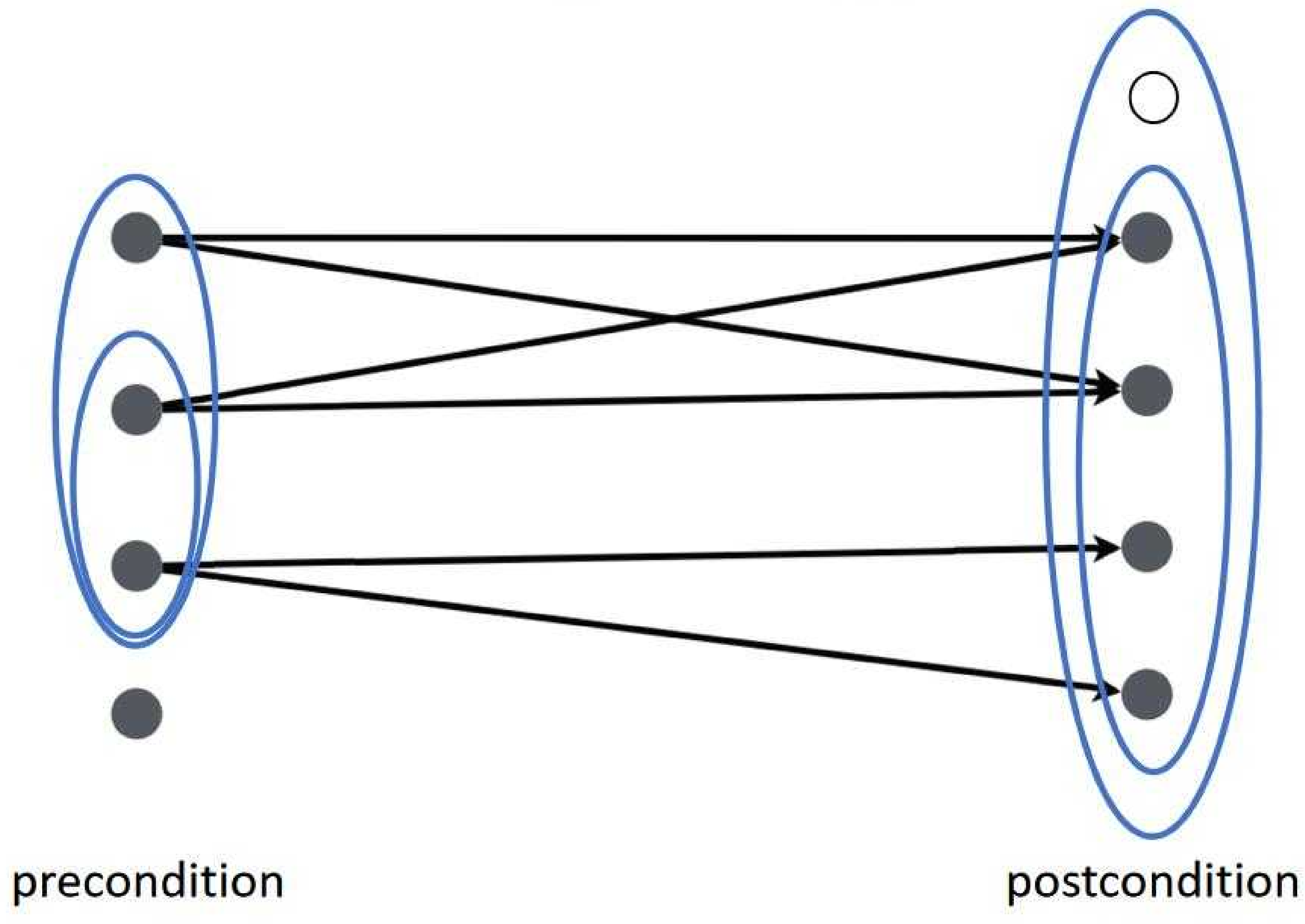
Incorrectness Logic (IL)

PETER W. O , HEARN,
Facebook and University College
London

Correctness Logic

- {precondition}code{postcondition}



Why we Need Incorrectness Logic?

Reasoning about correctness

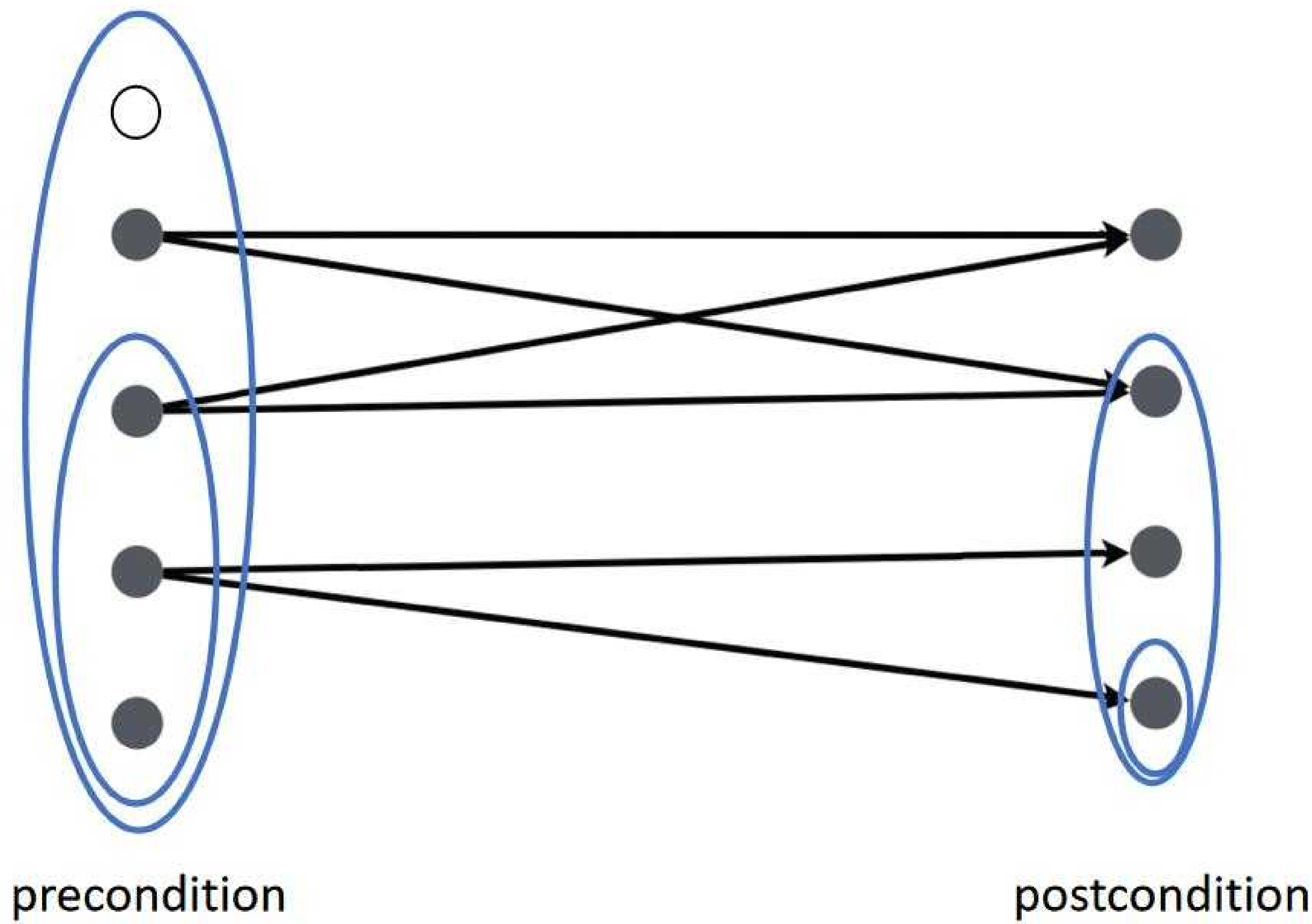
- *absence of bugs*
- *over-approximate approach*

- Tools based on over-approximation suffer from <u>false positives</u>

We need an under-approximate approach to prove the presence of bugs

Incorrectness Logic

- [presumption]code [result]



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

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* $post(C)p \subseteq q$

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

$post(C)p \subseteq q$

Incorrectness Logic (IL)

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

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

$post(C)p \supseteq q$

Incorrectness Logic (IL)

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

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

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

Incorrectness Logic (IL)

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

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* $post(C)p \supseteq q$

Incorrectness Logic (IL)

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

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* $post(C)p \supseteq q$

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* $post(C)p \subseteq q$

Incorrectness Logic (IL)

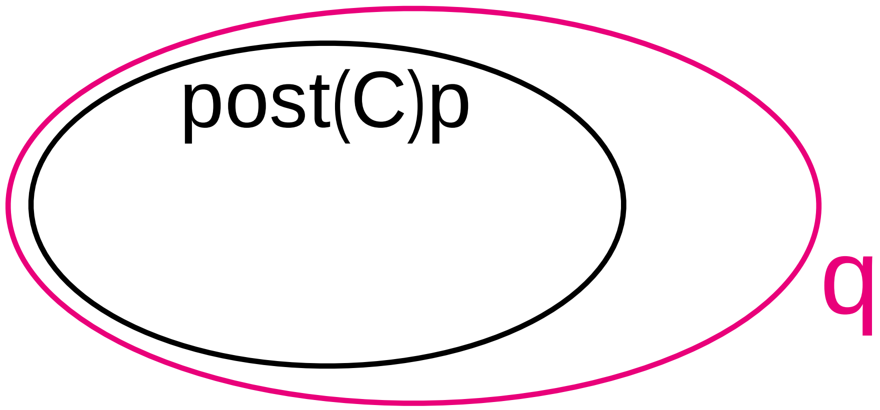
Hoare triples $\{p\}\ C\ \{q\}$ *iff* $\text{post}(C)p \subseteq q$

q *over-approximates* post(C)p

Incorrectness Logic (IL)

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

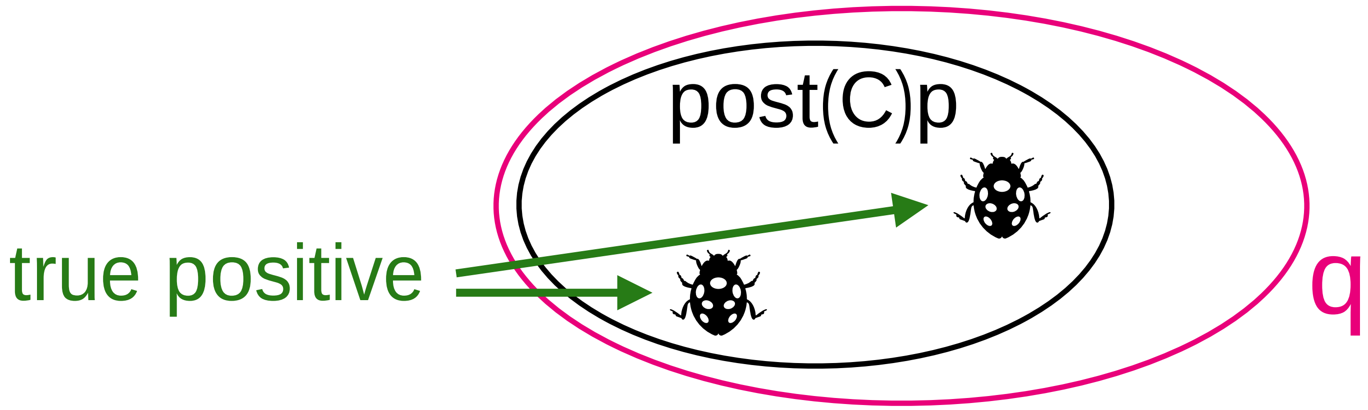
q *over-approximates* post(C)p



Incorrectness Logic (IL)

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

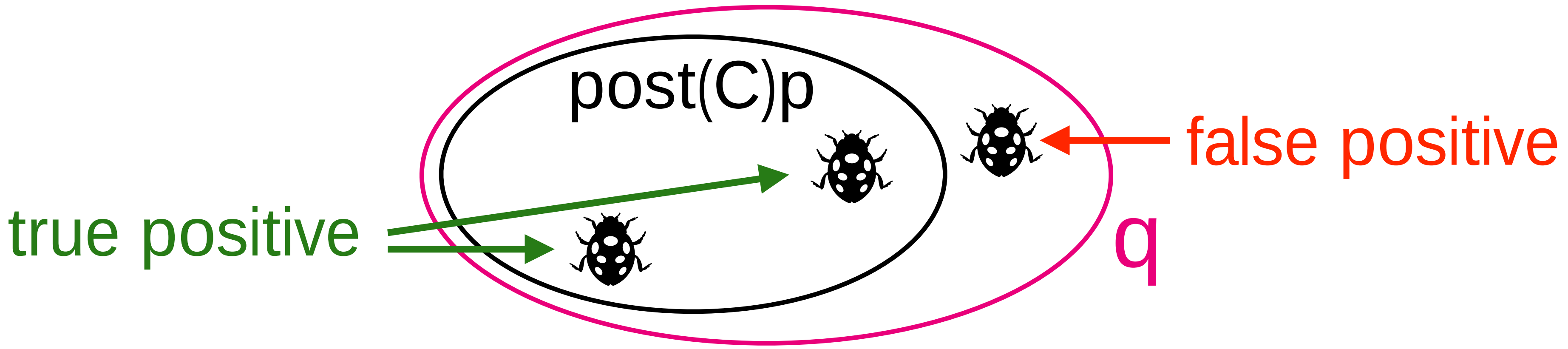
q *over-approximates* post(C)p



Incorrectness Logic (IL)

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

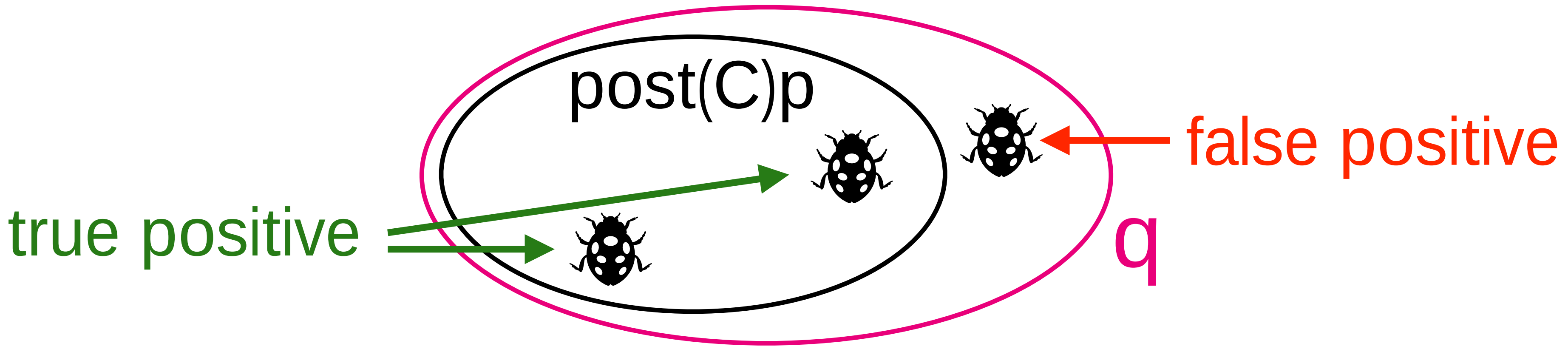
q *over-approximates* post(C)p



Incorrectness Logic (IL)

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

q *over-approximates* post(C)p



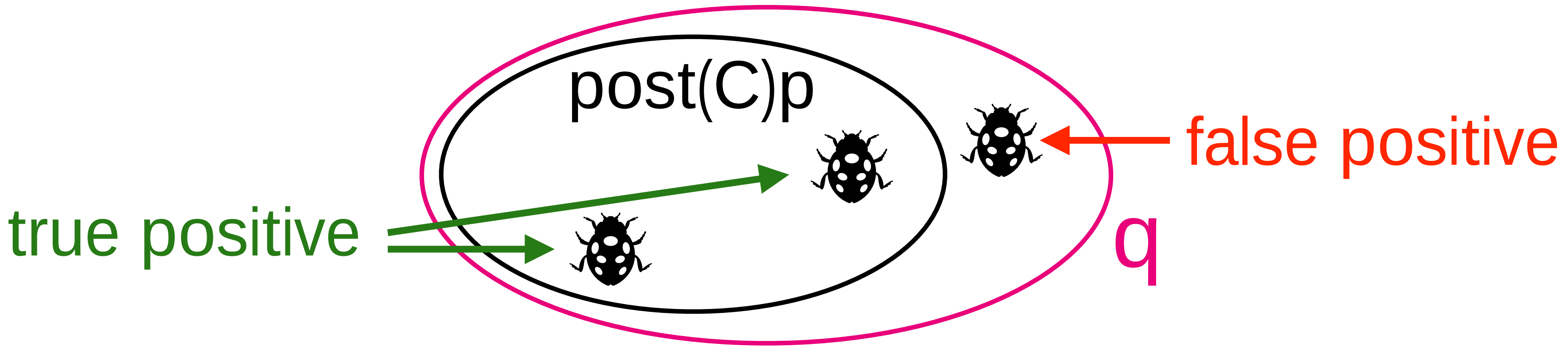
Incorrectness triples $[p]\ C\ [q]$ *iff* $\mathrm{post}(C)p \supseteq q$

q *under-approximates* post(C)p

Incorrectness Logic (IL)

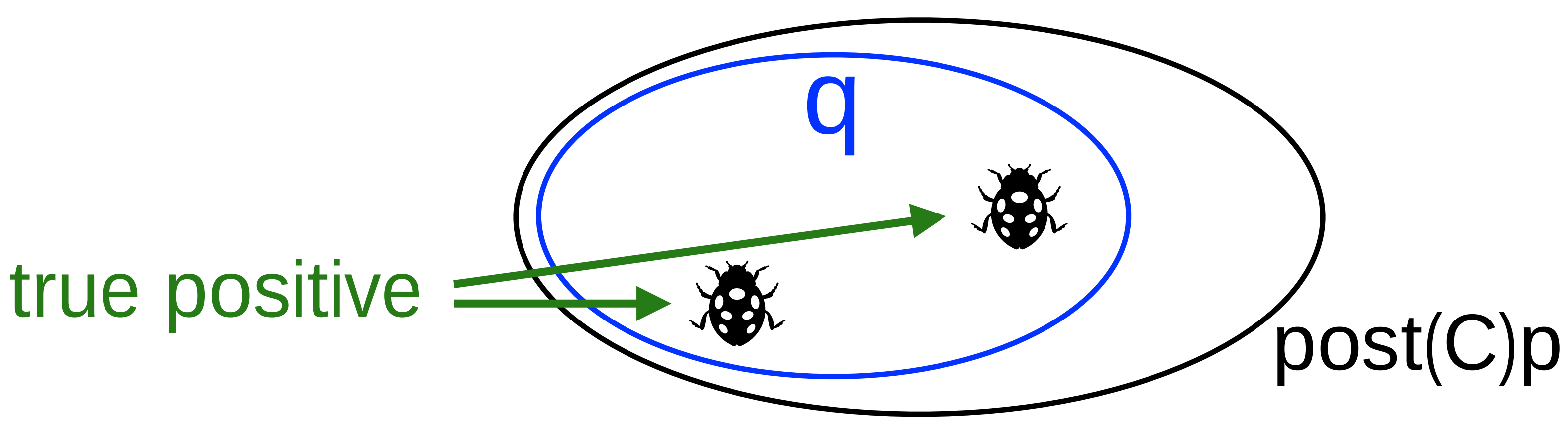
Hoare triples $\{p\}\ C\ \{q\}$ *iff* $post(C)p \subseteq q$

q *over-approximates* post(C)p



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

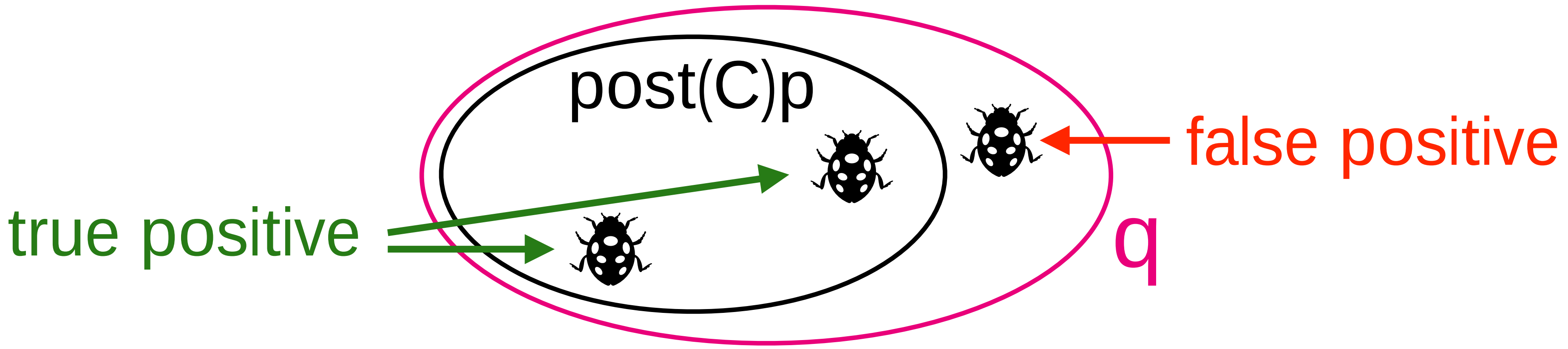
q *under-approximates* post(C)p



Incorrectness Logic (IL)

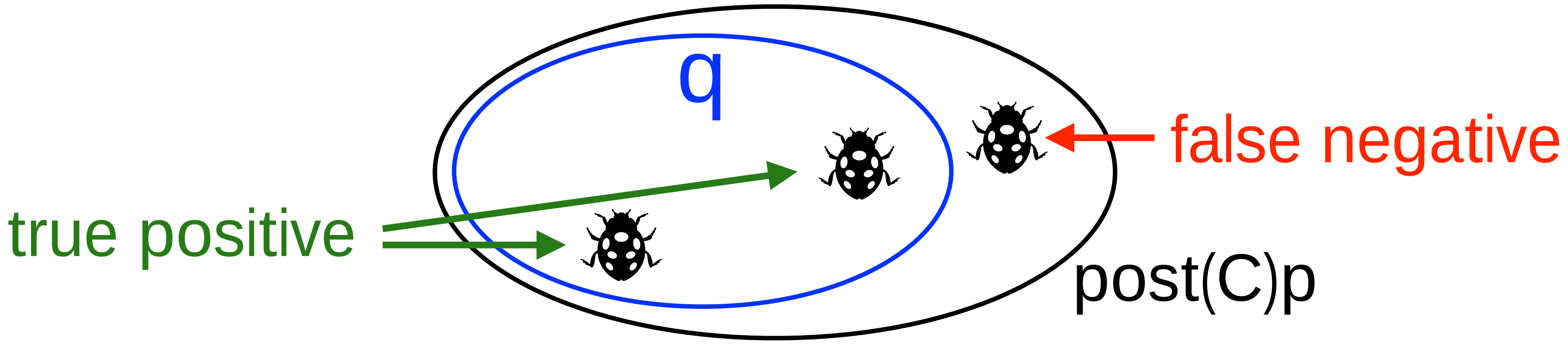
Hoare triples $\{p\}\ C\ \{q\}$ *iff* $post(C)p \subseteq q$

q *over-approximates* post(C)p



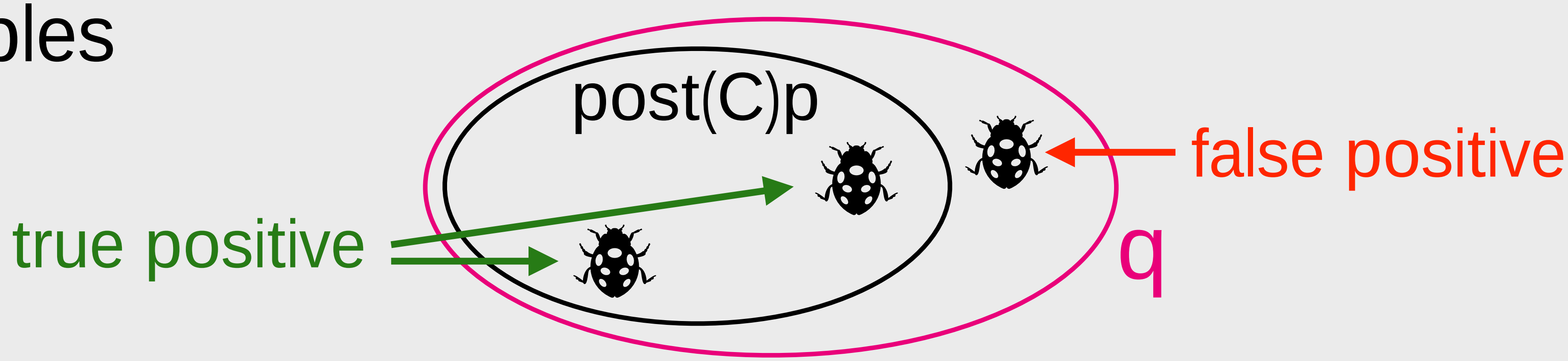
Incorrectness triples $[p]\ C\ [q]$ *iff* $post(C)p \supseteq q$

q *under-approximates* post(C)p



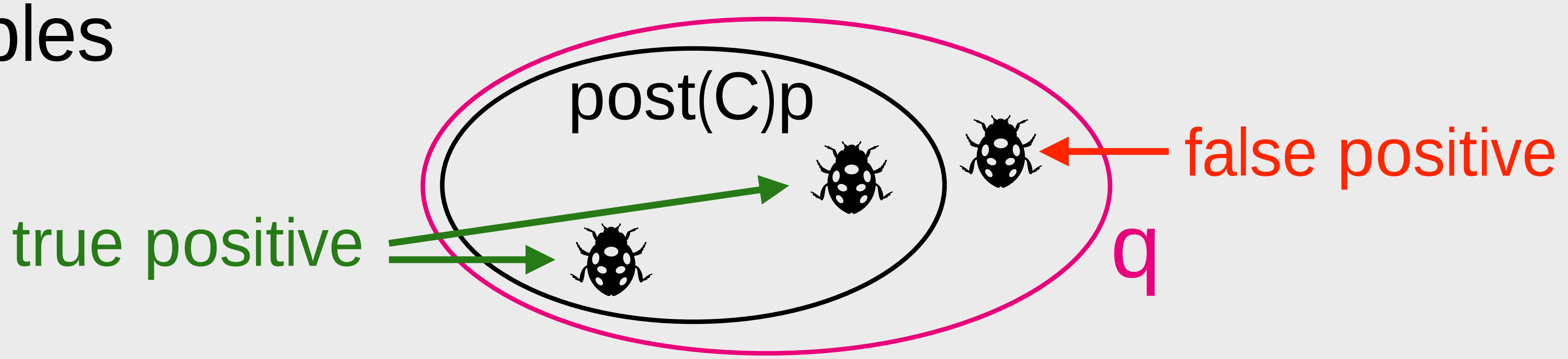
Incorrectness Logic (IL)

Hoare triples



Incorrectness Logic (IL)

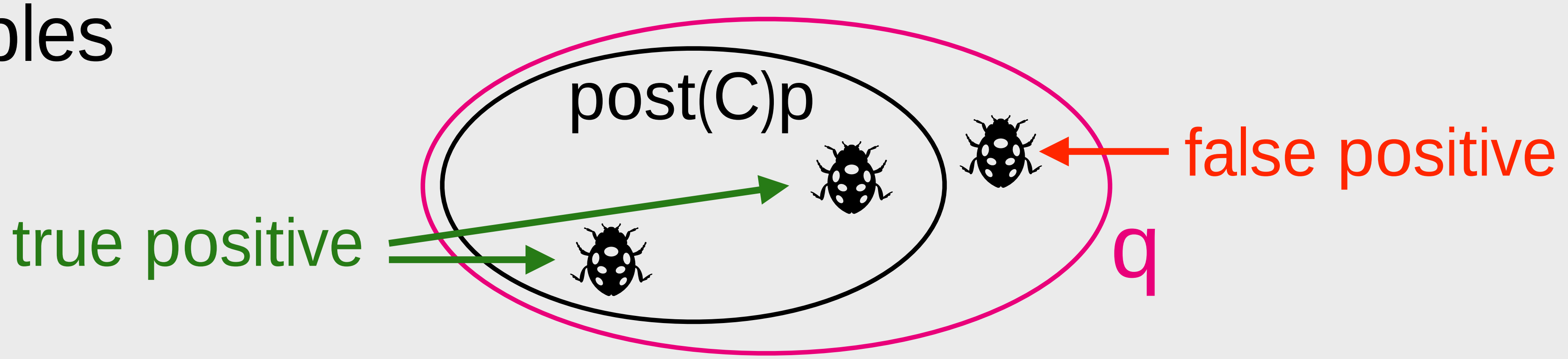
Hoare triples



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

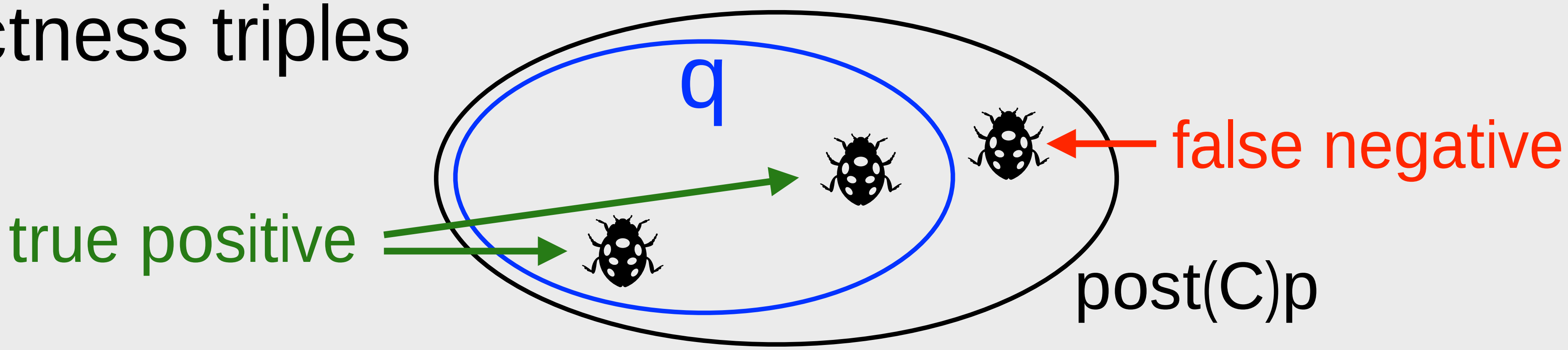
Incorrectness Logic (IL)

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)

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

ε: exit condition

ok: normal execution

er : erroneous execution

Incorrectness Logic (IL)

$[p]\ C\ [\varepsilon\text{: } q]$

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 $[\varepsilon\text{: } q]$

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]

Incorrectness Logic (IL)

$$[p]\ C\ [\epsilon\text{: } q]$$

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)

$[p]\ C\ [\varepsilon: q]$ *iff* $\mathrm{post}(C, \varepsilon)p \supseteq q$

Equivalent Definition (reachability)

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

Incorrectness Logic (IL)

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

```
/* presumes: [z==11] */
  if (x is even) {
     if (y is odd) {
        z=42
  }  }
/* 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]ε

[p]

[C]

[ok:q]

```
/* presumes: [z==11] */
  if (x is even) {
     if (y is odd) {
        z=42
  }  }
/* achieves: [z==42] */
```

?

(Slide courtesy of Peter O'Hearn)

Incorrectness Logic (IL)

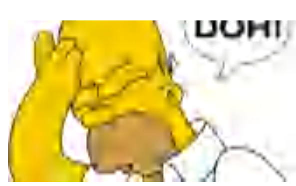
$[p]\ C\ [\varepsilon : q]$ *iff* $\forall s \in q.\ \exists s' \in p.\ (s',s) \in [C]\varepsilon$

[p]

[C]

[ok:q]

```
/* presumes: [z==11] */
if (x is even) {
   if (y is odd) {
      z=42
} }
/* achieves: [z==42] */
```



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

(Slide courtesy of Peter O’Hearn)

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

(Slide courtesy of Peter O’Hearn)

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

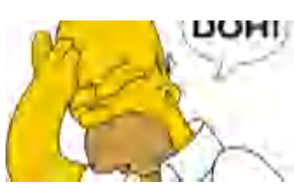
```
/* presumes: [z==11] */
  if (x is even) {
    if (y is odd) {
       z=42
  }  }
/* achieves: [z==42 && (x is even) && (y is odd) ] */
```

(Slide courtesy of Peter O’Hearn)

Incorrectness Logic (IL)

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

```
/* presumes: [z==11] */
  if (x is even) {
    if (y is odd) {
      z=42
  } }
/* achieves: [z==42] */
```



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

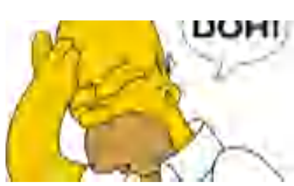
(Slide courtesy of Peter O'Hearn)

Incorrectness Logic (IL)

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

```
/* presumes: [z==11] */
  if (x is even) {
     if (y is odd) {
        z=42
  }  }
/* achieves: [z==42] */
```

✗

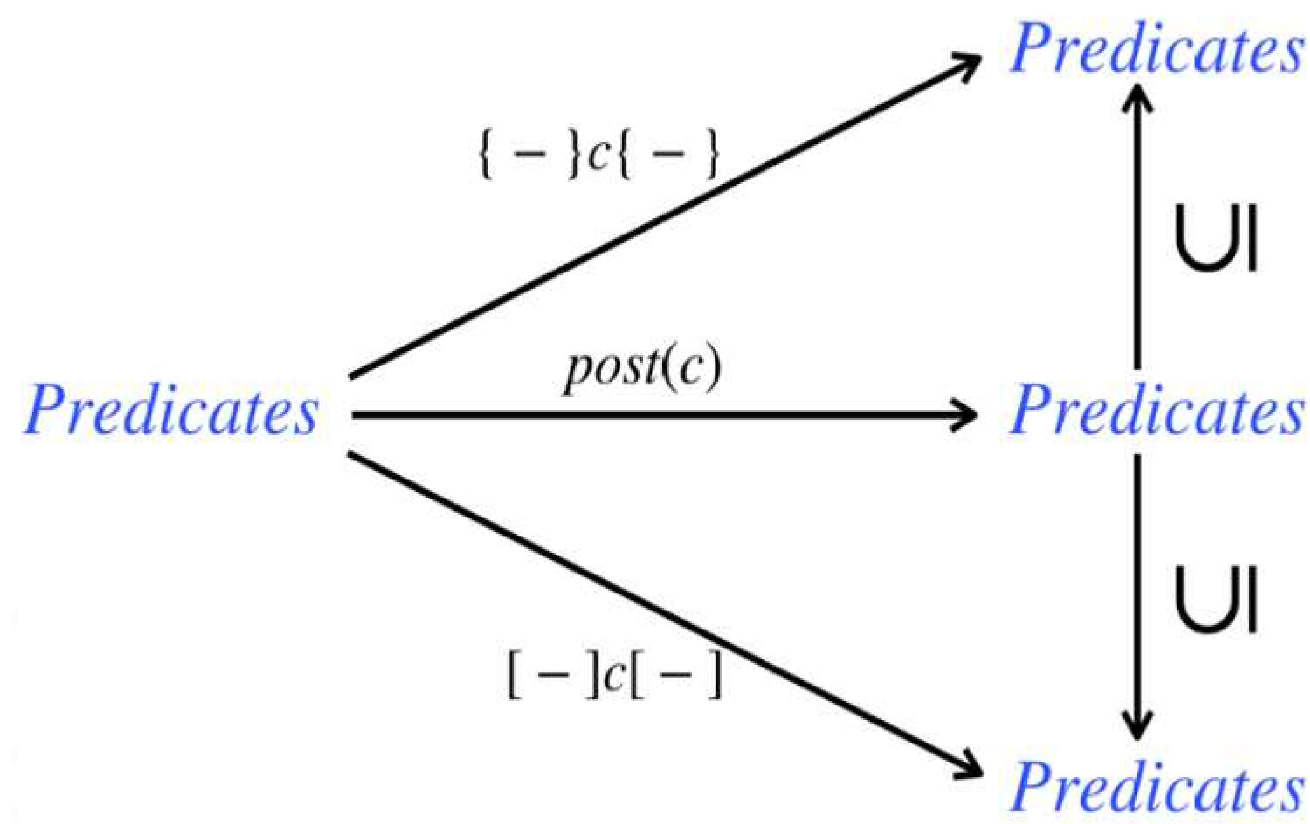


[z:42, x:1, y:2]

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

(Slide courtesy of Peter O’Hearn)

Incorrectness Logic (IL)



$$
\begin{array}{rlcl}
\wedge\vee\ \textit{Symmetry:} & [p]c[q_1] \wedge [p]c[q_2] & \iff & [p]c[q_1 \vee q_2] \\
 & \{p\}c\{q_1\} \wedge \{p\}c\{q_2\} & \iff & \{p\}c\{q_1 \wedge q_2\} \\
\Uparrow\Downarrow\ \textit{Symmetry:} & p' \Leftarrow p \wedge [p]c[q] \wedge q \Leftarrow q' & \implies & [p']c[q'] \\
 & p' \Rightarrow p \wedge \{p\}c\{q\} \wedge q \Rightarrow q' & \implies & \{p'\}c\{q'\} \\
\textit{Principle of Agreement:} & [u]c[u'] \wedge u \Rightarrow o \ \wedge\ \{o\}c\{o'\} & \implies & u' \Rightarrow o' \\
\textit{Principle of Denial:} & [u]c[u'] \wedge u \Rightarrow o \wedge \neg(u' \Rightarrow o') & \implies & \neg(\{o\}c\{o'\})
\end{array}
$$

Fig. 1. Correctness and Incorrectness Principles

Incorrectness Logic (IL)

$$\{p\}\ C\ \{q\} \quad \textit{iff} \quad post(C)p \subseteq q$$

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

Incorrectness Logic (IL)

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

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

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

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

Incorrectness Logic (IL)

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

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

Incorrectness Logic (IL)

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

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

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

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

Incorrectness Logic (IL)

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

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

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

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

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

Incorrectness Logic (IL)

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

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

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

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

Incorrectness Logic (IL)

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

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

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

Ignore Paths

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

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

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)

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

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

Incorrectness Logic Rules (Excerpt)

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

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

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

Incorrectness Logic Rules (Excerpt)

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

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

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

Incorrectness Logic Rules (Excerpt)

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

$[p]\ C^{\star}\ [ok\!: p]$ (Iterate — Zero)

Incorrectness Logic Rules (Excerpt)

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

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

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

Incorrectness Logic Rules (Excerpt)

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

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

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

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

Incorrectness Logic (Example)

3.2 Specifying Incorrectness

We can reason about errors by distinguishing result-assertion forms, for normal and erroneous or abnormal termination. For example, consider the following program, which is a variant of one used to illustrate buffer overflows by One [1996].

```
void foo(char* str)
/*  presumes: [*str[]==s]
    achieves: [er: *str[]==s && length(s) > 16 ] */
  { char buf[16];
    strcpy(buf,str);
  }

int main(int argc, char *argv[])
  { foo(argv[1]); }
```

The specification for `foo()` says that if the length of the input string is greater than 16 then we can get an error (in this case a buffer overflow). (There, we are writing `*str[]==s` to indicate that sequence s held as a null-terminated string starting at address str.)

Remark: it is not necessary that a segmentation fault must occur in C when a buffer overflows (worse can happen). Incorrectness is an abstraction that a programmer or tool engineer decides upon to help in engineering concerns for program construction. Logic provides a means to specify these assumptions, and then to perform sound reasoning based on them, but it does not set the assumptions.

Incorrectness Logic (Example)

3.3 Under-approximate Success

Even if we were mainly interested in incorrectness, under-approximate result assertions describing successful computations can help us soundly discover bugs that come after a procedure is called. In particular, if we were to have over-approximate assertions only for successful computations, then our reasoning could go wrong, as the following example illustrates.

```
void mkeven()
/*  presumes: [true],  wrong achieves: [ok: x==2 || x==4]      */
  { x=2; }

void usemkeven()
  {  mkeven();  if (x==4) {error();} }
```

We use `ok:` before an assertion to indicate that it describes a result for normal, not exceptional, termination of a program. The achieves assertion `mkeven()` describes an over-approximation of what the procedure produces, including a possibility (x==4) than cannot occur. If we were to use this wrong achieves assertion in `usemkeven()` to conclude that an error is possible then this would be a false positive warning.

For this reason, our formalism will include under-approximate achieves-assertions for both successful and erroneous termination. `mkeven()` achieves "`ok: x==2`", not "`ok: x==2 || x==4`".

The possibility of divergence gives rise to many more such examples. If we consider a version of `mkeven()` with body **`while`**`(true){};x=2`, then x==4 is an over-approximate post-condition, and it lines up perfectly with the condition to trip the error, but this would again be a false positive. Reasoning about termination is needed to be under-approximate; but, as we shall see, what is needed is not the same as showing termination on all inputs.

$[p]C[\epsilon{:}\, q]$: q under-approximates the states when C exits via ϵ starting from states in p.

We sometime write a quadruple

$[p]C[ok{:}\, q][er{:}\, r]$ as shorthand for $[p]C[ok{:}\, q]$ and $[p]C[er{:}\, r]$

Empty under-approximates

$$[p]C[\epsilon{:}\, \mathit{false}]$$

Consequence

$$\frac{p' \Leftarrow p \quad [p]C[\epsilon{:}\, q] \quad q \Leftarrow q'}{[p']C[\epsilon{:}\, q']}$$

Disjunction

$$\frac{[p_1]C[\epsilon{:}\, q_1] \quad [p_2]C[\epsilon{:}\, q_2]}{[p_1 \vee p_2]C[\epsilon{:}\, q_1 \vee q_2]}$$

Unit

$$[p]\texttt{skip}[ok{:}\, p][er{:}\, \mathit{false}]$$

Sequencing (short-circuit)

$$\frac{[p]C_1[er{:}\, r]}{[p]C_1; C_2[er{:}\, r]}$$

Sequencing (normal)

$$\frac{[p]C_1[ok{:}\, q] \quad [q]C_2[\epsilon{:}\, r]}{[p]C_1; C_2[\epsilon{:}\, r]}$$

Iterate zero

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

Iterate non-zero

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

Backwards Variant (where n fresh)

$$\frac{[p(n) \wedge nat(n)]C[ok{:}\, p(n+1) \wedge nat(n)]}{[p(0)]C^\star[ok{:}\, \exists n. p(n) \wedge nat(n)]}$$

Choice (where $i = 1$ or 2)

$$\frac{[p]C_i[\epsilon{:}\, q]}{[p]C_1 + C_2[\epsilon{:}\, q]}$$

Error

$$[p]\texttt{error}()[ok{:}\, \mathit{false}][er{:}\, p]$$

Assume

$$[p]\texttt{assume}\ B[ok{:}\, p \wedge B][er{:}\, \mathit{false}]$$

$$\texttt{while}\ B\ \texttt{do}\ C =_{def} (\texttt{assume}(B); C)^\star; \texttt{assume}(\neg B)$$

$$\texttt{if}\ B\ \texttt{then}\ C\ \texttt{else}\ C' =_{def} (\texttt{assume}(B); C) + (\texttt{assume}(\neg B); C')$$

$$\texttt{assert}(B) =_{def} \texttt{assume}(B) + (\texttt{assume}(\neg B); \texttt{error}())$$

Fig. 2. Generic Proof Rules of Incorrectness Logic

We do not include the dual of the disjunction rule, the rule with $\wedge$ in place of $\vee$, because it is unsound in incorrectness logic. For the program

$$C \quad = \quad (\texttt{assume}\ x == 1\ ;\ x = 88)\ +\ (\texttt{assume}\ x == 2\ ;\ x = 88)$$

we have $[x == 1]C[ok\colon x == 88]$ and $[x == 2]C[ok\colon x == 88]$, but the conjunction $x == 1 \wedge x == 2$ is inconsistent, and we don't have $[\mathit{false}]C[ok\colon x = 88]$: The strongest post of *false* is *false*, and $x == 88$ is not a subset of it.

Assignment

$$[p]x = e[ok\colon \exists x'.p[x'/x] \wedge x = e[x'/x]][er\colon \mathit{false}]$$

Nondet Assignment

$$[p]x = \texttt{nondet()}[ok\colon \exists x'.p[x'/x]][er\colon \mathit{false}]$$

Constancy

$$\frac{[p]C[\epsilon\colon q]}{[p \wedge r]C[\epsilon\colon q \wedge r]}\ \mathit{Mod}(C) \cap \mathit{Free}(r) = \emptyset$$

Local Variables

$$\frac{[p]C[\epsilon\colon q]}{[\exists x.p]\texttt{local}\ x.C[\epsilon\colon \exists x.q]}$$

Substitution

$$\frac{[p]C[\epsilon\colon q]}{([p]C[\epsilon\colon q])(e_1/x_1, \ldots, e_n/x_n)}\ \begin{array}{l} x_i \in \mathit{Mod}(C) \Rightarrow \\ e_i \text{ is a var not free in any other } e_j \end{array}$$

Fig. 3. Rules for Variables and Mutation

False Assertions in HL and IL

$\{p\}\ C\ \{q\} \quad \textit{iff} \quad post(C)p \subseteq q$

$\{false\}\ C\ \{q\}$ ✔ (vacuous)

False Assertions in HL and IL

$\{p\}\ C\ \{q\} \quad \textit{iff} \quad \mathrm{post}(C)p \subseteq q$

$\{\mathrm{false}\}\ C\ \{q\}$ ✔ (vacuous)

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

False Assertions in HL and IL

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

$\{\mathit{false}\}\ C\ \{q\}$ ✔ (vacuous)

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

✗ total correctness (unless $p \Rightarrow \mathit{false}$)

False Assertions in HL and IL

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

$\{false\}\ C\ \{q\}$ ✔ (vacuous)

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

✗ total correctness (unless $p \Rightarrow false$)

$[p]\ C\ [\varepsilon: q]$ *iff* $post(C, \varepsilon)p \supseteq q$

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

False Assertions in HL and IL

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

$\{false\}\ C\ \{q\}$ ✔ (vacuous)

$\{p\}\ C\ \{false\}$ ✔ partial correctness (non-termination)
✗ total correctness (unless $p \Rightarrow false$)

$[p]\ C\ [\varepsilon{:}\ q]$ *iff* $post(C, \varepsilon)p \supseteq q$

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

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

IL: Soundness & Completeness

Theorem. IL is sound and complete:

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