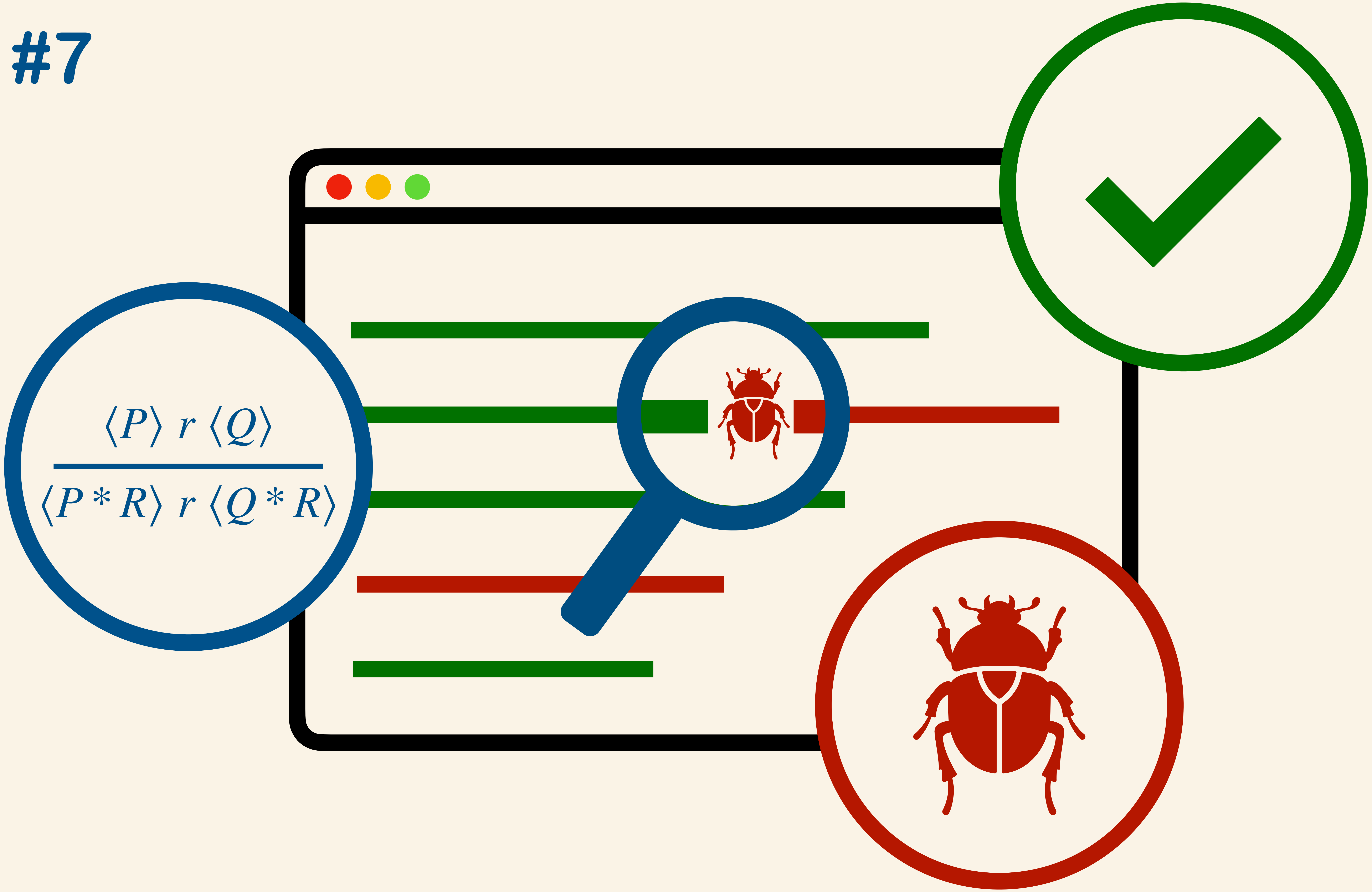
Program Analysis

Lecture #7

Roberto Bruni

Università di Pisa



SCAN ME

PhD Course
June 30 - July 4, 2025

GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

Footprint property

Footprint recipe

Given a command r

1. Derive the specification of r using **local axioms**
2. Apply rule frame to complete the specification

$$
\begin{array}{l}
\{P * x \mapsto \_\} \\
\quad \{x \mapsto \_\} \\
\qquad [x] := 1; \\
\quad \{x \mapsto 1\} \\
\qquad a := [x] \\
\quad \{x \mapsto 1 \land a = 1\} \\
\{P * x \mapsto 1 \land a = 1\}
\end{array}
$$

Footprint fails in SL

Counterexample:

We would like to derive

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

$$\{y \mapsto \_ * \mathsf{emp}\}$$

$$\{\mathsf{emp}\}$$

$$x := \mathsf{alloc}(\,);$$

$$\{\exists v.\ v \mapsto \_ \wedge x = v\}$$

$$\mathsf{free}(x)$$

$$\{\exists v.\ x = v\}$$

$$\{\exists v.\ y \mapsto \_ \wedge x = v\}$$ // strongest derivable spec: **incomplete spec**

lossy: x held a location!

$$\frac{}{\{x \mapsto \_\}\ \mathsf{free}(x)\ \{\mathsf{emp}\}}$$

Information loss

$\{x \mapsto \_\}\ [x] := y\ \{x \mapsto y\}$ // write

$\{y \mapsto v\}\ x := [y]\ \{x = v \land y \mapsto v\}$ // read

$\{\mathsf{emp}\}\ x := \mathsf{alloc}(\,)\ \{x \mapsto \_\}$ // alloc

$\{x \mapsto \_\}\ \mathsf{free}(x)\ \{\mathsf{emp}\}$ // dispose

resources can grow

but should never shrink!

Incorrectness Separation Logic (ISL)

CAV 2020

Local Reasoning About the Presence of Bugs: Incorrectness Separation Logic

Azalea Raad[1](✉), Josh Berdine[2], Hoang-Hai Dang[1], Derek Dreyer[1], Peter O'Hearn[2,3], and Jules Villard[2]

[1] Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern and Saarbrücken, Germany
{azalea,haidang,dreyer}@mpi-sws.org
[2] Facebook, London, UK
{jjb,peteroh,jul}@fb.com
[3] University College London, London, UK

Abstract. There has been a large body of work on local reasoning for proving the *absence* of bugs, but none for proving their *presence*. We present a new formal framework for local reasoning about the presence of bugs, building on two complementary foundations: 1) separation logic and 2) incorrectness logic. We explore the theory of this new *incorrectness separation logic* (ISL), and use it to derive a begin-anywhere, intra-procedural symbolic execution analysis that has no false positives *by construction*. In so doing, we take a step towards transferring modular, scalable techniques from the world of program verification to bug catching.

Keywords: Program logics · Separation logic · Bug catching

1 Introduction

There has been significant research on sound, local reasoning about the state for proving the absence of bugs (e.g., [2,13,26,29,30,41]). Locality leads to techniques that are compositional *both* in code (concentrating on a program component) and in the resources accessed (spatial locality), without tracking the entire global state or the global program within which a component sits. Compositionality enables reasoning to scale to large teams and codebases: reasoning can be done even when a global program is not present (e.g., a library, or during program construction), without having to write the analogue of a test or verification harness, and the results of reasoning about components can be composed efficiently [11].

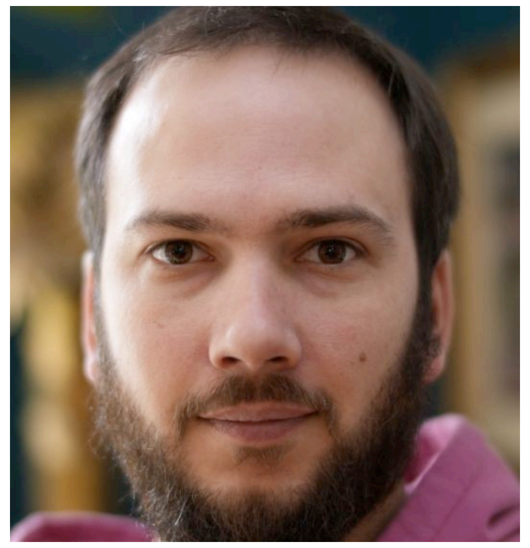
Meanwhile, many of the practical applications of symbolic reasoning have aimed at proving the *presence* of bugs (i.e., bug catching), rather than proving their absence (i.e., correctness). Logical bug catching methods include symbolic model checking [7,12] and symbolic execution for testing [9]. These methods are usually formulated as global analyses; but, the rationale of local reasoning holds just as well for bug catching as it does for correctness: it has the potential to

S. K. Lahiri and C. Wang (Eds.): CAV 2020, LNCS 12225, pp. 225–252, 2020.
https://doi.org/10.1007/978-3-030-53291-8_14

"bugs are a fundamental enough phenomenon to warrant a fundamental compositional theory for reasoning positively about their existence"





ISL = IL + SL

bug finding

IL

$$[P]\ r\ [\epsilon : Q]$$

local reasoning

SL

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

ISL

$$\frac{[P]\ r\ [\epsilon : Q]}{[P * R]\ r\ [\epsilon : Q * R]}$$

Regular commands

regular command

$$
\begin{aligned}
r \;::=\;& \; e \\
\mid\;& \; r_1; r_2 \\
\mid\;& \; r_1 + r_2 \\
\mid\;& \; r^\star
\end{aligned}
$$

atomic command

choice

Kleene star

$$
\begin{aligned}
e \;::=\;& \; \mathsf{skip} \\
\mid\;& \; b? \\
\mid\;& \; x := a \\
\mid\;& \; x := [y] && \text{// read} \\
\mid\;& \; [x] := y && \text{// write} \\
\mid\;& \; x := \mathsf{alloc}(\,) \\
\mid\;& \; \mathsf{free}(x) \\
\mid\;& \; \mathsf{error}(\,) \\
\mid\;& \; \ldots
\end{aligned}
$$

simplified

can fail

Disclaim

for the sake of presentation, some
technical aspects are not fully detailed

Local axioms: write

SL

$$\frac{}{\{x \mapsto \_\}\ [x] := y\ \{x \mapsto y\}}$$

$$\frac{}{[x \mapsto v]\ [x] := y\ [\mathsf{ok} : x \mapsto y]}$$

must put a value

$$\frac{}{[x = \mathsf{nil}]\ [x] := y\ [\mathsf{er} : x = \mathsf{nil}]}$$

null pointer dereference

Local axioms: read

SL

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

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

$$\frac{}{[y = \mathsf{nil}]\ x := [y]\ [\mathsf{er} : y = \mathsf{nil}]}$$

null pointer dereference

Local axioms: allocation

SL
$$\frac{}{\{\mathsf{emp}\}\ x := \mathsf{alloc}(\,)\ \{x \mapsto \_\}}$$

$$\frac{}{[x \doteq x']\ x := \mathsf{alloc}(\,)\ [\mathsf{ok} : x \mapsto \_]}$$

needed for footprint

Local axioms: dispose (1st try)

SL $\dfrac{}{\{x \mapsto \_\}\ \mathsf{free}(x)\ \{\mathsf{emp}\}}$

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

must put a value

$\dfrac{}{[x = \mathsf{nil}]\ \mathsf{free}(x)\ [\mathsf{er} : x = \mathsf{nil}]}$

null pointer dereference

Unsound Frame rule

$$\frac{[P]\ r\ [\epsilon : Q]}{[P * R]\ r\ [\epsilon : Q * R]}$$

$$\dfrac{\dfrac{}{[x \mapsto v]\ \mathsf{free}(x)\ [\mathsf{ok} : \mathsf{emp}]}\,[\mathsf{free}]}{\dfrac{[x \mapsto v * x \mapsto v]\ \mathsf{free}(x)\ [\mathsf{ok} : \mathsf{emp} * x \mapsto v]}{[\mathsf{false}]\ \mathsf{free}(x)\ [\mathsf{ok} : x \mapsto v]}\,[\mathsf{cons}]}\,[\mathsf{frame}]$$

the only sound under approximation can be false

this can be fixed by using a monotonic heap model (and we will recover completeness)

Solution

assertion

$$
\begin{aligned}
P ::= \;& \mathsf{true} \mid \mathsf{false} \mid a_1 < a_2 \mid a_1 = a_2 \mid \ldots \\
\mid \;& \neg P \mid P_1 \wedge P_2 \mid \exists x\,.\,P \mid \ldots \\
\mid \;& \mathsf{emp} \\
\mid \;& a_1 \mapsto a_2 \\
\mid \;& P_1 * P_2 \\
\mid \;& x \not\mapsto
\end{aligned}
$$

Boolean and classical assertions

structural assertions

track deallocated locations

Satisfaction: structural

$\langle s, h \rangle \vDash a_1 \mapsto a_2$ iff $\mathrm{dom}(h) = \{[\![a_1]\!]s\}$ and $h([\![a_1]\!]s) = [\![a_2]\!]s$

$\langle s, h \rangle \vDash \mathsf{emp}$ iff h is the empty map [ ]

$\langle s, h \rangle \vDash P_1 * P_2$ iff $\exists h_1, h_2 \,.\, \langle s, h_1 \rangle \vDash P_1$ and $\langle s, h_2 \rangle \vDash P_2$ and $h = h_1 \bullet h_2$

$\langle s, h \rangle \vDash x \not\mapsto$ iff $\mathrm{dom}(h) = \{s(x)\}$ and $h(s(x)) = \bot$

Notably: $x \mapsto v * x \not\mapsto \;\equiv\; \mathsf{false} \;\equiv\; x \not\mapsto * x \not\mapsto$

$y \mapsto v * x \not\mapsto \;\equiv\; y \mapsto v * x \not\mapsto \wedge x \neq y$

Local axioms: dispose

SL

$$\overline{\{x \mapsto \_\}\ \mathsf{free}(x)\ \{\mathsf{emp}\}}$$

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

must put a value

track deallocated locations

this way resources cannot shrink

$$\overline{[x = \mathsf{nil}]\ \mathsf{free}(x)\ [\mathsf{er} : x = \mathsf{nil}]}$$

null pointer dereference

Example

$$
\dfrac{
  \dfrac{
    \dfrac{}{[x \mapsto v]\ \mathsf{free}(x)\ [\mathsf{ok} : x \not\mapsto\ ]}\,[\mathsf{free}]
  }{
    [x \mapsto v * x \mapsto v]\ \mathsf{free}(x)\ [\mathsf{ok} : x \not\mapsto\ * x \mapsto v]
  }\,[\mathsf{frame}]
}{
  [\mathsf{false}]\ \mathsf{free}(x)\ [\mathsf{ok} : \mathsf{false}]
}\,[\mathsf{cons}]
$$

sound
under approximation!

Footprint holds in ISL

We would like to derive

$[y \mapsto \_]\ x := \text{alloc}(\,)\,;\ \text{free}(x)\ [\text{ok} : y \mapsto \_ \land x \not\mapsto \land y \neq x]$

$[y \mapsto \_ * \text{emp}]$

$\quad[\text{emp}]$

$\qquad x := \text{alloc}(\,);$

$\quad[x \mapsto v]$

$\qquad \text{free}(x)$

$\quad[x \not\mapsto\ ]$

$[\text{ok} : y \mapsto \_ * x \not\mapsto \land y \neq x]$

Additional local axioms

$$\frac{}{[x \not\mapsto\ ]\ [x] := y\ [\mathsf{er} : x \not\mapsto\ ]}$$

$$\frac{}{[y \not\mapsto\ ]\ x := [y]\ [\mathsf{er} : y \not\mapsto\ ]}$$

use after free errors

$$\frac{}{[x \not\mapsto\ ]\ \mathsf{free}(x)\ [\mathsf{er} : x \not\mapsto\ ]}$$

double free error

$$\frac{}{[y \not\mapsto\ ]\ x := \mathsf{alloc}(\,)\ [\mathsf{ok} : x \mapsto v \land x = y]}$$

reuse of deallocated locations

Soundness and completeness

Relational semantics

$[\![\mathsf{skip}]\!]\mathsf{ok} \triangleq \{(\sigma, \sigma)\}$

$[\![\mathsf{skip}]\!]\mathsf{er} \triangleq \varnothing$

$[\![b?]\!]\mathsf{ok} \triangleq \{(\sigma, \sigma) \mid \sigma = \langle s, h\rangle \wedge s \vDash b\}$

$[\![b?]\!]\mathsf{er} \triangleq \varnothing$

$[\![x := a]\!]\mathsf{ok} \triangleq \{(\langle s, h\rangle, \langle s[x \mapsto [\![a]\!]s], h\rangle)\}$

$[\![x := a]\!]\mathsf{er} \triangleq \varnothing$

$[\![\mathsf{error}(\,)]\!]\mathsf{ok} \triangleq \varnothing$

$[\![\mathsf{error}(\,)]\!]\mathsf{er} \triangleq \{(\sigma, \sigma)\}$

$[\![x := [y]]\!]\mathsf{ok} \triangleq \{(\langle s, h\rangle, \langle s[x \mapsto v], h\rangle) \mid v = h(s(y)) \in \mathbb{Z}\}$

$[\![x := [y]]\!]\mathsf{er} \triangleq \{(\langle s, h\rangle, \langle s, h\rangle) \mid s(y) = \mathsf{nil} \vee h(s(y)) = \bot\,\}$

$[\![[x] := y]\!]\mathsf{ok} \triangleq \{(\langle s, h\rangle, \langle s, h[s(x) \mapsto s(y)]\rangle) \mid h(s(x)) \in \mathbb{Z}\}$

$[\![[x] := y]\!]\mathsf{er} \triangleq \{(\langle s, h\rangle, \langle s, h\rangle) \mid s(x) = \mathsf{nil} \vee h(s(x)) = \bot\,\}$

$[\![x := \mathsf{alloc}(\,)]\!]\mathsf{ok} \triangleq \{(\langle s, h\rangle, \langle s[x \mapsto n], h[n \mapsto v]\rangle) \mid v \in \mathbb{Z} \wedge (n \notin \mathsf{dom}(h) \vee h(n) = \bot\,)\}$

$[\![x := \mathsf{alloc}(\,)]\!]\mathsf{er} \triangleq \varnothing$

$[\![\mathsf{free}(x)]\!]\mathsf{ok} \triangleq \{(\langle s, h \bullet [s(x) \mapsto v]\rangle, \langle s, h \bullet [s(x) \mapsto \bot\,]\rangle) \mid s(x) \in \mathbb{N} \wedge v \in \mathbb{Z}\}$

$[\![\mathsf{free}(x)]\!]\mathsf{er} \triangleq \{(\langle s, h\rangle, \langle s, h\rangle) \mid s(x) = \mathsf{nil} \vee h(s(x)) = \bot\,\}$

Actual ISL rules

Skip

$$\vdash [\mathsf{emp}]\ \texttt{skip}\ [ok: \mathsf{emp}]$$

Assign

$$\vdash [x = x']\ x := e\ [ok: x = e[x'/x]]$$

Havoc

$$\vdash [x = x']\ x := *\ [ok: x = v]$$

Assume

$$\vdash [\mathsf{emp}]\ \texttt{assume}(B)\ [ok: B]$$

Error

$$\vdash [\mathsf{emp}]\ \textsc{l}\colon \texttt{error}\ [er(\textsc{l}): \mathsf{emp}]$$

Seq1

$$\frac{\vdash [p]\ \mathbb{C}_1\ [er(\textsc{l}): q]}{\vdash [p]\ \mathbb{C}_1; \mathbb{C}_2\ [er(\textsc{l}): q]}$$

Seq2

$$\frac{\vdash [p]\ \mathbb{C}_1\ [ok: r] \qquad \vdash [r]\ \mathbb{C}_2\ [\epsilon: q]}{\vdash [p]\ \mathbb{C}_1; \mathbb{C}_2\ [\epsilon: q]}$$

Loop1

$$\vdash [p]\ \mathbb{C}^{\star}\ [ok: p]$$

Choice

$$\frac{\vdash [p]\ \mathbb{C}_i\ [\epsilon: q] \qquad \text{for some } i \in \{1, 2\}}{\vdash [p]\ \mathbb{C}_1 + \mathbb{C}_2\ [\epsilon: q]}$$

Exist

$$\frac{\vdash [p]\ \mathbb{C}\ [\epsilon: q] \qquad x \notin \mathsf{fv}(\mathbb{C})}{\vdash [\exists x. p]\ \mathbb{C}\ [\epsilon: \exists x. q]}$$

Loop2

$$\frac{\vdash [p]\ \mathbb{C}^{\star}; \mathbb{C}\ [\epsilon: q]}{\vdash [p]\ \mathbb{C}^{\star}\ [\epsilon: q]}$$

Cons

$$\frac{p' \Rightarrow p \qquad \vdash [p']\ \mathbb{C}\ [\epsilon: q'] \qquad q \Rightarrow q'}{\vdash [p]\ \mathbb{C}\ [\epsilon: q]}$$

Disj

$$\frac{\vdash [p_1]\ \mathbb{C}\ [\epsilon: q_1] \qquad \vdash [p_2]\ \mathbb{C}\ [\epsilon: q_2]}{\vdash [p_1 \vee p_2]\ \mathbb{C}\ [\epsilon: q_1 \vee q_2]}$$

Subst

$$\frac{\vdash [p]\ \mathbb{C}\ [\epsilon: q] \qquad y \notin \mathsf{fv}(p, \mathbb{C}, q)}{\vdash [p[y/x]]\ \mathbb{C}[y/x]\ [\epsilon: q[y/x]]}$$

Local

$$\frac{\vdash [p]\ \mathbb{C}\ [\epsilon: q]}{\vdash [\exists x.\ p]\ \texttt{local}\ x\ \texttt{in}\ \mathbb{C}\ [\epsilon: \exists x.\ q]}$$

Frame

$$\frac{\vdash [p]\ \mathbb{C}\ [\epsilon: q] \qquad \mathsf{mod}(\mathbb{C}) \cap \mathsf{fv}(r) = \emptyset}{\vdash [p * r]\ \mathbb{C}\ [\epsilon: q * r]}$$

Alloc1

$$\vdash [x = x']\ x := \texttt{alloc()}\ [ok: x \mapsto -]$$

Free

$$\vdash [x \mapsto e]\ \textsc{l}\colon \texttt{free}(x)\ [ok: x \not\mapsto]$$

Alloc2

$$\vdash [x = x' * y \not\mapsto]\ x := \texttt{alloc()}\ [ok: x = y * y \mapsto -]$$

FreeEr

$$\vdash [x \not\mapsto]\ \textsc{l}\colon \texttt{free}(x)\ [er(\textsc{l}): x \not\mapsto]$$

FreeNull

$$\vdash [x = \texttt{null}]\ \textsc{l}\colon \texttt{free}(x)\ [er(\textsc{l}): x = \texttt{null}]$$

Load

$$\vdash [x = x' * y \mapsto e]\ \textsc{l}\colon x := [y]\ [ok: x = e[x'/x] * y \mapsto e[x'/x]]$$

Store

$$\vdash [x \mapsto e]\ \textsc{l}\colon [x] := y\ [ok: x \mapsto y]$$

LoadEr

$$\vdash [y \not\mapsto]\ \textsc{l}\colon x := [y]\ [er(\textsc{l}): y \not\mapsto]$$

StoreEr

$$\vdash [x \not\mapsto]\ \textsc{l}\colon [x] := y\ [er(\textsc{l}): x \not\mapsto]$$

LoadNull

$$\vdash [y = \texttt{null}]\ \textsc{l}\colon x := [y]\ [er(\textsc{l}): y = \texttt{null}]$$

StoreNull

$$\vdash [x = \texttt{null}]\ \textsc{l}\colon [x] := y\ [er(\textsc{l}): x = \texttt{null}]$$

Correctness

Th. [*correctness*]
If $[P]\ r\ [\epsilon : Q]$ then $Q \subseteq [\![r]\!]\epsilon P$

Proof. By induction on the derivation.

Footprint theorem

Th. [*footprint*]
Any valid ISL triple $[\sigma_P]\ r\ [\epsilon : \sigma_Q]$ can be derived

Proof. See CAV2020 paper for details.

Final considerations on SL

ISL = IL + SL
for bug
catching!

ISL address compositional bug catching

targets memory safety bugs (use-after-free)

no-false-positives theorem: all bugs are true

Questions

Question 1

Is the axiom $[x \mapsto \_]\ [x] := y\ [\mathsf{ok} : x \mapsto y]$ sound?

$$\dfrac{(x \mapsto v) \Rightarrow (x \mapsto \_) \qquad \overline{[x \mapsto v]\ [x] := y\ [\mathsf{ok} : x \mapsto y]}}{[x \mapsto \_]\ [x] := y\ [\mathsf{ok} : x \mapsto y]}$$

Question 2

Prove that rule [*conj] is unsound

$$\frac{[P_1]\ r\ [Q_1] \qquad [P_2]\ r\ [Q_2]}{[P_1 * P_2]\ r\ [Q_1 * Q_2]}\text{[*conj]}$$

Consider $[x = 0]\ x := 1\ [x = 1]$ and $[x = 1]\ x := 1\ [x = 1]$

By rule [*conj] we could derive $[\mathsf{false}]\ x := 1\ [x = 1]$

which is not sound!