Local Reasoning about the Presence of Bugs: **Incorrectness Separation Logic (ISL)**

Azalea Raad[1,2] Josh Berdine[3] Hoang-Hai Dang[2]
Derek Dreyer[2] Peter O'Hearn[3,4] Jules Villard[3]

[1] Imperial College London
[2] Max Planck Institute for Software Systems (MPI-SWS)
[3] Facebook London
[4] University College London

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

State of the Art: ***Correctness***

- Lots of work on ***local reasoning*** for proving ***correctness***
  - ➡ Prove the ***absence of bugs***
  - ➡ ***Over-approximate*** reasoning

State of the Art: ***Correctness***

- Lots of work on ***local reasoning*** for proving ***correctness***
  - ➡ Prove the ***absence of bugs***
  - ➡ ***Over-approximate*** reasoning
  - ➡ ***Compositionality***

    in ***code*** ⇒ reasoning about ***incomplete components***

    in ***resources*** accessed ⇒ spatial locality

State of the Art: ***Correctness***

- Lots of work on ***local reasoning*** for proving ***correctness***
  - ➡ Prove the ***absence of bugs***
  - ➡ ***Over-approximate*** reasoning
  - ➡ ***Compositionality***

    in ***code*** ⇒ reasoning about ***incomplete components***

    in ***resources*** accessed ⇒ spatial locality
- ➡ ***Scalability*** to large teams and codebases

State of the Art: ***Bug Catching***

- Lots of work on ***bug catching***
  - Prove the ***presence of bugs***
  - E.g. symbolic model checking, symbolic execution for testing
  - ***Under-approximate*** reasoning

State of the Art: ***Bug Catching***

- Lots of work on ***bug catching***
  - Prove the ***presence of bugs***
  - E.g. symbolic model checking, symbolic execution for testing
  - ***Under-approximate*** reasoning
- Based on ***global*** reasoning

State of the Art: ***Bug Catching***

- Lots of work on ***bug catching***
  - Prove the ***presence of bugs***
  - E.g. symbolic model checking, symbolic execution for testing
  - ***Under-approximate*** reasoning
- Based on ***global*** reasoning
- Exceptions using ***local*** reasoning
  - e.g. ***Infer***

State of the Art: ***Bug Catching***

- Lots of work on ***bug catching***
  - Prove the ***presence of bugs***
  - E.g. symbolic model checking, symbolic execution for testing
  - ***Under-approximate*** reasoning
- Based on ***global*** reasoning
- Exceptions using ***local*** reasoning
  - e.g. ***Infer***
  - Using ***correctness***-based compositional analysis

State of the Art: ***Bug Catching***

❖ Lots of work on ***bug catching***

➡ Prove the ***presence of bugs***

➡ E.g. symbolic model checking, symbolic execution for testing

➡ ***U***

❖ Base

❖ Exce

➡ e.

➡ Us

Incorrectness Logic (O'Hearn)
Formal Foundations
for
Bug Catching

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)

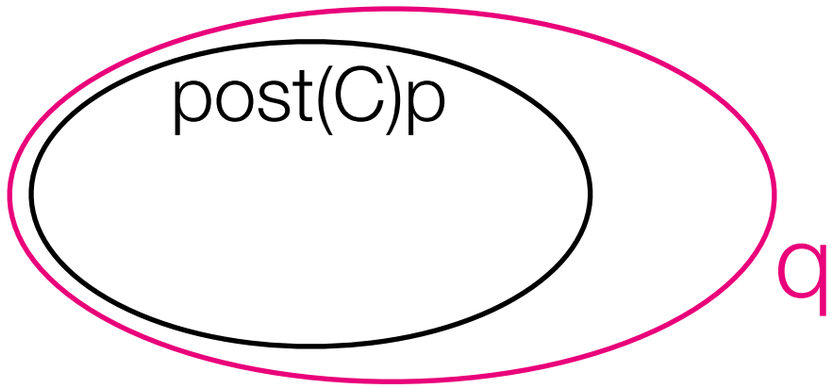
Hoare triples {p} C {q} *iff* post(C)p ⊆ q

q *over-approximates* post(C)p

Incorrectness Logic (IL)

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

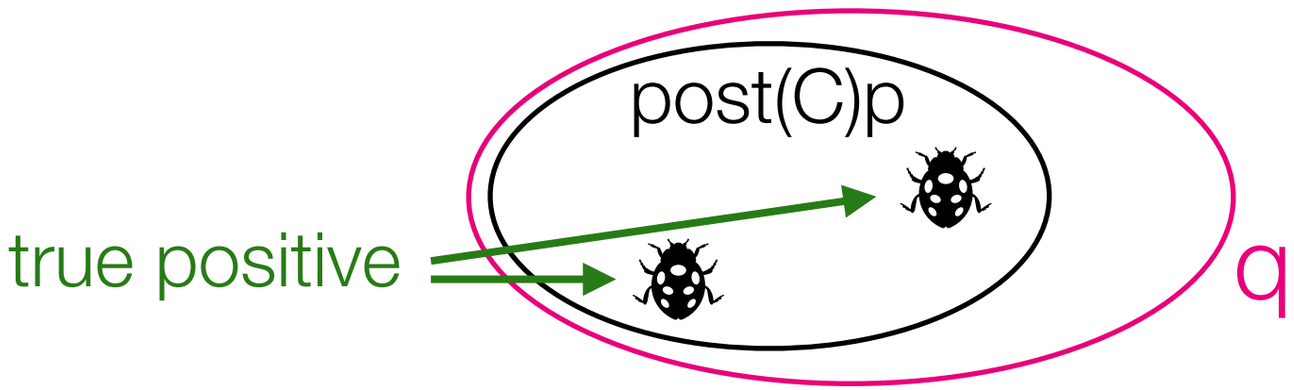
q *over-approximates* post(C)p



Incorrectness Logic (IL)

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

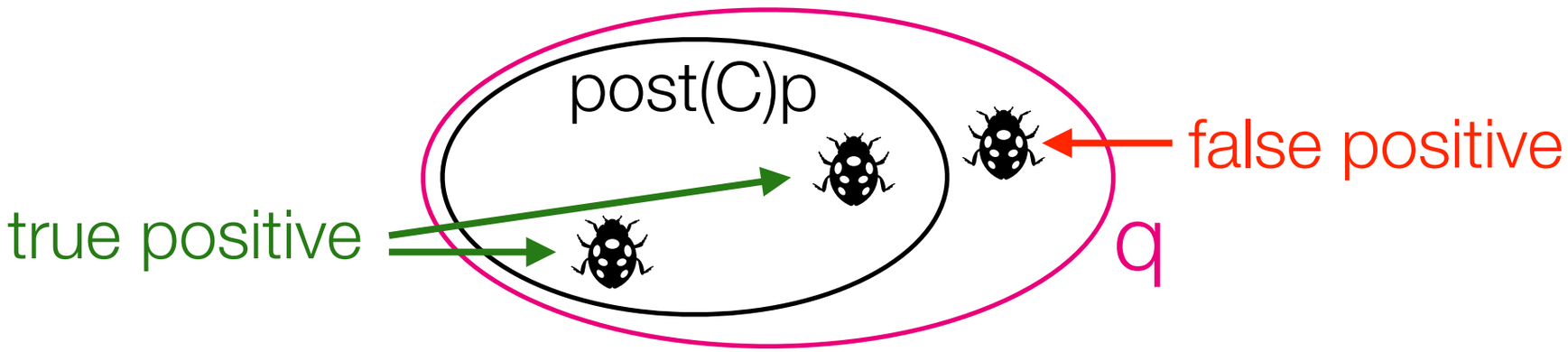
q *over-approximates* post(C)p



Incorrectness Logic (IL)

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

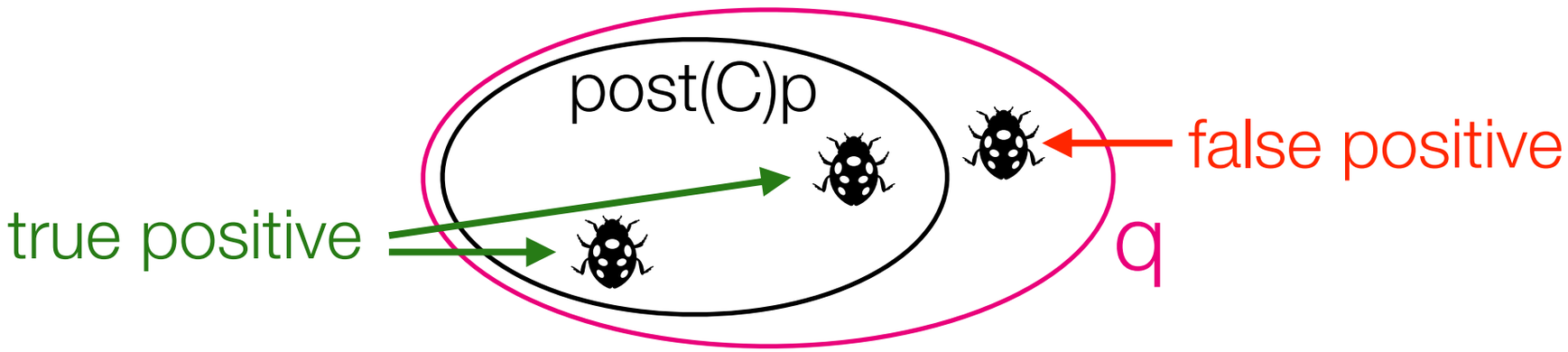
q *over-approximates* post(C)p



Incorrectness Logic (IL)

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

q *over-approximates* post(C)p



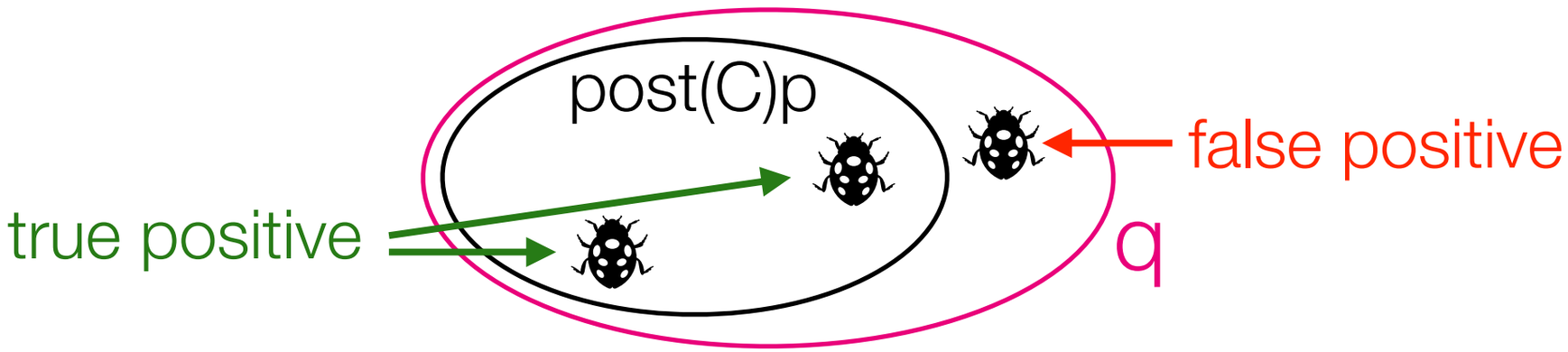
Incorrectness triples [p] C [q] *iff* post(C)p ⊇ q

q *under-approximates* post(C)p

Incorrectness Logic (IL)

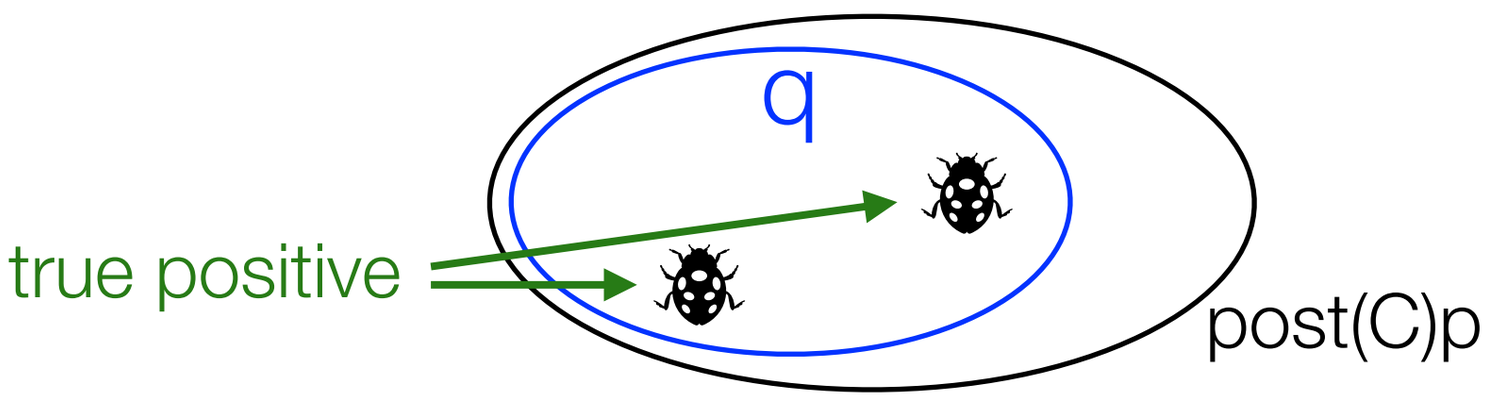
Hoare triples {p} C {q} *iff* post(C)p ⊆ q

q *over-approximates* post(C)p



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

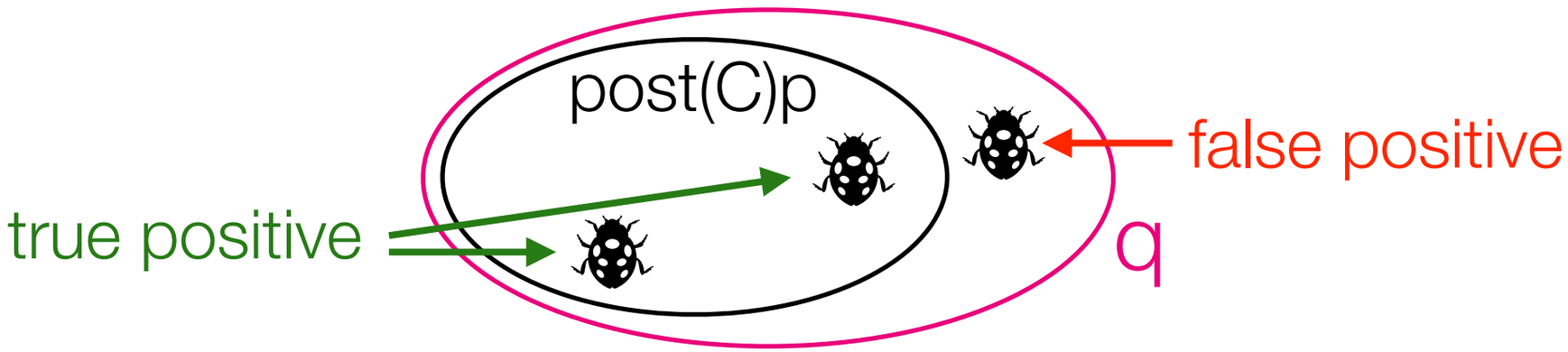
q *under-approximates* post(C)p



Incorrectness Logic (IL)

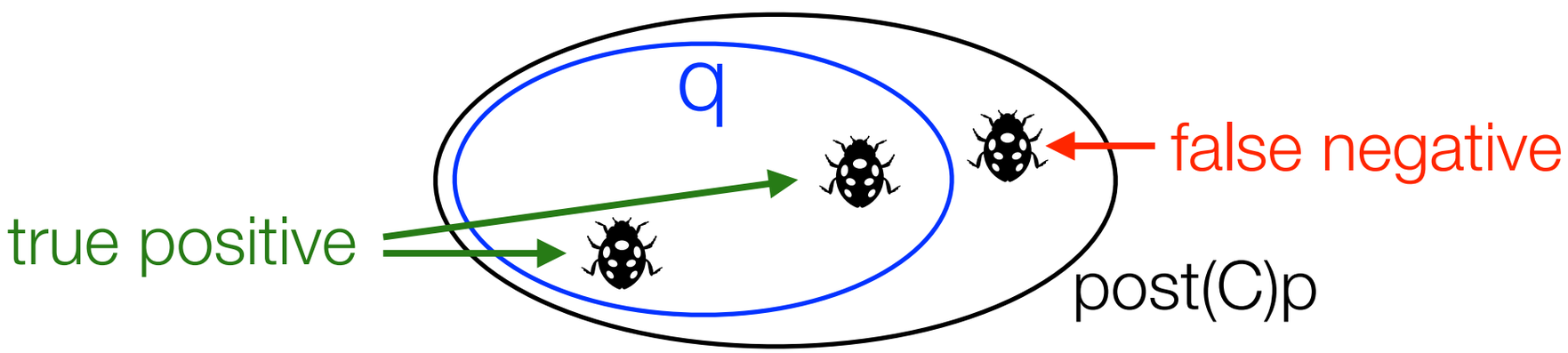
Hoare triples $\{p\}\ C\ \{q\}$ *iff* $post(C)p \subseteq q$

q *over-approximates* post(C)p



Incorrectness triples $[p]\ C\ [q]$ *iff* $post(C)p \supseteq q$

q *under-approximates* post(C)p



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

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

Incorrectness Logic (IL)

[p] C [ε: q]

ε: exit condition

ok: normal execution

er : erroneous execution

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

[p] error( ) [er: p]

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: Summary

+ ***Under-approximate*** analogue of Hoare Logic

+ Formal foundation for ***bug catching***

Incorrectness Logic: Summary

+ ***Under-approximate*** analogue of Hoare Logic

+ Formal foundation for ***bug catching***

– Global reasoning: ***non-compositional*** (as in original Hoare Logic)

– Cannot target ***memory safety bugs*** (e.g. use-after-free)

Incorrectness Logic: Summary

+ ***Under-approximate*** analogue of Hoare Logic

+ Formal foundation for ***bug catching***

– Glob Logic)

– Can

Our Solution

Incorrectness Separation Logic

Contributions

Contributions

- Incorrectness ***Separation*** Logic (ISL)
  - ➡ IL + SL for ***compositional bug catching***

Contributions

- Incorrectness ***Separation*** Logic (ISL)
  - ➡ IL + SL for ***compositional bug catching***
  - ➡ ***Under-approximate*** analogue of SL
  - ➡ Targets ***memory safety bugs*** (e.g. use-after-free)
  - ➡ ***Scalable***: inspired by Facebook Pulse

Contributions

- Incorrectness ***Separation*** Logic (ISL)
  - ➡ IL + SL for ***compositional bug catching***
  - ➡ ***Under-approximate*** analogue of SL
  - ➡ Targets ***memory safety bugs*** (e.g. use-after-free)
  - ➡ ***Scalable***: inspired by Facebook Pulse
- Combining IL+SL: not straightforward
  - ➡ ***invalid frame*** rule!

Contributions

- Incorrectness ***Separation*** Logic (ISL)
  - ➡ IL + SL for ***compositional bug catching***
  - ➡ ***Under-approximate*** analogue of SL
  - ➡ Targets ***memory safety bugs*** (e.g. use-after-free)
  - ➡ ***Scalable***: inspired by Facebook Pulse
- Combining IL+SL: not straightforward
  - ➡ ***invalid frame*** rule!
- Fix: a ***monotonic model*** for frame preservation

Contributions

- Incorrectness ***Separation*** Logic (ISL)
  - ➡ IL + SL for ***compositional bug catching***
  - ➡ ***Under-approximate*** analogue of SL
  - ➡ Targets ***memory safety bugs*** (e.g. use-after-free)
  - ➡ ***Scalable***: inspired by Facebook Pulse
- Combining IL+SL: not straightforward
  - ➡ ***invalid frame*** rule!
- Fix: a ***monotonic model*** for frame preservation
- Recovering the ***footprint property*** for completeness

Contributions

- Incorrectness ***Separation*** Logic (ISL)
  - ➡ IL + SL for ***compositional bug catching***
  - ➡ ***Under-approximate*** analogue of SL
  - ➡ Targets ***memory safety bugs*** (e.g. use-after-free)
  - ➡ ***Scalable***: inspired by Facebook Pulse
- Combining IL+SL: not straightforward
  - ➡ ***invalid frame*** rule!
- Fix: a ***monotonic model*** for frame preservation
- Recovering the ***footprint property*** for completeness
- ISL-based ***analysis***
  - ➡ ***No-false-positives theorem:***

    All bugs found are true bugs

Contributions

- Incorrectness ***Separation*** Logic (ISL)
  - ➡ IL + SL for ***compositional bug catching***
  - ➡ ***Under-approximate*** analogue of SL
  - ➡ Targets ***memory safety bugs*** (e.g. use-after-free)
  - ➡ ***Scalable***: inspired by Facebook Pulse
- Combining IL+SL: not straightforward
  - ➡ ***invalid frame*** rule!
- Fix: a ***monotonic model*** for frame preservation

This talk

- Recovering the ***footprint property*** for completeness
- ISL-based ***analysis***
  - ➡ ***No-false-positives theorem:***

    All bugs found are true bugs

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

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

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;
```

‘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

‘and ***separately***’

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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)

WRITE

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

ISL: Local Axioms (First Attempt)

WRITE

[x ↦ v’] [x]:= v [ok: x ↦ v]

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

ISL: Local Axioms (First Attempt)

WRITE

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

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

null-pointer dereference error

ISL: Local Axioms (First Attempt)

WRITE

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

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

null-pointer dereference error

READ

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

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

ISL: Local Axioms (First Attempt)

WRITE

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

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

null-pointer dereference error

READ

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

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

ALLOC

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

ISL: Local Axioms (First Attempt)

WRITE

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

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

null-pointer dereference error

READ

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

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

ALLOC

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

FREE

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

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

ISL: Local Axioms (First Attempt)

WRITE

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

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

null-pointer dereference error

READ

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

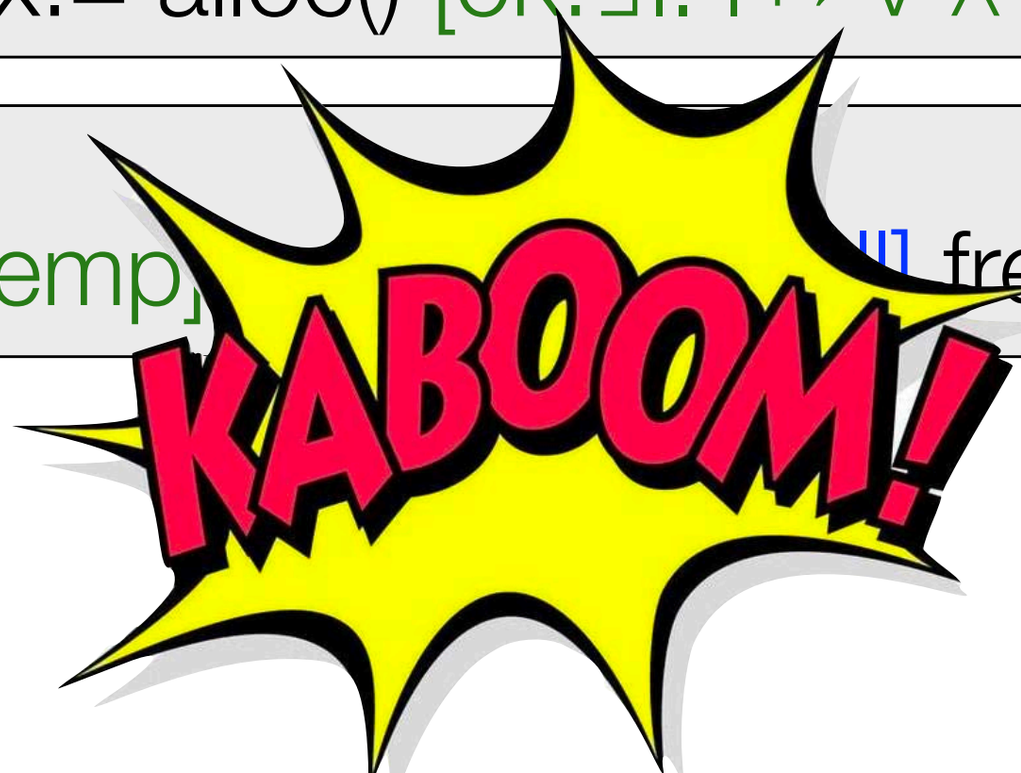
$[x=null]\ y:= [x]\ [er: x=null]$

ALLOC

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

FREE

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



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

ISL: Local Axioms (First Attempt)

ISL

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

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

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

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

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

ISL: Local Axioms (First Attempt)

ISL

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

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

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

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

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

ISL: Local Axioms (First Attempt)

ISL

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

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

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

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

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

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

ISL: Local Axioms (First Attempt)

ISL

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

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

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

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

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

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

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

ISL: Local Axioms (First Attempt)

ISL

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

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

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

Solution:

Track Deallocated Locations!

$$[x \mapsto v$$

(Frame)

$$\to v]$$

(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 $q \Rightarrow \text{false}$)

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!

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

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

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

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

ISL: Local Axioms

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

FREE

ISL: Local Axioms

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

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

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

ISL: Local Axioms

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

FREE

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

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

use-after-free error

ISL: Local Axioms

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

FREE

[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: $x \mapsto v$]

WRITE

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

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

ISL: Local Axioms

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

FREE

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

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

use-after-free error

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

WRITE

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

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

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

READ

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

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

ISL: Local Axioms

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

FREE

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

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

use-after-free error

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

WRITE

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

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

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

READ

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

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

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

ALLOC

ISL: Local Axioms

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

FREE

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

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

use-after-free error

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

WRITE

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

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

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

READ

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

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

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

ALLOC

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

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

Incorrectness Separation Logic

July 23rd @ 9:15

A. Raad J. Berdine H-H. Dang D. Dreyer P. O'Hearn J. Villard

Verification

Correctness

- Prove ***absence of bugs***
- ***Local*** reasoning
- ***Compositionality***
  - ➡ in code (incomplete code)
  - ➡ in resources accessed

Incorrectness Separation Logic

July 23rd @ 9:15

A. Raad J. Berdine H-H. Dang D. Dreyer P. O'Hearn J. Villard

Verification

Correctness

- Prove ***absence of bugs***
- ***Local*** reasoning
- ***Compositionality***
  - ➡ in code (incomplete code)
  - ➡ in resources accessed

Bug Catching (incorrectness)

- Prove ***presence of bugs***
- ***Global*** reasoning
  - ➡ e.g. symbolic model checking
- Local exceptions: e.g. Infer
  - ➡ Based on correctness

Incorrectness Separation Logic

July 23rd @ 9:15

A. Raad J. Berdine H-H. Dang D. Dreyer P. O'Hearn J. Villard

Verification

Correctness

- Prove ***absence of bugs***
- ***Local*** reasoning
- ***Compositionality***
  - ➡ in code (incomplete code)
  - ➡ in resources accessed

Bug Catching (incorrectness)

- Prove ***presence of bugs***
- ***Global*** reasoning
  - ➡ e.g. symbolic model checking
- Local exceptions: e.g. Infer
  - ➡ Based on correctness

Incorrectness Separation Logic (ISL)

Incorrectness logic: global reasoning for ***bug catching***

+

Separation logic: correctness-based ***local*** reasoning

Incorrectness Separation Logic

July 23rd @ 9:15

A. Raad J. Berdine H-H. Dang D. Dreyer P. O'Hearn J. Villard

Verification

Correctness

- Prove ***absence of bugs***
- ***Local*** reasoning
- ***Compositionality***
  - ➡ in code (incomplete code)
  - ➡ in resources accessed

Bug Catching (incorrectness)

- Prove ***presence of bugs***
- ***Global*** reasoning
  - ➡ e.g. symbolic model checking
- Local exceptions: e.g. Infer
  - ➡ Based on correctness

Incorrectness Separation Logic (ISL)

Incorrectness logic: global reasoning for ***bug catching***

+

Separation logic: correctness-based ***local*** reasoning

Formal foundation for ***local*** & ***compositional bug catching***