Is Peter Correct or Incorrect?

Patrick Cousot

Courant Institute, New York University
Peter’s Incorrectness Logic

• 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.
Peter’s Incorrectness Logic

• 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?
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.
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.

• That he doesn’t really like

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 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.

• That he doesn’t really like

  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?
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.
Singularities of Logics
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: \( \text{W} = \text{while (true) skip} \)
    - 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 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 $\{P\} W \{Q\}$

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 \( \{P\} W \{Q\} \)
  • i.e. 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 $\llbracket S \rrbracket_\bot$ of the programming language (in structural fixpoint form)
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)

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

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

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

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

3. Calculate the theory $\alpha(⟦S⟧\bot)$ 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 (HL)
I) Relational semantics
I. Angelic relational semantics $\llbracket S \rrbracket^e$

• Syntax*:

$$S \in \mathcal{S} ::= x = A \mid \text{skip} \mid S; S \mid \text{if (B) S else S} \mid \text{while (B) S}$$

• States: $\Sigma$

• Angelic relational semantics: $\llbracket S \rrbracket^e \in \wp(\Sigma \times \Sigma)$
1. Angelic relational semantics $\llbracket S \rrbracket$ (in deductive form)

- Notations using judgements:
  - $\sigma \vdash S \Rightarrow \sigma'$ 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
I. Angelic relational semantics $\llbracket S \rrbracket$ (in deductive form)

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

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

\[
\begin{align*}
\text{(a)} \quad & \sigma \vdash W \overset{i}{\Rightarrow} \sigma \\
\text{(b)} \quad & \frac{B[\text{B}]\sigma, \ \sigma \vdash S \overset{e}{\Rightarrow} \sigma', \ \sigma' \vdash W \overset{i}{\Rightarrow} \sigma''}{\sigma \vdash W \overset{i}{\Rightarrow} \sigma''} \quad (2) \\
\text{(a)} \quad & \sigma \vdash W \overset{i}{\Rightarrow} \sigma', \ B[\neg \text{B}]\sigma' \\
\text{(b)} \quad & \frac{\sigma \vdash W \overset{e}{\Rightarrow} \sigma'}{\sigma \vdash W \overset{e}{\Rightarrow} \sigma'} \quad (3)
\end{align*}
\]
I. Angelic relational semantics $\llbracket S \rrbracket$ (in fixpoint form)

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

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

- 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:

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

- **Hyper properties to properties abstraction:**

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

- **Post-image isomorphism:**

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

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

- **Hyper properties to properties abstraction:**

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

- **Post-image isomorphism:**

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

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

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

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

\[
\alpha_G(f) = \{ \langle P, f(P) \rangle \mid P \in \varphi(\Sigma) \}
\]

\[
\gamma_G(R) = \lambda P \cdot (Q \text{ such that } \langle P, S \rangle \in R)
\]
Abstraction

• Negation abstraction:

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

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

- The component wise approximation:

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

• The component wise approximation:

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

• Over-approximation:

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

- The component wise approximation:

\[ \langle x, y \rangle \sqsubseteq, \sqsubseteq \langle x', y' \rangle \triangleq x \sqsubseteq x' \land y \leq y' \]

- Over-approximation:

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

- Under-approximation:

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

• Strongest postcondition logic (SL): \[ \mathcal{T}(s) \triangleq \alpha_G \circ \text{post} \circ \alpha_C(\llbracket s \rrbracket) \]
  \[= \{ (P, \text{post} \llbracket s \rrbracket P) \mid P \in \wp(\Sigma) \} \]
Comparing logics through their theories

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

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

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

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

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

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

- **Hoare logic (HL):**
  \[ T_{\text{HL}}(W) \triangleq \text{post}(\supseteq \subseteq) \circ T(W) \]

- **Incorrectness logic (IL):**
  \[ T_{\text{IL}}(S) \triangleq \text{post}(\subseteq \supseteq) \circ T(S) \]

- **Hoare incorrectness logic (IH):**
  \[ T_{\text{IH}}(W) = \text{post}(\subseteq, \supseteq) \circ \alpha^- \circ T_{\text{HL}}(W) \]
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 \( \langle C, \sqsubseteq \rangle \leftrightarrow_{\alpha} \langle A, \preceq \rangle \) is a Galois connection between complete lattices \( \langle C, \sqsubseteq \rangle \) and \( \langle A, \preceq \rangle \), \( f \in C \xrightarrow{i} C \) and \( \bar{f} \in A \xrightarrow{i} A \) are increasing and commuting, that is, \( \alpha \circ f = \bar{f} \circ \alpha \), then \( \alpha(\text{lfp}^\sqsubseteq f) = \text{lfp}^\preceq \bar{f} \) (while semi-commutation \( \alpha \circ f \preceq \bar{f} \circ \alpha \) implies \( \alpha(\text{lfp}^\sqsubseteq f) \preceq \text{lfp}^\preceq \bar{f} \)).
2. Abstraction

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

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

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

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

\[
\mathcal{T}(W) \triangleq \{ \langle P, \text{post}[\neg B](lfp^\sqsubseteq \lambda X \cdot P \cup \text{post}([B] \land [S]^e)X) \rangle \mid P \in \wp(\Sigma) \}\]
Lemma 1.4 (Strongest postcondition over approximation).
\[
\begin{align*}
G \cap (\sigma & \cup \exists \subseteq Q) \subseteq (P, \sigma) \subseteq (P, Q') \subseteq \sigma \langle \exists \rangle X \subseteq \langle \exists \rangle \sigma P
\end{align*}
\]
Proof of Lemma 1.4.
\[
\begin{align*}
\text{Lemma 1.2 (Test). } & \text{post}([P = P \land B[b]]).
\end{align*}
\]
Proof of Lemma 1.2.
\[
\begin{align*}
\text{post}[P] & = \langle [P, post[\{P\} \land B[b]]] \rangle
\end{align*}
\]
Proof of Lemma 1.3.
\[
\begin{align*}
\text{Lemma 1.3 (Strongest postcondition). } & \text{T}(S) = a_0 = post[S] = \{P, post[\{P\} \land B[b]]\}.
\end{align*}
\]
Proof of Lemma 1.3.
\[
\begin{align*}
\text{Lemma 1.4 (Strongest postcondition over approximation). } \text{post}(\{S\}) & = \text{T}(S).
\end{align*}
\]
Proof of Lemma 1.4.
\[
\begin{align*}
\text{Lemma 1.5 (Consistency). } & \text{post}(P\{X\}) \subseteq \text{where } X \in \mu (E).
\end{align*}
\]
Proof of Lemma 1.5.
\[
\begin{align*}
\text{Lemma 1.6 (Pointwise commutation). } & \forall X \in \mu (E) = \mu (E\{X\}) = \mu (E\{X\}).
\end{align*}
\]
Proof of Lemma 1.6.
IV) Design of the proof system
Aczel correspondence
Aczel correspondence between deductive systems and fixpoints

- Rules: \( \frac{P}{c} \) (\( U \) universe, \( P \in \wp_{\text{fin}}(U) \) premiss, \( c \in U \) conclusion, \( \emptyset \) 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: \(R = \{\frac{P_i}{c_i} \mid i \in \Delta\}\), \(R \in \wp(\wp_{\text{fin}}(\mathcal{U}) \times \mathcal{U})\)
Aczel correspondence between deductive systems and fixpoints

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

- Deductive system: \( R = \left\{ \frac{P_i}{C_i} \mid i \in \Delta \right\}, \quad R \in \mathcal{P}(\mathcal{P}_{\text{fin}}(\mathcal{U}) \times \mathcal{U}) \)

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

\[
\{ t_n \in \mathcal{U} \mid \exists t_1, \ldots, t_{n-1} \in \mathcal{U} \ . \ \forall k \in [1, n] . \ \exists \frac{P}{c} \in R . \ P \subseteq \{ t_1, \ldots, t_{k-1} \} \land t_k = c \} = \text{lfp} \subseteq F(R) \quad \text{proof theoretic} \downarrow
\]

\[
F(R)X \triangleq \left\{ c \mid \exists \frac{P}{c} \in R . \ P \subseteq X \right\} \quad \text{model theoretic (gfp for coinduction)}
\]

\[
F(R)X \triangleq \left\{ c \mid \exists \frac{P}{c} \in R . \ P \subseteq X \right\} \quad \text{consequence operator}
\]
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:** \( R = \left\{ \frac{P_i}{c_i} \mid i \in \Delta \right\}, \quad R \in \wp(\wp_{\text{fin}}(\mathcal{U}) \times \mathcal{U}) \)

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

  \[
  \{ t_n \in \mathcal{U} \mid \exists t_1, \ldots, t_{n-1} \in \mathcal{U} . \forall k \in [1, n] . \exists \frac{P}{c} \in R . P \subseteq \{ t_1, \ldots, t_{k-1} \} \land t_k = c \}
  \]

  \( \Downarrow \) **proof theoretic**

  \[
  \text{lfp} \subseteq F(R)
  \]

  \( \Downarrow \) **model theoretic** (gfp for coinduction)

  \[
  F(R)X \triangleq \left\{ c \mid \exists \frac{P}{c} \in R . P \subseteq X \right\}
  \]

  \( \leftarrow \) **consequence operator**

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

- We get a sound and complete proof system

  BUT

  • your first prove the strongest consequence
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
Theorem H.3 (Non empty intersection with abstraction of least fixpoint). Assume that (1) \(L, \sqsubseteq, \perp, \top, \sqcap, \sqcup\) is an atomic complete lattice; (2) \(f \in L \to L\) preserves nonempty joins \(\sqcup\); (3) \(\langle L, \sqsubseteq \rangle \xrightarrow{\gamma} \langle \bar{L}, \leq, \wedge \rangle\); (4) \(\bar{Q} \in \bar{L} \setminus \{0\}\) where \(0 \triangleq \alpha(\perp)\); (5) There exists an inductive invariant \(I \in L\) of \(f\) (i.e. \(f(I) \sqsubseteq I\)); (6) \(\langle W, \sqsubseteq \rangle\) is a well-founded set and \(\nu \in \text{atoms}(I) \to W\) is a (variant) function; (7) There exists a sequence \((a_i \in \text{atoms}(I), i \in [1, \infty])\) that (7.a) \(a_1 \in f(\perp)\), (7.b) \(\forall i \in [1, \infty]. a_{i+1} \in \text{atoms}(f(a_i))\), (7.c) \(\forall i \in [1, \infty]. a_i \neq a_{i+1} \implies (\nu(a_i) > \nu(a_{i+1}))\), (7.d) \(\forall i \in [1, \infty]. (\nu(a_i) \neq \nu(a_{i+1}) \implies \alpha(a_i) \wedge \bar{Q} = 0\). Then, hypotheses (1) to (7) imply \(\alpha(lfp^{\sqsubseteq} f) \land \bar{Q} \neq 0\). Conversely (1) to (4) and \(lfp^{\sqsubseteq} f \land \gamma(\bar{Q}) \neq \perp\) imply (5) to (7).
Calculational design of the proof system
**Theorem 4.1 (Equivalent definitions of $\text{HL}$ theories).**

$$
\mathcal{T}_{\text{HL}}(\text{W}) \triangleq \text{post}(\subseteq, \supseteq) \circ \alpha^- \circ \mathcal{T}_{\text{HL}}(\text{W}) = \alpha^- \circ \mathcal{T}_{\text{HL}}(\text{W})
$$

$\text{W} = \text{while} (\text{B}) \text{ S}$

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

**Proof of Th. 4.1.**

$$
\mathcal{T}_{\text{HL}}(\text{W}) = \text{post}(\subseteq, \supseteq) \circ \alpha^- \circ \mathcal{T}_{\text{HL}}(\text{W}) \quad \text{(def. } \mathcal{T}_{\text{HL}})\,,
$$

$$
= \text{post}((\subseteq, \supseteq)(\neg \{P, Q\} | \text{post}[W]P \subseteq Q)) \quad \text{(Lem. 1.4 and def. (30) of } \alpha^-)\,,
$$

$$
= \text{post}(\subseteq, \supseteq)(\{P, Q\} | \neg (\text{post}[W]P \subseteq Q)) \quad \text{(def. } \neg)\,,
$$

$$
= \text{post}(\subseteq, \supseteq)(\{P, Q\} | \text{post}[W]P \cap \neg Q \neq \emptyset) \quad \text{(def. } \subseteq \text{ and } \neg)\,,
$$

$$
= \{P', Q'\} | \exists (P, Q) \in \{P, Q\} | \text{post}[W]P \cap \neg Q \neq \emptyset \cdot (P, Q) \subseteq, \supseteq (P', Q') \quad \text{(def. } \exists)\,,
$$

$$
= \{P', Q'\} | \exists (P, Q) \cdot \text{post}[W]P \cap \neg Q \neq \emptyset \cdot (P, Q) \subseteq, \supseteq (P', Q') \quad \text{(def. } \epsilon)\,,
$$

$$
= \{P', Q'\} | \exists Q \cdot \text{post}[W]P \cap \neg Q \neq \emptyset \cdot Q \supseteq Q' \quad \text{(component wise def. of } \subseteq, \supseteq)\,.
$$

$$
\{}(\epsilon) \text{ if } P \in P' \text{ then } \text{post}[W]P \subseteq \text{post}[W]P' \text{ by (12) so that } \text{post}[W]P \cap \neg Q \neq \emptyset \text{ implies } \text{post}[W]P' \cap \neg Q \neq \emptyset; \quad \text{(2) } \text{conversely, if } \exists Q \cdot \text{post}[W]P', \text{ then } \exists P \cdot \text{post}[W]P \cap \neg Q \neq \emptyset \cdot P \in P' \text{ by choosing } P = P'.\,
$$

$$
\{}(\epsilon) \text{ if } Q \supseteq Q' \text{ then } \neg Q' \supseteq \neg Q \text{ so } \text{post}[W]P' \cap \neg Q \neq \emptyset \text{ implies } \text{post}[W]P' \cap \neg Q' \neq \emptyset; \quad \text{(2) } \text{conversely post}[W]P' \cap \neg Q' \neq \emptyset \text{ implies } \exists Q \cdot \text{post}[W]P' \cap \neg Q \oplus Q \supseteq Q' \text{ by choosing } Q = Q'.\,
$$

$$
= \{P, Q\} | \neg (\text{post}[W]P \subseteq Q) \quad \text{(def. } \neg)\,,
$$

$$
= \alpha^- \circ \mathcal{T}_{\text{HL}}(\text{W}) \quad \text{(def. } \alpha^- \text{ and } \mathcal{T}_{\text{HL}} \text{ for Hoare logic)}\quad \Box
$$
Theorem 4.2 (Theory of $\text{HL}$).

\[ T_{\text{HL}}(w) = \{(P, Q) \mid \exists n \geq 1. \exists \sigma_i \in I, i \in [1, n] \}. \sigma_i \in P \land \forall i \in [1, n], \{[B[i] \land \{\sigma_i\}, \neg(\sigma_{i+1})]\} \in T_{\text{HL}}(S) \land \sigma_n \notin B[B] \land \sigma_n \notin Q \} \]

Proof of Th. 4.2.

\[ T_{\text{HL}}(w) = \{(P, Q) \mid \text{post-}[[B]](\text{hlf}^{*} F_{P}) \land \neg Q \land \emptyset \} \]

\[ \text{Lem. 1.3, where } F_{P}(X) \triangleq P \cup \text{post-}[[B]]( [S^{*}]) X \]

\[ \{(P, Q) \mid \text{hlf}^{*} P(I) \subseteq I \subseteq \exists \{W, \zeta\} \subseteq [w] Bf. \exists v \in I \rightarrow W. \exists \{\sigma_i \in I, i \in [1, \infty]\}. \sigma_i \in F_{\text{hlf}}(\emptyset) \land \forall i \in [1, \infty], \sigma_i \in F_{\text{hlf}}(\{\sigma_i\}) \land \forall i \in [1, \infty], (\sigma_i \neq \sigma_{i+1}) \rightarrow (v(\sigma_i) > v(\sigma_{i+1}) \land \forall i \in [1, \infty]). (v(\sigma_i) \triangleright v(\sigma_{i+1}) \rightarrow \{a_d\} \cap \text{pre}[[B]](\neg Q) = 0 \}
\]

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

\[ \{(P, Q) \mid \exists \{\sigma_i \in I, i \in [1, \infty]\}. \sigma_i \in P \land \forall i \in [1, \infty]. (\sigma_{i+1} \subseteq \text{post-}[[B]]([S^{*}])\{\sigma_i\}) \land \forall i \in [1, \infty]. (\sigma_i \neq \sigma_{i+1}) \rightarrow (v(\sigma_i) > v(\sigma_{i+1}) \land \forall i \in [1, \infty]). (v(\sigma_i) \triangleright v(\sigma_{i+1}) \rightarrow \sigma_i \in \text{pre}[[B]](\neg Q) \}
\]

\[ \text{def. } F_{P}(X) \triangleq P \cup \text{post-}[[B]]([S^{*}]) X, \zeta, \text{ and post, which is } \omega\text{-strict} \]

\[ \{(P, Q) \mid \exists \{\sigma_i \in I, i \in [1, \infty]\}. \sigma_i \in P \land \forall i \in [1, \infty]. (\sigma_{i+1} \subseteq \text{post-}[[B]]([S^{*}])\{\sigma_i\}) \land \forall i \in [1, \infty]. (\sigma_i \neq \sigma_{i+1}) \rightarrow (v(\sigma_i) > v(\sigma_{i+1}) \land \forall i \in [1, \infty]). (v(\sigma_i) \triangleright v(\sigma_{i+1}) \rightarrow \sigma_i \in \text{pre}[[B]](\neg Q) \}
\]

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

\[ \{(P, Q) \mid \exists \{\sigma_i \in I, i \in [1, n]\}. \sigma_i \in P \land \forall i \in [1, n]. (\sigma_{i+1} \subseteq \text{post-}[[B]]([S^{*}])\{\sigma_i\}) \land \sigma_n \in \text{pre}[[B]](\neg Q) \}
\]

\[ \text{def. } \neg \}

\[ \{(P, Q) \mid \exists \{\sigma_i \in I, i \in [1, n]\}. \sigma_i \in P \land \forall i \in [1, n]. (\sigma_{i+1} \subseteq \text{post-}[[B]]([S^{*}])\{\sigma_i\}) \land \sigma_n \in \text{pre}[[B]](\neg Q) \}
\]

\[ \text{def. } \neg \}

\[ \{(P, Q) \mid \exists \{\sigma_i \in I, i \in [1, n]\}. \sigma_i \in P \land \forall i \in [1, n]. (\sigma_{i+1} \subseteq \text{post-}[[B]]([S^{*}])\{\sigma_i\}) \land \sigma_n \in \text{pre}[[B]](\neg Q) \}
\]

\[ \text{def. } T_{\text{HL}}(S) \]
Theorem 4.3 (HL rules for conditional iteration).

\[
\exists (\sigma_i \in I, \ i \in [1,n]) . \ \sigma_1 \in P \land \forall i \in [1,n[ . (\mathcal{B}[B] \cap \{\sigma_i\}) \land (\neg\{\sigma_{i+1}\}) \land \sigma_n \notin B[B] \land \sigma_n \notin Q
\]

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

Proof of (3). We write \((P) \ S (Q) \equiv (P, Q) \in \overline{HL}(S)\);

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

By Aczel method, the (constant) fixpoint lfp \((P) S (Q)\) is defined by \(\{\emptyset \mid c \in S\}\);

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

Traditionally, the side condition is written as a premiss, to get (3). \hfill \Box
About incorrectness

- IL is **not** Hoare incorrectness logic (sufficient, not necessary)

\[ \neg (\{P\} s\{Q\}) \quad \not\Rightarrow \quad [P] s [\neg Q] \]

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

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

• Was Peter correct or incorrect?

• Of course he was correct

• BUT he took the hardest path

• Hoare incorrectness logic is the easiest and most popular way

• They are called debuggers

• It makes debugging a formal activity relying on a formal logic!
Conclusion

• Was Peter correct or incorrect?

• Of course he was correct

• **BUT** he took the hardest path
Conclusion

• Was Peter correct or incorrect?

• Of course 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?
• Of course 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?
• Of course 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?
• Of course 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?
- Of course 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