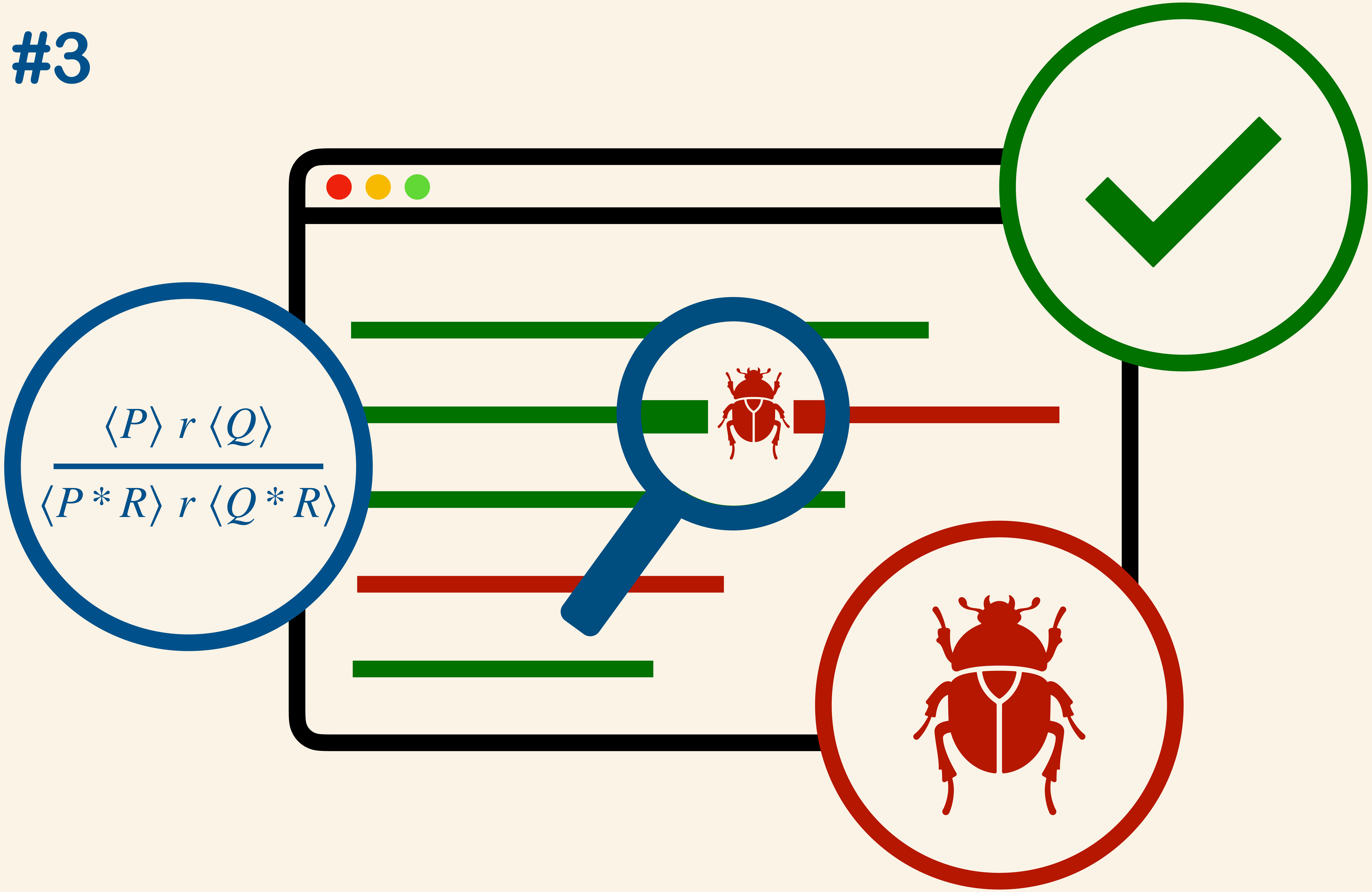
SCAN ME

Program Analysis

Lecture #3

Roberto Bruni

Università di Pisa

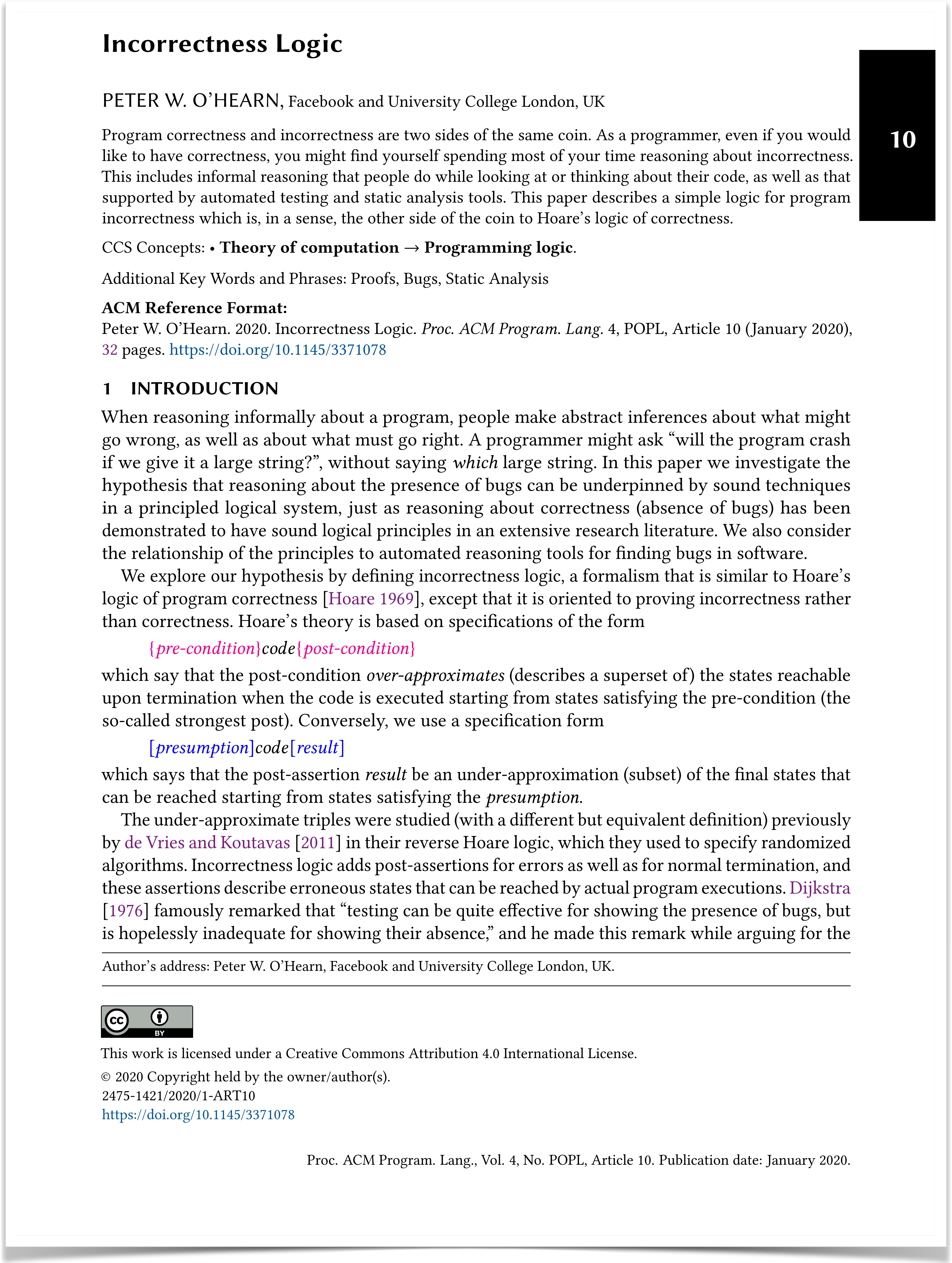


PhD Course
June 30 - July 4, 2025

GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

Program incorrectness: pragmatic motivations

POPL 2020



Incorrectness Logic

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

10

Program correctness and incorrectness are two sides of the same coin. As a programmer, even if you would like to have correctness, you might find yourself spending most of your time reasoning about incorrectness. This includes informal reasoning that people do while looking at or thinking about their code, as well as that supported by automated testing and static analysis tools. This paper describes a simple logic for program incorrectness which is, in a sense, the other side of the coin to Hoare's logic of correctness.

CCS Concepts: • **Theory of computation** → **Programming logic**.

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:
Peter W. O'Hearn. 2020. Incorrectness Logic. *Proc. ACM Program. Lang.* 4, POPL, Article 10 (January 2020), 32 pages. https://doi.org/10.1145/3371078

1 INTRODUCTION

When reasoning informally about a program, people make abstract inferences about what might go wrong, as well as about what must go right. A programmer might ask "will the program crash if we give it a large string?", without saying *which* large string. In this paper we investigate the hypothesis that reasoning about the presence of bugs can be underpinned by sound techniques in a principled logical system, just as reasoning about correctness (absence of bugs) has been demonstrated to have sound logical principles in an extensive research literature. We also consider the relationship of the principles to automated reasoning tools for finding bugs in software.

We explore our hypothesis by defining incorrectness logic, a formalism that is similar to Hoare's logic of program correctness [Hoare 1969], except that it is oriented to proving incorrectness rather than correctness. Hoare's theory is based on specifications of the form

$$\{\textit{pre-condition}\}\textit{code}\{\textit{post-condition}\}$$

which say that the post-condition *over-approximates* (describes a superset of) the states reachable upon termination when the code is executed starting from states satisfying the pre-condition (the so-called strongest post). Conversely, we use a specification form

$$[\textit{presumption}]\textit{code}[\textit{result}]$$

which says that the post-assertion *result* be an under-approximation (subset) of the final states that can be reached starting from states satisfying the *presumption*.

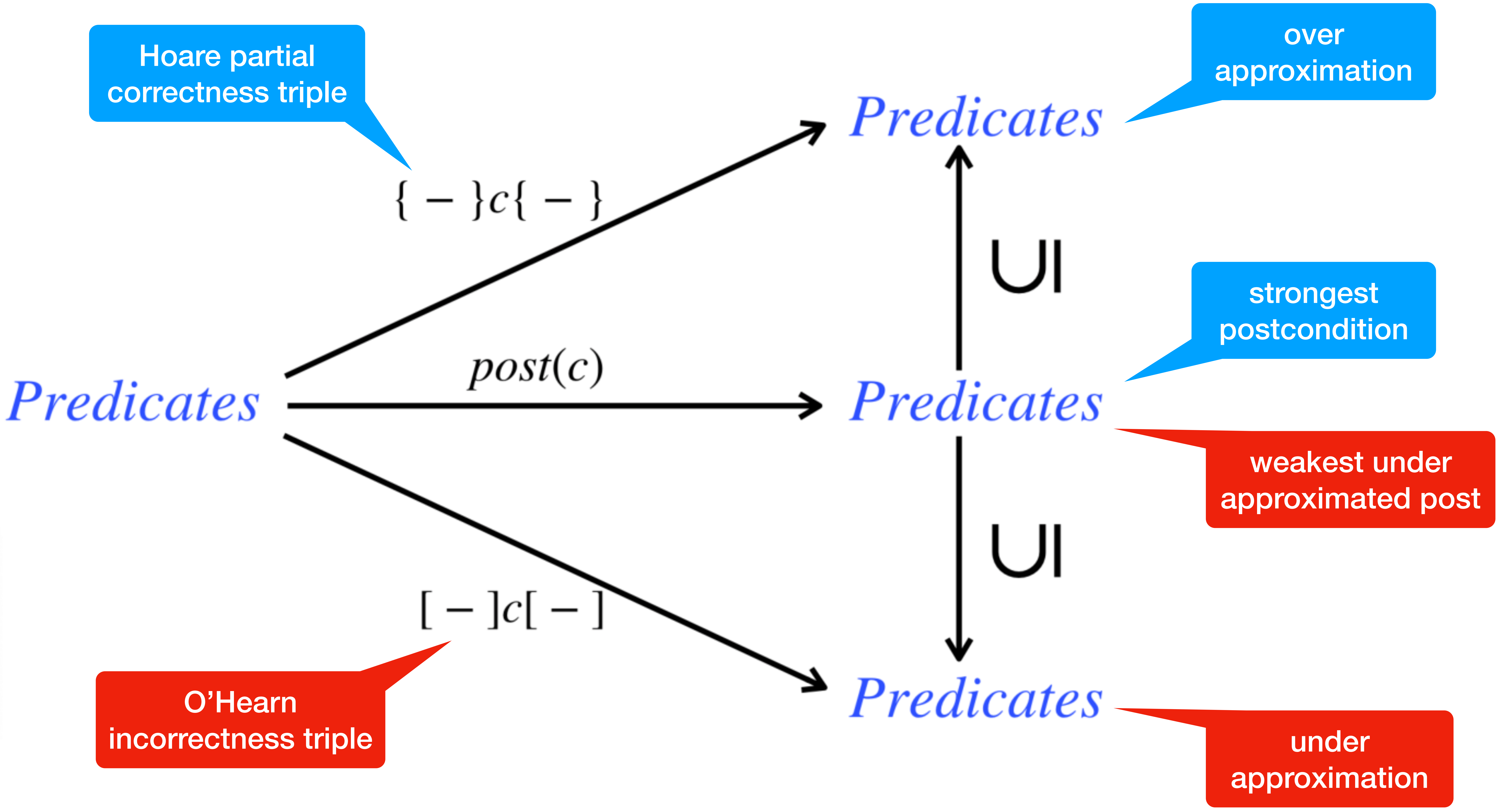
The under-approximate triples were studied (with a different but equivalent definition) previously by de Vries and Koutavas [2011] in their reverse Hoare logic, which they used to specify randomized algorithms. Incorrectness logic adds post-assertions for errors as well as for normal termination, and these assertions describe erroneous states that can be reached by actual program executions. Dijkstra [1976] famously remarked that "testing can be quite effective for showing the presence of bugs, but is hopelessly inadequate for showing their absence," and he made this remark while arguing for the

Author's address: Peter W. O'Hearn, Facebook and University College London, UK.

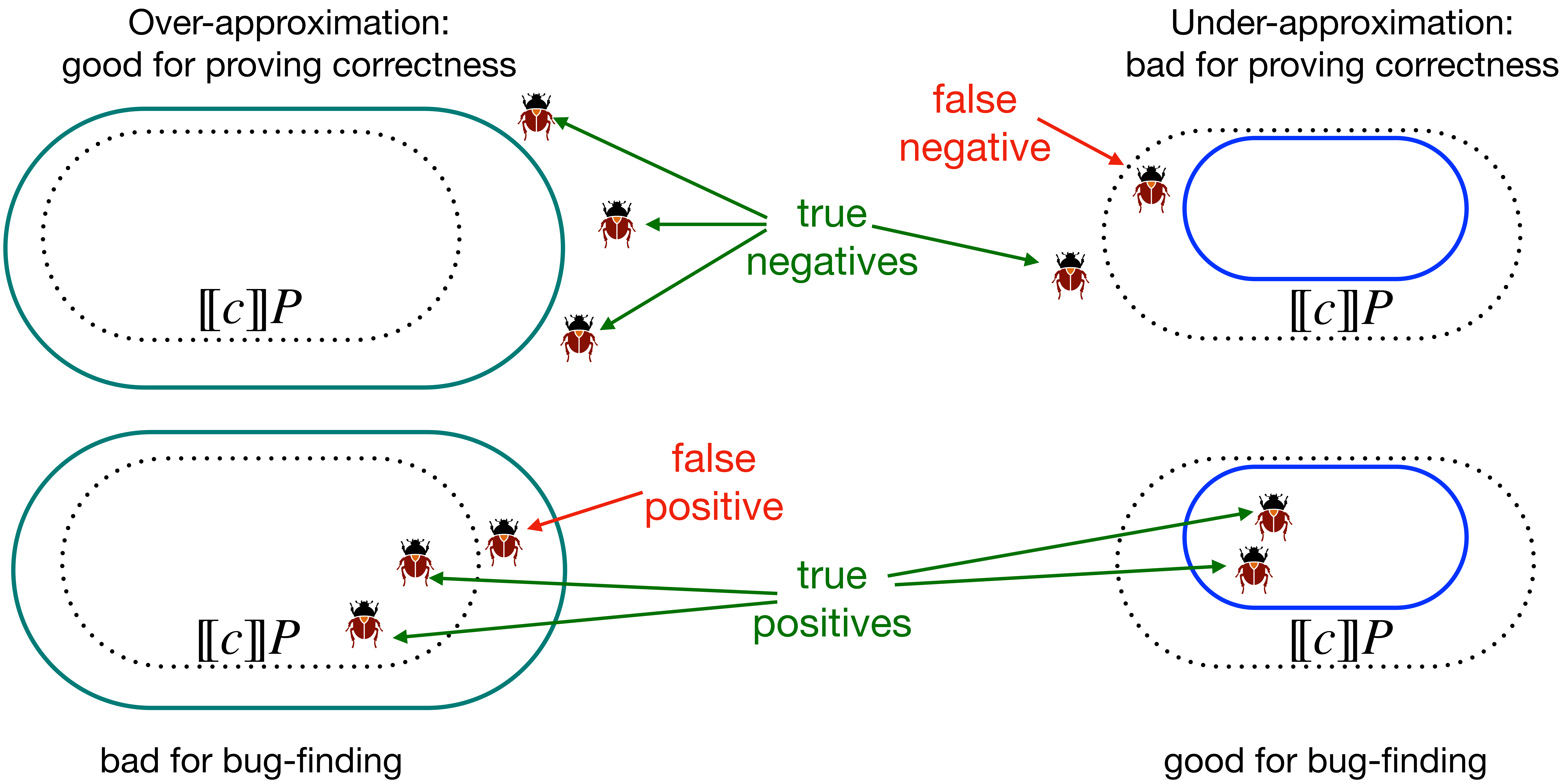
2475-1421/2020/1-ART10
https://doi.org/10.1145/3371078

"Program correctness and incorrectness are two sides of the same coin"
Peter O'Hearn (2020)

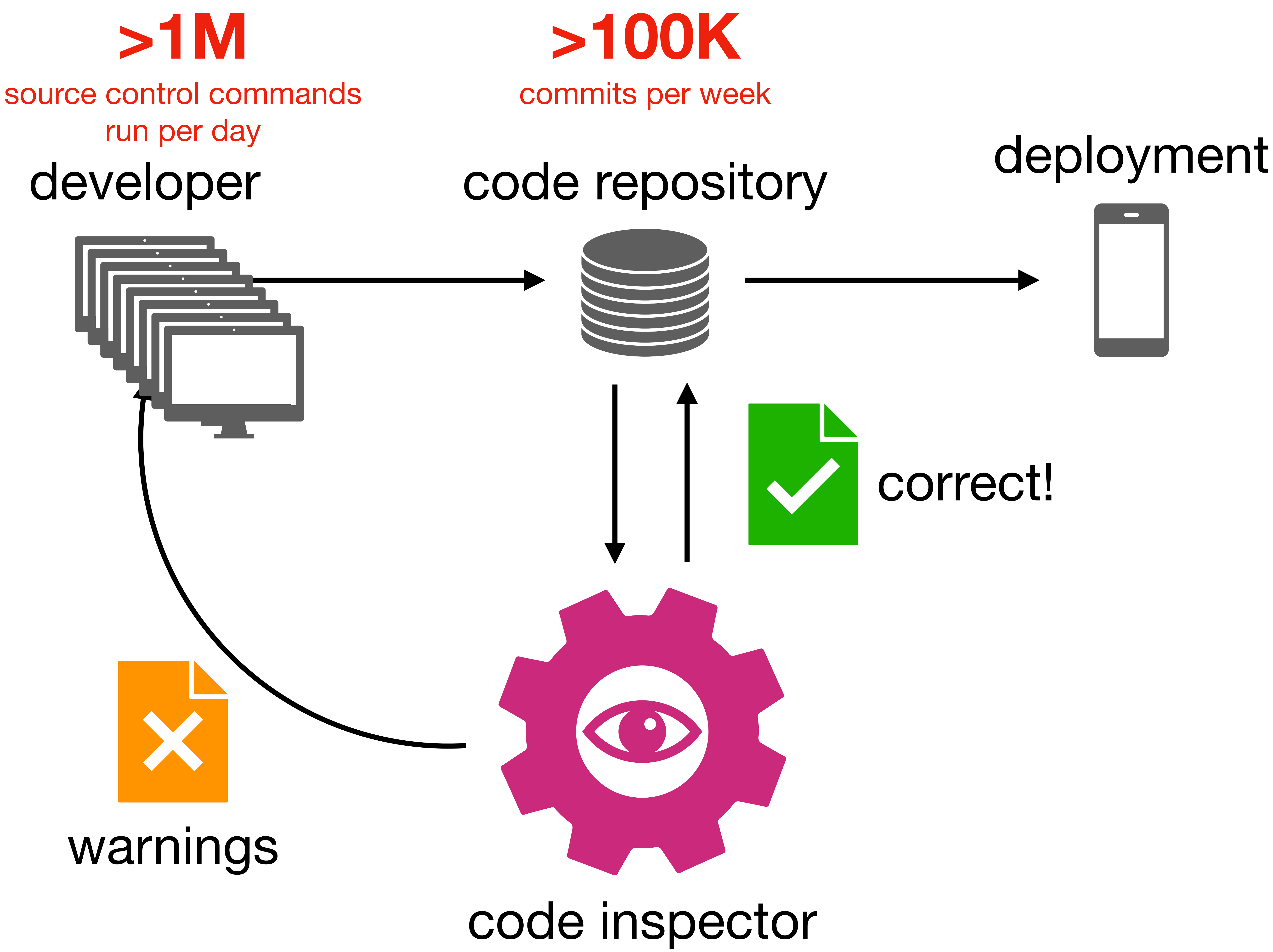
Picturing incorrectness



Correctness vs incorrectness



Correctness workflow, ideally



some scalability issues in a production environment:
analysis takes time (overnight?),
warnings are received late,
false positives mine credibility

Design principles

Low friction
do not rely on manual annotations

Act fast
able to report errors in less than 15’

Be compositional
whole program analysis is discouraged

Occam
do not use complex techniques (unless forced)

True positive theorem!
(under certain assumptions) the analyzer reports no false positives

“do not spam the developers!”

Incorrectness Logic
(IL)

Hoare's triples

pre
condition

$$\{P\}\ c\ \{Q\}$$

post
condition

for any input matching the precondition
executing the command establishes the postcondition

$$[\![c]\!]P \subseteq Q$$

over
approximation!

can include non
reachable states

O'Hearn's triples

pre
condition

$$[P]\ c\ [Q]$$

post
condition

any output matching the postcondition
can be reached by executing the command
on some input matching the precondition

$$[\![c]\!]P \supseteq Q$$

under
approximation!

includes just
reachable states

As first order formulas

$$\{P\}\ c\ \{Q\}$$

$$[\![c]\!]P \subseteq Q \qquad \equiv \qquad \forall \sigma \in P \,.\, \forall \sigma' \in [\![c]\!]\sigma \,.\, \sigma' \in Q$$

any reachable output satisfies the postcondition

$$[P]\ c\ [Q]$$

$$[\![c]\!]P \supseteq Q \qquad \equiv \qquad \forall \sigma' \in Q \,.\, \exists \sigma \in P \,.\, \sigma' \in [\![c]\!]\sigma$$

any output in the postcondition is reachable

Regular commands

regular command

atomic command

choice

Kleene star

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

$$
\begin{aligned}
e ::= \;& \mathsf{skip} \\
\mid \;& x := a \\
\mid \;& b? \\
\mid \;& \mathsf{error}(\,) \\
\mid \;& x := \mathsf{nondet}(\,) \\
\mid \;& \ldots
\end{aligned}
$$

Exit condition

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

ϵ is the exit condition

ok: normal execution

er: erroneous execution

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

$[y = v]$ error( ) $[\text{er} : y = v]$

Notation

$$[P]\ r\ [\mathsf{ok} : Q_1][\mathsf{er} : Q_2]$$

stands for

$$[P]\ r\ [\mathsf{ok} : Q_1] \text{ and } [P]\ r\ [\mathsf{er} : Q_2]$$

Floyd's axiom for assignment

$$\frac{}{[P]\ x := a\ [\mathsf{ok} : \exists x'.\, P[x'/x] \wedge x = a[x'/x]][\mathsf{er} : \mathsf{false}]}$$

$$[y = 42]\ \mathtt{x := 42}\ [\mathsf{ok} : x = y = 42]$$

Hoare's axiom for assignment?

$$\overline{[Q[a/x]]\ x := a\ [\mathsf{ok} : Q][\mathsf{er} : \mathsf{false}]}$$

$[y = 42]$ x := 42 $[\mathsf{ok} : x = y]$

unsound!

$\sigma \triangleq [x \mapsto 3, y \mapsto 3]$ not reachable

Other atomic commands

$$\frac{}{[P]\ \mathsf{skip}\ [\mathsf{ok}: P][\mathsf{er}: \mathsf{false}]}$$

$$\frac{}{[P]\ b?\ [\mathsf{ok}: P \wedge b][\mathsf{er}: \mathsf{false}]}$$

$$\frac{}{[P]\ \mathsf{error}()\ [\mathsf{ok}: \mathsf{false}][\mathsf{er}: P]}$$

$$\frac{}{[P]\ x := \mathsf{nondet}()\ [\mathsf{ok}: \exists x\,.\ P][\mathsf{er}: \mathsf{false}]}$$

Short circuiting of errors

$$\frac{[P]\ r_1\ [\mathsf{ok} : R] \qquad [R]\ r_2\ [\epsilon : Q]}{[P]\ r_1 ; r_2\ [\epsilon : Q]}$$

$$\frac{[P]\ r_1\ [\mathsf{er} : Q]}{[P]\ r_1 ; r_2\ [\mathsf{er} : Q]}$$

$[y = v]$ error( ) ; x := y $[\mathsf{er} : y = v]$

Dropping disjuncts

$$\frac{[P]\ r_1\ [\epsilon : Q]}{[P]\ r_1 + r_2\ [\epsilon : Q]} \qquad \frac{[P]\ r_2\ [\epsilon : Q]}{[P]\ r_1 + r_2\ [\epsilon : Q]}$$

sound under-approximation!
scalable bug detection

$[y = v]$ error( ) + x := y $[\text{er} : y = v]$

$[y = v]$ error( ) + x := y $[\text{ok} : x = y = v]$

Example

$[y = 0]$ if $even(x)$ then $y := 42$ $[\mathrm{ok} : y = 42]$

is it a valid IL triple?

$$(y = 42) \triangleq \{\ [x \mapsto 0, y \mapsto 42], [x \mapsto 1, y \mapsto 42], [x \mapsto 2, y \mapsto 42], \ldots\ \}$$

Example

$[y = 0]$ if $even(x)$ then $y := 42$ $[\mathsf{ok} : y = 42 \land even(x)]$

is it a valid IL triple?

$y = 42 \land even(x) \triangleq \{\ [x \mapsto 0, y \mapsto 42], [x \mapsto 2, y \mapsto 42], \ldots\ \}$

IL vs HL

$$[y = 0]\ \text{if } even(x) \text{ then } y := 42\ [\text{ok} : y = 42 \land even(x)]$$ ✓

$$\{y = 0\}\ \text{if } even(x) \text{ then } y := 42\ \{y = 42 \land even(x)\}$$ ✗

$$\{y = 0 \land even(x)\}\ \text{if } even(x) \text{ then } y := 42\ \{y = 42\}$$ ✓

$$[y = 0 \land even(x)]\ \text{if } even(x) \text{ then } y := 42\ [\text{ok} : y = 42]$$ ✗

Bounded loop unrolling

$$\frac{}{[P]\ r^\star\ [\mathsf{ok} : P]}$$

$$\frac{[P]\ r^\star; r\ [\epsilon : Q]}{[P]\ r^\star\ [\epsilon : Q]}$$

sound under-approximation!
scalable bug detection

$$[x = 0]\ (x := x + 1)^\star\ [\mathsf{ok} : x = 0]$$

$$[x = 0]\ (x := x + 1)^\star\ [\mathsf{ok} : x = 2]$$

Backwards variant (weak)

$$\frac{\forall n \in \mathbb{N}.\ [P_n]\ r\ [\mathsf{ok} : P_{n+1}]}{[P_0]\ r^\star\ [\mathsf{ok} : P_k]}$$

loop invariants are inherently over-approximations
sub-variants to reason about loop under-approximation

$[x = 0]\ (x := x + 1)^\star\ [\mathsf{ok} : x = 2^{42}]$ // $P_n \triangleq (x = n)$

$[x = 0]\ (x := x + 1)^\star\ ;\ \mathsf{if}\ (x = 2^{42})\ \mathsf{then}\ \mathsf{error}()\ [er : x = 2^{42}]$

Consequence rule

$$\frac{P' \Rightarrow P \quad [P']\, r\, [\epsilon : Q'] \quad Q \Rightarrow Q'}{[P]\, r\, [\epsilon : Q]}$$

shrink the post!
scalable bug detection

$$\frac{P \Rightarrow P' \quad \{P'\}\, r\, \{Q'\} \quad Q' \Rightarrow Q}{\{P\}\, r\, \{Q\}}$$

Some dualities

$$[P]\ r\ [Q_1] \ \wedge\ [P]\ r\ [Q_2] \quad \Leftrightarrow \quad [P]\ r\ [Q_1 \vee Q_2]$$

$$\{P\}\ r\ \{Q_1\} \wedge \{P\}\ r\ \{Q_2\} \quad \Leftrightarrow \quad \{P\}\ r\ \{Q_1 \wedge Q_2\}$$

Some dualities

dropping disjuncts (by conseq. rule)

$$\frac{[P]\ r\ [Q \vee R]}{[P]\ r\ [Q]}$$

dropping conjuncts (by conseq. rule)

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

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

Principle of agreement

Th.

If $[P'] \; r \; [Q'] \; \wedge$

$P' \Rightarrow P \; \wedge$

$\{P\} \; r \; \{Q\}$

then $Q' \Rightarrow Q$

Proof.

$Q' \subseteq$ // by IL

$[\![r]\!]P' \subseteq$ // $P' \Rightarrow P$

$[\![r]\!]P \subseteq$ // by HL

Q

partially correct programs cannot exhibit counterexamples

Principle of denial

Th.

If $[P'] \, r \, [Q'] \wedge$
$P' \Rightarrow P \wedge$
$\{P\} \, r \, \{Q\}$
then $Q' \Rightarrow Q$

Cor.

If $[P'] \, r \, [Q'] \wedge$
$P' \Rightarrow P \wedge$
$\neg(Q' \Rightarrow Q)$
then $\neg(\{P\} \, r \, \{Q\})$

any derivable counterexample witnesses program incorrectness

Examples

$[\text{true}]$

if $x \geq 0$ then

$[x \geq 0]$

skip

$[x \geq 0]$

else

$[x < 0]$

$x := -x$

$[\exists x'.\ x' < 0 \land x = -x'] \equiv [x > 0]$

$[\text{ok} : x \geq 0]$

Examples

$[z = 11]$

if $even(x)$ then

$[z = 11 \wedge even(x)]$

if $odd(y)$ then

$[z = 11 \wedge even(x) \wedge odd(y)]$

$z := 42$

$[z = 42 \wedge even(x) \wedge odd(y)]$

$[\text{ok} : z = 42 \wedge even(x) \wedge odd(y)]$

Finite unrolling of while loops

$$\text{while } b \text{ do } c \triangleq (b?; c)^\star; \neg b?$$

$$\frac{}{[P]\ \text{while } b \text{ do } c\ [\text{ok} : P \wedge \neg b]}$$

$$\frac{[P \wedge b]\ c\ [\text{ok} : Q]}{[P]\ \text{while } b \text{ do } c\ [\text{ok} : (P \vee Q) \wedge \neg b]}$$

Finite unrolling of while loops

$$\text{while } b \text{ do } c \triangleq (b?;c)^\star; \neg b?$$

$$\frac{\dfrac{}{[P]\ (b?;c)^\star\ [\text{ok} : P]} \qquad \dfrac{}{[P]\ \neg b?\ [\text{ok} : P \land \neg b]}}{[P]\ \text{while } b \text{ do } c\ [\text{ok} : P \land \neg b]}$$

Finite unrolling of while loops

$$\text{while } b \text{ do } c \triangleq (b?; c)^\star; \neg b?$$

$$r \triangleq b?; c$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{[P]\ r^\star\ [\text{ok} : P]} \quad \dfrac{\dfrac{}{[P]\ b?\ [\text{ok} : P \wedge b]} \quad [P \wedge b]\ c\ [\text{ok} : Q]}{[P]\ r\ [\text{ok} : Q]}}{[P]\ r^\star; r\ [\text{ok} : Q]}}{[P]\ r^\star\ [\text{ok} : Q]} \quad \dfrac{}{[Q]\ \neg b?\ [\text{ok} : Q \wedge \neg b]}}{[P]\ \text{while } b \text{ do } c\ [\text{ok} : (P \vee Q) \wedge \neg b]}$$

Examples

$[\text{true}]$

$n := \text{nondet}(\,);$

$[\text{true}]$

$x := 0;$

$[x = 0]$

$\text{while } n > 0 \text{ do } ($

$[x = 0 \land n > 0]$

$x := x + n;$

$[x = n \land n > 0]$

$n := \text{nondet}(\,)$

$[\exists n\,.\; x = n \land n > 0] \equiv [x > 0]$

$)\;\; [\text{ok} : x \geq 0 \land n \leq 0]$

$$\frac{[P \land b]\; c\; [\text{ok} : Q]}{[P]\; \text{while } b \text{ do } c\; [\text{ok} : (P \lor Q) \land \neg b]}$$

Validity, soundness, completeness

Validity

A IL triple $[P]\ r\ [Q]$ is **valid** if $Q \subseteq [\![r]\!]P$

Is $[x > 0]\ x := 10x\ [x > 10]$ valid?

Is $[x > 0, y > 0]\ x := yx\ [x \geq 0]$ valid?

Is $[x > 0, y > 0]\ x := yx\ [x = 42, y = 7]$ valid?

Is $[xy > 0]\ (x := yx)^\star\ [x > 0, y \neq 0]$ valid?

difficulty level: highest

Relational semantics

$$[\![r]\!] : \wp(\Sigma) \to \wp(\Sigma)$$

$$[\![r]\!]\epsilon \subseteq \Sigma \times \Sigma$$

$$[\![r]\!]\mathsf{ok} \subseteq \Sigma \times \Sigma$$

$$[\![r]\!]\mathsf{er} \subseteq \Sigma \times \Sigma$$

Semantics: atomic commands

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

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

$[\![b?]\!]\mathsf{ok} \triangleq \{(\sigma, \sigma) \mid \sigma \vDash b\}$

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

$[\![x := a]\!]\mathsf{ok} \triangleq \{(\sigma, \sigma[x \mapsto [\![a]\!]\sigma]) \mid \sigma \in \Sigma\}$

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

common
constructs

Semantics: atomic commands

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

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

$$[\![x := \mathsf{nondet}(\,)]\!]\mathsf{ok} \triangleq \{(\sigma, \sigma[x \mapsto v]) \mid \sigma \in \Sigma, v \in \mathbb{Z}\}$$

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

“exotic” constructs

Semantics: compositions

$S, T \subseteq \Sigma \times \Sigma$

$T \circ S \triangleq \{(\sigma_1, \sigma_2) \mid \exists \sigma \, . \, (\sigma_1, \sigma) \in S \land (\sigma, \sigma_2) \in T\} \subseteq \Sigma \times \Sigma$

$[\![r_1; r_2]\!]\mathrm{ok} \triangleq [\![r_2]\!]\mathrm{ok} \circ [\![r_1]\!]\mathrm{ok}$

$[\![r_1; r_2]\!]\mathrm{er} \triangleq [\![r_1]\!]\mathrm{er} \cup ([\![r_2]\!]\mathrm{er} \circ [\![r_1]\!]\mathrm{ok})$

$[\![r_1 + r_2]\!]\epsilon \triangleq [\![r_1]\!]\epsilon \cup [\![r_2]\!]\epsilon$

$[\![r^\star]\!]\epsilon \triangleq \bigcup_{k \in \mathbf{N}} [\![r^k]\!]\epsilon$ where $r^k \triangleq \underbrace{r; \cdots; r}_{k \text{ times}}$

Minimal set of rules

$$\frac{}{[P]\ e\ [\llbracket e \rrbracket P]}\ \text{[atom]}$$

$$\frac{[P]\ r_1\ [R] \qquad [R]\ r_2\ [Q]}{[P]\ r_1; r_2\ [Q]}\ \text{[seq]}$$

$$\frac{\forall i \in \{1,2\} \quad [P]\ r_i\ [Q_i]}{[P]\ r_1 + r_2\ [Q_1 \cup Q_2]}\ \text{[choice]}$$

$$\frac{\forall n \geq 0.\ [P_n]\ r\ [P_{n+1}]}{[P_0]\ r^\star\ [\exists k.\ P_k]}\ \text{[iter]}$$

$$\frac{P' \Rightarrow P \quad [P']\ r\ [Q'] \quad Q \Rightarrow Q'}{[P]\ r\ [Q]}\ \text{[cons]}$$

Auxiliary rules

$$\frac{[P_1]\ r\ [Q_1] \qquad [P_2]\ r\ [Q_2]}{[P_1 \vee P_2]\ r\ [Q_1 \vee Q_2]}\ \text{[disj]}$$

$$\frac{}{[P]\ r^\star\ [P]}\ \text{[iter0]}$$

$$\frac{[P]\ r^\star ; r\ [Q]}{[P]\ r^\star\ [Q]}\ \text{[unroll]}$$

assigned variables in r are disjoint from free variables in R

$$\frac{[P]\ r\ [Q]}{[P \wedge R]\ r\ [Q \wedge R]}\ \text{[frame]}$$

$$\frac{P' \Rightarrow P \qquad [P']\ r\ [Q]}{[P]\ r\ [Q]}\ \text{[weak]}$$

$$\frac{[P]\ r\ [Q'] \qquad Q \Rightarrow Q'}{[P]\ r\ [Q]}\ \text{[stren]}$$

Correctness

Th. Any derivable IL triple is valid

Proof. By induction on the derivation tree

(Relative) Completeness

independent of all but a finite number of varables

involving finitely-supported predicates

Th. Any valid IL triple can be derived.

Proof. (Assuming an oracle to decide implications.)

Roughly, by structural induction on the command r.

Atomic commands: [atom] + [cons]

Choice and sequence: by inductive hyp. + [disj] + [cons]

Kleene star: see O'Hearn's paper

For iteration first we do the proof for $\epsilon = ok$. Supposing $[p](C)^\star[ok{:}\, q]$ is true, we define

$$p(n) = \{\sigma \mid \text{you can get back from } \sigma \text{ to some state in } p \text{ by executing } C \text{ backwards } n \text{ times}\}.$$

Note that $p(0) = p$ by this definition. From the definition of $p(n)$ it is evident that

$$[p(n) \land nat(n)]C[ok{:}\, p(n+1) \land nat(n)]$$

is true, and hence it is provable by induction hypothesis. We apply the Backwards Invariant rule and then Consequence using $q \Rightarrow \exists n.p(n)$, which is a true implication because of the Characterization lemma. This shows that $[p](C)^\star[ok{:}\, q]$ is provable. (We use n to describe the number of iterations in a similar way to Harel [1979], except that he appeals to Gödel encoding, and to de Vries and Koutavas [2011], who use an infinitary disjunction.)

Now, for $\epsilon = er$ we use the idea is that if an error is thrown then some number of successful iterations happens first, followed by error happening on thenext (last) iteraiton. We use the rule Iterate non-zero to deal with this case. So, suppose $[p](C)^\star[er{:}\, q]$ is true and define *frontier* to be the reachable states for normal termination; i.e., $frontier = post([\![C]\!]ok)p$. By the just-proven completeness case for iteration and normal termination, we know that $[p](C)^\star[ok{:}\, frontier]$ is provable. Now, $[frontier]C[er{:}\, q]$ must be true (note the absence of $\star$), or else the beginning assumption that $[p](C)^\star[er{:}\, q]$ could not be. By induction hypothesis we know $[frontier]C[er{:}\, q]$ is provable, and we can use Sequencing (normal) and Iterate non-zero to conclude that $[p](C)^\star[er{:}\, q]$ is provable.

Questions

Question 1

Which IL triples are valid for any r and P ?

$[P]\ r\ [\mathsf{ok} : \mathsf{false}][\mathsf{er} : \mathsf{false}]$ ✓

$[P]\ r\ [\mathsf{ok} : \mathsf{true}]$ ✗

$[\mathsf{true}]\ r\ [\mathsf{ok} : P]$ ✗

$[wlp(r, P)]\ r\ [\mathsf{ok} : P]$ ✗

Question 2

Find a derivation for the IL triple

$[\text{true}]$ if $x \geq y$ then $z := x$ else $z := y$ $[ok : z = \max(x, y)]$

$[\text{true}]$

if $x \geq y$ then

$z := x$

else

$z := y$

Question 2

Find a derivation for the IL triple

[true] if $x \geq y$ then $z := x$ else $z := y$ [ok : $z = \max(x, y)$]

[true]

if $x \geq y$ then

[$x \geq y$]

$z := x$

[$z = x \geq y$] $\equiv$ [$x \geq y, z = \max(x, y)$]

else

[$x < y$]

$z := y$

[$z = y > x$] $\equiv$ [$y > x, z = \max(x, y)$]

[ok : $z = \max(x, y)$]

Question 3

Show that the following rule for assignment is not sound

$$\frac{}{[P]\ x := a\ [\mathsf{ok} : P[a/x]]}$$

syntax replacement

Consider the instance $[x = y]\ x := 0\ [\mathsf{ok} : y = 0]$

then $(x \mapsto 1, y \mapsto 0) \models (y = 0)$ but is not a reachable state!