h \text{ and } s,h' \models P \text{ then } s,h \bullet h' \models Q \\
 & & \\
s,h \models \mathtt{false} & & \text{never} \\
s,h \models P \Rightarrow Q & \text{iff} & \text{if } s,h \models P \ \text{ then }\ s,h \models Q \\
s,h \models \forall x.P & \text{iff} & \forall v \in \mathtt{Ints}.\,[s \mid x \mapsto v], h \models P
\end{array}
$$

$h \bullet h'$ = union when domains disjoint, undefined otherwise

$h \# h'$ means domains disjoint

Boolean constructs $\neg P$, $P \wedge Q$, $P \vee Q$, $\exists x.P$ defined in usual way from $\mathtt{false}$, $\forall$, $\Rightarrow$.

SL: Rules and Axioms

Frame Rule

$$\frac{\{p\}\ C\ \{q\}}{\{p * r\}\ C\ \{q * r\}}$$

$x \mapsto v * x \mapsto v' \Leftrightarrow \text{false}$ $\quad$ $x \mapsto v * \text{emp} \Leftrightarrow x \mapsto v$

Local Axioms

$\{x \mapsto -\}\ [x] := v\ \{x \mapsto v\}$

$\{x \mapsto v\}\ y := [x]\ \{x \mapsto v \wedge y = v\}$

$\{\text{emp}\}\ x := \text{alloc}()\ \{\exists l.\ l \mapsto - \wedge x = l\}$

$\{x \mapsto -\}\ \text{free}(x)\ \{\ \text{emp}\ \}$

The primitive statements of the language update or access only one memory cell each time. These local axioms are extremely concise and able to do local reasoning.

Frame rule allows to extend the reasoning from one cell to multiple cell.

SL: Proof Example

$$\frac{\{p\}\ C\ \{q\}}{\{p * r\}\ C\ \{q * r\}}$$

$$\{x \mapsto -\}\ [x] := v\ \{x \mapsto v\}$$

frame rule

write axiom

$$\{\, x \mapsto - * y \mapsto - * z \mapsto - \,\}$$

$$\{\, x \mapsto - \,\}$$

```
[x]:= 1;
```

$$\{\, x \mapsto 1 \,\}$$

$$\{\, x \mapsto 1 * y \mapsto - * z \mapsto - \,\}$$

```
[y]:= 2;
[z]:= 3;
```

$$\{\, x \mapsto 1 * y \mapsto 2 * z \mapsto 3 \,\}$$

SL: Soundness and Completeness

Theorem (Soundness)

SL is sound:

$$\{p\}C\{q\} \textit{ is provable} \Rightarrow \{p\}C\{q\} \textit{ is true}$$

SL is not complete however.

Incompleteness: Example

Counter-example.

would like to derive:

$\{y \mapsto -\}$ x:= alloc(); free(x) $\{\, y \mapsto - \wedge y \neq x \,\}$

$\{emp\}$
x:= alloc();
$\{\exists l.\ l \mapsto - \wedge x=l \}$
free(x);
$\{\exists l.\ x=l \}$

frame $y \mapsto -$ →

strongest derivable spec

$\{y \mapsto -\}$
x:= alloc(); free(x)
$\{\exists l.\ y \mapsto - \wedge x=l \}$

incomplete specification ✗

lossy: forgets that x held a location

$\{emp\}$ x:= alloc() $\{\exists l.\ l \mapsto - \wedge x=l \}$ $\boxed{\{x \mapsto -\} \text{ free(x) } \{ emp \}}$

ISL: Incorrectness Separation Logic

IL

$[p]\ C\ [\varepsilon: q]$

SL

$$\frac{\{p\}\ C\ \{q\}}{\{p * r\}\ C\ \{q * r\}}$$

$x \mapsto - \ * \ x \mapsto - \Leftrightarrow \mathrm{false}$

$x \mapsto v \ * \ \mathrm{emp} \Leftrightarrow x \mapsto v$

ISL

$$\frac{[p]\ C\ [\varepsilon: q]}{[p * r]\ C\ [\varepsilon: q * r]}$$

$x \mapsto v \ * \ x \mapsto v' \Leftrightarrow \mathrm{false}$

$x \mapsto v \ * \ \mathrm{emp} \Leftrightarrow x \mapsto v$

ISL: Problem of Combining

$[x \mapsto v']$ $[x] := v$ $[ok: x \mapsto v]$ $[x = null]$ $[x] := v$ $[er: x = null]$

null-pointer dereference error

$[x \mapsto v]$ $y := [x]$ $[ok: x \mapsto v \land y = v]$ $[x = null]$ $y := [x]$ $[er: x = null]$

$[emp]$ $x := alloc()$ $[ok: \exists l.\ l \mapsto v \land x = l]$

$[x \mapsto v]$ $free(x)$ $[ok: emp]$ $[x = null]$ $free(x)$ $[er: x = null]$

ISL: Problems of Combining

ISL

$$\frac{[p]\ C\ [\varepsilon: q]}{[p * r]\ C\ [\varepsilon: q * r]}$$

$x \mapsto v \ * \ x \mapsto v' \Leftrightarrow \text{false}$

$x \mapsto v \ * \ \text{emp} \Leftrightarrow x \mapsto v$

$$\cfrac{\cfrac{[x \mapsto v]\ \text{free}(x)\ [\text{ok: emp}]}{[x \mapsto v * x \mapsto v\ ]\ \text{free}(x)\ [\text{ok: emp} * x \mapsto v]}\ \text{(Frame)}}{[\text{false}]\ \text{free}(x)\ [\text{ok: } x \mapsto v]}\ \text{(Cons)}$$

$[p]\ C\ [\varepsilon: q]$ *iff* $\forall\, s \in q.\ \exists\, s' \in p.\ (s', s) \in [C]\varepsilon$

ISL: Solution

The solution for the frame rule is to remember that we allocated memory for the pointer even if we freed them.

Old free rule:

$$[x \mapsto v]\mathit{free}(x)[\mathit{ok} : \mathit{emp}]$$

New free rule:

$$[x \mapsto v]\mathit{free}(x)[\mathit{ok} : x \not\mapsto]$$

$$x \mapsto v \ast x \mapsto v' \Leftrightarrow \text{false}$$
$$x \mapsto v \ast \text{emp} \Leftrightarrow x \mapsto v$$
$$x \mapsto v \ast x \not\mapsto \Leftrightarrow \text{false}$$
$$x \not\mapsto \ast x \not\mapsto \Leftrightarrow \text{false}$$

ISL: Target Programs and Rules

$$\textsc{Comm} \ni \mathbb{C} ::= \texttt{skip} \mid x := e \mid x := * \mid \texttt{assume}(B) \mid \texttt{local}\ x\ \texttt{in}\ \mathbb{C} \mid \mathbb{C}_1; \mathbb{C}_2 \mid \mathbb{C}_1 + \mathbb{C}_2 \mid \mathbb{C}^\star$$
$$\mid x := \texttt{alloc()} \mid \text{L}: \texttt{free}(x) \mid \text{L}: x := [y] \mid \text{L}: [x] := y \mid \text{L}: \texttt{error}$$

$$\begin{aligned}
\texttt{if}\ B\ \texttt{then}\ \mathbb{C}_1\ \texttt{else}\ \mathbb{C}_2 &\triangleq (\texttt{assume}(B); \mathbb{C}_1) + (\texttt{assume}(!B); \mathbb{C}_2) \\
\texttt{while}(B)\ \mathbb{C} &\triangleq (\texttt{assume}(B); \mathbb{C})^\star; \texttt{assume}(!B) \\
\texttt{assert}(B) &\triangleq (\texttt{assume}(!B); \texttt{error}) + \texttt{assume}(B) \\
x := \texttt{malloc()} &\triangleq x := \texttt{alloc()}\ +\ x := \texttt{null}
\end{aligned}$$

$\textsc{FreeEr}$
$\vdash [x \not\mapsto\ ]\ \text{L}: \texttt{free}(x)\ [er(\text{L}): x \not\mapsto\ ]$

$\textsc{FreeNull}$
$\vdash [x = \texttt{null}]\ \text{L}: \texttt{free}(x)\ [er(\text{L}): x = \texttt{null}]$

$\textsc{Load}$
$\vdash [x = x' * y \mapsto e]\ \text{L}: x := [y]\ [ok: x = e[x'/x] * y \mapsto e[x'/x]]$

$\textsc{Store}$
$\vdash [x \mapsto e]\ \text{L}: [x] := y\ [ok: x \mapsto y]$

$\textsc{LoadEr}$
$\vdash [y \not\mapsto\ ]\ \text{L}: x := [y]\ [er(\text{L}): y \not\mapsto\ ]$

$\textsc{StoreEr}$
$\vdash [x \not\mapsto\ ]\ \text{L}: [x] := y\ [er(\text{L}): x \not\mapsto\ ]$

$\textsc{LoadNull}$
$\vdash [y = \texttt{null}]\ \text{L}: x := [y]\ [er(\text{L}): y = \texttt{null}]$

$\textsc{StoreNull}$
$\vdash [x = \texttt{null}]\ \text{L}: [x] := y\ [er(\text{L}): x = \texttt{null}]$

ISL: Semantic

Adapt original semantic incorrectness logic to ISL semantic

$$[\![.]\!] : \textsc{Comm} \to \textsc{Exit} \to \mathcal{P}(\textsc{State} \times \textsc{State}) \qquad \sigma \in \textsc{State} \triangleq \textsc{Store} \times \textsc{Heap}$$

$$s \in \textsc{Store} \triangleq \textsc{Var} \stackrel{\text{fin}}{\to} \textsc{Val} \qquad h \in \textsc{Heap} \triangleq \textsc{Loc} \stackrel{\text{fin}}{\rightharpoonup} \textsc{Val} \uplus \{\bot\} \qquad l \in \textsc{Loc} \subseteq \textsc{Val}$$

$$[\![\texttt{skip}]\!]ok \triangleq \{(\sigma,\sigma) \mid \sigma \in \textsc{State}\} \qquad [\![\texttt{skip}]\!]er(-) \triangleq \emptyset$$

$$[\![x := e]\!]ok \triangleq \{((s,h),(s[x \mapsto s(e)],h))\} \qquad [\![x := e]\!]er(-) \triangleq \emptyset$$

$$[\![x := *]\!]ok \triangleq \{((s,h),(s[x \mapsto v],h)) \mid v \in \textsc{Val}\} \qquad [\![x := *]\!]er(-) \triangleq \emptyset$$

$$[\![\texttt{assume}(B)]\!]ok \triangleq \{(\sigma,\sigma) \mid \sigma=(s,h) \land s(B)\neq 0\} \qquad [\![\texttt{assume}(B)]\!]er(-) \triangleq \emptyset$$

$$[\![\textsc{l}\colon \texttt{error}]\!]ok \triangleq \emptyset \qquad [\![\textsc{l}\colon \texttt{error}]\!]er(\textsc{l}') \triangleq \{(\sigma,\sigma) \mid \textsc{l}=\textsc{l}'\}$$

$$[\![\mathbb{C}_1;\mathbb{C}_2]\!]\epsilon \triangleq \left\{(\sigma,\sigma') \,\middle|\, \begin{array}{l} \epsilon \neq ok \land (\sigma,\sigma') \in [\![\mathbb{C}_1]\!]\epsilon \\ \lor\ \exists \sigma''.\ (\sigma,\sigma'') \in [\![\mathbb{C}_1]\!]ok \land (\sigma'',\sigma') \in [\![\mathbb{C}_2]\!]\epsilon \end{array}\right\}$$

$$[\![\texttt{local}\ \ x\ \ \texttt{in}\ \ \mathbb{C}]\!]\epsilon \triangleq \{((s[x \mapsto v],h),(s'[x \mapsto v],h')) \mid ((s,h),(s',h')) \in [\![\mathbb{C}]\!]\epsilon\}$$

$$[\![\mathbb{C}_1 + \mathbb{C}_2]\!]\epsilon \triangleq [\![\mathbb{C}_1]\!]\epsilon \cup [\![\mathbb{C}_2]\!]\epsilon$$

$$[\![\mathbb{C}^\star]\!]\epsilon \triangleq \bigcup_{i\in\mathbb{N}} [\![\mathbb{C}^i]\!]\epsilon \quad \text{with} \quad \mathbb{C}^0 \triangleq \texttt{skip} \quad \text{and} \quad \mathbb{C}^{i+1} \triangleq \mathbb{C};\mathbb{C}^i$$

ISL: Semantic

Semantic related to memory operations.

$$\llbracket x := \texttt{alloc()} \rrbracket ok \triangleq \left\{ (\sigma, (s[x \mapsto l], h[l \mapsto v])) \;\middle|\; \begin{array}{l} \sigma = (s, h) \wedge v \in \textsc{Val} \\ \wedge\ (l \notin dom(h) \vee h(l) = \bot) \end{array} \right\}$$

$$\llbracket x := \texttt{alloc()} \rrbracket er(-) \triangleq \emptyset$$

$$\llbracket \mathrm{L}\colon \texttt{free}(x) \rrbracket ok \triangleq \{ (\sigma, (s, h[s(x) \mapsto \bot])) \mid \sigma = (s, h) \wedge h(s(x)) \in \textsc{Val} \}$$

$$\llbracket \mathrm{L}\colon \texttt{free}(x) \rrbracket er(\mathrm{L}') \triangleq \{ (\sigma, \sigma) \mid \mathrm{L} = \mathrm{L}' \wedge \sigma = (s, h) \wedge (s(x) = \mathbf{null} \vee h(s(x)) = \bot) \}$$

$$\llbracket \mathrm{L}\colon x := [y] \rrbracket ok \triangleq \{ (\sigma, (s[x \mapsto v], h)) \mid \sigma = (s, h) \wedge h(s(y)) = v \in \textsc{Val} \}$$

$$\llbracket \mathrm{L}\colon x := [y] \rrbracket er(\mathrm{L}') \triangleq \{ (\sigma, \sigma) \mid \mathrm{L} = \mathrm{L}' \wedge \sigma = (s, h) \wedge (s(y) = \mathbf{null} \vee h(s(y)) = \bot) \}$$

$$\llbracket \mathrm{L}\colon [x] := y \rrbracket ok \triangleq \{ (\sigma, (s, h[s(x) \mapsto s(y)])) \mid \sigma = (s, h) \wedge h(s(x)) \in \textsc{Val} \}$$

$$\llbracket \mathrm{L}\colon [x] := y \rrbracket er(\mathrm{L}') \triangleq \{ (\sigma, \sigma) \mid \mathrm{L} = \mathrm{L}' \wedge \sigma = (s, h) \wedge (s(x) = \mathbf{null} \vee h(s(x)) = \bot) \}$$

ISL: Soundness

Theorem (Soundness of ISL)

For all $p, \mathbb{C}, q, \epsilon$, *if* $\vdash [p]\mathbb{C}[\epsilon : q]$, *the* $\models [p]\mathbb{C}[\epsilon : q]$.

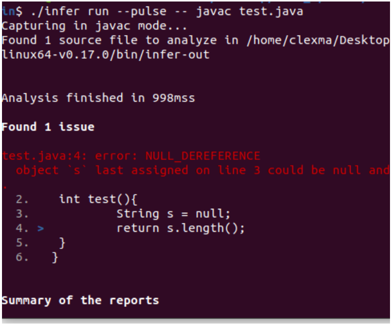
The definition of $\models [p]\mathbb{C}[\epsilon : q]$ is $\forall \sigma_q \in q. \exists \sigma_p \in p. (\sigma_p, \sigma_q) \in [[\mathbb{C}]]\epsilon$

Tool: Infer

Current Version: 0.17.0

Only able to do inference with separation logic.

`./infer run -- javac test.java`



- CONSTANT_ADDRESS_DEREFERENCE
- MEMORY_LEAK
- NULLPTR_DEREFERENCE
- STACK_VARIABLE_ADDRESS_ESCAPE
- USE_AFTER_DELETE
- USE_AFTER_FREE
- USE_AFTER_LIFETIME
- VECTOR_INVALIDATION

In the doc of next version:

`./infer --pulse --pulse-intraprocedural-only`