Outcome Logic: A Unifying Foundation for Correctness and Incorrectness Reasoning

Noam Zilberstein
Cornell University

Derek Dreyer
MPI-SWS

Alexandra Silva
Cornell University

“Program correctness and incorrectness are two sides of the same coin.”

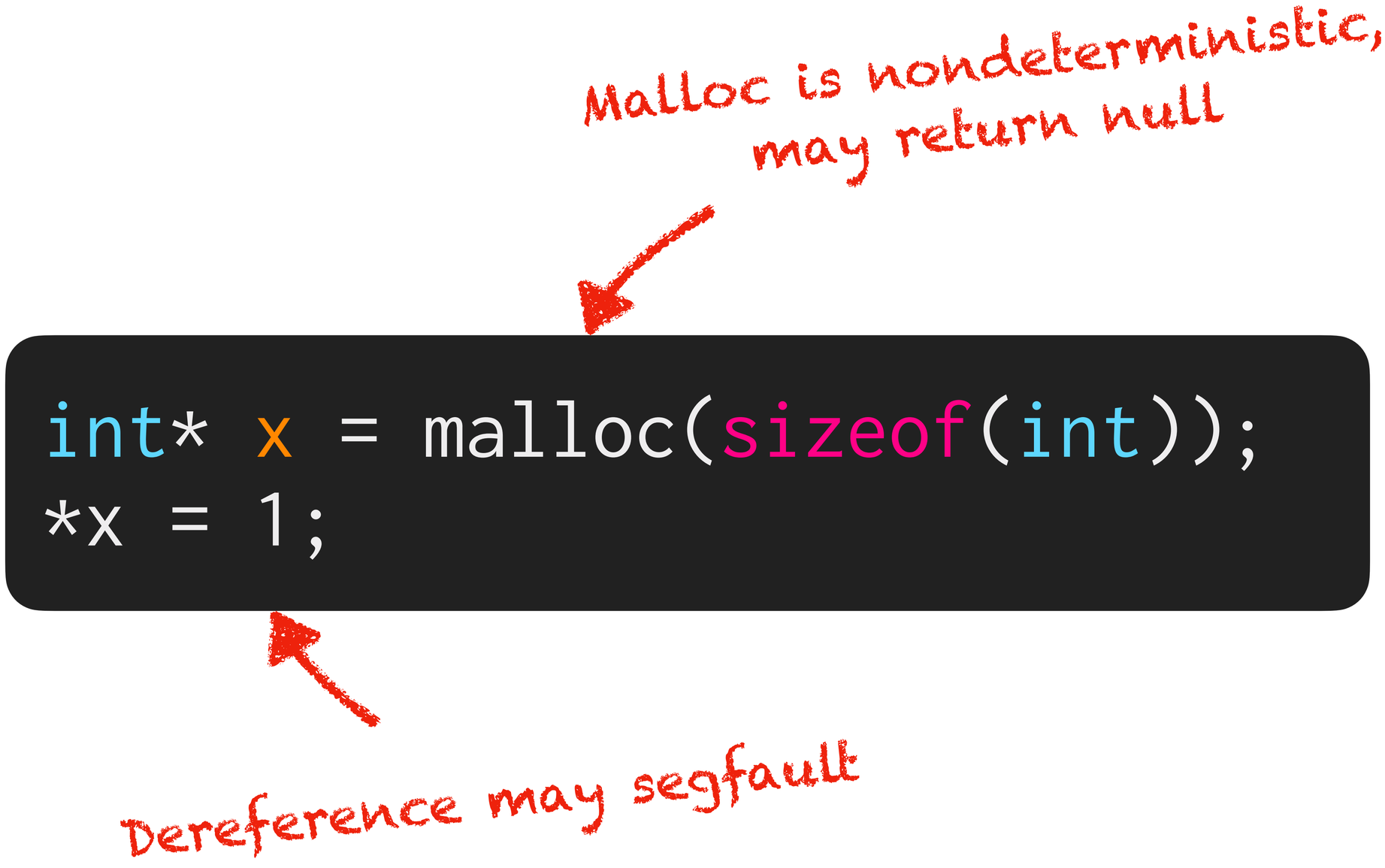
— Peter O’Hearn [2020]

Can a single program logic handle correctness and incorrectness?

What is Incorrectness?

- **True positives**
  Reported bugs should actually be possible

- **Under-approximation**
  Find bugs without inspecting the entire program

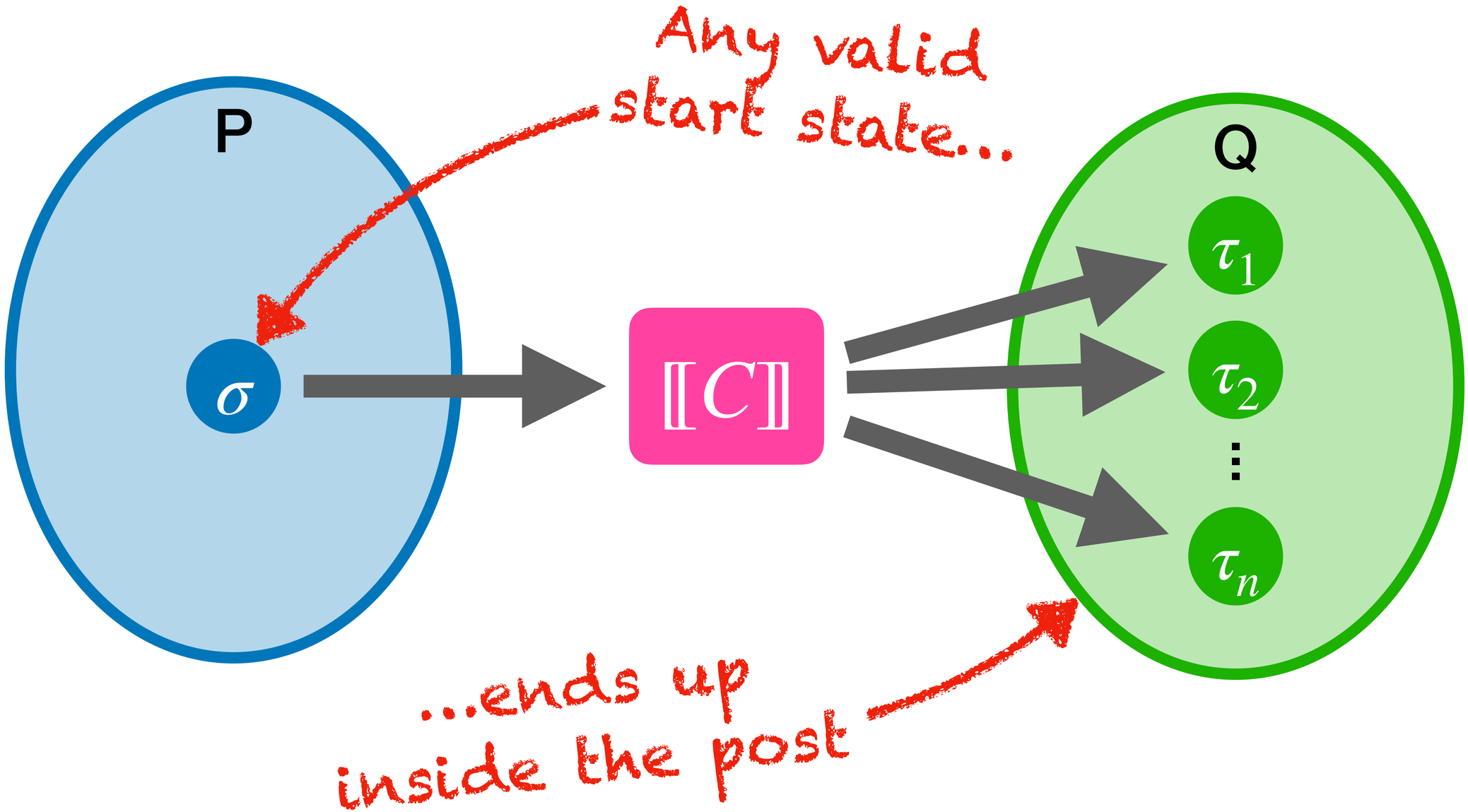

Malloc is nondeterministic,
may return null
int* x = malloc(sizeof(int));
*x = 1;
Dereference may segfault

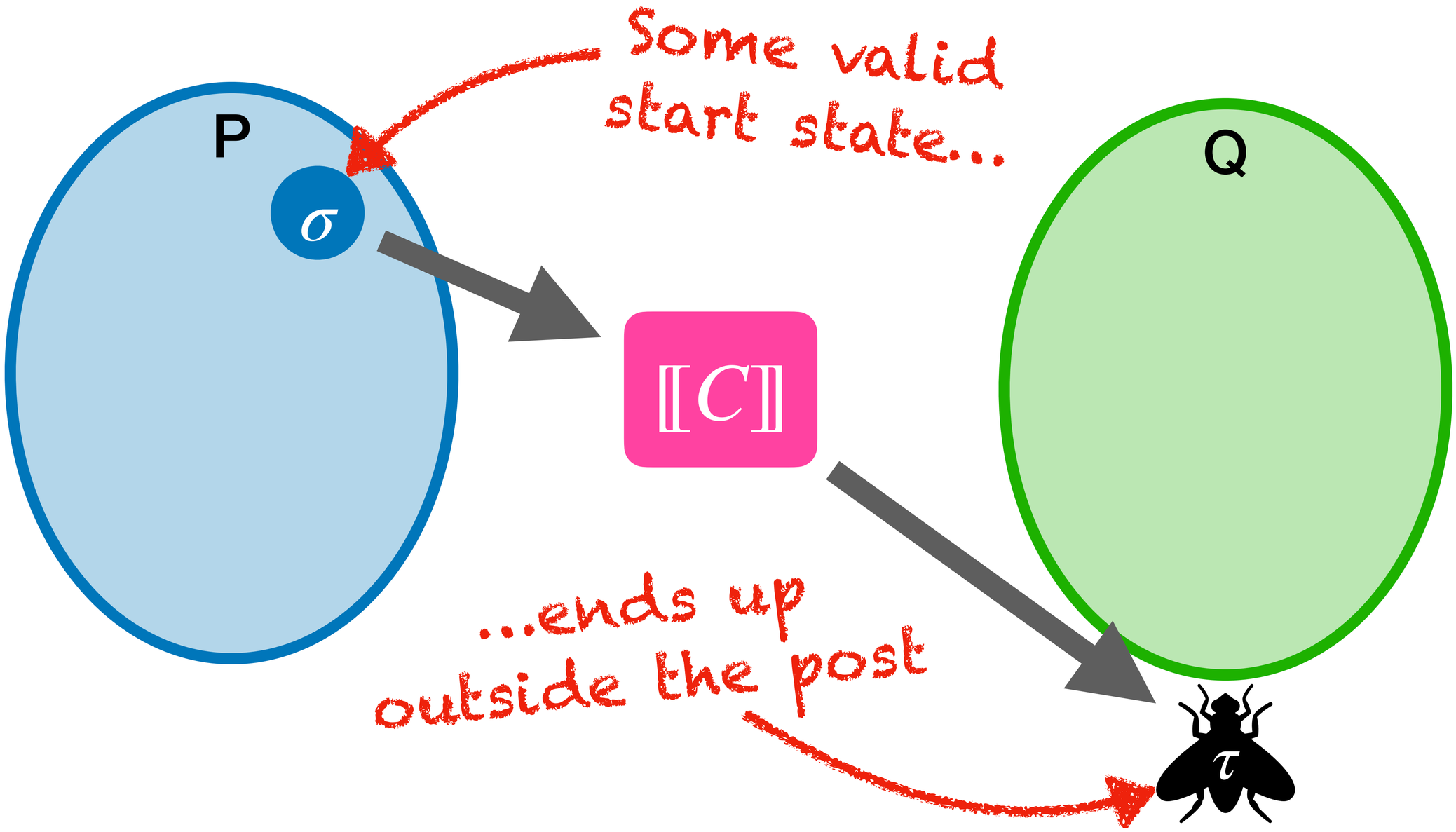
Incorrectness + Hoare Logic

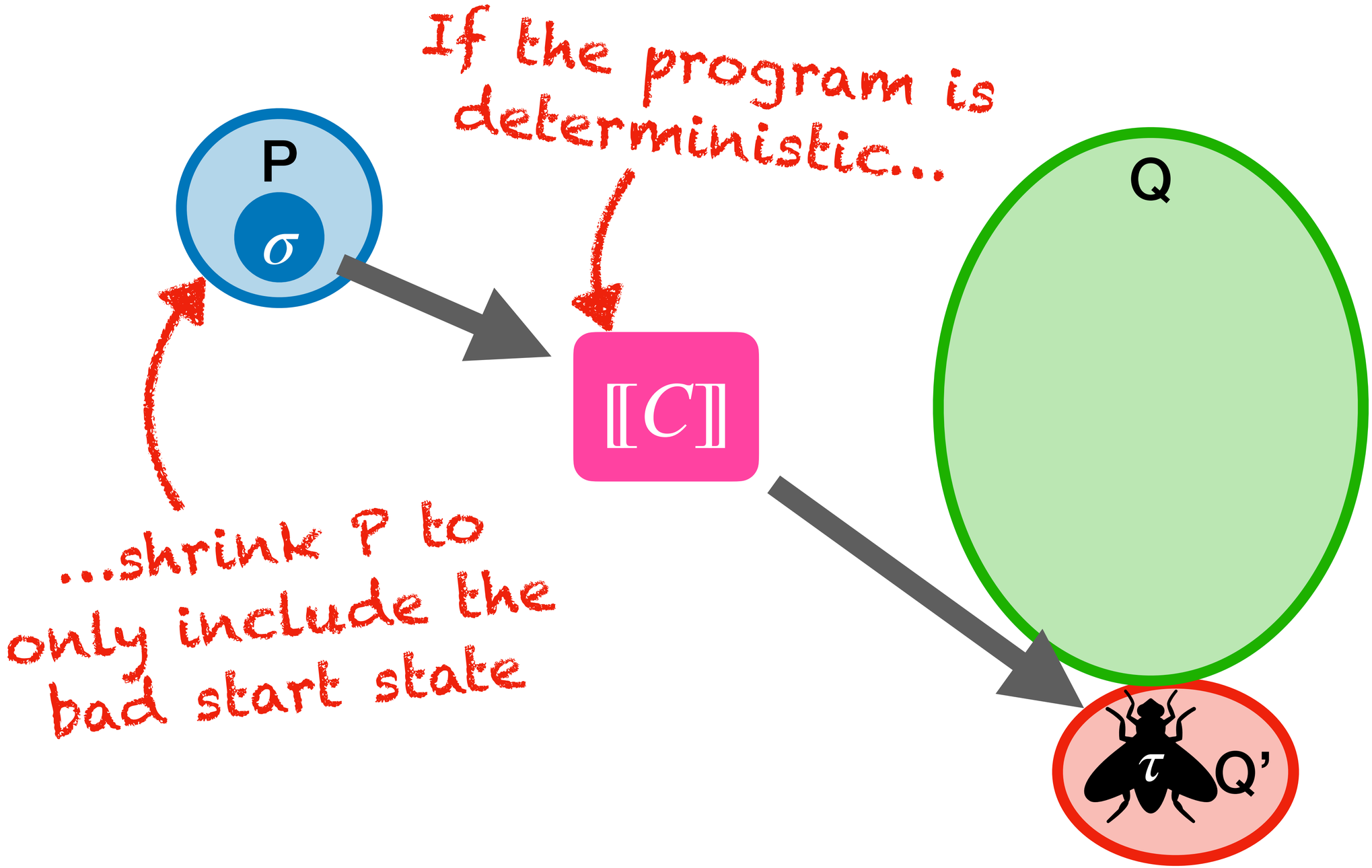
```
{true}
   int* x = malloc(sizeof(int));
   *x = 1;
{(ok : x ↦ 1) ∨ (er : x = null)}
```

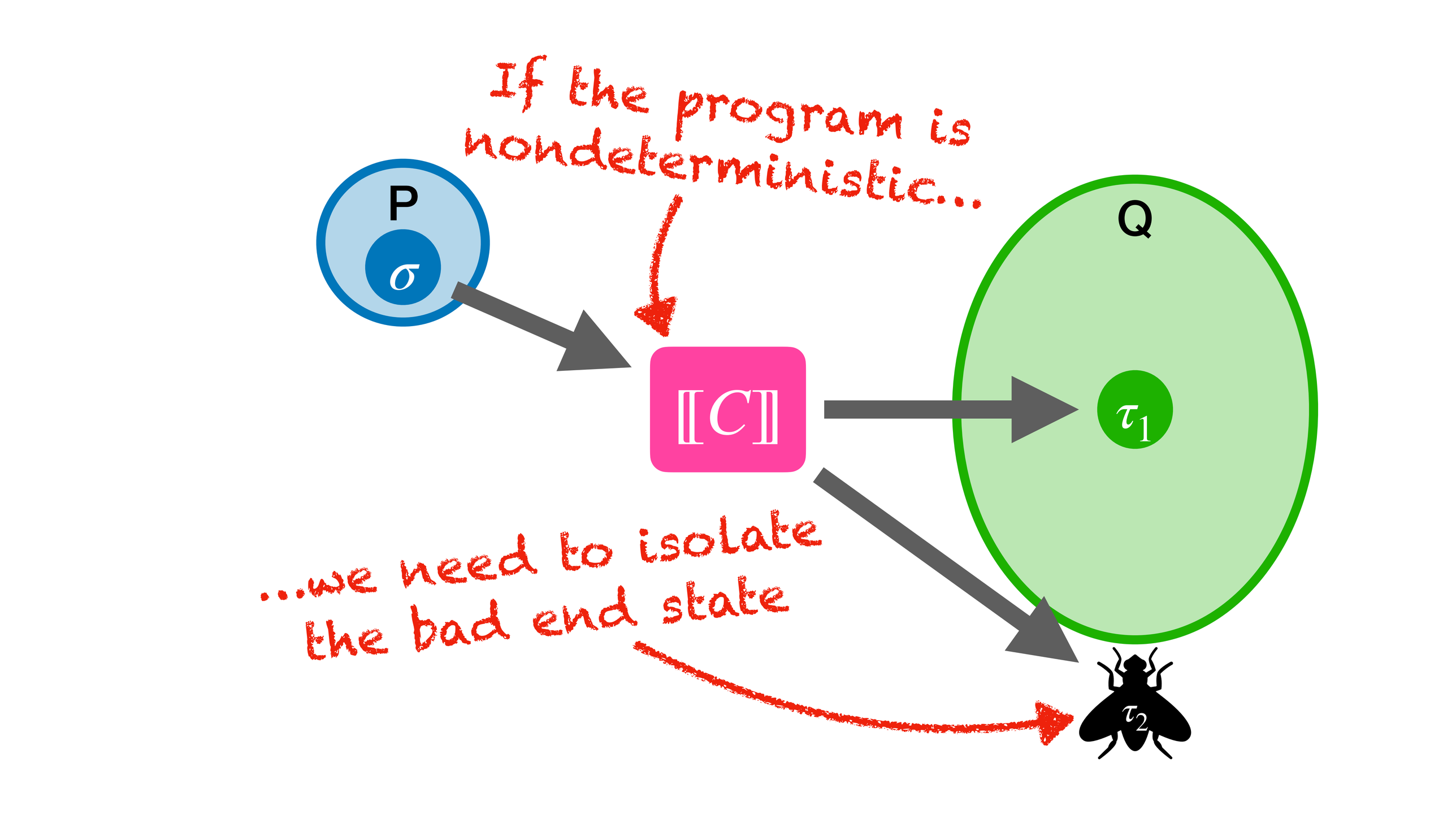
Does this spec characterize the bug?

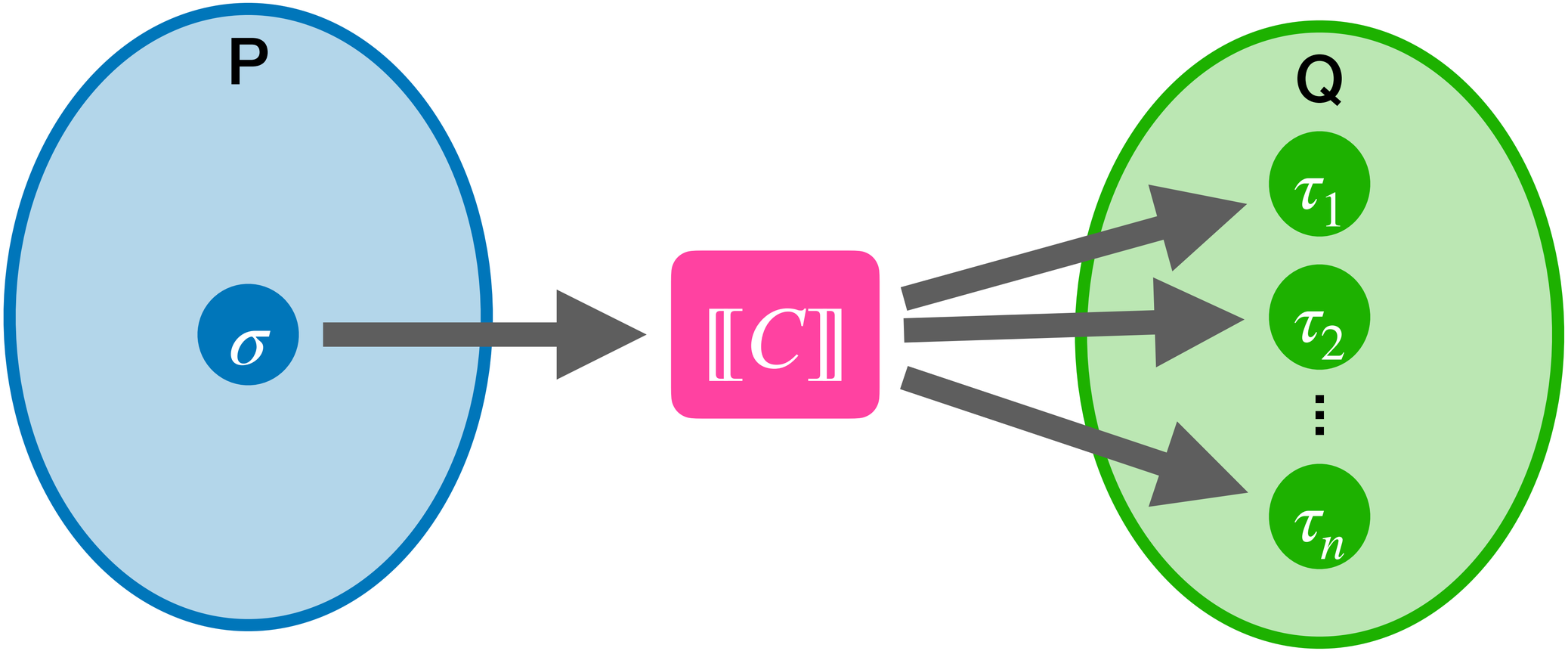
No! We don't know if the error is reachable

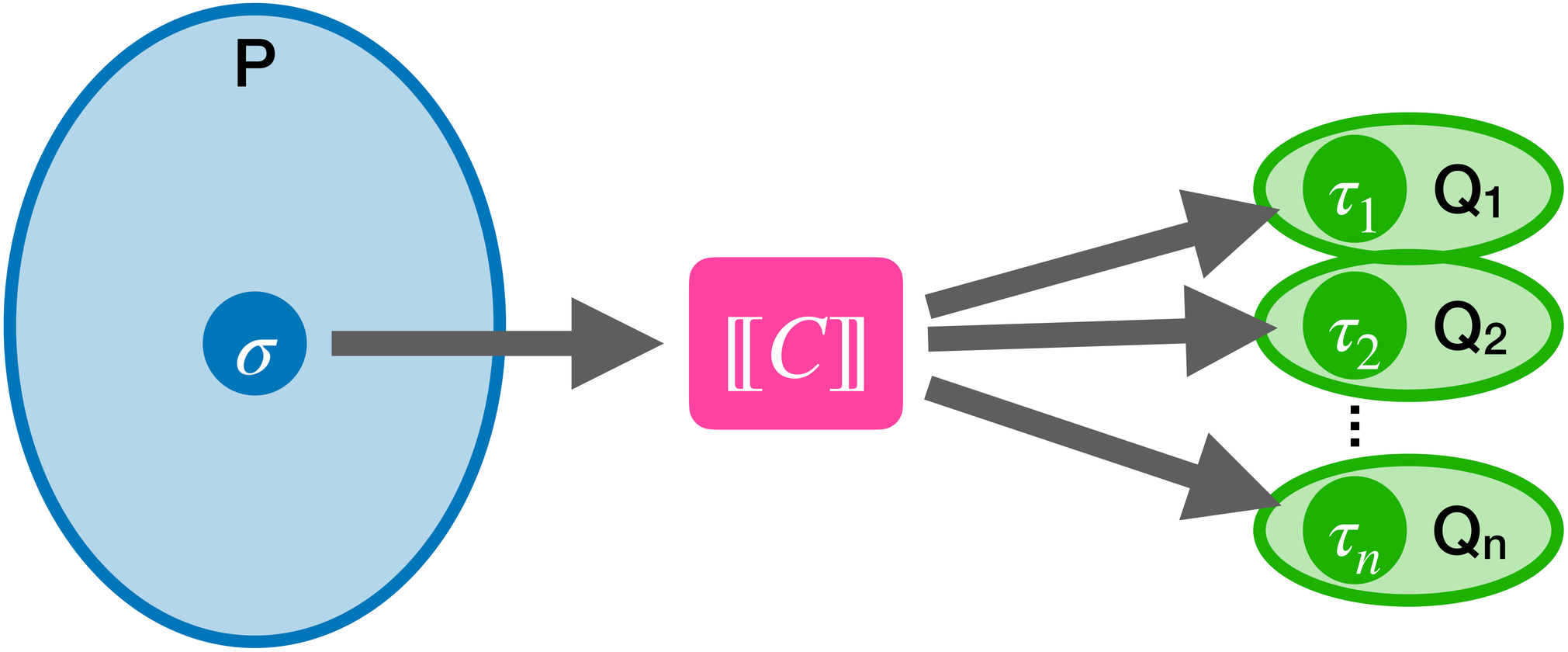

Any valid start state...
P
Q
σ
$[\![C]\!]$
τ_1
τ_2
τ_n
...ends up inside the post


Some valid
start state...
P
σ
Q
$[\![C]\!]$
...ends up
outside the post
τ


If the program is deterministic...
P
σ
Q
⟦C⟧
...shrink P to only include the bad start state
τ
Q'


If the program is nondeterministic...
P
σ
$[\![C]\!]$
Q
τ_1
τ_2
...we need to isolate the bad end state


P
σ
$[\![C]\!]$
Q
τ_1
τ_2
τ_n


P
σ
$[\![C]\!]$
τ_1
Q1
τ_2
Q2
τ_n
Qn

Outcome Logic

$$\vDash \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \text{iff} \quad \forall S\,.\, S \vDash \varphi \implies [\![C]\!](S) \vDash \psi$$

Outcome Assertions

$$
\begin{aligned}
\varphi ::= \; & \top \\
\mid \; & \bot \\
\mid \; & \varphi \wedge \psi \\
\mid \; & \varphi \vee \psi \\
& \vdots \\
\mid \; & \varphi \oplus \psi \\
\mid \; & P
\end{aligned}
$$

$S \vDash \varphi \oplus \psi$ iff $\exists S_1, S_2 \,.\, S = S_1 \cup S_2$ and $S_1 \vDash \varphi$ and $S_2 \vDash \psi$

$S \vDash P$ iff $S \neq \varnothing$ and $S \subseteq P$

Outcome Logic and Incorrectness

$\langle \mathsf{ok} : \mathsf{true} \rangle$

```
  int* x = malloc(sizeof(int));
  *x = 1;
```

$\langle (\mathsf{ok} : x \mapsto 1) \oplus (\mathsf{er} : x = \mathsf{null}) \rangle$

But this outcome is irrelevant

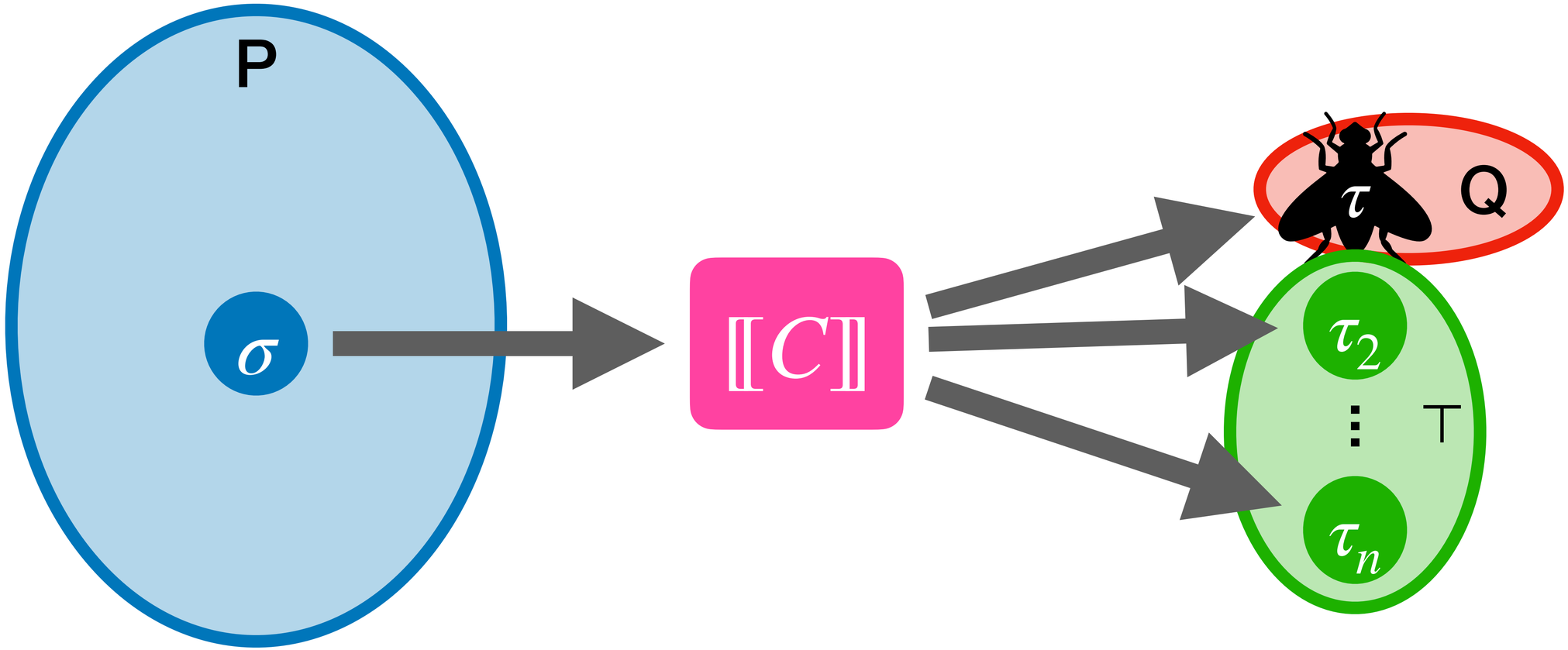
This outcome must be reachable

Dropping Outcomes

```
⟨ok : true⟩
  int* x = malloc(sizeof(int));
  *x = 1;
⟨(er : x = null) ⊕ T⟩
```

Still reachable

But we dropped the extra info


P
σ
$[\![C]\!]$
τ
Q
τ_2
T
τ_n

Outcome Logic and Incorrectness

$$\not\models \langle \varphi \rangle\ C\ \langle \psi \rangle$$

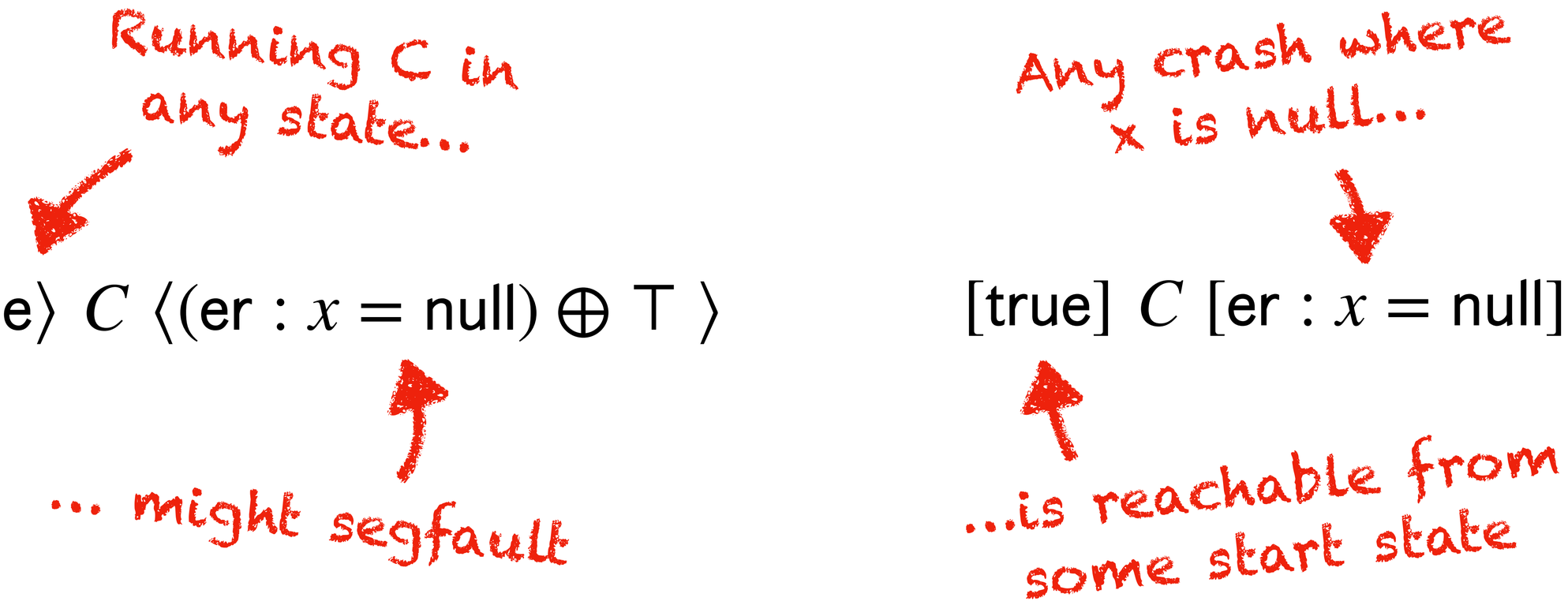
Any "correctness" specification...

iff

...can be disproven in Outcome Logic

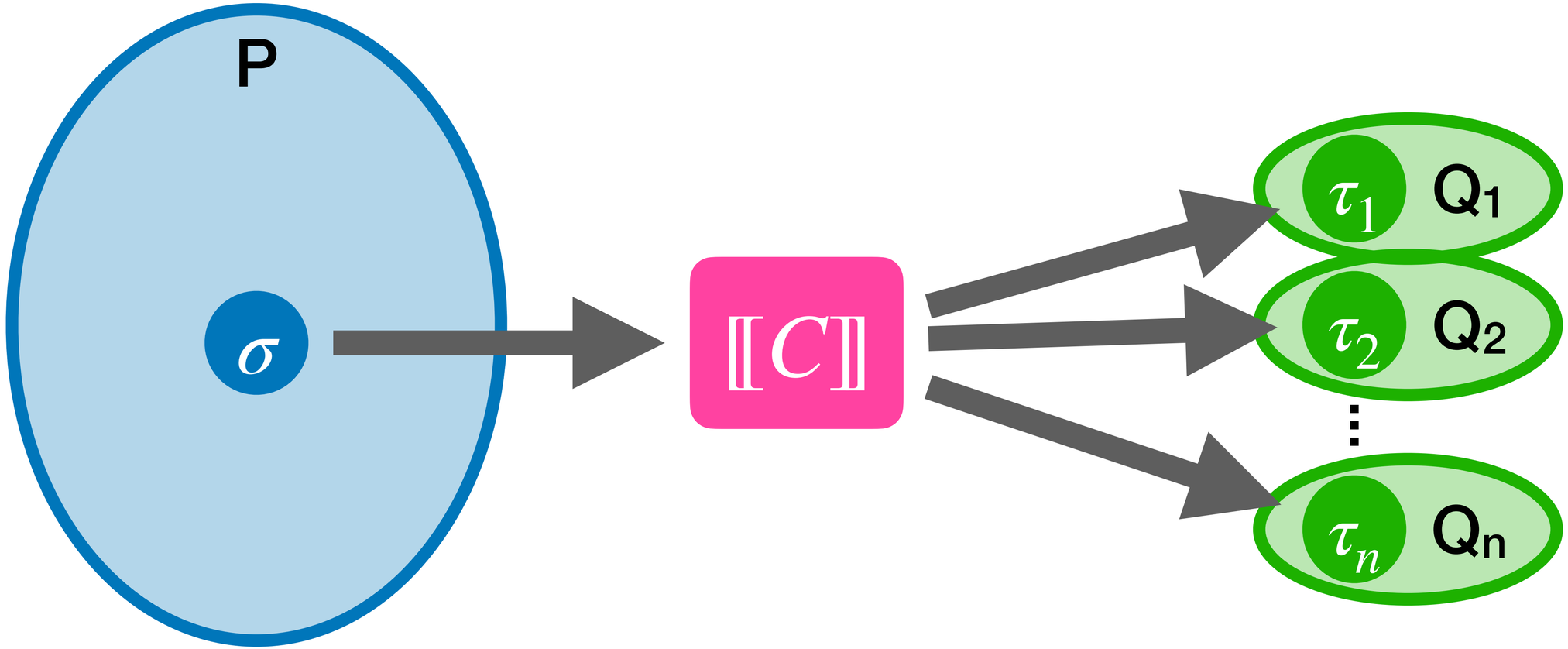
$$\exists \varphi' \Rightarrow \varphi\,.\quad \mathsf{sat}(\varphi') \quad \text{and} \quad \models \langle \varphi' \rangle\ C\ \langle \neg\psi \rangle$$

Incorrectness Logic [O'Hearn 2019]

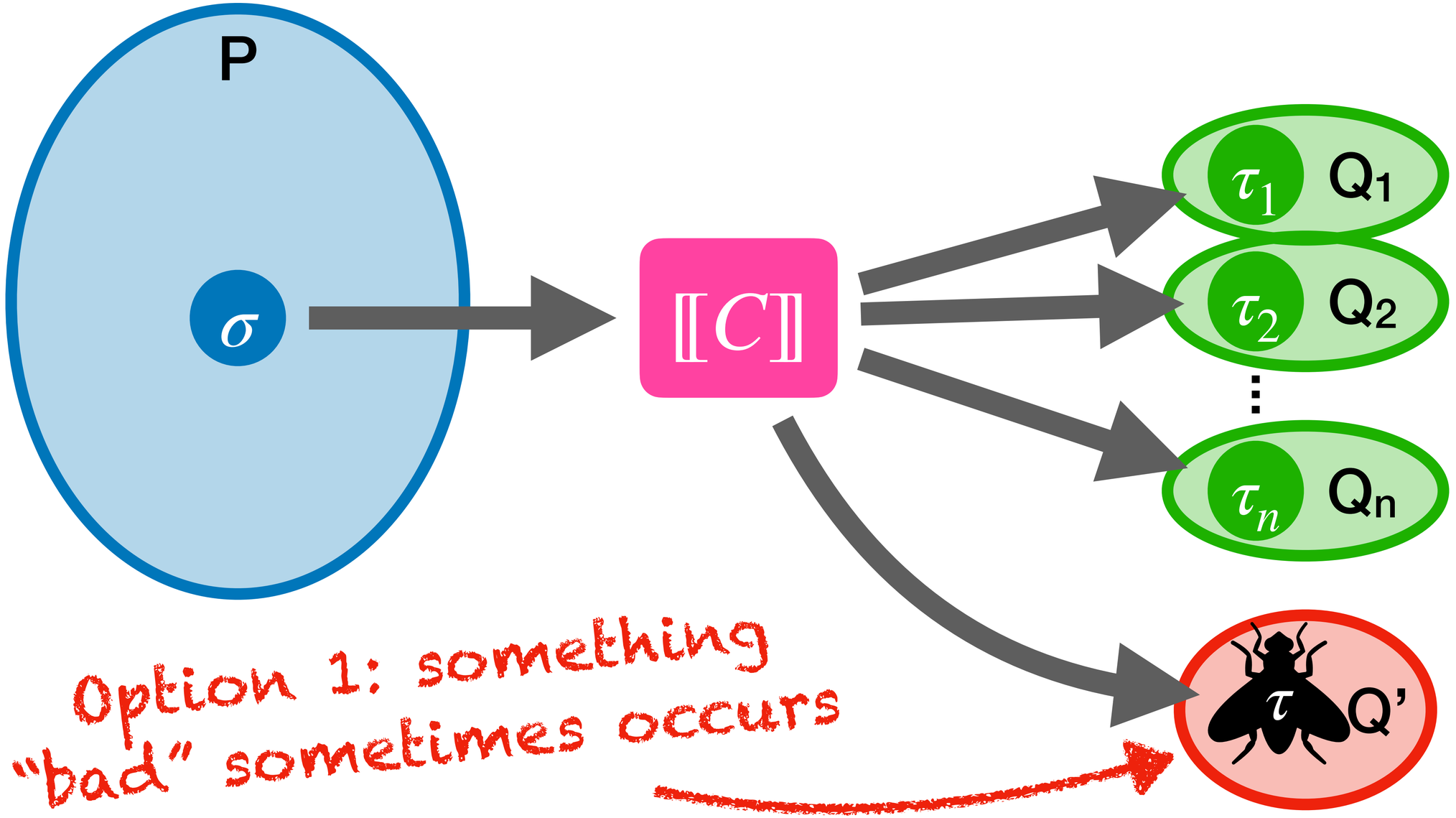


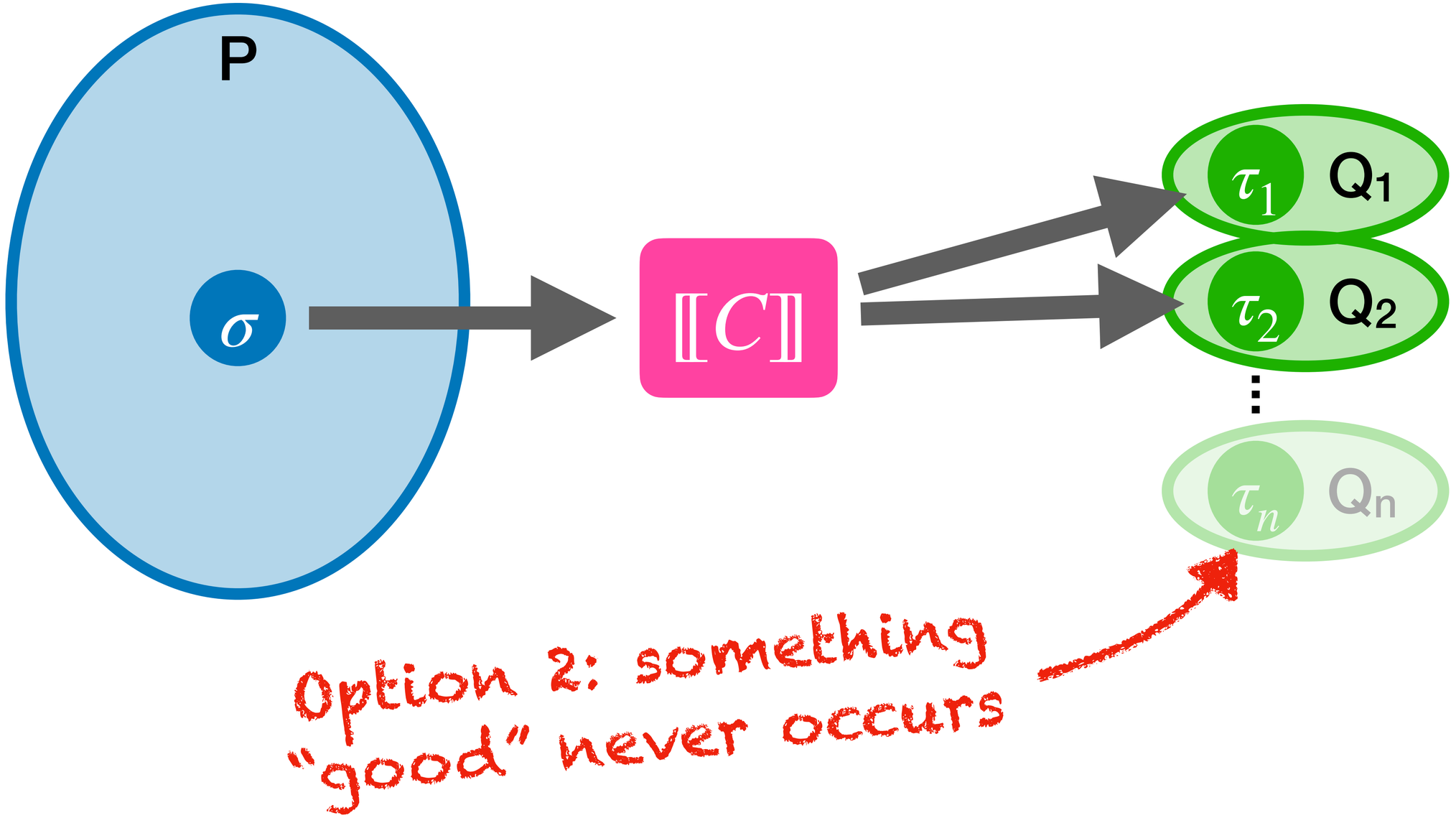
Manifest errors:

Which start states force the bug to appear? [Le et al. 2022]



How can this spec be false?


P
σ
$[\![C]\!]$
τ_1 Q1
τ_2 Q2
τ_n Qn
τ Q'
Option 1: something
"bad" sometimes occurs


P
σ
$[\![C]\!]$
τ_1
Q_1
τ_2
Q_2
τ_n
Q_n
Option 2: something
"good" never occurs

Can a single program logic handle correctness and incorrectness...

...with computational effects?

Outcome Logic

$$\vDash \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \text{iff} \quad \forall m \in M\Sigma\,.\ m \vDash \varphi \implies [\![C]\!](m) \vDash \psi$$

M is a monad
(with some extra properties)

Probabilistic Programs

Network is unreliable, may drop message

$\langle \mathsf{true} \rangle$

```
int x = ping(192.0.2.1);
```

$\langle \Pr[x = \mathsf{ok}] = 99\% \ \oplus \ \Pr[x = \mathsf{er}] = 1\% \rangle$

Program succeeds 99% of the time

“Program correctness and incorrectness are two sides of the same coin.”

— Peter O’Hearn [2020]

*Program correctness and incorrectness are two **usages** of the same **program logic**.*

Conclusion

Can a single program logic handle correctness and incorrectness?

Incorrectness Reasoning

- True positives
- Under-approximation

Outcome Logic

- Semantics parametric on *monadic* representation of effects
- Any false triple can be *disproven*
- Outcome Logic can identify *more* types of bugs than IL
- *Manifest errors*: it's useful to know which start states force a bug