Is Peter Correct or Incorrect?

Patrick Cousot

Courant Institute, New York University
In POPL 2020, Peter O’Hearn introduced the nonconformist idea of an incorrectness logic.

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.
• In POPL 2020, Peter O’Hearn introduced the nonconformist idea of an incorrectness logic

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.

• Is it?
• And he moderately enjoyed other approaches to incorrectness
• Such as ``necessary preconditions”

The concept of necessary precondition [Cousot et al. 2013] is related. A necessary precondition for a program is a predicate which, whenever falsified, leads to divergence or an error, but never to successful termination.
Peter’s Incorrectness Logic

• And he moderately enjoyed other approaches to incorrectness
• Such as ``necessary preconditions’’

  The concept of necessary preconditon [Cousot et al. 2013] is related. A necessary precondition for a program is a predicate which, whenever falsified, leads to divergence or an error, but never to successful termination.

• But he doesn’t really like it!

  ... Finally, there are programs for which no non-trivial necessary pre-condition exists (e.g., skip + error()), but where perfectly fine presumptions exist for incorrectness logic.
Peter’s Incorrectness Logic

• And he moderately enjoyed other approaches to incorrectness

• Such as ``necessary preconditions”

  The concept of necessary preconditon [Cousot et al. 2013] is related. A necessary precondition for a program is a predicate which, whenever falsified, leads to divergence or an error, but never to successful termination.

• But he doesn’t really like it!

  … Finally, there are programs for which no non-trivial necessary pre-condition exists (e.g., skip + error()), but where perfectly fine presumptions exist for incorrectness logic.

• Should he?
Peter’s Incorrectness Logic

In summary, there is a rich variety of problems for both experimental and theoretical work to bring the foundations of reasoning about program incorrectness onto a par with the extensively developed foundations for correctness.
An A Parte on

Singularities of Logics
Emptiness versus Universality

- **Emptiness**: some programs satisfy no formula of the logic
- **Ex. 1**: a potentially nonterminating program satisfies no formula of the Manna-Pnueli total correctness logic
- **Ex. 2**: Peter’s example for “necessary preconditions”
Emptiness versus Universality

• **Emptiness**: some programs satisfy no formula of the logic
  • Ex. 1: a potentially nonterminating program satisfies no formula of the Manna-Pnueli total correctness logic
  • Ex. 2: Peter’s example for “necessary preconditions”

• **Universality**: some programs satisfy all formulas of the logic
  • Ex. 1: \( W = \text{while (true) skip} \) satisfies all Hoare triples \( \{P\} W \{Q\} \)
    - i.e. false is always false and true is always true
Emptiness versus Universality

• **Emptiness**: some programs satisfy no formula of the logic
  • Ex. 1: a potentially nonterminating satisfies no formula of the Manna-Pnueli total correctness logic
  • Ex. 2: Peter’s example for “necessary preconditions”

• **Universality**: some programs satisfy all formulae of the logic
  • Ex. 1: $W = \text{while (true) skip}$ satisfies all Hoare triples $\{P\} W \{Q\}$
  • Same in logic: false is never satisfied and true is always satisfied
Foundations of Reasoning on Logics
Method to design a program transformational logics

1. Define the natural relational semantics $\llbracket S \rrbracket_\perp$ of the programming language (in structural fixpoint form)
Method to design a program transformational logics

1. Define the natural relational semantics $\llbracket S \rrbracket_{\bot}$ of the programming language (in structural fixpoint form)

2. Define the theory of the logics as an abstraction $\alpha(\llbracket S \rrbracket_{\bot})$ of the collecting semantics $\{\llbracket S \rrbracket_{\bot}\}$ (strongest (hyper) property)

Theory of a logic = the subset of all true formulas
Method to design a program transformational logics

1. Define the natural relational semantics $\llbracket S \rrbracket_\bot$ of the programming language (in structural fixpoint form)

2. Define the theory of the logics as an abstraction $\alpha(\llbracket S \rrbracket_\bot)$ of the collecting semantics $\{\llbracket S \rrbracket_\bot\}$ (strongest (hyper) property)

3. Calculate the theory $\alpha(\llbracket S \rrbracket_\bot)$ in structural fixpoint form by fixpoint abstraction

Theory of a logic = the subset of all true formulas
Method to design a program transformational logics

1. Define the natural relational semantics $⟦S⟧_⊥$ of the programming language (in structural fixpoint form)
2. Define the theory of the logics as an abstraction $\alpha(\{⟦S⟧_⊥\})$ of the collecting semantics $\{⟦S⟧_⊥\}$ (strongest (hyper) property)
3. Calculate the theory $\alpha(\{⟦S⟧_⊥\})$ in structural fixpoint form by fixpoint abstraction
4. Calculate the proof system by fixpoint induction and Aczel correspondence between fixpoints and deductive systems

Theory of a logic = the subset of all true formulas
The Design of
Hoare Incorrectness Logic ($\overline{H\!L}$)
I) Relational semantics
I. Angelic relational semantics $[S]^e$

- **Syntax*:**
  \[
  S \in \mathcal{S} ::= x = A \mid \text{skip} \mid S;S \mid \text{if } (B) \ S \text{ else } S \mid \text{while } (B) \ S
  \]

- **States:** $\Sigma$

- **Angelic relational semantics:** $[S]^e \in \wp(\Sigma \times \Sigma)$
I. Angelic relational semantics $\llbracket S \rrbracket$ (in deductive form)

- Notations using judgements:
  - $\sigma \vdash S \Rightarrow^e \sigma'$ for $\langle \sigma, \sigma' \rangle \in [S]^e$
  - $\sigma \vdash \text{while}(B) S \Rightarrow^i \sigma'$ for $\sigma$ leads to $\sigma'$ after 0 or more iterations
1. Angelic relational semantics $\llbracket S \rrbracket$ (in deductive form)

- Notations using judgements:
  - $\sigma \vdash S \Rightarrow ^e \sigma'$ for $\langle \sigma, \sigma' \rangle \in \llbracket S \rrbracket^e$
  - $\sigma \vdash \text{while}(B) \ S \Rightarrow ^i \sigma'$ for $\sigma$ leads to $\sigma'$ after 0 or more iterations

- Semantics of the conditional iteration$^*$ $W = \text{while}(B) \ S$:
  - (a) $\sigma \vdash W \Rightarrow ^i \sigma$
  - (b) $\frac{B[\llbracket B \rrbracket \sigma, \quad \sigma \vdash S \Rightarrow ^e \sigma', \quad \sigma' \vdash W \Rightarrow ^i \sigma''}{\sigma \vdash W \Rightarrow ^i \sigma''}$ (2)
  - (a) $\sigma \vdash W \Rightarrow ^i \sigma'$, $B[\neg B] \sigma'$$
  - (b) $\frac{\sigma \vdash W \Rightarrow ^e \sigma'}{\sigma \vdash W \Rightarrow ^e \sigma'}$ (3)

---

O'Hearn Fest, POPL 2024, London
I. Angelic relational semantics \([S]\) (in fixpoint form)

- **Semantics of the conditional iteration**\(^* \) \( W = \text{while}(B) \ S : \)

  \[
  F^e(X) \triangleq \text{id} \cup ([B] \ ; \ [S]^e \ ; X), \quad X \in \wp(\Sigma \times \Sigma)
  \]

  \[
  [\text{while } (B) \ S]^e \triangleq \text{lfp} \subseteq F^e \ ; \ [\neg B]
  \]

- **Derived using Aczel correspondence between deductive systems and set-theoretic fixpoints (forthcoming)**
II) Abstraction of the semantics to the theory
Exact abstractions
Abstraction

- Hyper properties to properties abstraction:

\[
\begin{align*}
\langle \varphi(\varphi(\Sigma \times \Sigma)), \subseteq \rangle &\xrightarrow{\alpha_C} \langle \varphi(\Sigma \times \Sigma), \subseteq \rangle \\
\alpha_C(P) &\equiv \bigcup P \\
\gamma_C(S) &\equiv \varphi(S)
\end{align*}
\]
Abstraction

• Hyper properties to properties abstraction:

\[
\langle \varphi(\varphi(\Sigma \times \Sigma)) \rangle, \subseteq \xleftarrow{\gamma_C} \langle \varphi(\Sigma \times \Sigma) \rangle, \subseteq \xrightarrow{\alpha_C} \bigcup P \quad \gamma_C(S) \triangleq \varphi(S)
\]

• Post-image isomorphism:

\[
\langle \varphi(\Sigma \times \Sigma) \rangle, \subseteq \xleftarrow{\text{pre}_\text{post}} \langle \varphi(\Sigma) \rightarrow \varphi(\Sigma) \rangle, \subseteq \xrightarrow{\text{post}} \langle \varphi(\Sigma) \rangle, \subseteq \quad \text{post}(R) \triangleq \lambda P \cdot \{ \sigma' \mid \exists \sigma \in P \land \langle \sigma, \sigma' \rangle \in R \}
\]

\[
\text{pre}(R) \triangleq \lambda X \cdot \{ \sigma \mid \forall \sigma' \in Q . \langle \sigma, \sigma' \rangle \in R \}
\]
Abstraction

- Hyper properties to properties abstraction:

  \[ \langle \varphi(\varphi(\Sigma \times \Sigma)), \subseteq \rangle \xleftrightarrow{\gamma_C} \alpha_C \xrightarrow{\alpha_C} \langle \varphi(\Sigma \times \Sigma), \subseteq \rangle \quad \alpha_C(P) = \bigcup P \quad \gamma_C(S) = \varphi(S) \]

- Post-image isomorphism:

  \[ \langle \varphi(\Sigma \times \Sigma), \subseteq \rangle \xleftrightarrow{\text{post}} \langle \varphi(\Sigma) \rightarrow \varphi(\Sigma), \subseteq \rangle \quad \text{post}(R) = \lambda P \cdot \{ \sigma' \mid \exists \sigma \in P \wedge (\sigma, \sigma') \in R \} \]

  \[ \text{pre}(R) = \lambda X \cdot \{ \sigma \mid \forall \sigma' \in Q . (\sigma, \sigma') \in R \} \]

- Graph isomorphism (a function is isomorphic to its graph, which is a functional relation):

\[ \langle \varphi(\Sigma) \rightarrow \varphi(\Sigma), = \rangle \xleftrightarrow{\alpha_G} \langle \varphi_{\text{fun}}(\varphi(\Sigma) \times \varphi(\Sigma)), = \rangle \quad f \in \varphi(\Sigma) \rightarrow \varphi(\Sigma) \]

\[ \alpha_G(f) = \{ (P, f(P)) \mid P \in \varphi(\Sigma) \} \quad \gamma_G(R) = \lambda P \cdot (Q \text{ such that } (P, S) \in R) \]
Abstraction

• Negation abstraction:

\[ X \in \wp(\mathcal{X}), \alpha^{-}(X) \triangleq \neg X \text{ (where } \neg X \triangleq \mathcal{X} \setminus X) \]

\[ \langle \wp(\mathcal{X}), \subseteq \rangle \iff_{\alpha^{-}} \langle \wp(\mathcal{X}), \supseteq \rangle \quad \text{and} \quad \langle \wp(\mathcal{X}), \supseteq \rangle \iff_{\alpha^{-}} \langle \wp(\mathcal{X}), \subseteq \rangle \]
Consequence approximation
Approximation abstraction

• The component wise approximation:

\[ \langle x, y \rangle \subseteq, \leq \langle x', y' \rangle \iff x \subseteq x' \land y \leq y' \]
Approximation abstraction

- The component wise approximation:

\[ \langle x, y \rangle \subseteq, \leq \langle x', y' \rangle \iff x \subseteq x' \land y \leq y' \]

- Over-approximation:

\[
\text{post}(\subseteq, \supseteq) = \lambda R \cdot \{ \langle P, Q \rangle \mid \exists \langle P', Q' \rangle \in R \cdot P \subseteq P' \land Q' \subseteq Q \}
\]
Approximation abstraction

• The component wise approximation:

\[ \langle x, y \rangle \subseteq \leq \langle x', y' \rangle \iff x \subseteq x' \land y \leq y' \]

• Over-approximation:

\[ \text{post}(\subseteq, \supseteq) = \lambda R \cdot \{ \langle P, Q \rangle | \exists \langle P', Q' \rangle \in R . P \subseteq P' \land Q' \subseteq Q \} \]

• Under-approximation:

\[ \text{post}(\supseteq, \subseteq) = \lambda R \cdot \{ \langle P, Q \rangle | \exists \langle P', Q' \rangle \in R . P' \subseteq P \land Q \subseteq Q' \} \]
Comparing logics through their theories
Comparing logics through their theories

- Strongest postcondition logic (SL):

\[ \mathcal{T}(S) \triangleq \alpha_G \circ \text{post} \circ \alpha_C (\{ [S] \}) \]
\[ = \{ \langle P, \text{post}[S]P \rangle | P \in \wp(\Sigma) \} \]
Comparing logics through their theories

- **Strongest postcondition logic (SL):**
  \[
  \mathcal{T}(S) \triangleq \alpha_G \circ \text{post} \circ \alpha_C\left(\llbracket S \rrbracket\right)
  \]
  \[
  = \{ \langle P, \text{post}[S]P \rangle \mid P \in \wp(\Sigma) \}
  \]

- **Hoare logic (HL):**
  \[
  \mathcal{T}_{HL}(S) \triangleq \text{post}(\supseteq) \circ \mathcal{T}(S)
  \]
Comparing logics through their theories

• Strongest postcondition logic (SL): \[ T(S) \triangleq \alpha_G \circ \text{post} \circ \alpha_C([S]) \]
  \[ = \{ (P, \text{post}[S]P) \mid P \in \wp(\Sigma) \} \]

• Hoare logic (HL): \[ T_{\text{HL}}(S) \triangleq \text{post}(\supseteq \subseteq) \circ T(S) \]

• Incorrectness logic (IL): \[ T_{\text{IL}}(S) \triangleq \text{post}(\subseteq \supseteq) \circ T(S) \]
Comparing logics through their theories

• Strongest postcondition logic (SL):  \( \mathcal{T}(S) \triangleq \alpha_G \circ \text{post} \circ \alpha_C(\{[S]\}) \)
  \[
  = \{ \langle P, \text{post}[S]P \rangle \mid P \in \varphi(\Sigma) \}
  \]

• Hoare logic (HL):  \( \mathcal{T}_{HL}(S) \triangleq \text{post}(\unleq \subseteq) \circ \mathcal{T}(S) \)

• Incorrectness logic (IL):  \( \mathcal{T}_{IL}(S) \triangleq \text{post}(\subseteq \unleq) \circ \mathcal{T}(S) \)

• Hoare incorrectness logic (\(\overline{HL}\)):  \( \mathcal{T}_{\overline{HL}}(S) \triangleq \text{post}(\unleq \subseteq) \circ \alpha^- \circ \mathcal{T}_{HL}(S) \)
Comparing logics through their theories

Fig. 3. Hierarchical taxonomy of transformational assertional logics
Fixpoint abstraction
2. Abstraction

- The abstraction of a fixpoint is a fixpoint (POPL 79)

Theorem II.2.1 (Fixpoint abstraction). If \( \langle C, \sqsubseteq \rangle \leftarrow \alpha \rightarrow \langle A, \preceq \rangle \) is a Galois connection between complete lattices \( \langle C, \sqsubseteq \rangle \) and \( \langle A, \preceq \rangle \), \( f \in C \xrightarrow{i} C \) and \( \tilde{f} \in A \xrightarrow{i} A \) are increasing and commuting, that is, \( \alpha \circ f = \tilde{f} \circ \alpha \), then \( \alpha(\text{lfp}^\sqsubseteq f) = \text{lfp}^\preceq \tilde{f} \) (while semi-commutation \( \alpha \circ f \preceq \tilde{f} \circ \alpha \) implies \( \alpha(\text{lfp}^\sqsubseteq f) \preceq \text{lfp}^\preceq \tilde{f} \)).
2. Abstraction

- The abstraction of a fixpoint is a fixpoint (POPL 79)

**Theorem II.2.1 (Fixpoint abstraction).** If \( \langle C, \sqsubseteq \rangle \xrightarrow{\alpha} \langle A, \preceq \rangle \) is a Galois connection between complete lattices \( \langle C, \sqsubseteq \rangle \) and \( \langle A, \preceq \rangle \), \( f \in C \xrightarrow{\alpha} C \) and \( \bar{f} \in A \xrightarrow{\alpha} A \) are increasing and commuting, that is, \( \alpha \circ f = \bar{f} \circ \alpha \), then \( \alpha(\mathrm{lfp} \subseteq f) = \mathrm{lfp} \preceq \bar{f} \) (while semi-commutation \( \alpha \circ f \preceq \bar{f} \circ \alpha \) implies \( \alpha(\mathrm{lfp} \subseteq f) \preceq \mathrm{lfp} \preceq \bar{f} \)).

- We get a fixpoint definition of the theory of strongest postconditions logic (SL)

- For the iteration \( W = \text{while } (B) S : \)

\[
\mathcal{T}(W) \defeq \{ \langle P, \text{post}[-B][\lambda X \cdot P \cup \text{post}(\lceil B \rceil ; \lceil S \rceil^e) X) \rangle \mid P \in \wp(\Sigma) \}
\]
Lemma 1.1 (Composition). \( \text{post}(X \circ Y) = \text{post}(Y) \circ \text{post}(X) \).

Proof of Lemma 1.1.

\( \text{post}(X \circ Y) \)

- \( \text{AP} \) \( \{a' \} \) \( \exists \sigma \in P \cup \{(a', a') \} \times X \circ Y \) \( \{ \text{def. post} \} \)
- \( \text{AP} \) \( \{a' \} \) \( \exists \sigma \in P \cup \{a', a' \} \times X \cup Y \) \( \{ \text{def. AP} \} \)
- \( \text{AP} \) \( \{a' \} \) \( \exists \sigma \in \{a' \} \cup \{a', a' \} \times (X \cup Y) \) \( \{ \text{def. AP} \} \)
- \( \text{AP} \) \( \{a' \} \) \( \exists \sigma \in \text{post}(X \circ Y) \) \( \{ \text{def. post} \} \)
- \( \text{AP} \) \( \{a' \} \) \( \exists \sigma \in \text{post}(X) \circ \text{post}(Y) \) \( \{ \text{def. function composition} \} \)

Lemma 1.2 (Test). \( \text{post}[b] P = P \land \{b \} \).

Proof of Lemma 1.2.

\( \text{post}[b] P \)

- \( \{a' \} \) \( \exists \sigma \in P \cup \{a', a' \} \) \( \{ \text{def. post} \} \)
- \( \{ \sigma \in P \times \{a', a' \} \} \) \( \{ \text{def. \{a', a' \}} \} \)
- \( \text{AP} \) \( \{a' \} \) \( \exists \sigma \in \{a' \} \times (X \times X) \) \( \{ \text{def. AP} \} \)
- \( \text{AP} \) \( \{a' \} \) \( \exists \sigma \in \text{post}[b] P \) \( \{ \text{def. post} \} \)
- \( \text{AP} \) \( \{a' \} \) \( \exists \sigma \in \text{post}(X) \times \text{post}(Y) \) \( \{ \text{def. function composition} \} \)

Lemma 1.3 (Strongest postcondition). \( T(s) = a_0 = \text{post}[s] = \{lfp, \text{post}[s] P \mid P \in \mathcal{P}(s)\} \).

Proof of Lemma 1.3.

\( T(s) \)

- \( a_0 = \text{post} a_0 = \text{post}[s] \{s\} \) \( \{ \text{def. T} \} \)
- \( a_0 = \text{post}(\{s\}, X \times X) \) \( \{ \text{def. post} \} \)
- \( a_0 = \text{post}[s] \) \( \{ \text{def. post} \} \)
- \( a_0 = \text{post}(s) \) \( \{ \text{def. post} \} \)
- \( a_0 = \text{post}[s] P \) \( \{ \text{def. post} \} \)
- \( \{lfp, \text{post}[s] P \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. T} \} \)

Lemma 1.4 (Strongest postcondition over approximation). \( T_m(s) = \text{post} a_0 = \text{post}[s] = \{lfp, \text{post}[s] P \mid P \in \mathcal{P}(s)\} \).

Proof of Lemma 1.4.

\( T_m(s) \)

- \( \text{post} a_0 = \text{post}[s] \) \( \{ \text{def. function composition} \} \)
- \( \text{post} a_0 = \text{post}[s] = \{lfp, \text{post}[s] P \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. post} \} \)
- \( \{lfp, \text{post}[s] P \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. post} \} \)
- \( \{lfp, \text{post}[s] P \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. post} \} \)
- \( \{lfp, \text{post}[s] P \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. post} \} \)
- \( \{lfp, \text{post}[s] P \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. post} \} \)

Corollary 1.8 (Conditional iteration strongest postcondition graph). \( T(W) = \{(P, \text{post}[a] \{lfp, \text{post}[a] P \mid P \in \mathcal{P}(s)\}) \mid P \in \mathcal{P}(s)\} \).

Proof of Corollary 1.8.

\( T(W) \)

- \( \text{post}[a] = \text{post}[b] \) \( \{ \text{def. post} \} \)
- \( \text{post}[a] = \text{post}[b] \) \( \{ \text{def. post} \} \)
- \( \{lfp, \text{post}[a] \{lfp, \text{post}[a] P \mid P \in \mathcal{P}(s)\} \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. post} \} \)
- \( \{lfp, \text{post}[a] \{lfp, \text{post}[a] P \mid P \in \mathcal{P}(s)\} \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. post} \} \)
- \( \{lfp, \text{post}[a] \{lfp, \text{post}[a] P \mid P \in \mathcal{P}(s)\} \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. post} \} \)
- \( \{lfp, \text{post}[a] \{lfp, \text{post}[a] P \mid P \in \mathcal{P}(s)\} \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. post} \} \)
- \( \{lfp, \text{post}[a] \{lfp, \text{post}[a] P \mid P \in \mathcal{P}(s)\} \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. post} \} \)
- \( \{lfp, \text{post}[a] \{lfp, \text{post}[a] P \mid P \in \mathcal{P}(s)\} \mid P \in \mathcal{P}(s)\} \) \( \{ \text{def. post} \} \)
IV) Design of the proof system
Aczel correspondence
Aczel correspondence between deductive systems and fixpoints

- Rules: \( \frac{P}{\mathcal{C}} \) (\( \mathcal{U} \) universe, \( P \in \mathcal{F}_\text{fin}(\mathcal{U}) \) premiss, \( c \in \mathcal{U} \) conclusion, \( \emptyset \mathcal{C} \) axiom)
Aczel correspondence between deductive systems and fixpoints

- **Rules:** \( \frac{P}{c} \) (\( \mathcal{U} \) universe, \( P \in \wp_{\text{fin}}(\mathcal{U}) \) premiss, \( c \in \mathcal{U} \) conclusion, \( \emptyset \) axiom)

- **Deductive system:** \( R = \left\{ \frac{P_i}{c_i} \mid i \in \Delta \right\} \), \( R \in \wp(\wp_{\text{fin}}(\mathcal{U}) \times \mathcal{U}) \)
Aczel correspondence between deductive systems and fixpoints

- **Rules:** \( \frac{P}{c} \) (\( \mathcal{U} \) universe, \( P \in \wp(\mathcal{U}) \) premiss, \( c \in \mathcal{U} \) conclusion, \( \wp \) axiom)

- **Deductive system:** \( R = \left\{ \frac{P_i}{c_i} \mid i \in \Delta \right\}, \quad R \in \wp(\wp(\wp(\mathcal{U}) \times \mathcal{U}) \times \mathcal{U}) \)

- **Subset of the universe \( \mathcal{U} \) defined by \( R \):**

\[
\begin{align*}
\{ t_n \in \mathcal{U} \mid & \exists t_1, \ldots, t_{n-1} \in \mathcal{U} \cdot \forall k \in [1, n] \cdot \exists \frac{P}{c} \in R \cdot P \subseteq \{ t_1, \ldots, t_{k-1} \} \land t_k = c \} \\
= & \text{ lfp} \subseteq F(R) \quad \text{\textarrow{\textup{proof theoretic}}} \\
F(R)X & \triangleq \left\{ c \mid \exists \frac{P}{c} \in R \cdot P \subseteq X \right\} \quad \text{\textarrow{\textup{model theoretic (gfp for coinduction)}}} \\
\end{align*}
\]

\( \text{\textarrow{\textup{consequence operator}}} \)
Aczel correspondence between deductive systems and fixpoints

- Rules: $\frac{P}{c}$ (universal, $P \in \varnothing_{\text{fin}}(U)$ premiss, $c \in U$ conclusion, $\varnothing$ axiom)

- Deductive system: $R = \left\{ \frac{P_i}{c_i} \mid i \in \Delta \right\}$, $R \in \varnothing(\varnothing_{\text{fin}}(U) \times U)$

- Subset of the universe $U$ defined by $R$:
  \[
  \begin{align*}
  \{ t_n \in U \mid \exists t_1, \ldots, t_{n-1} \in U . \forall k \in [1, n] . \exists \frac{P}{c} \in R . P \subseteq \{ t_1, \ldots, t_{k-1} \} \land t_k = c \} \\
  &= \text{lfp} \subseteq F(R) \\
  F(R)X &\triangleq \left\{ c \mid \exists \frac{P}{c} \in R . P \subseteq X \right\}
  \end{align*}
  \]

- Deductive system defining $\text{lfp} \subseteq F$:
  $R_F \triangleq \left\{ \frac{P}{c} \mid P \subseteq U \land c \in F(P) \right\}$
Why not using Aczel method to get the proof system at this point?

- We get a sound and complete proof system

- BUT impractical:
  - your first prove the strongest consequence, and then
  - the consequence rule to approximate!
Why not using Aczel method to get the proof system at this point?

• We get a sound and complete proof system

• **BUT** impractical:
  
  • you first prove the strongest postcondition, and then
  
  • use the consequence rule to approximate!
Fixpoint induction
Fixpoint induction

**Theorem H.3 (Non empty intersection with abstraction of least fixpoint).** Assume that (1) \( \langle L, \sqsubseteq, \bot, \top, \sqcap, \sqcup \rangle \) is a complete lattice; (2) \( f \in L \rightarrow L \) preserves nonempty joins \( \sqcup \); (3) \( \langle L, \sqsubseteq \rangle \xrightarrow{\gamma} \langle \bar{L}, \preceq, \land \rangle \); (4) \( \bar{Q} \in \bar{L} \setminus \{ \omega \} \) where \( 0 \preceq \alpha(\bot) \); (5) There exists an inductive invariant \( I \in L \) of \( f \) (i.e. \( f(I) \subseteq I \)); (6) \( \langle W, \leq \rangle \) is a well-founded set and \( v \in \text{atoms}(I) \rightarrow W \) is a (variant) function; (7) There exists a sequence \( (a_i \in \text{atoms}(I), i \in [1, \infty]) \) that (7.a) \( a_1 \in f(\bot) \), (7.b) \( \forall i \in [1, \infty] \cdot a_{i+1} \in \text{atoms}(f(a_i)) \), (7.c) \( \forall i \in [1, \infty] \cdot (a_i \neq a_{i+1}) \Rightarrow (v(a_i) > v(a_{i+1})) \), (7.d) \( \forall i \in [1, \infty] \cdot (v(a_i) \nless v(a_{i+1}) \Rightarrow \alpha(a_i) \land \bar{Q} \neq 0) \); Then, hypotheses (1) to (7) imply \( \alpha(\text{lfp}^E f) \land \bar{Q} \neq 0 \). Conversely (1) to (4) and \( \text{lfp}^E f \sqcap \gamma(\bar{Q}) \neq \bot \) imply (5) to (7).
Calculational design of the proof system
HL does not need a consequence rule

Theorem 4.1 (Equivalent definitions of $\overline{HL}$ theories).

$$\overline{T_{HL}}(S) \triangleq \text{post}(\preceq, \succeq) \circ \alpha^\succeq \circ T_{HL}(S) = \alpha^\succeq \circ T_{HL}(S)$$

Observe that Th. 4.1 shows that $\text{post}(\preceq, \succeq)$ can be dispensed with. This implies that the consequence rule is useless for Hoare incorrectness logic.

Proof of Th. 4.1.

$$\begin{align*}
\overline{T_{HL}}(S) & \triangleq \text{post}(\preceq, \succeq) \circ \alpha^\succeq \circ T_{HL}(S) \\
& = \text{post}(\preceq, \succeq)(\neg \{\langle P, Q \rangle | \text{post}[S]P \subseteq Q\}) \quad \{\text{def. } \overline{T_{HL}}\} \\
& = \text{post}(\preceq, \succeq)(\{\langle P, Q \rangle | \neg \text{post}[S]P \subseteq Q\}) \quad \{\text{Lem. 1.4 and def. (30)}\} \\
& = \text{post}(\preceq, \succeq)(\{\langle P, Q \rangle | \text{post}[S]P \cap \neg Q \neq \emptyset\}) \quad \{\text{def. } \neg\} \\
& = \{\langle P', Q' \rangle | \exists (P, Q) \in \{\langle P, Q \rangle | \text{post}[S]P \cap \neg Q \neq \emptyset\} \}. \langle P, Q \rangle \preceq \succeq \langle P', Q' \rangle\} \quad \{\text{def. } \text{post}\} \\
& = \{\langle P', Q' \rangle | \exists (P, Q) . \text{post}[S]P \cap \neg Q \neq \emptyset \land \langle P, Q \rangle \preceq \succeq \langle P', Q' \rangle\} \quad \{\text{def. } \epsilon\} \\
& = \{\langle P', Q' \rangle | \exists Q . \text{post}[S]P \cap \neg Q \neq \emptyset \land Q \supseteq Q'\} \quad \{\text{component wise def. of } \preceq, \succeq\} \\
& = \{\langle P', Q' \rangle | \exists Q . \text{post}[S]P' \cap \neg Q \neq \emptyset \land Q \supseteq Q'\} \\
& \quad \{\epsilon\} \quad \text{if } P \subseteq P' \text{ then } \text{post}[S]P \subseteq \text{post}[S]P' \text{ by (12) so that } \text{post}[S]P \cap \neg Q \neq \emptyset \text{ implies } \text{post}[S]P' \cap \neg Q \neq \emptyset; \\
& \quad \{\succeq\} \quad \text{conversely, if } \exists Q . \text{post}[S]P' \text{, then } \exists P . \text{post}[S]P \cap \neg Q \neq \emptyset \land P \subseteq P' \text{ by choosing } P = P'.\} \\
& = \{\langle P', Q' \rangle | \text{post}[S]P' \cap \neg Q' \neq \emptyset\} \\
& \quad \{\epsilon\} \quad \text{if } Q \supseteq Q' \text{ then } \neg Q' \supseteq \neg Q \text{ so } \text{post}[S]P' \cap \neg Q \neq \emptyset \text{ implies } \text{post}[S]P' \cap \neg Q' \neq \emptyset; \\
& \quad \{\succeq\} \quad \text{conversely post}[S]P' \cap \neg Q' \neq \emptyset \text{ implies } \exists Q . \text{post}[S]P' \cap \neg Q \neq \emptyset \land Q \supseteq Q' \text{ by choosing } Q = Q'.\} \\
& = \{\langle P, Q \rangle | \neg (\text{post}[S]P \subseteq Q)\} \quad \{\text{def. } \epsilon \text{ and } \neg\} \\
& = \alpha^\succeq \circ T_{HL}(S) \quad \{\text{def. } \alpha^\succeq \text{ and } T_{HL} \text{ for Hoare logic}\} \quad \square
\end{align*}
\[ \mathcal{T}_{\Pi}(w) = \{(P, Q) \mid \exists n \geq 1. 3(\sigma_i \in I, i \in [1,n]) \cdot \sigma_i \in P \land \forall i \in [1,n]. \{B[i] \cap \{\sigma_i\}, \neg\{\sigma_{i+1}\}\} \in \mathcal{T}_{\Pi}(S) \land \sigma_n \notin B[i] \land \sigma_n \notin \emptyset \} \]

**Proof of Th. 4.2.** \( W = \text{while} (B) S \)

\[ \mathcal{T}_{\Pi}(w) = \{(P, Q) \mid \text{post}[-][\emptyset][f]_P \cap \neg Q \cap \emptyset \} \]

(Lem. 1.3, where \( f_0(X) \triangleq P \cup \text{post}(B) \cap [s^*]X \))

\[ \{(P, Q) \mid \text{post}[-][\emptyset][f]_P \cap \neg Q \cap \emptyset \} \]

\[ \{(P, Q) \mid \exists i \in I, \exists W, \exists \notin B[i], \exists i \in I \rightarrow W. \exists (\sigma_i \in I, i \in [1,\infty]) \cdot \sigma_i \in f_0(X) \cap \forall i \in [1,\infty] \cdot \sigma_i \in f_0(X) \cap \forall i \in [1,\infty] \cdot (\sigma_i \neq \sigma_{i+1}) \Rightarrow (v(\sigma_i) > v(\sigma_{i+1}) \cap \forall i \in [1,\infty] \cdot (v(\sigma_i) > v(\sigma_{i+1}) \Rightarrow \sigma_i \in \text{pre}[-][\emptyset](\neg Q)) \}
\]

\[ \text{induction principle Th. H.3} \]

\[ \{(P, Q) \mid \exists i \in I, \exists W, \notin B[i], \exists i \in I \rightarrow W. \exists (\sigma_i \in I, i \in [1,\infty]) \cdot \sigma_i \in P \land \forall i \in [1,\infty] \cdot (\sigma_i \neq \sigma_{i+1}) \Rightarrow (v(\sigma_i) > v(\sigma_{i+1}) \cap \forall i \in [1,\infty] \cdot (v(\sigma_i) > v(\sigma_{i+1}) \Rightarrow \sigma_i \in \text{pre}[-][\emptyset](\neg Q)) \}
\]

\[ \text{since if } \sigma_{i+1} \in P, \text{we can equivalently consider the sequence } \sigma_i \in I, i \in [1,\infty]) \}

\[ \{(P, Q) \mid \exists i \in I, \exists W, \notin B[i], \exists i \in I \rightarrow W. \exists (\sigma_i \in I, i \in [1,\infty]) \cdot \sigma_i \in P \land \forall i \in [1,\infty] \cdot (\sigma_i \neq \sigma_{i+1}) \Rightarrow (v(\sigma_i) > v(\sigma_{i+1}) \cap \forall i \in [1,\infty] \cdot (v(\sigma_i) > v(\sigma_{i+1}) \Rightarrow \sigma_i \in \text{pre}[-][\emptyset](\neg Q)) \}
\]

\[ \text{(def. } \text{pre} \text{)} \]

\[ \{(P, Q) \mid \exists i \in I, \exists W, \notin B[i], \exists i \in I \rightarrow W. \exists (\sigma_i \in I, i \in [1,\infty]) \cdot \sigma_i \in P \land \forall i \in [1,\infty] \cdot (\sigma_i \neq \sigma_{i+1}) \Rightarrow (v(\sigma_i) > v(\sigma_{i+1}) \cap \forall i \in [1,\infty] \cdot (v(\sigma_i) > v(\sigma_{i+1}) \Rightarrow \sigma_i \in \text{pre}[-][\emptyset](\neg Q)) \}
\]

\[ \text{(def. } \text{pre} \text{)} \]

\[ \{(P, Q) \mid \exists n \geq 1. 3(\sigma_i \in I, i \in [1,n]) \cdot \sigma_i \in P \land \forall i \in [1,n]. \{\sigma_{i+1} \leq \text{post}(B) \cap [s^*] \sigma_i \land \sigma_n \notin B[i] \land \sigma_n \notin \emptyset \} \}
\]

\[ I \text{ is not used and can always be chosen to be } \Sigma \}

\[ \{(P, Q) \mid \exists n \geq 1. 3(\sigma_i \in I, i \in [1,n]) \cdot \sigma_i \in P \land \forall i \in [1,n]. \{\sigma_{i+1} \leq \text{post}(B) \cap [s^*] \sigma_i \land \sigma_n \notin B[i] \land \sigma_n \notin \emptyset \} \}
\]

\[ \{(P, Q) \mid \exists n \geq 1. 3(\sigma_i \in I, i \in [1,n]) \cdot \sigma_i \in P \land \forall i \in [1,n]. \{\sigma_{i+1} \leq \text{post}(B) \cap [s^*] \sigma_i \land \sigma_n \notin B[i] \land \sigma_n \notin \emptyset \} \}
\]

\[ \{(P, Q) \mid \exists n \geq 1. 3(\sigma_i \in I, i \in [1,n]) \cdot \sigma_i \in P \land \forall i \in [1,n]. \{\sigma_{i+1} \leq \text{post}(B) \cap [s^*] \sigma_i \land \sigma_n \notin B[i] \land \sigma_n \notin \emptyset \} \}
\]

\[ \{(P, Q) \mid \exists n \geq 1. 3(\sigma_i \in I, i \in [1,n]) \cdot \sigma_i \in P \land \forall i \in [1,n]. \{\sigma_{i+1} \leq \text{post}(B) \cap [s^*] \sigma_i \land \sigma_n \notin B[i] \land \sigma_n \notin \emptyset \} \}
\]

\[ \{(P, Q) \mid \exists n \geq 1. 3(\sigma_i \in I, i \in [1,n]) \cdot \sigma_i \in P \land \forall i \in [1,n]. \{\sigma_{i+1} \leq \text{post}(B) \cap [s^*] \sigma_i \land \sigma_n \notin B[i] \land \sigma_n \notin \emptyset \} \}
\]

\[ \{(P, Q) \mid \exists n \geq 1. 3(\sigma_i \in I, i \in [1,n]) \cdot \sigma_i \in P \land \forall i \in [1,n]. \{\sigma_{i+1} \leq \text{post}(B) \cap [s^*] \sigma_i \land \sigma_n \notin B[i] \land \sigma_n \notin \emptyset \} \}
\]
Proof system of \( \overline{HL} \)

**Theorem 4.3 (\( \overline{HL} \) rules for conditional iteration).**

\[
\exists (\sigma_i \in I, i \in [1, n]) \cdot \sigma_1 \in P \land \forall i \in [1, n[ . (B[B] \cap \{\sigma_i\}) S (\lnot\{\sigma_{i+1}\}) \land \sigma_n \notin B[B] \land \sigma_n \notin Q \hfill (3)
\]

\[\langle P \rangle \text{while (B) } S (\langle Q \rangle)\]

**Proof of (3).** We write \( \langle P \rangle S (\langle Q \rangle) = \langle P, Q \rangle \in \overline{HL}(S) \).

By structural induction (S being a strict component of \( \text{while (B) } S \)), the rule for \( \langle P \rangle S (\langle Q \rangle) \) have already been defined;

By **Aczel method**, the (constant) fixpoint \( \text{lfp}^\subseteq \lambda X \cdot S \) is defined by \( \{ \emptyset : c \in S \} \);

So for \( \text{while (B) } S \) we have an axiom \( \langle P \rangle \text{while (B) } S (\langle Q \rangle) \) with side condition \( \exists (\sigma_i \in I, i \in [1, n]) \cdot \sigma_1 \in P \land \forall i \in [1, n[ . (B[B] \cap \{\sigma_i\}) S (\lnot\{\sigma_{i+1}\}) \land \sigma_n \notin B[B] \land \sigma_n \notin Q \) where \( \langle B[B] \cap \{\sigma_i\} \rangle S (\lnot\{\sigma_{i+1}\}) \) is well-defined by structural induction;

Traditionally, the side condition is written as a premiss, to get (3).
• IL is **not** Hoare incorrectness logic (sufficient, not necessary)

\[
\neg (\{P\} S \{Q\}) \quad \not\iff \quad [P] S \neg Q
\]

\[
\iff \quad \exists R \in \wp(\Sigma). [P] S [R] \land R \cap \neg Q \neq \emptyset
\]

\[
\iff \quad \exists \sigma \in \Sigma. [P] S \{\{\sigma\}\} \land \sigma \notin Q
\]
Conclusion

- Was Peter correct or incorrect?

- He took the hardest path
- Hoare's incorrectness logic is the easiest and most popular way
- It has proof verifiers and theorem provers
- They are called debuggers
- It makes debugging a formal activity relying on a formal logic!
Conclusion

• Was Peter correct or incorrect?
• In a certain sense, he was correct
• **BUT** he took the hardest path

Hoare incorrectness logic is the easiest and most popular way. It has proof verifiers and theorem provers. They are called debuggers. It makes debugging a formal activity relying on a formal logic!
Conclusion

• Was Peter correct or incorrect?
• In a certain sense, he was correct
• BUT he took the hardest path
• Hoare incorrectness logic is the easiest and most popular way
Conclusion

- Was Peter correct or incorrect?
- In a certain sense, he was correct
- **BUT** he took the hardest path
- Hoare incorrectness logic is the easiest and most popular way
  - It has **proof verifiers** and **theorem provers**
Conclusion

• Was Peter correct or incorrect?

• In a certain sense, he was correct

• **BUT** he took the hardest path

• Hoare incorrectness logic is the easiest and most popular way
  • It has proof verifiers and theorem provers
  • They are called debuggers
Conclusion

• Was Peter correct or incorrect?

• In a certain sense, he was correct

• **BUT** he took the hardest path

• Hoare incorrectness logic is the easiest and most popular way
  • It has **proof verifiers and theorem provers**
  • They are called **debuggers**
  • Which are therefore **formal tools based on a formal logic!**
Conclusion

• Was Peter correct or incorrect?
• In a certain sense, he was correct
• **BUT** he took the hardest path
• Hoare incorrectness logic is the easiest and most popular way
  • It has proof verifiers and theorem provers
  • They are called **debuggers**
  • Which are therefore formal tools based on a formal logic! 😅
The End, Thank You
The End, Thank You
Happy Sixties to Peter