Incorrectness Separation Logic (ISL)

Azalea Raad

Imperial College London
Max Planck Institute for Software Systems (MPI-SWS)

Joint work with: Peter O'Hearn, Josh Berdine, Jules Villard @Facebook
Derek Dreyer, Hoang-Hai Dang @MPI-SWS

azalea@imperial.ac.uk SoundAndComplete.org @azalearaad

Part I.

Incorrectness Logic (IL)

Incorrectness Logic (IL)

Hoare triples {p} C {q}

Incorrectness Logic (IL)

Hoare triples {p} C {q}

For all states s *in* p
if running C *on* s *terminates in* s', *then* s' *is in* q

Incorrectness Logic (IL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

For all states s *in* p
if running C *on* s *terminates in* s', *then* s' *is in* q

Incorrectness Logic (IL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

For all states s *in* p
if running C *on* s *terminates in* s', *then* s' *is in* q

post(C)p ⊆ q

Incorrectness Logic (IL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

For all states s *in* p
if running C *on* s *terminates in* s', *then* s' *is in* q

post(C)p ⊇ q

Incorrectness Logic (IL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

For all states s *in* p
if running C *on* s *terminates in* s', *then* s' *is in* q

[p] C [q] *iff* post(C)p ⊇ q

Incorrectness Logic (IL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

For all states s *in* p
if running C *on* s *terminates in* s', *then* s' *is in* q

Incorrectness triples [p] C [q] *iff* post(C)p ⊇ q

Incorrectness Logic (IL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

For all states s *in* p
if running C *on* s *terminates in* s', *then* s' *is in* q

Incorrectness triples [p] C [q] *iff* post(C)p ⊇ q

For all states s *in* q
s *can be reached by running* C *on some* s' *in* p

Incorrectness Logic (IL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

Incorrectness Logic (IL)

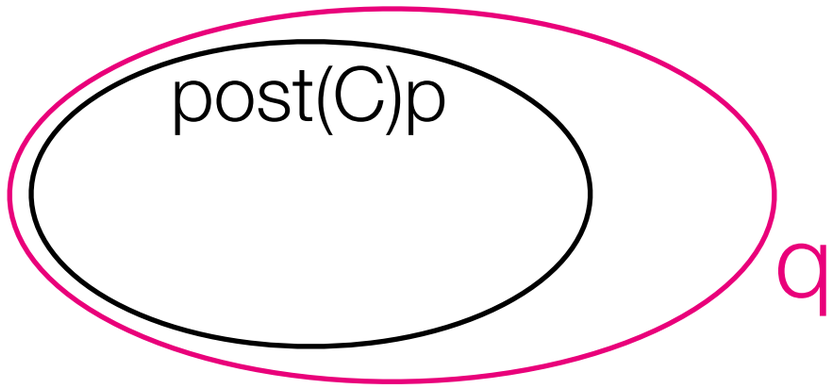
Hoare triples {p} C {q} *iff* post(C)p ⊆ q

q *over-approximates* post(C)p

Incorrectness Logic (IL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

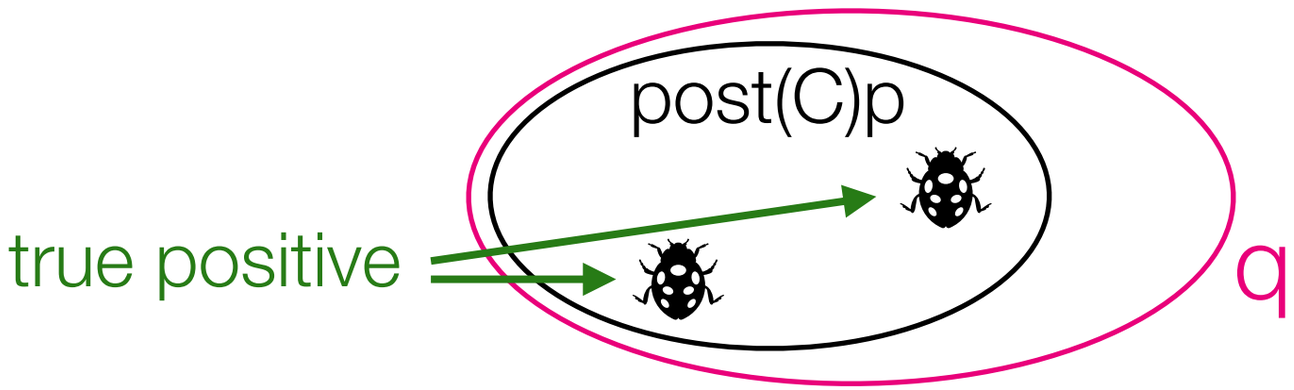
q *over-approximates* post(C)p



Incorrectness Logic (IL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

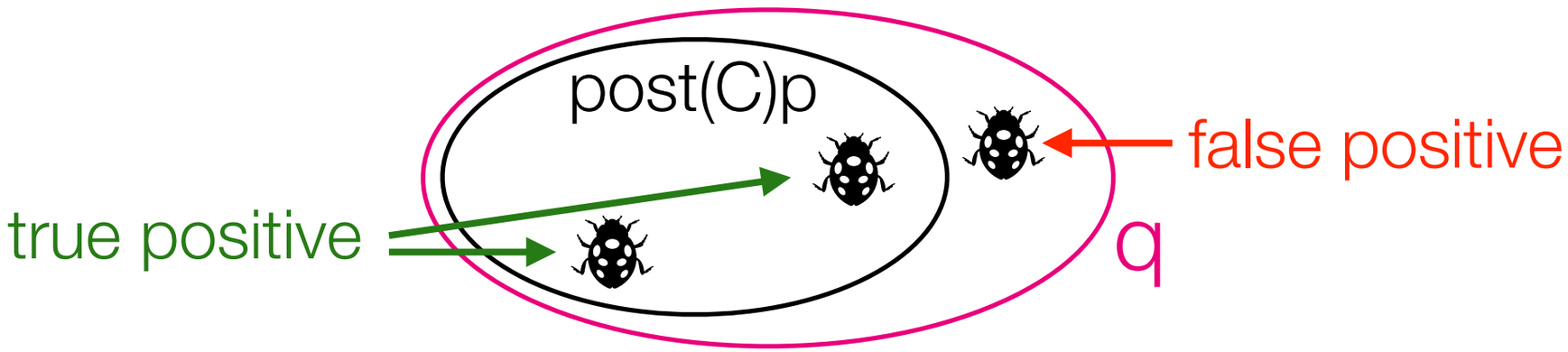
q *over-approximates* post(C)p



Incorrectness Logic (IL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

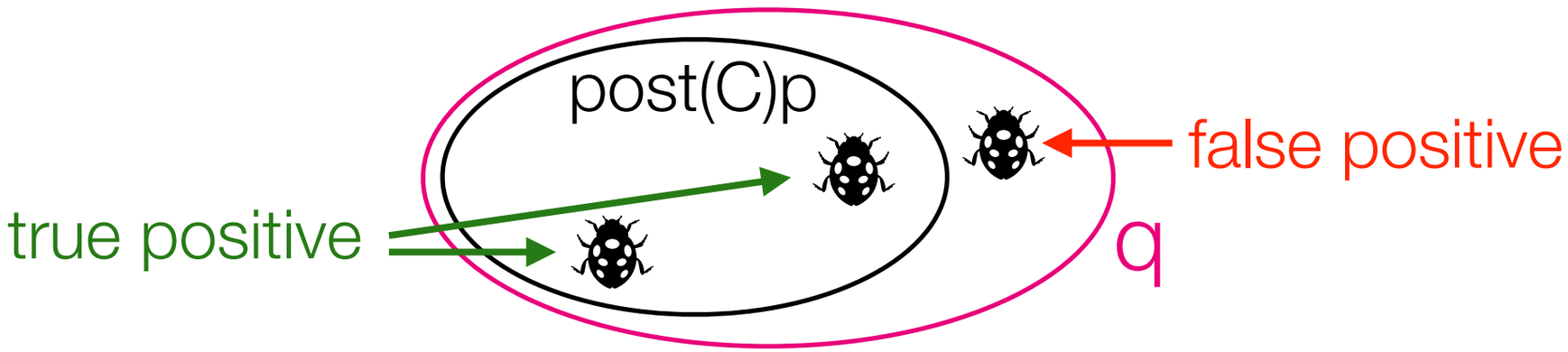
q *over-approximates* post(C)p



Incorrectness Logic (IL)

Hoare triples {p} C {q} *iff* post(C)p ⊆ q

q *over-approximates* post(C)p



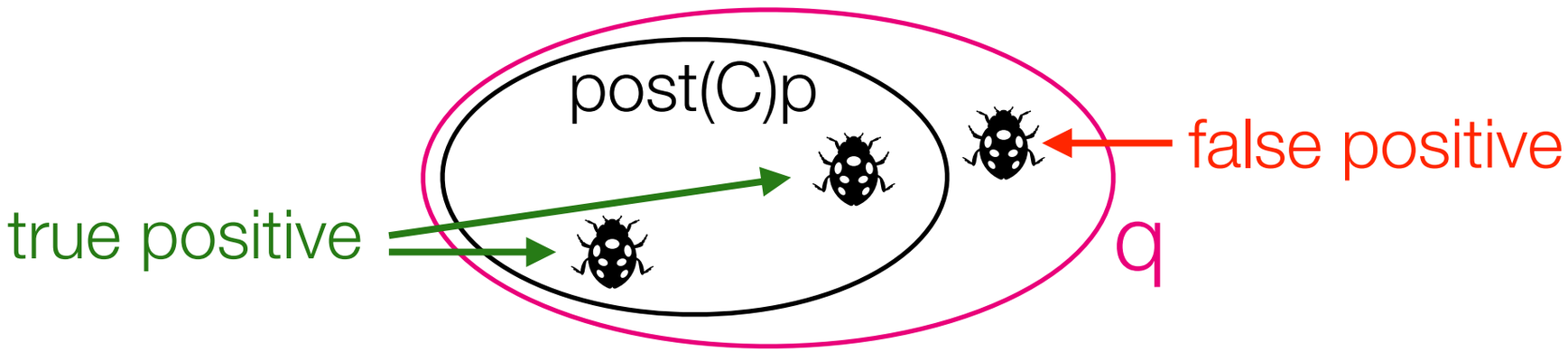
Incorrectness triples [p] C [q] *iff* post(C)p ⊇ q

q *under-approximates* post(C)p

Incorrectness Logic (IL)

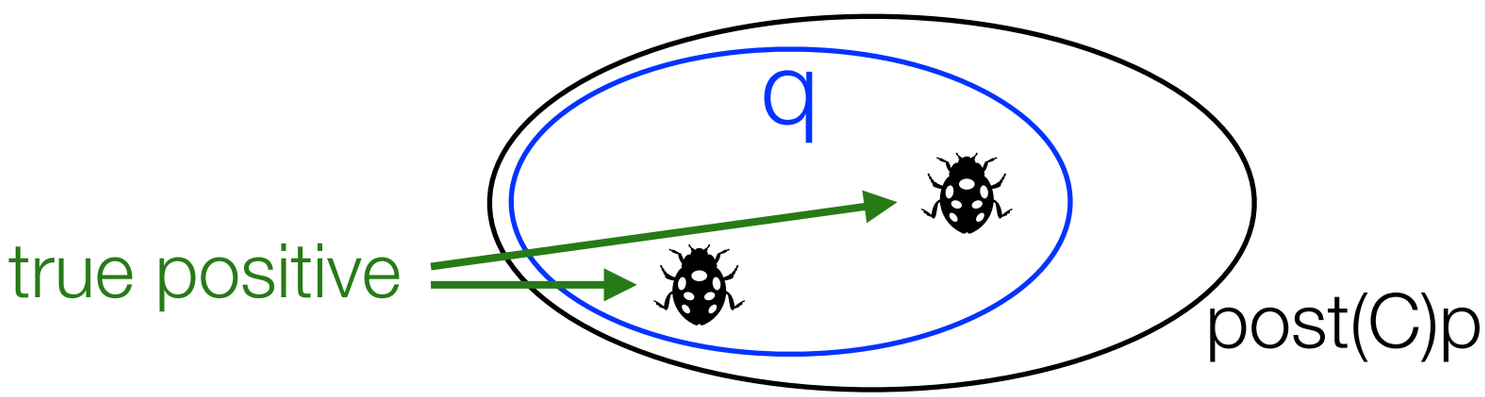
Hoare triples {p} C {q} *iff* post(C)p ⊆ q

q *over-approximates* post(C)p



Incorrectness triples [p] C [q] *iff* post(C)p ⊇ q

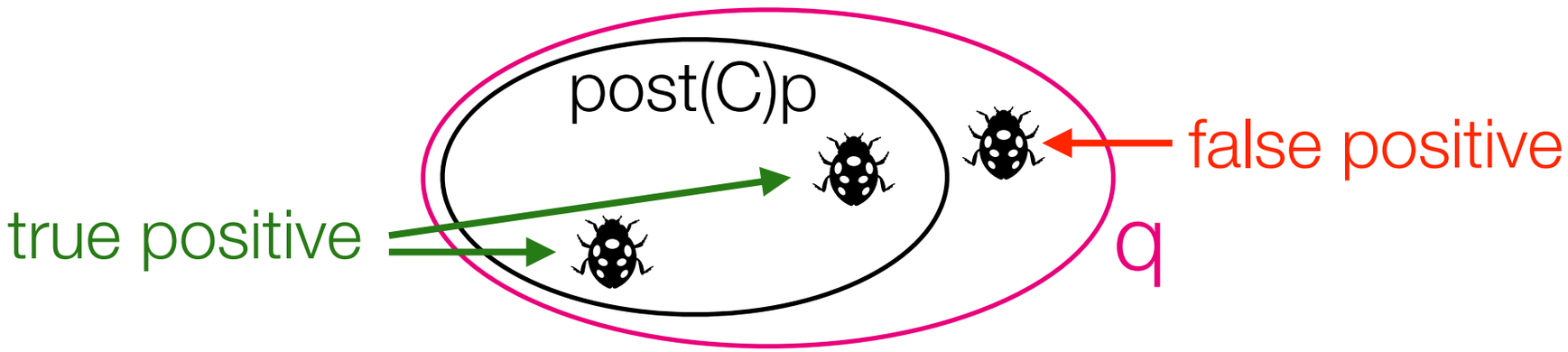
q *under-approximates* post(C)p



Incorrectness Logic (IL)

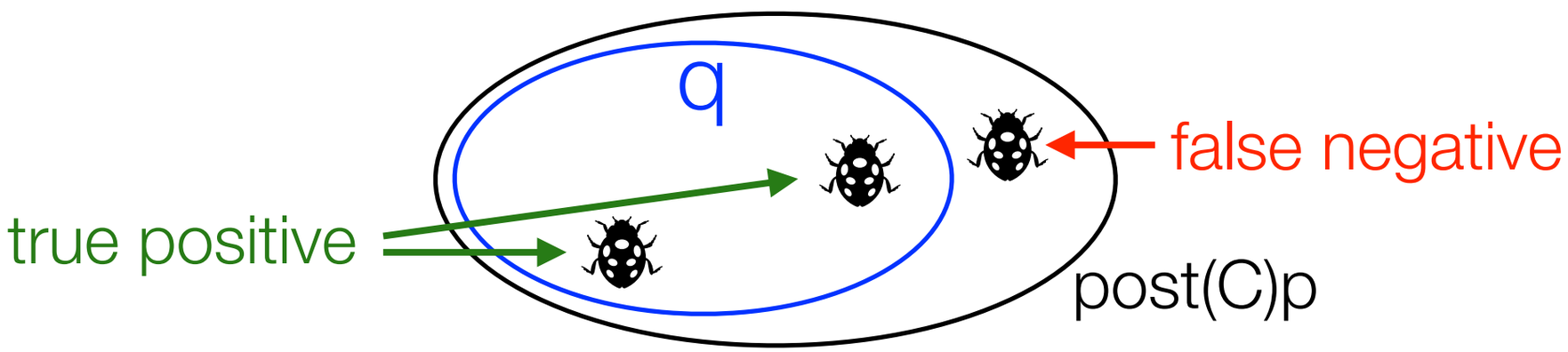
Hoare triples {p} C {q} *iff* post(C)p ⊆ q

q *over-approximates* post(C)p



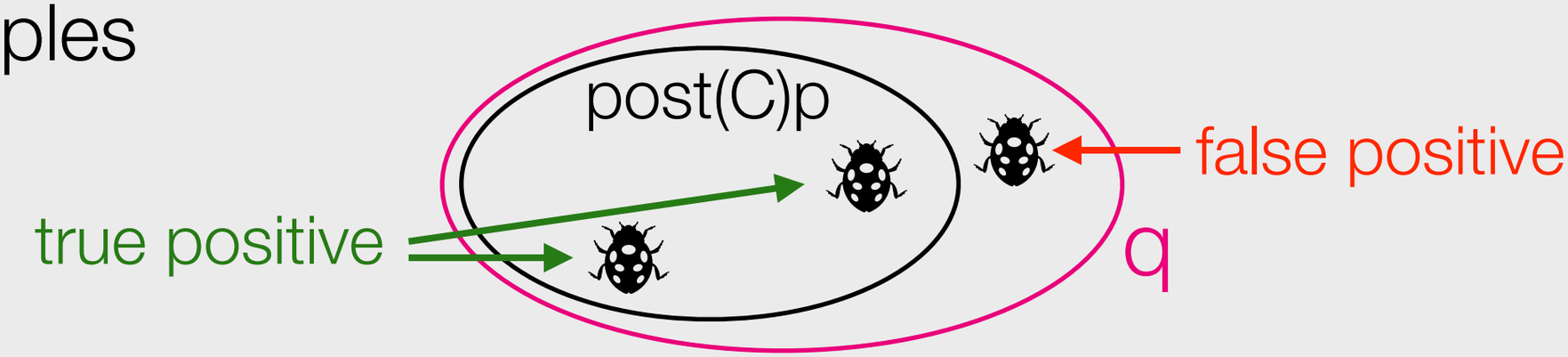
Incorrectness triples [p] C [q] *iff* post(C)p ⊇ q

q *under-approximates* post(C)p



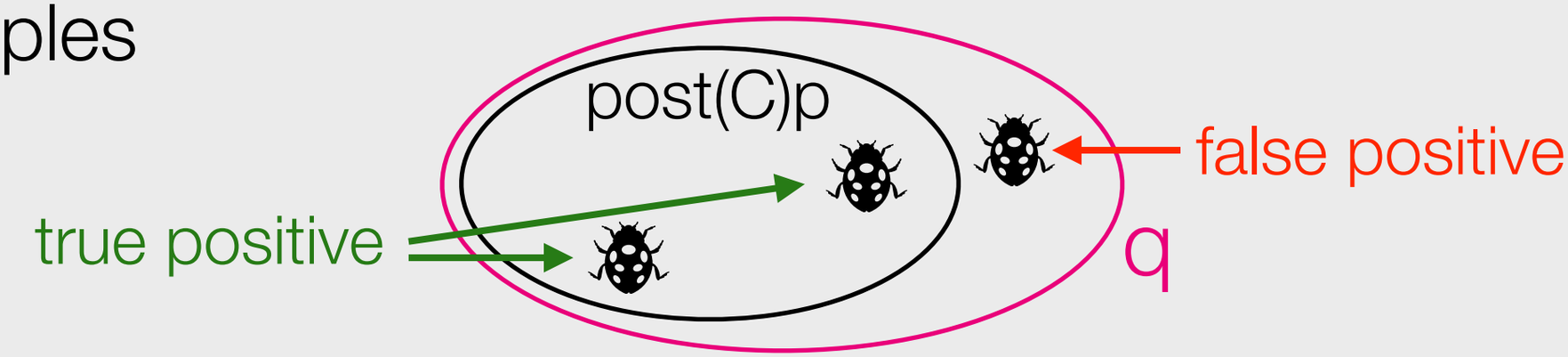
Incorrectness Logic (IL)

Hoare triples



Incorrectness Logic (IL)

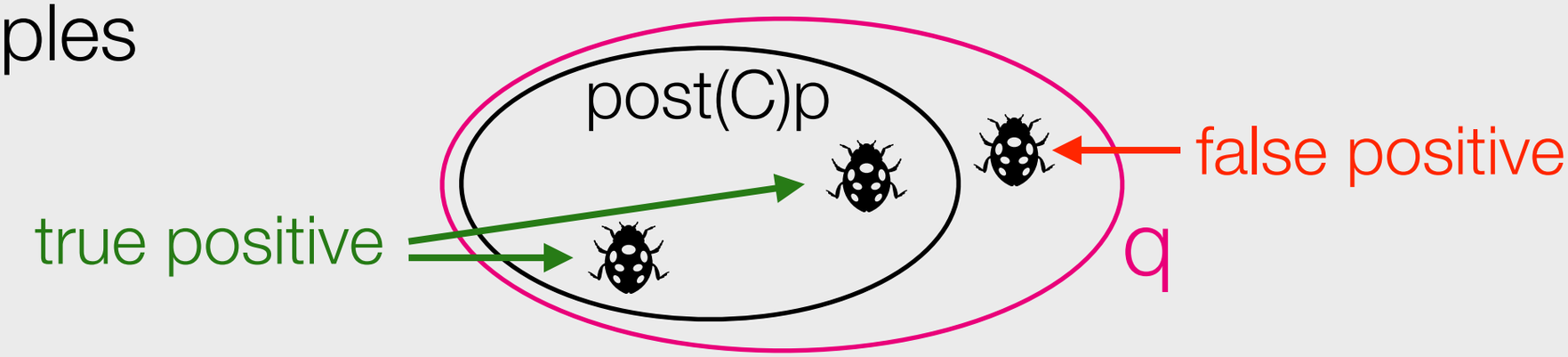
Hoare triples



False positive: Reported bugs may not be bugs (Infer @Facebook)

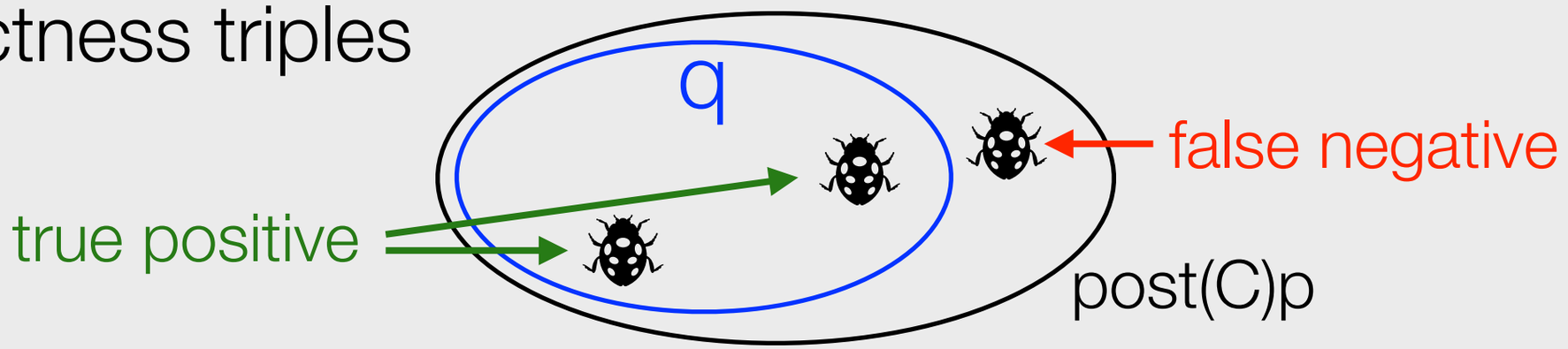
Incorrectness Logic (IL)

Hoare triples



False positive: Reported bugs may not be bugs (Infer @Facebook)

Incorrectness triples



No false positive: Bugs reported are definitely bugs (Pulse, RacerD @Facebook)

Incorrectness Logic (IL)

[p] C [ε: q]

ε: exit condition

ok: normal execution

er : erroneous execution

Incorrectness Logic (IL)

[p] C [ε: q]

ε: exit condition

ok: normal execution

er : erroneous execution

[true] x:=y [ok: x=y]

Incorrectness Logic (IL)

[p] C [ε: q]

ε: exit condition

ok: normal execution

er : erroneous execution

[true] x:=y [ok: x=y]

[y=v] x:=y [ok: x=y=v]

Incorrectness Logic (IL)

[p] C [ε: q]

ε: exit condition

ok: normal execution

er : erroneous execution

[true] x:=y [ok: x=y]

[y=v] x:=y [ok: x=y=v]

[p] error( ) [er: p]

Incorrectness Logic (IL)

[p] C [ε: q]

ε: exit condition

ok: normal execution

er : erroneous execution

[true] x:=y [ok: x=y]

[p] error( ) [er: p]

[y=v] x:=y [ok: x=y=v]

[p] x:=y [ok: ∃x'. p[x'/x] ∧ x=y]

Incorrectness Logic (IL)

$$[p]\ C\ [\varepsilon\colon q] \quad \textit{iff} \quad \mathrm{post}(C, \varepsilon)p \supseteq q$$

Equivalent Definition (reachability)

$$[p]\ C\ [\varepsilon\colon q] \quad \textit{iff} \quad \forall\, s \in q.\ \exists\, s' \in p.\ (s', s) \in [C]\varepsilon$$

Incorrectness Logic (IL)

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

```
1   /* presumes: [z==11] */
2      if (x is even) {
3         if (y is odd) {
4            z=42
5      }  }
6   /* achieves: [z==42] */
```

(Slide courtesy of Peter O'Hearn)

Incorrectness Logic (IL)

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

[p]

[C]

[ok:q]

```
1   /* presumes: [z==11] */
2   if (x is even) {
3       if (y is odd) {
4           z=42
5   } }
6   /* achieves: [z==42] */
```

(Slide courtesy of Peter O'Hearn)

Incorrectness Logic (IL)

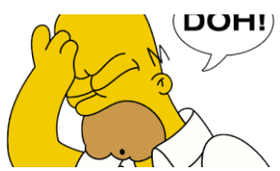
[p] C [ε: q] *iff* ∀ s ∈ q. ∃ s' ∈ p. (s',s) ∈ [C]ε

[p]

[C]

[ok:q]

```
1   /* presumes: [z==11] */
2   if (x is even) {
3       if (y is odd) {
4           z=42
5   } }
6   /* achieves: [z==42] */
```



[z:42, x:1, y:2]

(Slide courtesy of Peter O'Hearn)

$$[p]\ C\ [\varepsilon\colon q] \quad \textit{iff} \quad \forall\, s \in q.\ \exists\, s' \in p.\ (s', s) \in [C]\varepsilon$$

(Slide courtesy of Peter O’Hearn)

$$[p]\ C\ [\varepsilon: q] \quad \textit{iff} \quad \forall\, s \in q.\ \exists\, s' \in p.\ (s', s) \in [C]\varepsilon$$

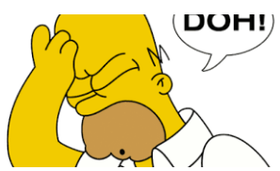
```
1   /* presumes: [z==11] */
2      if (x is even) {
3         if (y is odd) {
4            z=42
5      }  }
6   /* achieves: [z==42 && (x is even) && (y is odd) ] */
```

(Slide courtesy of Peter O'Hearn)

Incorrectness Logic (IL)

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

```
1   /* presumes: [z==11] */
2      if (x is even) {
3         if (y is odd) {
4            z=42
5      }  }
6   /* achieves: [z==42] */
```



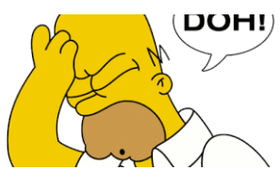
[z:42, x:1, y:2]

(Slide courtesy of Peter O'Hearn)

Incorrectness Logic (IL)

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

```
1   /* presumes: [z==11] */
2      if (x is even) {
3         if (y is odd) {
4            z=42
5      }  }
6   /* achieves: [z==42] */
```



[z:42, x:1, y:2]

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

(Slide courtesy of Peter O'Hearn)

Incorrectness Logic (IL)

$\{p\}\ C\ \{q\}$ *iff* $\text{post}(C)p \subseteq q$

$$\frac{p \Rightarrow p' \quad \{p'\}\ C\ \{q'\} \quad q' \Rightarrow q}{\{p\}\ C\ \{q\}}\ \text{(HL-Cons)}$$

Incorrectness Logic (IL)

$\{p\}\ C\ \{q\}$ *iff* $\text{post}(C)p \subseteq q$

$$\frac{p \Rightarrow p' \quad \{p'\}\ C\ \{q'\} \quad q' \Rightarrow q}{\{p\}\ C\ \{q\}}\ \text{(HL-Cons)}$$

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

$$\frac{p \Leftarrow p' \quad [p']\ C\ [\varepsilon\colon q'] \quad q' \Leftarrow q}{[p]\ C\ [\varepsilon\colon q]}\ \text{(IL-Cons)}$$

Incorrectness Logic (IL)

$$\frac{p \Rightarrow p' \quad \{p'\}\ C\ \{q'\} \quad q' \Rightarrow q}{\{p\}\ C\ \{q\}}\ \text{(HL-Cons)}$$

$$\frac{p \Leftarrow p' \quad [p']\ C\ [\varepsilon : q'] \quad q' \Leftarrow q}{[p]\ C\ [\varepsilon : q]}\ \text{(IL-Cons)}$$

Incorrectness Logic (IL)

$$\frac{p \Rightarrow p' \quad \{p'\}\ C\ \{q'\} \quad q' \Rightarrow q}{\{p\}\ C\ \{q\}}\ \text{(HL-Cons)}$$

$$\frac{\{p\}\ C\ \{q \wedge r\}}{\{p\}\ C\ \{q\}}$$ ✔

$$\frac{p \Leftarrow p' \quad [p']\ C\ [\varepsilon: q'] \quad q' \Leftarrow q}{[p]\ C\ [\varepsilon: q]}\ \text{(IL-Cons)}$$

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

Incorrectness Logic (IL)

$$\frac{p \Rightarrow p' \quad \{p'\}\ C\ \{q'\} \quad q' \Rightarrow q}{\{p\}\ C\ \{q\}}\ \text{(HL-Cons)}$$

$$\frac{\{p\}\ C\ \{q \wedge r\}}{\{p\}\ C\ \{q\}}$$ ✔

$$\frac{p \Leftarrow p' \quad [p']\ C\ [\varepsilon\colon q'] \quad q' \Leftarrow q}{[p]\ C\ [\varepsilon\colon q]}\ \text{(IL-Cons)}$$

$$\frac{[p]\ C\ [\varepsilon\colon q \wedge r]}{[p]\ C\ [\varepsilon\colon q]}$$ ✗

$$\frac{[p]\ C\ [\varepsilon\colon q \vee r]}{[p]\ C\ [\varepsilon\colon q]}$$ ✔

Incorrectness Logic (IL)

$$\frac{p \Rightarrow p' \quad \{p'\}\ C\ \{q'\} \quad q' \Rightarrow q}{\{p\}\ C\ \{q\}}\ \text{(HL-Cons)}$$

$$\frac{\{p\}\ C\ \{q \wedge r\}}{\{p\}\ C\ \{q\}}\ \checkmark \qquad \frac{\{p\}\ C\ \{q \vee r\}}{\{p\}\ C\ \{q\}}\ \times$$

$$\frac{p \Leftarrow p' \quad [p']\ C\ [\varepsilon: q'] \quad q' \Leftarrow q}{[p]\ C\ [\varepsilon: q]}\ \text{(IL-Cons)}$$

$$\frac{[p]\ C\ [\varepsilon: q \wedge r]}{[p]\ C\ [\varepsilon: q]}\ \times \qquad \frac{[p]\ C\ [\varepsilon: q \vee r]}{[p]\ C\ [\varepsilon: q]}\ \checkmark$$

Incorrectness Logic (IL)

$$\frac{p \Rightarrow p' \quad \{p'\}\ C\ \{q'\} \quad q' \Rightarrow q}{\{p\}\ C\ \{q\}}\ \text{(HL-Cons)}$$

$$\frac{\{p\}\ C\ \{q \wedge r\}}{\{p\}\ C\ \{q\}}\ \checkmark \qquad \frac{\{p\}\ C\ \{q \vee r\}}{\{p\}\ C\ \{q\}}\ \times$$

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

Ignore Paths

$$\frac{p \Leftarrow p' \quad [p']\ C\ [\varepsilon : q']\ \quad q' \Leftarrow q}{[p]\ C\ [\varepsilon : q]}\ \text{(IL-Cons)}$$

$$\frac{[p]\ C\ [\varepsilon : q \wedge r]}{[p]\ C\ [\varepsilon : q]}\ \times \qquad \checkmark$$

Infer.Pulse

- Analyzer for C++ lifetimes, numbers on 100s kLOC codebase
- 20 disjunct limit versus 50 disjuncts (5 unrollings each)
- 20 is 2.75x wall clock faster than 50
- 3.1x user time faster
- 20 find 97% of issues of 50

(Slide courtesy of Peter O’Hearn)

A duality

For correctness reasoning

You **get to forget** information as you go along a path, but you **must remember** all the paths.

For incorrectness reasoning

You **must remember** information as you go along a path, but you **get to forget** some of the paths

(Slide courtesy of Peter O’Hearn)

Incorrectness Logic Rules (Excerpt)

$$[p]\ C\ [\varepsilon\colon q] \quad \textit{iff} \quad \forall\, s \in q.\ \exists\, s' \in p.\ (s', s) \in [C]\varepsilon$$

$$\frac{[p]\ C_1\ [ok\colon r] \qquad [r]\ C_2\ [\varepsilon\colon q]}{[p]\ C_1; C_2\ [\varepsilon\colon q]}\ \text{(Seq — normal)}$$

Incorrectness Logic Rules (Excerpt)

$$[p]\ C\ [\varepsilon\colon q] \quad \textit{iff} \quad \forall\, s \in q.\ \exists\, s' \in p.\ (s', s) \in [C]\varepsilon$$

$$\frac{[p]\ C_1\ [ok\colon r] \qquad [r]\ C_2\ [\varepsilon\colon q]}{[p]\ C_1; C_2\ [\varepsilon\colon q]}\ \text{(Seq — normal)}$$

$$\frac{[p]\ C_1\ [er\colon q]}{[p]\ C_1; C_2\ [er\colon q]}\ \text{(Seq — short-circuit)}$$

Incorrectness Logic Rules (Excerpt)

$$[p]\ C\ [\varepsilon\colon q] \quad \textit{iff} \quad \forall\, s \in q.\ \exists\, s' \in p.\ (s', s) \in [C]\varepsilon$$

$$\frac{[p]\ C_1\ [ok\colon q]}{[p]\ C_1 + C_2\ [ok\colon q]}\ \text{(Choice — Left)}$$

$$\frac{[p]\ C_2\ [ok\colon q]}{[p]\ C_1 + C_2\ [ok\colon q]}\ \text{(Choice — Right)}$$

Incorrectness Logic Rules (Excerpt)

$$[p]\ C\ [\varepsilon\colon q] \quad \textit{iff} \quad \forall\, s \in q.\ \exists\, s' \in p.\ (s', s) \in [C]\varepsilon$$

$$[p]\ C^{\star}\ [ok\colon p] \qquad \text{(Iterate — Zero)}$$

Incorrectness Logic Rules (Excerpt)

$$[p]\ C\ [\varepsilon\colon q] \quad \textit{iff} \quad \forall\, s \in q.\ \exists\, s' \in p.\ (s', s) \in [C]\varepsilon$$

$$[p]\ C^{\star}\ [ok\colon p] \qquad \text{(Iterate — Zero)}$$

$$\frac{[p]\ C^{\star};\ C\ [ok\colon q]}{[p]\ C^{\star}\ [ok\colon q]} \qquad \text{(Iterate — Non-zero)}$$

Incorrectness Logic Rules (Excerpt)

$$[p]\ C\ [\varepsilon\colon q] \quad \textit{iff} \quad \forall\, s \in q.\ \exists\, s' \in p.\ (s', s) \in [C]\varepsilon$$

$$[p]\ C^{\bigstar}\ [ok\colon p] \qquad \text{(Iterate — Zero)}$$

$$\frac{[p]\ C^{\bigstar};\ C\ [ok\colon q]}{[p]\ C^{\bigstar}\ [ok\colon q]} \qquad \text{(Iterate — Non-zero)}$$

$$\frac{\forall n \in \mathbb{N}.\ \ [p(n)]\ C\ [ok\colon p(n+1)]}{[p(0)]\ C^{\bigstar}\ [ok\colon \exists n.\ p(n)]} \qquad \text{(Iterate — Variant)}$$

False Assertions in HL and IL

{p} C {q} *iff* post(C)p ⊆ q

{false} C {q} ✔ (vacuous)

False Assertions in HL and IL

{p} C {q} *iff* post(C)p ⊆ q

{false} C {q} ✔ (vacuous)

{p} C {false} ✔ partial correctness (non-termination)

False Assertions in HL and IL

{p} C {q} *iff* post(C)p ⊆ q

{false} C {q} ✔ (vacuous)

{p} C {false} ✔ partial correctness (non-termination)

✗ total correctness (unless p ⇒ false)

False Assertions in HL and IL

{p} C {q} *iff* post(C)p ⊆ q

{false} C {q} ✔ (vacuous)

{p} C {false} ✔ partial correctness (non-termination)

✗ total correctness (unless p ⇒ false)

[p] C [ε: q] *iff* post(C, ε)p ⊇ q

[p] C [ε: false] ✔ (vacuous)

False Assertions in HL and IL

{p} C {q} *iff* post(C)p ⊆ q

{false} C {q} ✔ (vacuous)

{p} C {false} ✔ partial correctness (non-termination)

✗ total correctness (unless p ⇒ false)

[p] C [ε: q] *iff* post(C, ε)p ⊇ q

[p] C [ε: false] ✔ (vacuous)

[false] C [ε: q] ✗ (unless q ⇒ false)

IL: Soundness & Completeness

Theorem. IL is sound and complete:

[p] C [ε: q] is true iff it is provable

Part II.
Incorrectness **Separation** Logic (ISL)

What Is Separation Logic (SL)?

SL : ***Local*** & ***compositional*** reasoning via ***ownership*** & ***separation***

☛ ideal for heap-manipulating programs with ***aliasing***

What Is Separation Logic (SL)?

SL : ***Local*** & ***compositional*** reasoning via ***ownership*** & ***separation***

☛ ideal for heap-manipulating programs with ***aliasing***

```
[x]:= 1;
[y]:= 2;
[z]:= 3;
```

What Is Separation Logic (SL)?

SL : ***Local*** & ***compositional*** reasoning via ***ownership*** & ***separation***

☛ ideal for heap-manipulating programs with ***aliasing***

```
[x]:= 1;
[y]:= 2;
[z]:= 3;
```

post: {x = 1 ∧ y = 2 ∧ z = 3}

What Is Separation Logic (SL)?

SL : ***Local*** & ***compositional*** reasoning via ***ownership*** & ***separation***

☛ ideal for heap-manipulating programs with ***aliasing***

pre: $\{x \neq y \wedge x \neq z \wedge y \neq z\}$

```
[x]:= 1;
[y]:= 2;
[z]:= 3;
```

post: $\{x = 1 \wedge y = 2 \wedge z = 3\}$

What Is Separation Logic (SL)?

SL : ***Local*** & ***compositional*** reasoning via ***ownership*** & ***separation***

☛ ideal for heap-manipulating programs with ***aliasing***

pre: { $x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge \ldots$ }

```
[x1]:= 1;
[x2]:= 2;
…
[xn]:= n;
```

post: { $x_1 = 1 \wedge \ldots \wedge x_n = n$ }

What Is Separation Logic (SL)?

SL : ***Local*** & ***compositional*** reasoning via ***ownership*** & ***separation***

☛ ideal for heap-manipulating programs with ***aliasing***

pre: { $x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge \ldots$ }

```
[x1]:= 1;
[x2]:= 2;
…
[xn]:= n;
```

post: { $x_1 = 1 \wedge \ldots \wedge x_n = n$ }

n!/2 conjuncts !

What Is Separation Logic (SL)?

SL : ***Local*** & ***compositional*** reasoning via ***ownership*** & ***separation***

☛ ideal for heap-manipulating programs with ***aliasing***

pre: { x ↦ - ∗ y ↦ - ∗ z ↦ - }

```
[x]:= 1;
[y]:= 2;
[z]:= 3;
```

post: { x ↦ 1 ∗ y ↦ 2 ∗ z ↦ 3 }

What Is Separation Logic (SL)?

SL : ***Local*** & ***compositional*** reasoning via ***ownership*** & ***separation***

☛ ideal for heap-manipulating programs with ***aliasing***

pre: $\{ x \mapsto - * y \mapsto - * z \mapsto - \}$

```
[x]:= 1;
[y]:= 2;
[z]:= 3;
```

‘and ***separately***’

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

What Is Separation Logic (SL)?

SL : ***Local*** & ***compositional*** reasoning via ***ownership*** & ***separation***

☛ ideal for heap-manipulating programs with ***aliasing***

pre: $\{ x \mapsto - * y \mapsto - * z \mapsto - \}$

ownership of heap cell at x

```
[x]:= 1;
[y]:= 2;
[z]:= 3;
```

'and ***separately***'

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

What Is Separation Logic (SL)?

SL : ***Local*** & ***compositional*** reasoning via ***ownership*** & ***separation***

☛ ideal for heap-manipulating programs with ***aliasing***

pre: $\{ x \mapsto - * y \mapsto - * z \mapsto - \}$

```
[x]:= 1;
[y]:= 2;
[z]:= 3;
```

ownership of heap cell at x

'and ***separately***'

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

$\forall x,v,v'.\ x \mapsto v * x \mapsto v' \Rightarrow \text{false}$

The Essence of Separation Logic (SL)

Frame Rule

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

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

The Essence of Separation Logic (SL)

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

The Essence of Separation Logic (SL)

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

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

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

The Essence of Separation Logic (SL)

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

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

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

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

The Essence of Separation Logic (SL)

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 Essence of Separation Logic (SL)

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

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

The Essence of Separation Logic (SL)

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

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

$\{\ x \mapsto -\ *\ y \mapsto -\ *\ Z \mapsto -\ \}$

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

The Essence of Separation Logic (SL)

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

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

$\{\ x \mapsto - * y \mapsto - * Z \mapsto -\ \}$

write axiom ↑

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

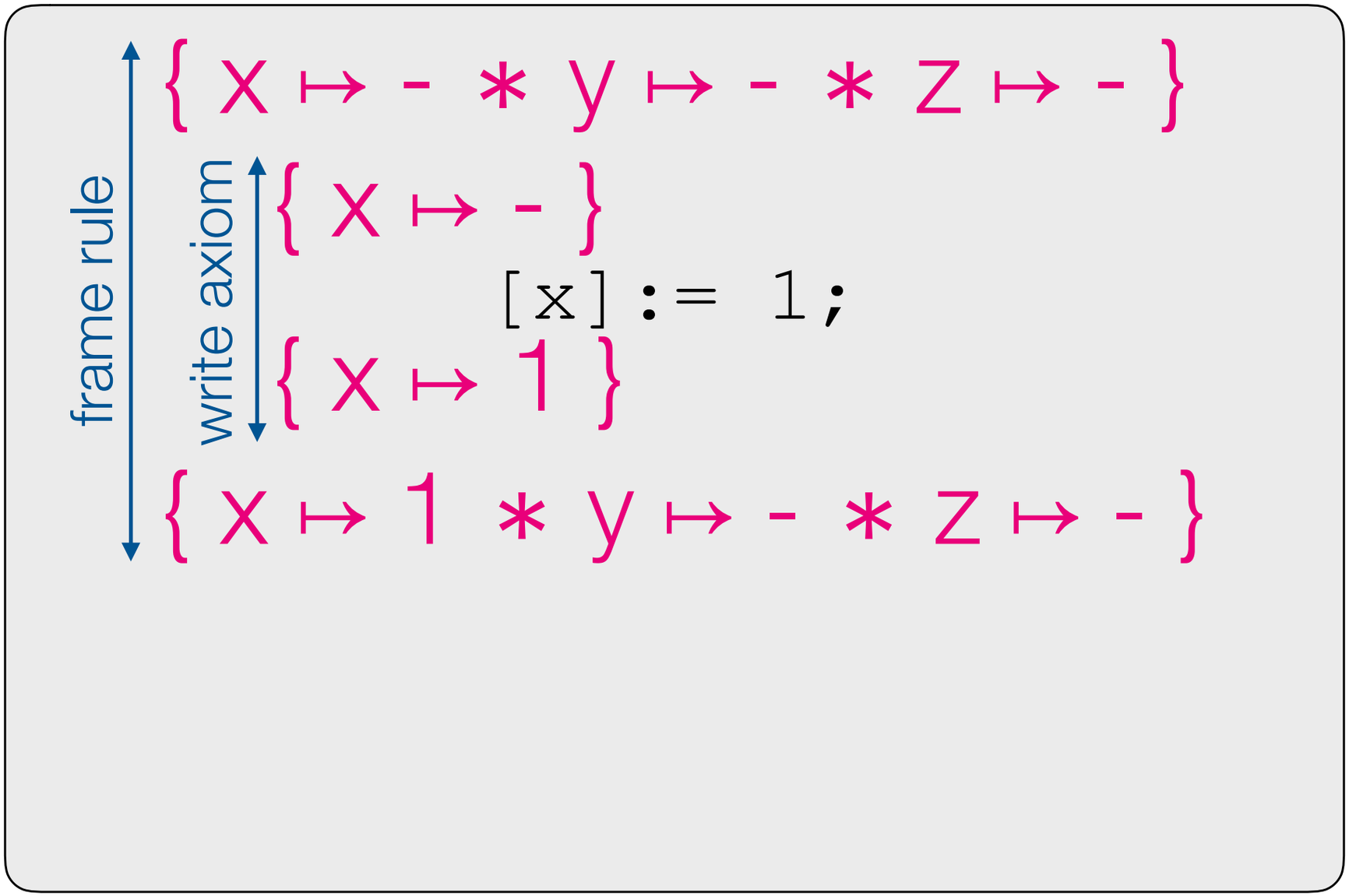
```
[x]:= 1;
```

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

The Essence of Separation Logic (SL)

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

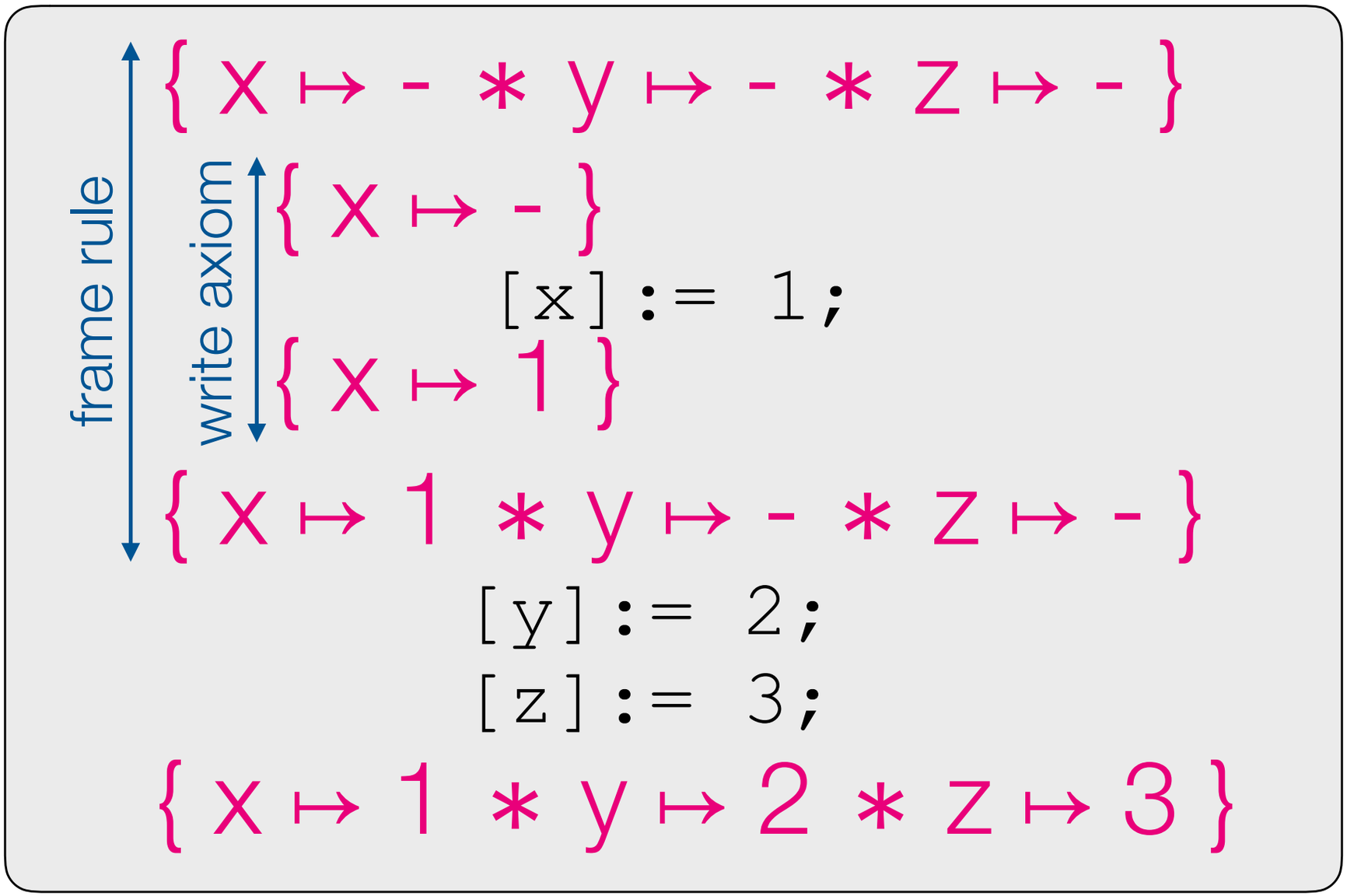
$\{x \mapsto -\}\ [x] := v\ \{x \mapsto v\}$



The Essence of Separation Logic (SL)

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

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



SL: Soundness & Completeness

Theorem. SL is sound:

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

SL: Soundness & Completeness

Theorem. SL is sound:

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

SL ***Completeness***:

{p} C {q} is true $\overset{?}{\Rightarrow}$ {p} C {q} is provable

SL: Soundness & Completeness

Theorem. SL is sound:

{p} C {q} is provable ⇒ {p} C {q} is true

SL ***Completeness***:

{p} C {q} is true $\overset{?}{\Rightarrow}$ {p} C {q} is provable ✗

SL: Soundness & Completeness

Theorem. SL is sound:

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

SL ***Completeness***:

{p} C {q} is true $\overset{?}{\Rightarrow}$ {p} C {q} is provable ✗

SL does not satisfy ***footprint property***

Incorrectness Separation Logic (ISL)

IL

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

SL

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

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

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

Incorrectness Separation Logic (ISL)

IL

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

SL

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

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

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

ISL

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

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

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

ISL: Local Axioms (First Attempt)

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

ISL: Local Axioms (First Attempt)

[x ↦ v’] [x]:= v [ok: x ↦ v] [x=null] [x]:= v [er: x=null]

ISL: Local Axioms (First Attempt)

[x ↦ v'] [x]:= v [ok: x ↦ v]

[x=null] [x]:= v [er: x=null]

null-pointer dereference error

ISL: Local Axioms (First Attempt)

[x ↦ v'] [x]:= v [ok: x ↦ v]

[x=null] [x]:= v [er: x=null]

null-pointer dereference error

[x ↦ v] y:= [x] [ok: x ↦ v∧y=v]

[x=null] y:= [x] [er: x=null]

ISL: Local Axioms (First Attempt)

[x ↦ v'] [x]:= v [ok: x ↦ v]

[x=null] [x]:= v [er: x=null]

null-pointer dereference error

[x ↦ v] y:= [x] [ok: x ↦ v∧y=v]

[x=null] y:= [x] [er: x=null]

[emp] x:= alloc() [ok:∃l. l ↦ v ∧ x=l]

ISL: Local Axioms (First Attempt)

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

$[x=null]\ [x]:= v\ [er: x=null]$

null-pointer dereference error

$[x \mapsto v]\ y:= [x]\ [ok: x \mapsto v \wedge y=v]$

$[x=null]\ y:= [x]\ [er: x=null]$

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

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

$[x=null]\ free(x)\ [er: x=null]$

ISL: Local Axioms (First Attempt)

[x ↦ v'] [x]:= v [ok: x ↦ v]

[x=null] [x]:= v [er: x=null]

null-pointer dereference error

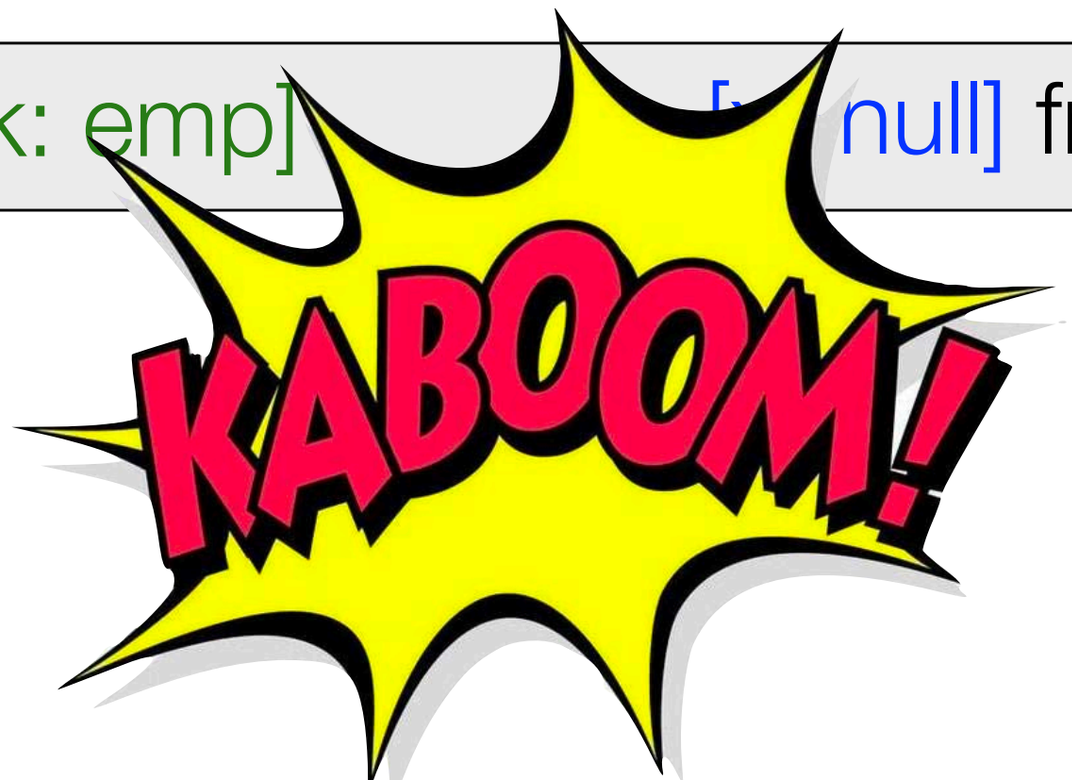
[x ↦ v] y:= [x] [ok: x ↦ v∧y=v]

[x=null] y:= [x] [er: x=null]

[emp] x:= alloc() [ok:∃l. l ↦ v ∧ x=l]

[x ↦ v] free(x) [ok: emp]

[x=null] free(x) [er: x=null]



ISL: Local Axioms (First Attempt)

ISL

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

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

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

$[x \mapsto v]\ \text{free}(x)\ [\text{ok}\colon \text{emp}]$

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

ISL: Local Axioms (First Attempt)

ISL

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

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

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

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

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

ISL: Local Axioms (First Attempt)

ISL

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

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

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

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

$$\frac{}{[\text{false}]\ \text{free}(x)\ [\text{ok: } x \mapsto v]}\ \text{(Cons)}$$

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

ISL: Local Axioms (First Attempt)

ISL

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

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

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

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

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

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

$[\text{false}]\ C\ [\varepsilon: q]$ ✗ (unless $q \Rightarrow \text{false}$)

ISL: Local Axioms (First Attempt)

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

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

Frame preservations:

All r $*$-compatible with q must be $*$-compatible with p

ISL: Local Axioms (First Attempt)

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

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

Frame preservations:

All r $*$-compatible with q must be $*$-compatible with p

$[x \mapsto v']\ [x]:= v\ [ok: x \mapsto v]$ $\qquad$ $*$-comp(q) = $*$-comp(p)

ISL: Local Axioms (First Attempt)

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

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

Frame preservations:

All r $*$-compatible with q must be $*$-compatible with p

$[x \mapsto v']\ [x]:= v\ [ok: x \mapsto v]$	$*$-comp(q) = $*$-comp(p)
$[x \mapsto v]\ y:= [x]\ [ok: x \mapsto v \wedge y=v]$	$*$-comp(q) = $*$-comp(p)
$[emp]\ x:= alloc()\ [ok: \exists l.\ l \mapsto v \wedge x=l]$	$*$-comp(q) $\subset$ $*$-comp(p)

ISL: Local Axioms (First Attempt)

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

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

Frame preservations:

All r $*$-compatible with q must be $*$-compatible with p

$[x \mapsto v']\ [x]:= v\ [ok: x \mapsto v]$	$*$-comp(q) = $*$-comp(p)
$[x \mapsto v]\ y:= [x]\ [ok: x \mapsto v \wedge y=v]$	$*$-comp(q) = $*$-comp(p)
$[emp]\ x:= alloc()\ [ok: \exists l.\ l \mapsto v \wedge x=l]$	$*$-comp(q) $\subset$ $*$-comp(p)
$[x \mapsto v]\ free(x)\ [ok: emp]$	$*$-comp(q) $\supset$ $*$-comp(p) $\Uplus$ $x \mapsto v$

ISL: Local Axioms (First Attempt)

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

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

Frame preservations:

All r $*$-compatible with q must be $*$-compatible with p

$[x \mapsto v']\ [x]:= v\ [ok: x \mapsto v]$	$*$-comp(q) = $*$-comp(p)
$[x \mapsto v]\ y:= [x]\ [ok: x \mapsto v \wedge y=v]$	$*$-comp(q) = $*$-comp(p)
$[emp]\ x:= alloc()\ [ok: \exists l.\ l \mapsto v \wedge x=l]$	$*$-comp(q) $\subset$ $*$-comp(p)
$[x \mapsto v]\ free(x)\ [ok: emp]$	$*$-comp(q) $\supset$ $*$-comp(p) $\uplus$ $x \mapsto v$ / $*$-comp(p) $\uplus$ $x \mapsto v$

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

ISL: Local Axioms (First Attempt)

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

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

Frame preservations:

All r $*$-compatible with q must be $*$-compatible with p

$[x \mapsto v']\ [x]:= v\ [ok: x \mapsto v]$	$*$-comp(q) = $*$-comp(p)
$[x \mapsto v]\ y:= [x]\ [ok: x \mapsto v \wedge y=v]$	$*$-comp(q) = $*$-comp(p)
$[emp]\ x:= alloc()\ [ok: \exists l.\ l \mapsto\text{-} \wedge x=l]$	$*$-comp(q) $\subset$ $*$-comp(p)
$[x \mapsto v]\ free(x)\ [ok: emp]$	$*$-comp(q) $\supset$ $*$-comp(p)

forgetting (shrinking) resource

ISL: Local Axioms (First Attempt)

Solution:

Track Deallocated Locations!

Solution: Track Deallocated Locations!

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

Solution: Track Deallocated Locations!

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

Solution: Track Deallocated Locations!

Solution: Track Deallocated Locations!

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

x is ***deallocated***

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

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

Solution: Track Deallocated Locations!

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

x is ***deallocated***

$x \mapsto v * x \mapsto v' \Leftrightarrow$ false

$x \mapsto v * emp \Leftrightarrow x \mapsto v$

$x \mapsto v * x \not\mapsto \Leftrightarrow$ false

$x \not\mapsto * x \not\mapsto \Leftrightarrow$ false

Solution: Track Deallocated Locations!

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

Solution: Track Deallocated Locations!

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

Solution: Track Deallocated Locations!

$$\frac{[x \mapsto v]\ \text{free(x)}\ [\text{ok: } x \not\mapsto\ ]}{[x \mapsto v * x \mapsto v\ ]\ \text{free(x)}\ [\text{ok: } x \not\mapsto\ * x \mapsto v]}$$

$$[\text{false}]\ \text{free(x)}\ [\text{ok: false}]$$ ✔

$$[p]\ C\ [\varepsilon\text{: } q] \quad \textit{iff} \quad \forall\ s \in q.\ \exists\ s' \in p.\ (s', s) \in [C]\varepsilon$$

$$[p]\ C\ [\varepsilon\text{: false}]$$ ✔ (vacuous)

ISL: Local Axioms

$[x \mapsto v]$ free(x) [ok: $x \not\mapsto$] $\qquad$ [x=null] free(x) [er: x=null]

ISL: Local Axioms

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

$[x{=}null]$ free(x) [er: x=null]

$[x \not\mapsto]$ free(x) [er: $x \not\mapsto$]

ISL: Local Axioms

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

$[x{=}null]$ free(x) [er: x=null]

$[x \not\mapsto]$ free(x) [er: $x \not\mapsto$]

use-after-free error

ISL: Local Axioms

[x ↦ v] free(x) [ok: x ↦̸]

[x=null] free(x) [er: x=null]

[x ↦̸] free(x) [er: x ↦̸]

use-after-free error

[x ↦ v'] [x]:= v [ok: x ↦ v]

[x=null] [x]:= v [er: x=null]

[x ↦̸] [x]:= v [er: x ↦̸]

ISL: Local Axioms

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

$[x=null]$ free(x) $[er: x=null]$

$[x \not\mapsto]$ free(x) $[er: x \not\mapsto]$

use-after-free error

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

$[x=null]$ [x]:= v $[er: x=null]$

$[x \not\mapsto]$ [x]:= v $[er: x \not\mapsto]$

$[x \mapsto v]$ y:= [x] $[ok: x \mapsto v \wedge y=v]$

$[x=null]$ y:= [x] $[er: x=null]$

$[x \not\mapsto]$ y:= [x] $[er: x \not\mapsto]$

ISL: Local Axioms

[x ↦ v] free(x) [ok: x ↦̸]

[x=null] free(x) [er: x=null]

[x ↦̸] free(x) [er: x ↦̸]

use-after-free error

[x ↦ v'] [x]:= v [ok: x ↦ v]

[x=null] [x]:= v [er: x=null]

[x ↦̸] [x]:= v [er: x ↦̸]

[x ↦ v] y:= [x] [ok: x ↦ v∧y=v]

[x=null] y:= [x] [er: x=null]

[x ↦̸] y:= [x] [er: x ↦̸]

[emp] x:= alloc() [ok:∃l. l ↦ v ∧ x=l]

ISL: Local Axioms

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

$[x=null]$ free(x) $[er: x=null]$

$[x \not\mapsto]$ free(x) $[er: x \not\mapsto]$

use-after-free error

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

$[x=null]$ [x]:= v $[er: x=null]$

$[x \not\mapsto]$ [x]:= v $[er: x \not\mapsto]$

$[x \mapsto v]$ y:= [x] $[ok: x \mapsto v \wedge y=v]$

$[x=null]$ y:= [x] $[er: x=null]$

$[x \not\mapsto]$ y:= [x] $[er: x \not\mapsto]$

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

$[y \not\mapsto]$ x:= alloc() $[ok: y \mapsto v \wedge x=y]$

Based on ISL!

Infer.Pulse

- Analyzer for C++ lifetimes, numbers on 100s kLOC codebase
- 20 disjunct limit versus 50 disjuncts (5 unrollings each)
- 20 is 2.75x wall clock faster than 50
- 3.1x user time faster
- 20 find 97% of issues of 50

(Slide courtesy of Peter O’Hearn)

SL: Soundness & Completeness

Theorem. SL is sound:

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

SL ***Completeness***:

{p} C {q} is true $\overset{?}{\Rightarrow}$ {p} C {q} is provable ✗

SL does not satisfy ***footprint property***

SL: Footprint Property

Given C:

SL: Footprint Property

Given C:

1. Derive spec of C using ***local axioms***

SL: Footprint Property

Given C:

1. Derive spec of C using ***local axioms***
2. Extend with ***frame***

SL: Footprint Property

Given C:

1. Derive spec of C using ***local axioms***
2. Extend with ***frame***

   $\Rightarrow$ ***complete*** C spec

SL: Footprint Property

Given C:

1. Derive spec of C using ***local axioms***
2. Extend with ***frame***

   $\Rightarrow$ ***complete*** C spec

Example.

[x]:= 1; a:= [x];

SL: Footprint Property

Given C:

1. Derive spec of C using ***local axioms***
2. Extend with ***frame***

   $\Rightarrow$ ***complete*** C spec

Example.

$\{x \mapsto -\}$ [x]:= 1; a:= [x]; $\{x \mapsto 1 \wedge a=1\}$

SL: Footprint Property

Given C:

1. Derive spec of C using ***local axioms***
2. Extend with ***frame***

   $\Rightarrow$ ***complete*** C spec

Example.

$\{x \mapsto -\}$ [x]:= 1; a:= [x]; $\{x \mapsto 1 \wedge a=1\}$

$\{p * x \mapsto -\}$ [x]:= 1; a:= [x]; $\{p * x \mapsto 1 \wedge a=1\}$

SL: Footprint Property

Given C:

1. Derive spec of C using ***local axioms***
2. Extend with ***frame***

   $\Rightarrow$ ***complete*** C spec

Example.

$\{x \mapsto -\}$ [x]:= 1; a:= [x]; $\{x \mapsto 1 \wedge a=1\}$

$\{p * x \mapsto -\}$ [x]:= 1; a:= [x]; $\{p * x \mapsto 1 \wedge a=1\}$

complete specification ✔

SL: Footprint Property

Counter-example.

would like to derive:

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

SL: Footprint Property

Counter-example.

would like to derive:

{y ↦ -} x:= alloc(); free(x) { y ↦ - ∧ y ≠ x }

{emp} x:= alloc() {∃l. l ↦ -∧ x=l } {x ↦ -} free(x) { emp }

SL: Footprint Property

Counter-example.

would like to derive:

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

$\{emp\}$

x:= alloc();

$\{\exists l.\ l \mapsto - \wedge x{=}l \}$

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

SL: Footprint Property

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);

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

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

SL: Footprint Property

Counter-example.

would like to derive:

{y ↦ -} x:= alloc(); free(x) { y ↦ - ∧ y ≠ x }

{emp}

x:= alloc();

{∃l. l ↦ -∧ x=l }

frame y ↦ -

free(x);

{∃l. x=l }

{emp} x:= alloc() {∃l. l ↦ -∧ x=l } {x ↦ -} free(x) { emp }

SL: Footprint Property

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)
$\{\exists l.\ y \mapsto - \wedge x=l \}$

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

SL: Footprint Property

Counter-example.

would like to derive:

{y ↦ -} x:= alloc(); free(x) { y ↦ - ∧ y ≠ x }

{emp}
x:= alloc();
{∃l. l ↦ -∧ x=l }
free(x);
{∃l. x=l }

frame y ↦ -

strongest derivable spec

{y ↦ -}
x:= alloc(); free(x)
{∃l. y ↦ -∧ x=l }

incomplete specification ✗

{emp} x:= alloc() {∃l. l ↦ -∧ x=l } {x ↦ -} free(x) { emp }

SL: Footprint Property

Counter-example.

would like to derive:

{y ↦ -} x:= alloc(); free(x) { y ↦ - ∧ y ≠ x }

{emp}
x:= alloc();
{∃l. l ↦ -∧ x=l }
free(x);
{∃l. x=l }

frame y ↦ -

strongest derivable spec

{y ↦ -}
x:= alloc(); free(x)
{∃l. y ↦ -∧ x=l }

incomplete specification ✗

lossy: forgets that x held a location

{emp} x:= alloc() {∃l. l ↦ -∧ x=l }

{x ↦ -} free(x) { emp }

ISL: Footprint Property

[x ↦ v'] [x]:= v [ok: x ↦ v]	∗-comp(q) = ∗-comp(p)
[x ↦ v] y:= [x] [ok: x ↦ v∧y=v]	∗-comp(q) = ∗-comp(p)
[emp] x:= alloc() [ok:∃l. l ↦- ∧ x=l]	∗-comp(q) ⊂ ∗-comp(p)
[x ↦ v] free(x) [ok: x↛]	∗-comp(q) = ∗-comp(p)

ISL: Footprint Property

$[x \mapsto v']$ [x]:= v $[ok: x \mapsto v]$ $*\text{-comp}(q) = *\text{-comp}(p)$

$[x \mapsto v]$ y:= [x] $[ok: x \mapsto v \wedge y=v]$ $*\text{-comp}(q) = *\text{-comp}(p)$

[emp] x:= alloc() $[ok: \exists l.\ l \mapsto\text{-} \wedge x=l]$ $*\text{-comp}(q) \subset *\text{-comp}(p)$

$[x \mapsto v]$ free(x) $[ok: x \not\mapsto]$ $*\text{-comp}(q) = *\text{-comp}(p)$

Observation

resource ***may grow***, but ***never shrinks!***

⇓

resource monotonicity

ISL: Footprint Property

[x ↦ v'] [x]:= v [ok: x ↦ v]	$*$-comp(q)	= $*$-comp(p)
[x ↦ v] y:= [x] [ok: x ↦ v∧y=v]	$*$-comp(q)	= $*$-comp(p)
[emp] x:= alloc() [ok:∃l. l ↦- ∧ x=l]	$*$-comp(q)	⊂ $*$-comp(p)
[x ↦ v] free(x) [ok: x↛]	$*$-comp(q)	= $*$-comp(p)

Observation

resource ***may grow***, but ***never shrinks!***

⇓

resource monotonicity

⇓

footprint property

ISL: Footprint Property

SL

{y ↦ -} x:= alloc(); free(x) { y ↦ -∧ y ≠ x }

✗

ISL: Footprint Property

SL

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

✗

ISL

$[y \mapsto -]$ x:= alloc(); free(x) $[\text{ok}: y \mapsto - * x \not\mapsto]$

$[\text{emp}]$ x:= alloc() $[\text{ok}: \exists l.\ l \mapsto - \wedge x=l]$ $[x \mapsto -]$ free(x) $[\text{ok}: x \not\mapsto]$

ISL: Footprint Property

SL

{y ↦ -} x:= alloc(); free(x) { y ↦ -∧ y ≠ x }

✗

ISL

[y ↦ -] x:= alloc(); free(x) [ok: y ↦ - ∗ x ↦̸]

⇓

ISL

[y ↦ -] x:= alloc(); free(x) [ok: y ↦ - ∗ x ↦̸ ∧ y ≠ x]

✔

[emp] x:= alloc() [ok: ∃l. l ↦ -∧ x=l] [x ↦ -] free(x) [ok: x ↦̸]

Part III.
Concurrent ISL (CISL)
Teaser

Concurrent Incorrectness Separation Logic (CISL)

ISL

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

Concurrent Incorrectness Separation Logic (CISL)

ISL

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

CSL

$$\frac{\{p_1\}\ C_1\ \{q_1\} \qquad \{p_2\}\ C_2\ \{q_2\}}{\{p_1 * p_2\}\ C_1 \,\|\, C_2\ \{q_1 * q_2\}}$$

Concurrent Incorrectness Separation Logic (CISL)

ISL

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

CSL

$$\frac{\{p_1\}\ C_1\ \{q_1\} \qquad \{p_2\}\ C_2\ \{q_2\}}{\{p_1 * p_2\}\ C_1 \parallel C_2\ \{q_1 * q_2\}}$$

CISL

$$\frac{[p_1]\ C_1\ [\varepsilon\colon q_1] \qquad [p_2]\ C_2\ [\varepsilon\colon q_2]}{[p_1 * p_2]\ C_1 \parallel C_2\ [\varepsilon\colon q_1 * q_2]}$$

$$\frac{[p_1]\ C_1\ [\varepsilon : q_1] \qquad [p_2]\ C_2\ [\varepsilon : q_2]}{[p_1 * p_2]\ C_1 \parallel C_2\ [\varepsilon : q_1 * q_2]}\ \text{(Par)}$$

$$\frac{[p_1]\ C_1\ [\varepsilon: q_1] \qquad [p_2]\ C_2\ [\varepsilon: q_2]}{[p_1 * p_2]\ C_1 \,\|\, C_2\ [\varepsilon: q_1 * q_2]}\ \text{(Par)}$$

$$\frac{[p]\ C_{11}\ [ok: r] \qquad [r]\ C_{12} \,\|\, C_2\ [\varepsilon: q]}{[p]\ (C_{11}; C_{12}) \,\|\, C_2\ [\varepsilon: q]}\ \text{(ParInterleaveL)}$$

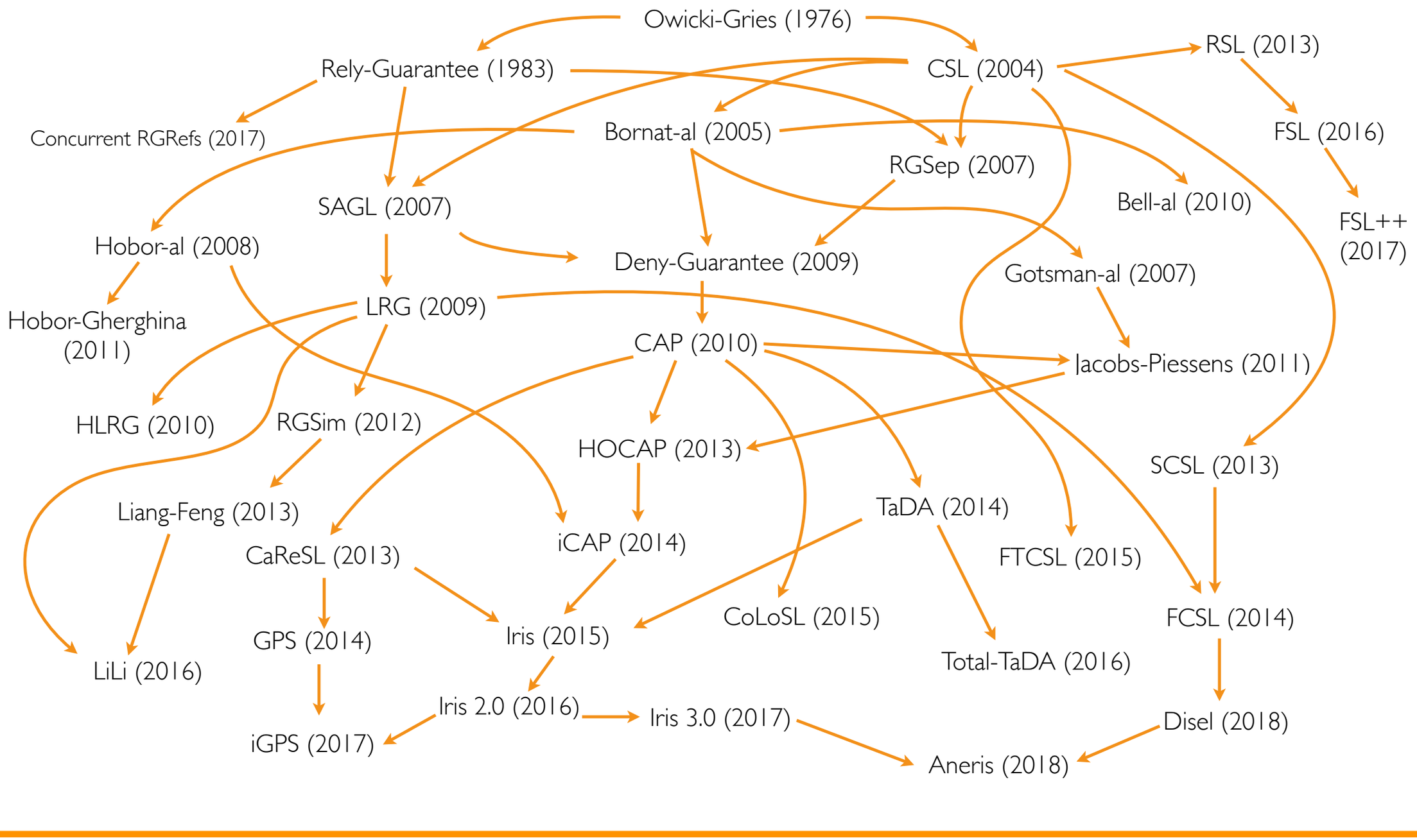
$$\frac{[p_1]\ C_1\ [\varepsilon: q_1] \quad [p_2]\ C_2\ [\varepsilon: q_2]}{[p_1 * p_2]\ C_1 \,\|\, C_2\ [\varepsilon: q_1 * q_2]}\ \text{(Par)}$$

$$\frac{[p]\ C_{11}\ [ok: r] \quad [r]\ C_{12} \,\|\, C_2\ [\varepsilon: q]}{[p]\ (C_{11}; C_{12}) \,\|\, C_2\ [\varepsilon: q]}\ \text{(ParInterleaveL)}$$

$$\frac{[p]\ C_{21}\ [ok: r] \quad [r]\ C_1 \,\|\, C_{22}\ [\varepsilon: q]}{[p]\ C_1 \,\|\, (C_{21}; C_{22})\ [\varepsilon: q]}\ \text{(ParInterleaveR)}$$

Which CSL?

CSL Family Tree



(Graph courtesy of Ilya Sergey)

CISL

A ***general*** framework

for

Concurrent ISL

CISL

A ***general*** framework
for
Concurrent ISL

instantiated for:

- ➡ race detection (à la RacerD)
- ➡ deadlock detection
- ➡ disjoint concurrency
- ➡ shared concurrency (à la CSL)
- ➡ …

Conclusions

- Incorrectness Logic (IL)
  - ➡ A rigorous foundation for bug catching
  - ➡ A unifying theory of testing and verification

Conclusions

- Incorrectness Logic (IL)
  - ➡ A rigorous foundation for bug catching
  - ➡ A unifying theory of testing and verification

- Incorrectness ***Separation*** Logic (ISL)
  - ➡ Combining IL and SL for ***compositional bug catching***
  - ➡ A monotonic model for frame preservation
  - ➡ Recovering the ***footprint property*** for completeness

Conclusions

- Incorrectness Logic (IL)
  - ➡ A rigorous foundation for bug catching
  - ➡ A unifying theory of testing and verification

- Incorrectness ***Separation*** Logic (ISL)
  - ➡ Combining IL and SL for ***compositional bug catching***
  - ➡ A monotonic model for frame preservation
  - ➡ Recovering the ***footprint property*** for completeness

- Future work
  - ➡ ***Concurrent*** Incorrectness Separation Logic (CISL)
    Extending ISL with concurrency
    Tools for deadlock detection, race detection, …

Conclusions

- Incorrectness Logic (IL)
  - ➡ A rigorous foundation for bug catching
  - ➡ A unifying theory of testing and verification

- Incorrectness ***Separation*** Logic (ISL)
  - ➡ Combining IL and SL for ***compositional bug catching***
  - ➡ A monotonic model for frame preservation
  - ➡ Recovering the ***footprint property*** for completeness

- Future work
  - ➡ ***Concurrent*** Incorrectness Separation Logic (CISL)
    Extending ISL with concurrency
    Tools for deadlock detection, race detection, …

Thank You for Listening!

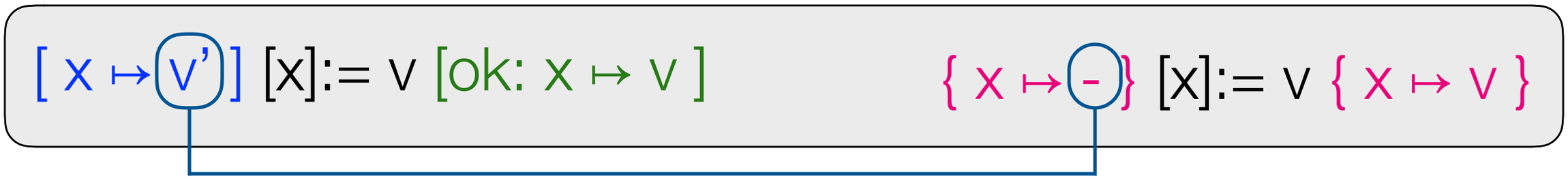
azalea@imperial.ac.uk SoundAndComplete.org @azalearaad

ISL vs. SL Local Axioms

[x ↦ v'] [x]:= v [ok: x ↦ v]

{ x ↦ - } [x]:= v { x ↦ v }

ISL vs. SL Local Axioms



ISL vs. SL Local Axioms

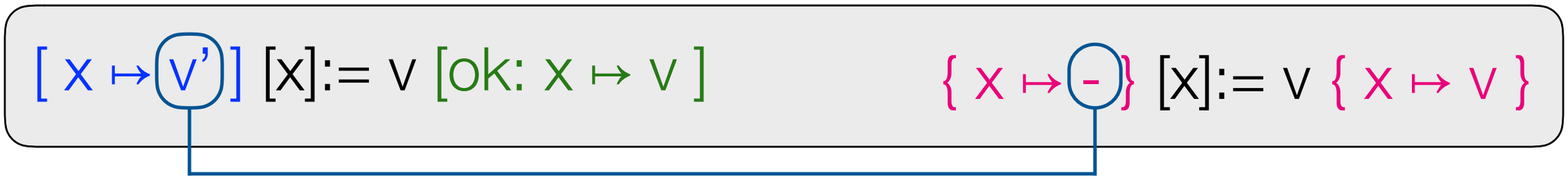
[x ↦ v'] [x]:= v [ok: x ↦ v]

{ x ↦ - } [x]:= v { x ↦ v }

Suppose instead we had the axiom:

[x ↦ -] [x]:= v [ok: x ↦ v]

ISL vs. SL Local Axioms



Suppose instead we had the axiom:

[x ↦ -] [x]:= v [ok: x ↦ v]

Let us prove:

[x ↦ 1] [x]:= 2 [ok: x ↦ 2]

ISL vs. SL Local Axioms

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

Suppose instead we had the axiom:

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

Let us prove:

$[x \mapsto 1]\ [x]{:=}\ 2\ [ok:\ x \mapsto 2]$ apply Cons →

$$\frac{p \Leftarrow p' \quad [p']\ C\ [\varepsilon:\ q'] \quad q' \Leftarrow q}{[p]\ C\ [\varepsilon:\ q]}\ (\text{Cons})$$

ISL vs. SL Local Axioms

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

Suppose instead we had the axiom:

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

Let us prove:

$[x \mapsto 1]\ [x]{:=}\ 2\ [ok:\ x \mapsto 2]$ —apply Cons→ Show: $x \mapsto -\ \Longrightarrow\ x \mapsto 1$

$$\frac{p \Leftarrow p' \quad [p']\ C\ [\varepsilon:\ q'] \quad q' \Leftarrow q}{[p]\ C\ [\varepsilon:\ q]}\ (\text{Cons})$$

ISL vs. SL Local Axioms

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

Suppose instead we had the axiom:

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

Let us prove:

$[x \mapsto 1]\ [x]{:=}\ 2\ [ok:\ x \mapsto 2]$ —apply Cons→ Show: $x \mapsto -\ \Longrightarrow\ x \mapsto 1$ ✗

$$\frac{p \Leftarrow p' \quad [p']\ C\ [\varepsilon:\ q'] \quad q' \Leftarrow q}{[p]\ C\ [\varepsilon:\ q]}\ \text{(Cons)}$$