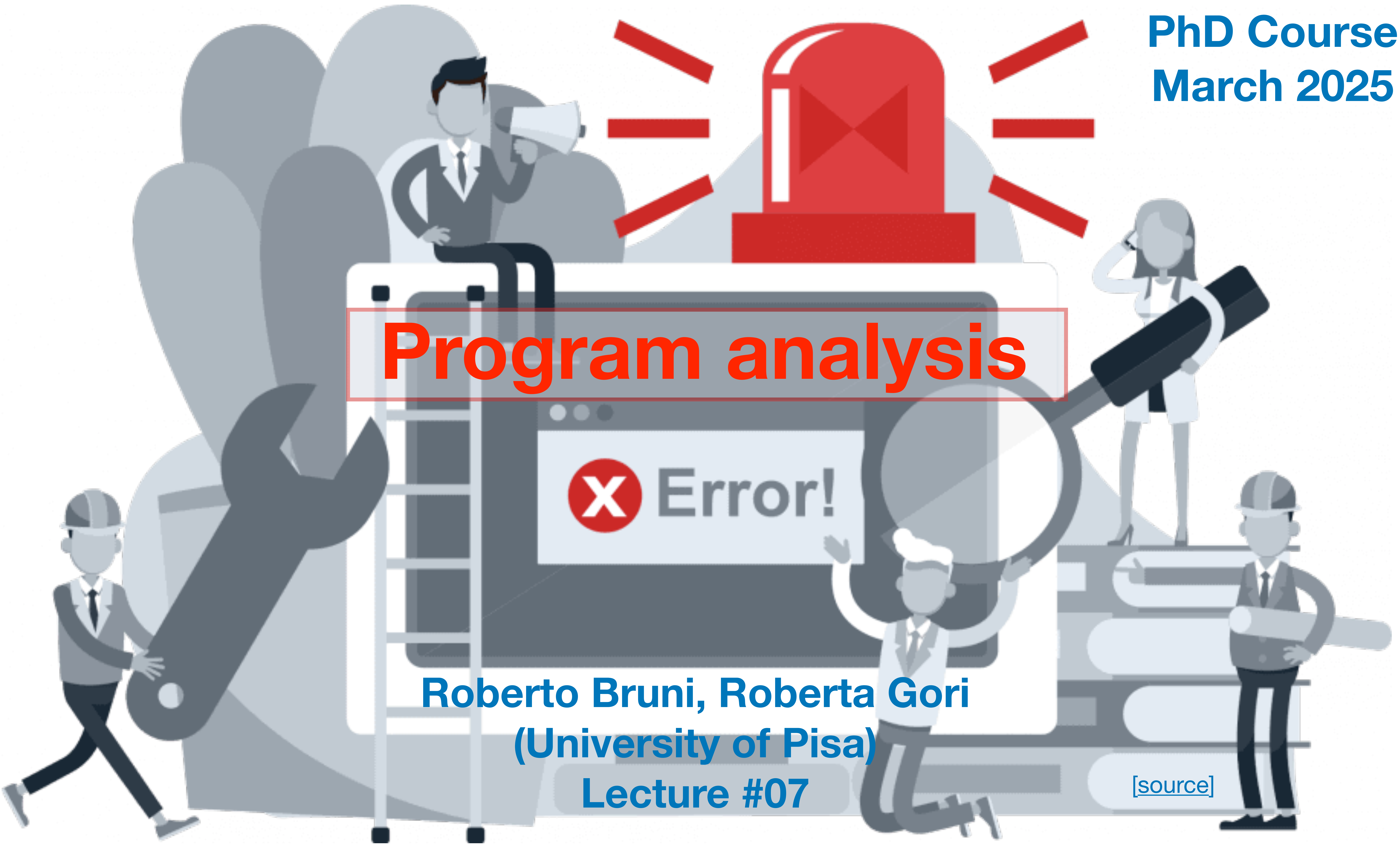
PhD Course
March 2025

Program analysis



Roberto Bruni, Roberta Gori
(University of Pisa)

Lecture #07

[source]

Taxonomy of program logics

Different logics for different purposes!!

Hoare Logic

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

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

$$\forall \sigma \in P\,.\, \forall \delta \in [\![c]\!]\sigma\,.\, \delta \in Q$$

can prove the absence of bugs

(any execution of c from P is correct)

Example

$\{x \leq 0, y = 1\}$

while(x≤5) do x:=x+y;

Example

$\{x \leq 0, y = 1\}$

while(x≤5) do x:=x+y;

$\{x = 6\}$

Example

$\{x \leq 0\}$

while(x≤5) do x:=x+y;

$??\{x = 6\}$

Example

$\{x \leq 0\}$

while(x≤5) do x:=x+y;

$\{x \geq 6\}$

Example

$\{x \leq 0\}$

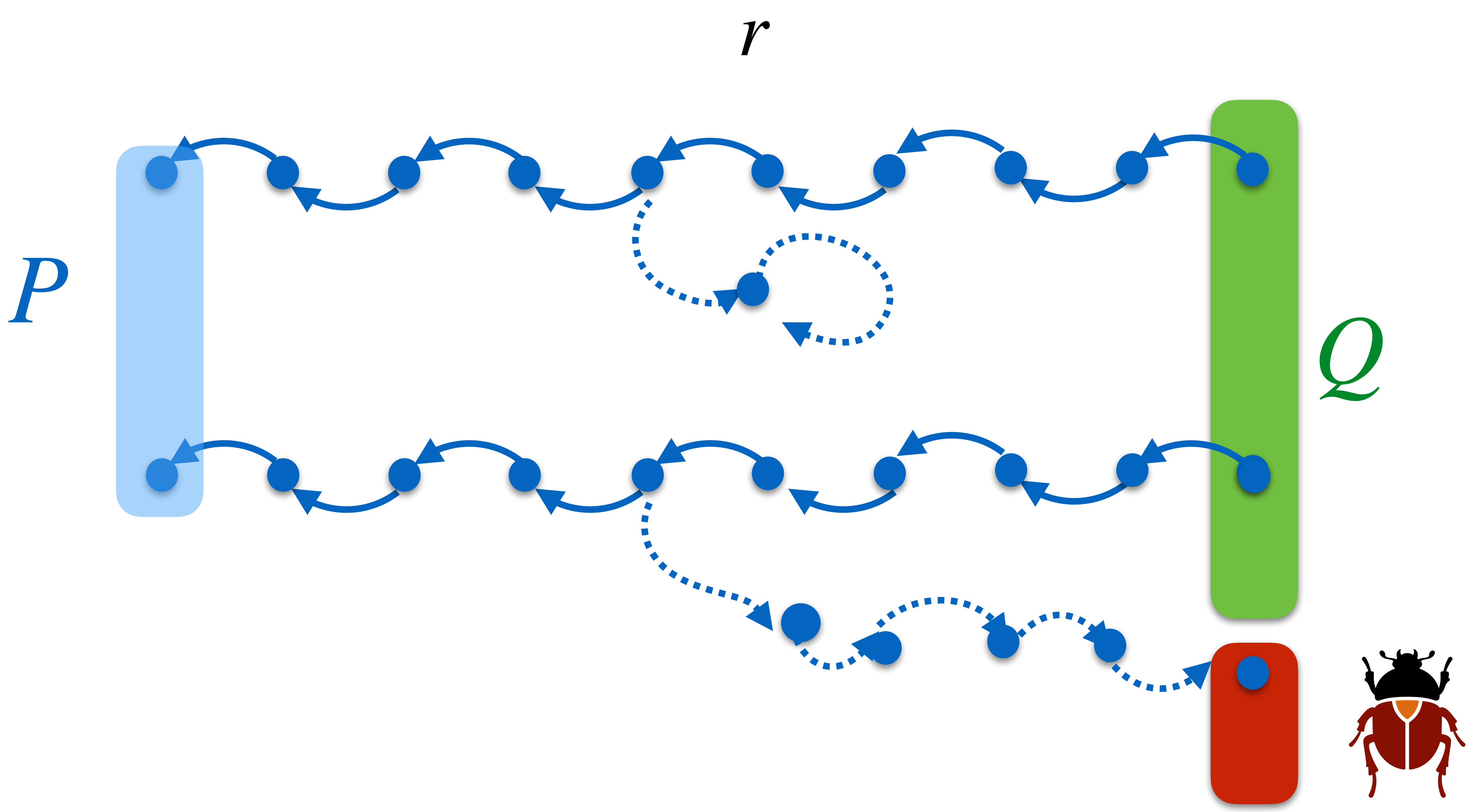
while(x≤5) do x:=x+y;

$\{x \geq 0\}$

Backward semantics

$$[\![\overleftarrow{r}]\!]\sigma' \triangleq \{\sigma \mid \sigma' \in [\![r]\!]\sigma\}$$

$$\sigma \in [\![\overleftarrow{r}]\!]\sigma' \Leftrightarrow \sigma' \in [\![r]\!]\sigma$$



Necessary condition

$$(P)\ c\ (Q)$$

validity: $P \supseteq [\![\overleftarrow{c}]\!] Q$

$$\forall \delta \in Q \,.\, \forall \sigma \in [\![\overleftarrow{c}]\!] \delta \,.\, \sigma \in P$$

express necessary conditions for correctness

(any execution of c from outside P is incorrect)

Example

```
while(x≤5) do x:=x+y;
```

$(x = 6)$

Example

$(x \leq 6 \wedge \exists n \,.\, n * y = 6 - x)$

```
while(x≤5) do x:=x+y;
```

$(x = 6)$

Example

$(x \leq 6)$

```
while(x≤5) do x:=x+y;
```

$(x = 6)$

Incorrectness Logic (BUA)

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

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

$$\forall \delta \in Q\ .\ \exists \sigma \in P\ .\ \delta \in [\![c]\!]\sigma$$

can prove the presence of bugs

(any error in Q is reachable executing c)

Example

$[x \leq 0]$

```
while(x≤5) do x:=x+y;
```

$??[x = 6]$

Reachable??

Example

$[x \leq 0]$

```
while(x≤5) do x:=x+y;
```

$??[x = 6]$

Reachable??

$No!! \; x = 6 \land y = -1$

Example

$[x \leq 0]$

```
while(x≤5) do x:=x+y;
```

$[x = 6 \wedge y > 0]$

Reachable??

Example

$[x \leq 0]$

```
while(x≤5) do x:=x+y;
```

$[x = 6 \wedge y > 0]$

Reachable??

$$\forall y > 0, \exists min\ n\ .\ \ 6 - (n * y) \leq 0,\ x = 6 - (n * y)$$

Example

$[x \leq 8]$

```
while(x≤5) do x:=x+y;
```

$[x = 6 \wedge y > 0]$

Reachable??

Sufficient Incorrectness Logic (FUA)

$$\langle P \rangle \; c \; \langle Q \rangle$$

validity: $P \subseteq [\![\overleftarrow{c}]\!] Q$

$$\forall \sigma \in P \, . \, \exists \delta \in Q \, . \, \delta \in [\![c]\!] \sigma$$

express sufficient conditions for incorrectness

(any state in P can lead to $er : Q$)

Example

```
while(x≤5) do x:=x+y;
```

$\langle x = 6 \rangle$

Example

$$\langle x \leq 6 \wedge \exists n \,.\, n * y = 6 - x \rangle$$

while(x≤5) do x:=x+y;

$$\langle x = 6 \rangle$$

Example

$$< x \leq 6 \wedge y = 6 - x >$$

```
while(x≤5) do x:=x+y;
```

$$\langle x = 6 \rangle$$

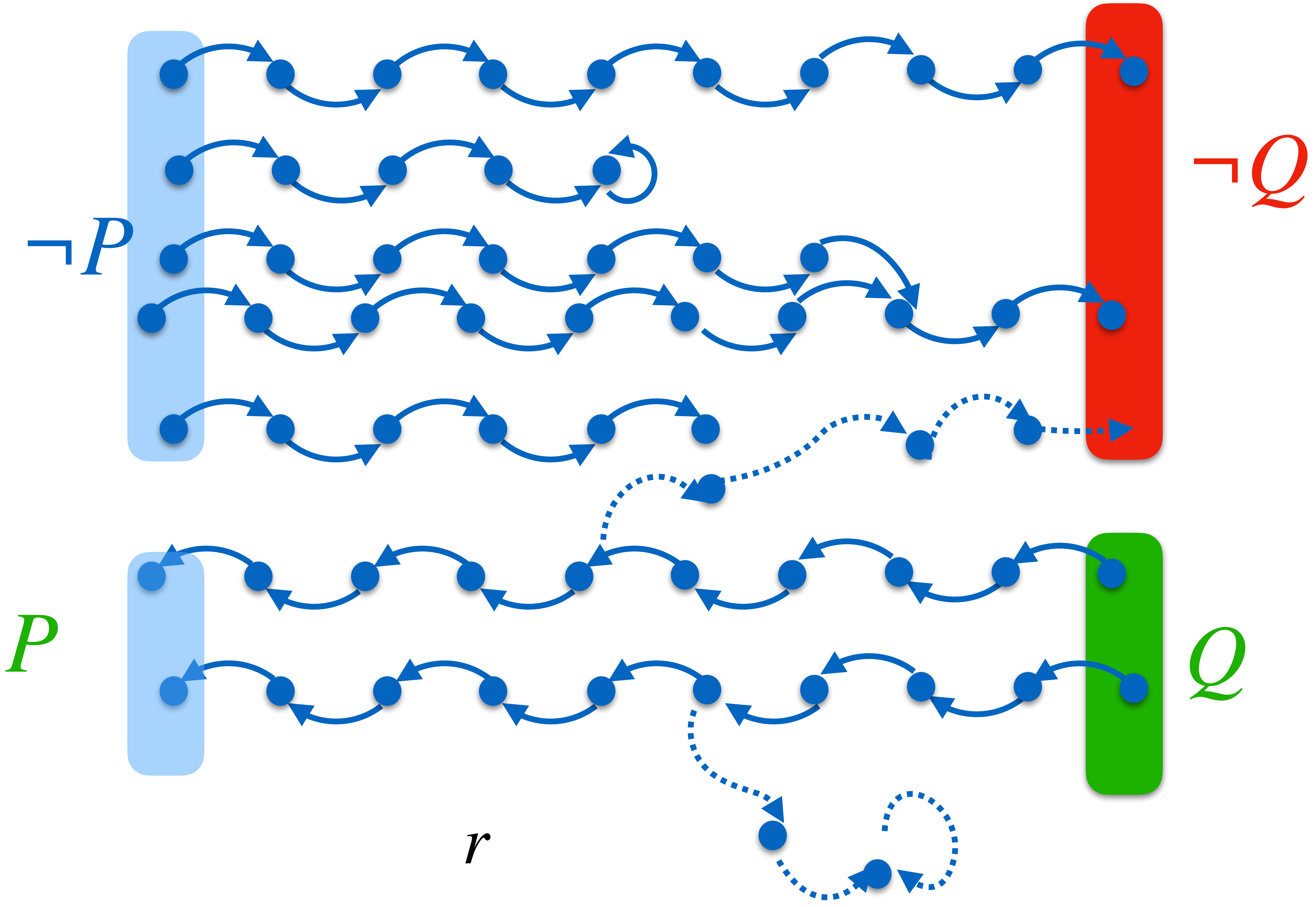
The taxonomy

	Forward	Backward
Over	HL $\{P\}\ c\ \{Q\}$ $[\![c]\!]P \subseteq Q$	NC $(P)\ c\ (Q)$ $P \supseteq [\![\overleftarrow{c}]\!]Q$
Under	IL $[P]\ c\ [Q]$ $[\![c]\!]P \supseteq Q$	SIL $\langle P\rangle\ c\ \langle Q\rangle$ $P \subseteq [\![\overleftarrow{c}]\!]Q$

Compare logics along the approximation axis

	Forward	Backward
Over	{HL} $[\![r]\!]P \subseteq Q$	(NC) $[\![\overleftarrow{r}]\!]Q \subseteq P$
Under	[IL] $[\![r]\!]P \supseteq Q$	⟨⟨SIL⟩⟩ $[\![\overleftarrow{r}]\!]Q \supseteq P$

NC vs HL



$$\{\neg P\}\ r\ \{\neg Q\} \iff (P)\ r\ (Q)$$

$$[\![r]\!]\neg P \subseteq \neg Q \iff [\![\overleftarrow{r}]\!]Q \subseteq P$$

Compare logics along the approximation axis

	Forward	Backward
Over	{HL} $[\![r]\!]P \subseteq Q$	(NC) $[\![\overleftarrow{r}]\!]Q \subseteq P$
Under	[IL] $[\![r]\!]P \supseteq Q$	⟪SIL⟫ $[\![\overleftarrow{r}]\!]Q \supseteq P$

Compare logics along the approximation axis

	Forward	Backward
Over	{HL} $[\![r]\!]P \subseteq Q$	(NC) $[\![\overleftarrow{r}]\!]Q \subseteq P$
Under	[IL] $[\![r]\!]P \supseteq Q$	⟨⟨SIL⟩⟩ $[\![\overleftarrow{r}]\!]Q \supseteq P$

SIL vs IL

c_{42}:

```
if even (x) {
        if  odd(y) { z := 42; }
}
```

Safe $z \neq 42$
E.g., x:=1/(42- z)

No relations!

Given a specification of the possible errors

$Q \triangleq \{ z = 42 \}$

With IL one can prove

$[\, z{=}11 \,]\ c_{42}\ [\, z{=}42 \wedge odd(y) \wedge even(x) \,]$

Expressing that the postcondition is reachable

With SIL one can prove

$\langle\!\langle z{=}11 \wedge \mathrm{odd}(y) \wedge \mathrm{even}(x) \rangle\!\rangle\ c_{42}\ \langle\!\langle z{=}42 \rangle\!\rangle$

Expressing a precondition that leads to error states

Compare logics according to the consequence rule

	Forward	Backward
Over	{HL} $[\![r]\!]P \subseteq Q$	(NC) $[\![\overleftarrow{r}]\!]Q \subseteq P$
Under	[IL] $[\![r]\!]P \supseteq Q$	⟪SIL⟫ $[\![\overleftarrow{r}]\!]Q \supseteq P$

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

Consequence rules follows the diagonal of the schema, so they suggest relations between HL-SIL and IL-NC

Compare logics according to the consequence rule

	Forward	Backward
Over	{HL} $[\![r]\!]P \subseteq Q$	(NC) $[\![\overleftarrow{r}]\!]Q \subseteq P$
Under	[IL] $[\![r]\!]P \supseteq Q$	⟪SIL⟫ $[\![\overleftarrow{r}]\!]Q \supseteq P$

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

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

Consequence rules follows the diagonal of the schema, so they suggest relations between HL-SIL and IL-NC

Relations following the diagonals

NC-IL: no relation

HL-SIL: loosely related,

r deterministic and terminating: SIL equivalent to HL

$$\langle\!\langle P \rangle\!\rangle\ r\ \langle\!\langle Q \rangle\!\rangle \Leftrightarrow \{P\}\ r\ \{Q\}$$

SIL vs HL

c_{42} :

```
x := nondet();
if even (x) {
        if  odd(y) { z := 42; }
   }
```

Safe $z \neq 42$
E.g., x:=1/(42- z)

Given a specification of the possible errors
$Q \triangleq \{ z = 42 \}$

With SIL one can prove

$\langle\langle \text{odd(y)} \rangle\rangle \; c_{42} \; \langle\langle z=42 \rangle\rangle$

Expressing a precondition that leads to error states

With HL one can prove

$\{ z=42 \} \; c_{42} \; \{ z=42 \}$

Sufficient incorrectness logic (SIL)

OOPSLA 2025

Revealing Sources of (Memory) Errors via Backward Analysis

FLAVIO ASCARI, University of Pisa, Italy
ROBERTO BRUNI, University of Pisa, Italy
ROBERTA GORI, University of Pisa, Italy
FRANCESCO LOGOZZO, Meta Platforms, USA

Sound over-approximation methods are effective for proving the absence of errors, but inevitably produce false alarms that can hamper programmers. In contrast, under-approximation methods focus on bug detection and are free from false alarms. In this work, we present two novel proof systems designed to locate the source of errors via backward under-approximation, namely Sufficient Incorrectness Logic (SIL) and its specialization for handling memory errors, called Separation SIL. The SIL proof system is minimal, sound and complete for Lisbon triples, enabling a detailed comparison of triple-based program logics across various dimensions, including negation, approximation, execution order, and analysis objectives. More importantly, SIL lays the foundation for our main technical contribution, by distilling the inference rules of Separation SIL, a sound and (relatively) complete proof system for automated backward reasoning in programs involving pointers and dynamic memory allocation. The completeness result for Separation SIL relies on a careful crafting of both the assertion language and the rules for atomic commands.

CCS Concepts: • **Theory of computation** → **Logic and verification**; *Proof theory*; *Hoare logic*; **Separation logic**; *Programming logic*.

Additional Key Words and Phrases: Sufficient Incorrectness Logic, Incorrectness Logic, Outcome Logic

ACM Reference Format:
Flavio Ascari, Roberto Bruni, Roberta Gori, and Francesco Logozzo. 2025. Revealing Sources of (Memory) Errors via Backward Analysis. *Proc. ACM Program. Lang.* 9, OOPSLA1, Article 127 (April 2025), 28 pages.
https://doi.org/10.1145/3720486

1 Introduction

Formal methods aim to automate the improvement of software reliability and security. Notable success stories are, e.g., the Astrée static analyzer [Blanchet et al. 2003], the SLAM model checker [Ball and Rajamani 2001], the certified C compiler CompCert [Leroy 2009], VCC for safety properties verification [Cohen et al. 2009], and the Frama-C platform for the integration of many C code analyses [Baudin et al. 2021]. Despite that, effective program correctness methods struggle to reach mainstream adoption, mostly because they exploit over-approximation to handle decidability issues and false positives are seen as a distraction by expert programmers. Being free from false positives is possibly the reason why *under-approximation* approaches for bug-finding, such as testing and bounded model checking, are preferred in industrial applications. Incorrectness Logic (IL) [O'Hearn 2020] is a new program logic for bug-finding: *any error state found in the post can be produced by some input states that satisfy the pre.* However, IL triples are not able to characterize precisely *the input states that are responsible for a given error.* This is possibly rooted in the *forward* flavor of the under-approximation, which follows the ordinary direction of code execution.

Authors' Contact Information: Flavio Ascari, University of Pisa, Pisa, Italy, flavio.ascari@phd.unipi.it; Roberto Bruni, University of Pisa, Pisa, Italy, bruni@di.unipi.it; Roberta Gori, University of Pisa, Pisa, Italy, roberta.gori@unipi.it; Francesco Logozzo, Meta Platforms, Seattle, USA, logozzo@meta.com.

ACM 2475-1421/2025/4-ART127
https://doi.org/10.1145/3720486

"SIL can characterise the source of errors"

Sufficient Incorrectness Logic (SIL)

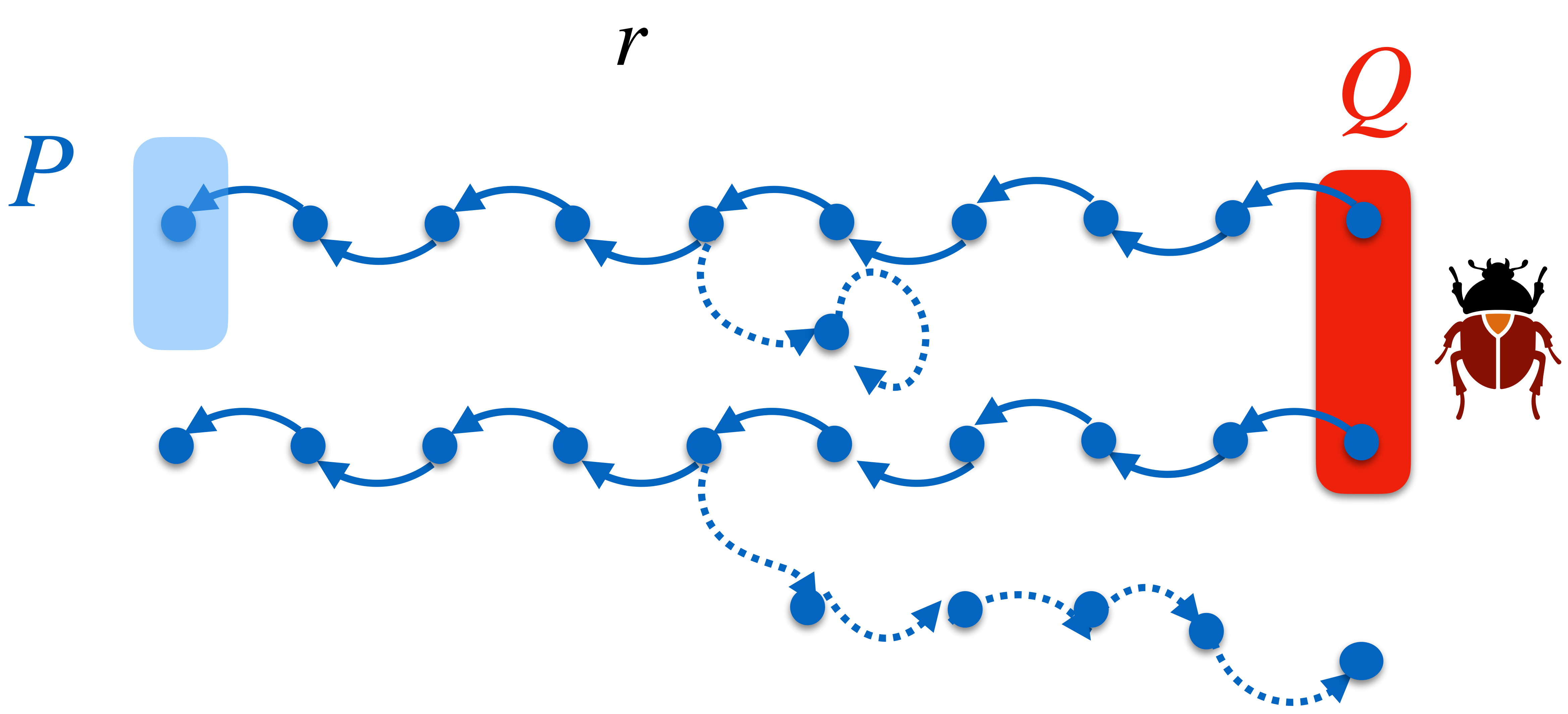
Given Q a specification of the possible errors

$\langle\!\langle P \rangle\!\rangle \; c \; \langle\!\langle Q \rangle\!\rangle$ is valid when $[\![\overleftarrow{r}]\!] Q \supseteq P$

It is an under-approximation!

means

$\forall \sigma \in P \, \exists \sigma' \in Q \, . \, \sigma' \in [\![r]\!] \sigma$



An under-approximating logic designed to devise the initial states leading to errors

Manifest errors

An error is manifest if it occurs independently of the context and is therefore particularly interesting to point out to programmers

Manifest errors cannot be characterised with IL

But they can be easily characterised with SIL

$\langle\!\langle\, true \,\rangle\!\rangle \; r \; \langle\!\langle\, Q \,\rangle\!\rangle$ is valid $\Leftrightarrow$ Q is a manifest error

SIL Rules

The proof system favours backward analysis starting from the (error) postconditions

Hoare's axiom for assignment

$$\frac{}{\langle\!\langle Q[a/x] \rangle\!\rangle\, x := a\, \langle\!\langle Q \rangle\!\rangle} \langle\!\langle atom - a \rangle\!\rangle$$

$$\langle\!\langle y > 0 \rangle\!\rangle \quad x := y - 1 \quad \langle\!\langle x \geq 0 \rangle\!\rangle$$

$$\langle\!\langle y \neq 43 \rangle\!\rangle \quad x := y - 1 \quad \langle\!\langle x \neq 42 \rangle\!\rangle$$

SIL Rules

The proof system favours backward analysis starting from the (error) postconditions

$$\frac{}{\langle\langle Q \cap b \rangle\rangle\ b?\ \langle\langle Q \rangle\rangle}\ \langle\langle atom - g \rangle\rangle$$

$$\langle\langle \varnothing \rangle\rangle \quad (x > 0)? \qquad \langle\langle x = -42 \rangle\rangle$$

$$\langle\langle x = 42 \rangle\rangle \quad (x > 0)? \qquad \langle\langle x = 42 \rangle\rangle$$

SIL Rules

The proof system favours backward analysis starting from the (error) postconditions

Same conditions for both branches

$$\frac{\langle\!\langle P_1 \rangle\!\rangle r_1 \langle\!\langle Q \rangle\!\rangle \qquad \langle\!\langle P_2 \rangle\!\rangle r_2 \langle\!\langle Q \rangle\!\rangle}{\langle\!\langle P_1 \cup P_2 \rangle\!\rangle\ r_1 + r_2\ \langle\!\langle Q \rangle\!\rangle} \langle\!\langle choice \rangle\!\rangle$$

$$\langle\!\langle y = 43 \vee y = 42 \rangle\!\rangle \qquad (x := y - 1) + (x := y) \qquad \langle\!\langle x = 42 \rangle\!\rangle$$

$$\langle\!\langle \mathit{true} \rangle\!\rangle = \langle\!\langle y \neq 43 \vee y \neq 42 \rangle\!\rangle \qquad (x := y - 1) + (x := y) \qquad \langle\!\langle x \neq 42 \rangle\!\rangle$$

$$\langle\!\langle y \neq 43 \rangle\!\rangle \quad (x := y - 1) + (x := 42) \qquad \langle\!\langle x \neq 42 \rangle\!\rangle$$

SIL Rules

The proof system favours backward analysis starting from the (error) postconditions

Backward iteration starting from final state Q_0

$$\frac{\forall n \geq 0. \langle\!\langle Q_{n+1} \rangle\!\rangle \; r \; \langle\!\langle Q_n \rangle\!\rangle}{\langle\!\langle \bigcup_{n \geq 0} Q_n \rangle\!\rangle \; r^* \; \langle\!\langle Q_0 \rangle\!\rangle} \; \langle\!\langle iter \rangle\!\rangle$$

$$\langle\!\langle x \leq 42 \rangle\!\rangle = \langle\!\langle \ldots \vee x = 41 \vee x = 42 \rangle\!\rangle \quad (x := x + 1)^* \quad \langle\!\langle x = 42 \rangle\!\rangle$$

SIL Rules

The proof system favours backward analysis starting from the (error) postconditions

SIL can drop disjunction going backward:

$$\frac{}{\langle\!\langle \emptyset \rangle\!\rangle \; r \; \langle\!\langle Q \rangle\!\rangle} \quad \langle\!\langle empty \rangle\!\rangle \qquad\qquad \frac{\langle\!\langle P \cup P' \rangle\!\rangle \; r \; \langle\!\langle Q \rangle\!\rangle}{\langle\!\langle P \rangle\!\rangle \; r \; \langle\!\langle Q \rangle\!\rangle} \quad \langle\!\langle cons' \rangle\!\rangle$$

$$\langle\!\langle x = 41 \lor x = 42 \rangle\!\rangle \quad (x := x + 1)^* \quad \langle\!\langle x = 42 \rangle\!\rangle$$

Validity, soundness and completeness

A proof system for SIL

Core rules

$$\frac{}{\langle\!\langle [\![\overleftarrow{\mathsf{c}}]\!] Q \rangle\!\rangle \ \mathsf{c} \ \langle\!\langle Q \rangle\!\rangle} \ \langle\!\langle \text{atom} \rangle\!\rangle \qquad \frac{P \subseteq P' \quad \langle\!\langle P' \rangle\!\rangle \ \mathsf{r} \ \langle\!\langle Q' \rangle\!\rangle \quad Q' \subseteq Q}{\langle\!\langle P \rangle\!\rangle \ \mathsf{r} \ \langle\!\langle Q \rangle\!\rangle} \ \langle\!\langle \text{cons} \rangle\!\rangle$$

$$\frac{\langle\!\langle P_1 \rangle\!\rangle \ \mathsf{r}_1 \ \langle\!\langle Q \rangle\!\rangle \quad \langle\!\langle P_2 \rangle\!\rangle \ \mathsf{r}_2 \ \langle\!\langle Q \rangle\!\rangle}{\langle\!\langle P_1 \cup P_2 \rangle\!\rangle \ \mathsf{r}_1 + \mathsf{r}_2 \ \langle\!\langle Q \rangle\!\rangle} \ \langle\!\langle \text{choice} \rangle\!\rangle \qquad \frac{\langle\!\langle P \rangle\!\rangle \ \mathsf{r}_1 \ \langle\!\langle R \rangle\!\rangle \quad \langle\!\langle R \rangle\!\rangle \ \mathsf{r}_2 \ \langle\!\langle Q \rangle\!\rangle}{\langle\!\langle P \rangle\!\rangle \ \mathsf{r}_1 ; \mathsf{r}_2 \ \langle\!\langle Q \rangle\!\rangle} \ \langle\!\langle \text{seq} \rangle\!\rangle$$

$$\frac{\forall n \geq 0 \,.\, \langle\!\langle Q_{n+1} \rangle\!\rangle \ \mathsf{r} \ \langle\!\langle Q_n \rangle\!\rangle}{\langle\!\langle \bigcup_{n \geq 0} Q_n \rangle\!\rangle \ \mathsf{r}^* \ \langle\!\langle Q_0 \rangle\!\rangle} \ \langle\!\langle \text{iter} \rangle\!\rangle$$

Additional rules

$$\frac{}{\langle\!\langle \emptyset \rangle\!\rangle \ \mathsf{r} \ \langle\!\langle Q \rangle\!\rangle} \ \langle\!\langle \text{empty} \rangle\!\rangle \qquad \frac{\langle\!\langle P_1 \rangle\!\rangle \ \mathsf{r} \ \langle\!\langle Q_1 \rangle\!\rangle \quad \langle\!\langle P_2 \rangle\!\rangle \ \mathsf{r} \ \langle\!\langle Q_2 \rangle\!\rangle}{\langle\!\langle P_1 \cup P_2 \rangle\!\rangle \ \mathsf{r} \ \langle\!\langle Q_1 \cup Q_2 \rangle\!\rangle} \ \langle\!\langle \text{disj} \rangle\!\rangle$$

$$\frac{}{\langle\!\langle Q \rangle\!\rangle \ \mathsf{r}^* \ \langle\!\langle Q \rangle\!\rangle} \ \langle\!\langle \text{iter0} \rangle\!\rangle \qquad \frac{\langle\!\langle P \rangle\!\rangle \ \mathsf{r}^* ; \mathsf{r} \ \langle\!\langle Q \rangle\!\rangle}{\langle\!\langle P \rangle\!\rangle \ \mathsf{r}^* \ \langle\!\langle Q \rangle\!\rangle} \ \langle\!\langle \text{unroll} \rangle\!\rangle$$

$$\frac{\langle\!\langle P \rangle\!\rangle \ \mathsf{r}^* ; \mathsf{r} \ \langle\!\langle Q_1 \rangle\!\rangle}{\langle\!\langle P \cup Q_2 \rangle\!\rangle \ \mathsf{r}^* \ \langle\!\langle Q_1 \cup Q_2 \rangle\!\rangle} \ \langle\!\langle \text{unroll-split} \rangle\!\rangle$$

Soundness and completeness

SIL validity of a triple : $[\![\overleftarrow{r}]\!]Q \supseteq P$

Th. [*Soundness*]
All provable triples (including additional rules) are valid

Th. [*Completeness*]
All valid triples are provable (using the core rules)

Questions

Question 1

Which SIL triples are valid for any r and P ?

$\langle\!\langle \mathsf{false} \rangle\!\rangle \; r \; \langle\!\langle P \rangle\!\rangle$ ✓

$\langle\!\langle \mathsf{true} \rangle\!\rangle \; r \; \langle\!\langle \mathsf{true} \rangle\!\rangle$ ✗

$\langle\!\langle P \rangle\!\rangle \; r^* \; \langle\!\langle P \vee x = 0 \rangle\!\rangle$ ✓

$\langle\!\langle wlp(r, P) \rangle\!\rangle \; r \; \langle\!\langle P \rangle\!\rangle$ ✗

Question 2

Prove that rule [conj] is unsound for SIL

$$\frac{\langle\langle P_1 \rangle\rangle\ r\ \langle\langle Q_1 \rangle\rangle \quad \langle\langle P_2 \rangle\rangle\ r\ \langle\langle Q_2 \rangle\rangle}{\langle\langle P_1 \wedge P_2 \rangle\rangle\ r\ \langle\langle Q_1 \wedge Q_2 \rangle\rangle}\ \text{[conj]}$$

Consider $\langle\langle x = 0 \rangle\rangle\ x := \text{nondet}()\ \langle\langle x = 0 \rangle\rangle$
and $\langle\langle x = 0 \rangle\rangle\ x := \text{nondet}()\ \langle\langle x = 1 \rangle\rangle$
By rule [conj] we could derive $\langle\langle x = 0 \rangle\rangle\ x := 1\ \langle\langle \text{false} \rangle\rangle$
which is not sound!

Question 3

Prove or disprove the validity of the following axiom in SIL

$$\frac{}{\langle\!\langle P \rangle\!\rangle \ (b)? \ \langle\!\langle P \wedge b \rangle\!\rangle}$$

Consider the following triple $\langle\!\langle x \geq 0 \rangle\!\rangle \ (x > 1)? \ \langle\!\langle x \geq 2 \rangle\!\rangle$

which is not a valid triple since from x=0 we cannot reach x≥2

Exercise

```
// function r
x := nondet();
if (x=1) {
  if (y≤100) {
          C }}
```

```
// function C-McCarthy 91 function
while (x>0) {
    if (y>100) {
          y :=y-10; x := x-1  }
    else
          y := y+11; x := x+1 } }
```

	SIL	IL	HL	NC
$[\mathrm{true}]\ \mathrm{r}\ [y = 91 \wedge x \neq 1]$	✗	✓	✗	✓
$\langle\!\langle y \leq 100 \rangle\!\rangle\ \mathrm{r}\ \langle\!\langle y = 91 \wedge x \neq 1 \rangle\!\rangle$	✓	✓	✗	✓
$\langle\!\langle y \leq 100 \rangle\!\rangle\ \mathrm{r}\ \langle\!\langle y = 91 \rangle\!\rangle$	✓	✗	✗	✓
$\langle\!\langle y < 91 \rangle\!\rangle\ \mathrm{r}\ \langle\!\langle y = 91 \rangle\!\rangle$	✓	✗	✗	✗