2 Hoare Logic

Hoare Logic is a proof system for establishing *partial correctness* of imperative programs. Introduced by Tony Hoare in his influential 1969 paper *"An Axiomatic Basis for Computer Programming"*, it provided one of the first compositional and mathematically rigorous frameworks for reasoning about program behaviour. A Hoare proof does not guarantee termination; rather, it guarantees that *if* a program terminates, then the final state satisfies the desired postcondition. This distinction between partial and total correctness has played a central role in the development of modern program logics and predicate–transformer semantics.

This section presents the meaning of Hoare triples, the standard proof rules for the while language we defined in Section 1, several representative examples illustrating backward reasoning, and a soundness and completeness proof with respect to the semantics fixed in Section 1. In doing so, we follow the classical tradition initiated by Hoare and further developed by Floyd, Dijkstra, and many others in the foundations of program verification.

2.1 Hoare triples and partial correctness meaning

Definition 2.1. Assertions are predicates, i.e., sets of states $P, Q \subseteq \Sigma$. We write $\sigma \models P$ for $\sigma \in P$.

We also use the usual logical notation $(\wedge, \vee, \neg, \Rightarrow)$ as shorthand for the corresponding set-theoretic operations on assertions.

Definition 2.2. A Hoare triple has the form

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

where P is the precondition, c is a command, and Q is the postcondition.

Definition 2.3. We interpret Hoare triples using the relational semantics $[\![c]\!] \subseteq \Sigma \times \Sigma$ from Section 1. The triple $\{P\}c\{Q\}$ is *(partially) correct* if every terminating execution of c starting from a state satisfying P ends in a state satisfying Q:

$$\models \{P\}\, c\, \{Q\} \quad \Longleftrightarrow \quad \forall \sigma, \sigma'.\ \big(\sigma \models P \wedge (\sigma, \sigma') \in [\![c]\!]\big) \Rightarrow \sigma' \models Q.$$

Equivalently, using the post-image operator $\mathrm{post}(c)(P)$:

$$\models \{P\}\, c\, \{Q\} \quad \Longleftrightarrow \quad \mathrm{post}(c)(P) \subseteq Q.$$

This is the "over-approximation" direction: all reachable final states from P are contained in Q.

Total correctness (not the focus here). Total correctness strengthens partial correctness by additionally requiring termination from all P-states. Since our later discussion of incorrectness logic and quantum program logics is most naturally aligned with partial correctness, we focus on partial correctness in this thesis unless stated otherwise.

2.2 Proof rules

We present a standard Hoare system for the command language of Section 1. A judgment

$$\vdash \{P\}\, c\, \{Q\}$$

means that the Hoare triple is derivable in the proof system.

Entailment and substitution (semantic). For assertions $P, Q \subseteq \Sigma$, we write $P \Rightarrow Q$ for semantic entailment, i.e. $P \subseteq Q$ (equivalently, $\forall \sigma.\ \sigma \models P \Rightarrow \sigma \models Q$).

For an assertion $Q \subseteq \Sigma$ and an arithmetic expression a, define the assertion $Q[x \leftarrow a] \subseteq \Sigma$ by:

$$Q[x \leftarrow a] \triangleq \{\sigma \in \Sigma \mid \sigma[x \mapsto [\![a]\!](\sigma)] \models Q\}.$$

Equivalently,

$$\sigma \models Q[x \leftarrow a] \quad \Longleftrightarrow \quad \sigma[x \mapsto [\![a]\!](\sigma)] \models Q.$$

$$\frac{}{\vdash \{P\}\ \texttt{skip}\ \{P\}}\ (\textsc{Skip}) \qquad \frac{}{\vdash \{Q[x \leftarrow a]\}\ x := a\ \{Q\}}\ (\textsc{Assign})$$

$$\frac{\vdash \{P\}\ c_1\ \{R\} \qquad \vdash \{R\}\ c_2\ \{Q\}}{\vdash \{P\}\ c_1; c_2\ \{Q\}}\ (\textsc{Seq})$$

$$\frac{\vdash \{P \wedge b\}\ c_1\ \{Q\} \qquad \vdash \{P \wedge \neg b\}\ c_2\ \{Q\}}{\vdash \{P\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{Q\}}\ (\textsc{If})$$

$$\frac{\vdash \{I \wedge b\}\ c\ \{I\}}{\vdash \{I\}\ \texttt{while}\ b\ \texttt{do}\ c\ \{I \wedge \neg b\}}\ (\textsc{While})$$

$$\frac{P' \Rightarrow P \qquad \vdash \{P\}\ c\ \{Q\} \qquad Q \Rightarrow Q'}{\vdash \{P'\}\ c\ \{Q'\}}\ (\textsc{Conseq})$$

The rules above form the standard Hoare Logic proof system for partial correctness. The Skip and Assign rules describe the effect of basic commands. In particular, the assignment rule relies on the substitution property stated above: to ensure a postcondition Q after executing $x := a$, it suffices to require the precondition $Q[x \leftarrow a]$ beforehand. This is the essence of backward reasoning. The Seq rule expresses compositionality: correctness of $c_1; c_2$ follows from correctness of each component with a suitable intermediate assertion. The If rule splits reasoning along the two branches of the conditional, interpreting the guard b as a state predicate via expression evaluation $\sigma \models b \triangleq [\![b]\!](\sigma) = \texttt{tt}$. The While rule introduces a loop invariant I, which must hold initially and be preserved by the body whenever the guard is true. This captures inductive reasoning over an unbounded number of iterations. Finally, the Conseq rule allows strengthening preconditions and weakening postconditions, enabling the adjustment of assertions to fit the structure of a larger proof.

2.3 Examples (backward reasoning)

In the examples below, we intentionally keep programs small while showing the main proof patterns.

Example 1 Consider the program that enforces non-negativity: $c \equiv \texttt{if}\ (x < 0)\ \texttt{then}\ x := -x\ \texttt{else skip}$. We prove: $\vdash \{\texttt{true}\}\ c\ \{x \geq 0\}$. By (If), it suffices to prove two branches:

Then-branch. Goal: $\vdash \{x < 0\}\ x := -x\ \{x \geq 0\}$. By (Assign), it suffices to show: $x < 0 \Rightarrow (x \geq 0)[x \leftarrow -x]$, i.e. $x < 0 \Rightarrow (-x \geq 0)$, which is valid over integers.

Else-branch. Goal: $\vdash \{x \geq 0\}\ \texttt{skip}\ \{x \geq 0\}$, which follows from (Skip).

Conclusion. Apply (If) and optionally (Conseq) (with true as the overall precondition) to conclude the result.

Example 2 We verify a standard loop that computes $1 + \cdots + n$ for $n \geq 0$. Let the program be:

$$c \equiv s := 0;\ i := 0;\ \texttt{while}\ (i < n)\ \texttt{do}\ \big(i := i + 1;\ s := s + i\big).$$

We prove the partial correctness specification:

$$\vdash \{n \geq 0\}\ c\ \{s = \tfrac{n(n+1)}{2}\}.$$

Invariant choice. A natural invariant relates s and i is as follows (the intuition is after i iterations, the program has accumulated $1 + \cdots + i$):

$$I \triangleq (0 \leq i \leq n)\ \wedge\ \left(s = \tfrac{i(i+1)}{2}\right).$$

Stage 1: initialization establishes I. After $s := 0;\ i := 0$, we want I. Using (Seq) twice and (Assign) backward:

- For $i := 0$, sufficient precondition is $I[i \leftarrow 0]$.
- For $s := 0$, sufficient precondition is $(I[i \leftarrow 0])[s \leftarrow 0]$.

Compute:

$$I[i \leftarrow 0] \equiv (0 \leq 0 \leq n) \wedge \left(s = \tfrac{0 \cdot 1}{2}\right) \equiv (0 \leq n) \wedge (s = 0).$$

Then

$$(I[i \leftarrow 0])[s \leftarrow 0] \equiv (0 \leq n) \wedge (0 = 0),$$

which is implied by $n \geq 0$. Hence

$$\vdash \{n \geq 0\}\ s := 0;\ i := 0\ \{I\}.$$

Stage 2: body preserves I. Let the body be $d \equiv i := i + 1;\ s := s + i$. We must prove:

$$\vdash \{I \wedge i < n\}\ d\ \{I\}.$$

Use (Seq) with an intermediate assertion R after $i := i+1$. A convenient choice is to "update" the summation index:

$$R \triangleq (1 \leq i \leq n)\ \wedge\ \left(s = \tfrac{(i-1)i}{2}\right).$$

This R expresses that after incrementing i, s is still the sum up to $i - 1$.

First subcommand: prove $\vdash \{I \wedge i < n\}\ i := i + 1\ \{R\}$. By (Assign), it suffices to show:

$$(I \wedge i < n) \Rightarrow R[i \leftarrow i + 1].$$

Compute:

$$R[i \leftarrow i + 1] \equiv \big(1 \leq i + 1 \leq n\big) \wedge \left(s = \tfrac{(i+1-1)(i+1)}{2}\right) \equiv \big(0 \leq i < n\big) \wedge \left(s = \tfrac{i(i+1)}{2}\right).$$

This is implied by $I \wedge i < n$.

Second subcommand: prove $\vdash \{R\}\ s := s + i\ \{I\}$. By (Assign), it suffices to show:

$$R \Rightarrow I[s \leftarrow s + i].$$

Compute:

$$I[s \leftarrow s + i] \equiv (0 \leq i \leq n) \wedge \left(s + i = \tfrac{i(i+1)}{2}\right).$$

From R we have $1 \leq i \leq n$ and $s = \frac{(i-1)i}{2}$, hence

$$s + i = \tfrac{(i-1)i}{2} + i = \tfrac{(i-1)i+2i}{2} = \tfrac{i(i+1)}{2}.$$

Also $1 \le i$ implies $0 \le i$. Thus $R \Rightarrow I[s \leftarrow s + i]$.

Combining by (Seq), we obtain $\vdash \{I \wedge i < n\}\ d\ \{I\}$.

Stage 3: conclude postcondition at loop exit. By (While):

$$\vdash \{I\}\ \texttt{while}\ (i < n)\ \texttt{do}\ d\ \{I \wedge \neg(i < n)\}.$$

At exit, $\neg(i < n)$ means $i \ge n$. Together with I giving $i \le n$, we obtain $i = n$. Substituting into I yields $s = \frac{n(n+1)}{2}$. Using (Conseq) to weaken the final postcondition, we conclude:

$$\vdash \{I\}\ \texttt{while}\ (i < n)\ \texttt{do}\ d\ \{s = \tfrac{n(n+1)}{2}\}.$$

Finally, combine with initialization using (Seq) to derive:

$$\vdash \{n \ge 0\}\ c\ \{s = \tfrac{n(n+1)}{2}\}.$$

2.4 Soundness of Hoare Logic

We now prove that the proof system above is sound with respect to the operational/relational semantics of Section 1. Soundness means: if a triple is derivable syntactically, then it is valid semantically.

Theorem 2.4 (Soundness of Hoare Logic)**.** *For all assertions P, Q and commands c,*

$$\vdash \{P\}\, c\, \{Q\} \quad \Longrightarrow \quad \models \{P\}\, c\, \{Q\}.$$

Proof. By induction on the derivation of $\vdash \{P\}\, c\, \{Q\}$. We consider each last rule used.

(Skip). We must show $\models \{P\}\, \texttt{skip}\, \{P\}$. Take any σ, σ' with $\sigma \models P$ and $(\sigma, \sigma') \in [\![\texttt{skip}]\!]$. By definition of $[\![\texttt{skip}]\!]$, we have $\sigma' = \sigma$. Hence $\sigma' \models P$.

(Assign). The derivation ends with

$$\vdash \{Q[x \leftarrow a]\}\ x := a\ \{Q\}.$$

We show $\models \{Q[x \leftarrow a]\}\ x := a\ \{Q\}$. Let $\sigma \models Q[x \leftarrow a]$ and suppose $(\sigma, \sigma') \in [\![x := a]\!]$. By the semantics of assignment,

$$\sigma' = \sigma[x \mapsto [\![a]\!](\sigma)].$$

By the defining property of substitution, $\sigma \models Q[x \leftarrow a]$ implies $\sigma' \models Q$. This proves validity.

(Seq). The derivation ends with:

$$\frac{\vdash \{P\}\ c_1\ \{R\} \qquad \vdash \{R\}\ c_2\ \{Q\}}{\vdash \{P\}\ c_1; c_2\ \{Q\}}.$$

By IH, $\models \{P\}\, c_1\, \{R\}$ and $\models \{R\}\, c_2\, \{Q\}$. Take $\sigma \models P$ and $(\sigma, \sigma') \in [\![c_1; c_2]\!]$. By relational composition, there exists σ_1 such that $(\sigma, \sigma_1) \in [\![c_1]\!]$ and $(\sigma_1, \sigma') \in [\![c_2]\!]$. From $\models \{P\}\, c_1\, \{R\}$ we get $\sigma_1 \models R$; then from $\models \{R\}\, c_2\, \{Q\}$ we get $\sigma' \models Q$. Hence $\models \{P\}\, c_1; c_2\, \{Q\}$.

(If). The derivation ends with:

$$\frac{\vdash \{P \wedge b\}\ c_1\ \{Q\} \qquad \vdash \{P \wedge \neg b\}\ c_2\ \{Q\}}{\vdash \{P\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{Q\}}.$$

By IH, both premises are semantically valid. Take $\sigma \models P$ and $(\sigma, \sigma') \in [\![\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2]\!]$. By the big-step semantics, either $[\![b]\!](\sigma) = \mathsf{tt}$ and $(\sigma, \sigma') \in [\![c_1]\!]$, or $[\![b]\!](\sigma) = \mathsf{ff}$ and $(\sigma, \sigma') \in [\![c_2]\!]$. In the first case, $\sigma \models P \wedge b$, hence $\sigma' \models Q$ by IH. In the second case, $\sigma \models P \wedge \neg b$, hence $\sigma' \models Q$ by IH. Thus the conditional rule is sound.

(Conseq). The derivation ends with:

$$\frac{P' \Rightarrow P \qquad \vdash \{P\}\ c\ \{Q\} \qquad Q \Rightarrow Q'}{\vdash \{P'\}\ c\ \{Q'\}}.$$

By IH, $\models \{P\}\, c\, \{Q\}$. Let $\sigma \models P'$ and $(\sigma, \sigma') \in [\![c]\!]$. Since $P' \Rightarrow P$, we have $\sigma \models P$, so $\sigma' \models Q$ by validity. Since $Q \Rightarrow Q'$, we get $\sigma' \models Q'$. Hence $\models \{P'\}\, c\, \{Q'\}$.

(While). The derivation ends with:

$$\frac{\vdash \{I \wedge b\}\ c\ \{I\}}{\vdash \{I\}\ \texttt{while}\ b\ \texttt{do}\ c\ \{I \wedge \neg b\}}.$$

By IH, $\models \{I \wedge b\}\, c\, \{I\}$. We show $\models \{I\}\, \texttt{while}\ b\ \texttt{do}\ c\, \{I \wedge \neg b\}$.

Take any σ, σ' such that $\sigma \models I$ and $\langle \texttt{while}\ b\ \texttt{do}\ c, \sigma\rangle \Downarrow \sigma'$. We prove $\sigma' \models I \wedge \neg b$ by induction on the given big-step derivation of the loop.

Base (guard false). If the derivation ends with the rule

$$\frac{[\![b]\!](\sigma) = \mathsf{ff}}{\langle \texttt{while}\ b\ \texttt{do}\ c, \sigma\rangle \Downarrow \sigma},$$

then $\sigma' = \sigma$. Since $\sigma \models I$ and $[\![b]\!](\sigma) = \mathsf{ff}$, we have $\sigma' \models I \wedge \neg b$.

Step (guard true). Otherwise the derivation ends with

$$\frac{[\![b]\!](\sigma) = \mathsf{tt} \quad \langle c, \sigma\rangle \Downarrow \sigma_1 \quad \langle \texttt{while}\ b\ \texttt{do}\ c, \sigma_1\rangle \Downarrow \sigma_2}{\langle \texttt{while}\ b\ \texttt{do}\ c, \sigma\rangle \Downarrow \sigma_2}.$$

Here $\sigma' = \sigma_2$. Since $\sigma \models I$ and the guard is true, we have $\sigma \models I \wedge b$. Also $(\sigma, \sigma_1) \in [\![c]\!]$. By $\models \{I \wedge b\}\, c\, \{I\}$ we obtain $\sigma_1 \models I$. Now apply the induction hypothesis to the sub-derivation $\langle \texttt{while}\ b\ \texttt{do}\ c, \sigma_1\rangle \Downarrow \sigma_2$ to conclude $\sigma_2 \models I \wedge \neg b$. Thus $\sigma' \models I \wedge \neg b$.

This establishes $\models \{I\}\, \texttt{while}\ b\ \texttt{do}\ c\, \{I \wedge \neg b\}$.

All cases preserve semantic validity, so the theorem follows. □

Remark. Soundness is the foundational guarantee that Hoare proofs are trustworthy with respect to the chosen semantics. In later chapters, we will mirror this pattern for incorrectness logic and for quantum program logics: define a semantics first, state a proof system, and then prove that every rule is semantics-preserving.

2.5 Completeness of Hoare Logic

Soundness alone does not guarantee that the proof system is expressive enough: a semantically valid triple might still be non-derivable due to limitations of the proof rules or the assertion language. Completeness addresses this concern.

Hoare Logic reasons about program states using an *assertion language* (e.g. first-order arithmetic over integers). Since this assertion language is necessarily limited in expressive power, one cannot hope for an *unconditional* completeness theorem: if the assertion language is too weak to express certain semantic properties of programs, then no proof system can derive all valid Hoare triples. This limitation was already observed in early work by Floyd, Hoare, and later formalized by Cook, who showed that full completeness would imply solving undecidable problems.

For this reason, completeness results for Hoare Logic are stated in a *relative* form. A proof system is said to be relatively complete if, *assuming* the assertion language is expressive enough to describe weakest preconditions (or equivalently, strongest postconditions) and to prove all valid implications between assertions, then every semantically valid Hoare triple is derivable in the proof system. In other words, the only source of incompleteness comes from the expressive limitations of the assertion language, not from the rules of Hoare Logic itself. A standard reference for this result is Cook's relative completeness theorem for Hoare logic.

Expressiveness assumptions on the assertion logic Let $\vdash_{\mathcal{A}} \varphi$ denote provability in the underlying assertion logic $\mathcal{A}$ (e.g. Peano arithmetic, or any sufficiently strong theory). We assume:

(1) (*Soundness of assertions*) If $\vdash_{\mathcal{A}} \varphi$ then φ is true in the intended model of states.

(2) (*Sufficient expressiveness*) For each command c and assertion Q, the predicate $\mathrm{wlp}(c, Q)$ is expressible in the assertion language. That is, there exists a formula (assertion) also written $\mathrm{wlp}(c, Q)$ such that for all states σ,

$$\sigma \models \mathrm{wlp}(c, Q) \quad \Longleftrightarrow \quad \forall \sigma'.\ (\sigma, \sigma') \in [\![c]\!] \Rightarrow \sigma' \models Q.$$

(For a while-language, this is a non-trivial assumption; it is exactly what relative completeness isolates.)

(3) (*Ability to prove needed implications*) Whenever $P \subseteq P'$ as sets of states, the implication $P \Rightarrow P'$ is derivable in $\mathcal{A}$, and similarly for postconditions.

The next theorem is the standard form of relative completeness for partial correctness of a while-language.

Theorem 2.5 (Relative completeness of Hoare Logic)**.** *Assume the assertion logic $\mathcal{A}$ can express and reason about weakest liberal preconditions as in Section 2.5. Then for all assertions P, Q and commands c,*

$$\models \{P\}\, c\, \{Q\} \quad \Longrightarrow \quad \vdash \{P\}\, c\, \{Q\}.$$

Proof. Fix P, Q, c and assume $\models \{P\}\, c\, \{Q\}$. By the equivalence from Section 1,

$$\models \{P\}\, c\, \{Q\} \quad \Longleftrightarrow \quad P \subseteq \mathrm{wlp}(c, Q).$$

By the "ability to prove needed implications" assumption on $\mathcal{A}$, we may treat $P \Rightarrow \mathrm{wlp}(c, Q)$ as an available (provable) implication.

It therefore suffices to show the following *key lemma*:

$$\vdash \{\mathrm{wlp}(c, Q)\}\, c\, \{Q\}.$$

Indeed, once this lemma holds, we obtain the desired triple $\vdash \{P\}\, c\, \{Q\}$ by a single application of (Conseq):

$$\frac{P \Rightarrow \mathrm{wlp}(c, Q) \qquad \vdash \{\mathrm{wlp}(c, Q)\}\, c\, \{Q\} \qquad Q \Rightarrow Q}{\vdash \{P\}\, c\, \{Q\}}.$$

Proof of the lemma. We prove $\vdash \{\mathrm{wlp}(c, Q)\}\, c\, \{Q\}$ by structural induction on the command c.

- $c = \texttt{skip}$**.** We have $\mathrm{wlp}(\texttt{skip}, Q) = Q$. Then $\vdash \{Q\}\, \texttt{skip}\, \{Q\}$ by (Skip).
- $c = (x := a)$**.** One shows that $\mathrm{wlp}(x := a, Q)$ is equivalent to the substitution $Q[x \leftarrow a]$. Formally, for all σ,

  $$\sigma \models \mathrm{wlp}(x := a, Q) \Longleftrightarrow \sigma[x \mapsto [\![a]\!](\sigma)] \models Q \Longleftrightarrow \sigma \models Q[x \leftarrow a].$$

  Hence $\mathrm{wlp}(x := a, Q) \equiv Q[x \leftarrow a]$, and $\vdash \{Q[x \leftarrow a]\}\ x := a\ \{Q\}$ follows from (Assign).
- $c = (c_1; c_2)$**.** Let $R \triangleq \mathrm{wlp}(c_2, Q)$. Then, by standard properties of wlp,

  $$\mathrm{wlp}(c_1; c_2, Q) = \mathrm{wlp}(c_1, \mathrm{wlp}(c_2, Q)) = \mathrm{wlp}(c_1, R).$$

  By IH applied to c_1 with postcondition R, we have $\vdash \{\mathrm{wlp}(c_1, R)\}\ c_1\ \{R\}$. By IH applied to c_2 with postcondition Q, we have $\vdash \{R\}\ c_2\ \{Q\}$. Then (Seq) yields

  $$\vdash \{\mathrm{wlp}(c_1; c_2, Q)\}\ c_1; c_2\ \{Q\}.$$

- $c = \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2$. For partial correctness, wlp satisfies:

$$\mathrm{wlp}(\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, Q) = (b \wedge \mathrm{wlp}(c_1, Q)) \ \vee\ (\neg b \wedge \mathrm{wlp}(c_2, Q)).$$

  By IH we have $\vdash \{\mathrm{wlp}(c_1, Q)\}\, c_1\, \{Q\}$ and $\vdash \{\mathrm{wlp}(c_2, Q)\}\, c_2\, \{Q\}$. Using (Conseq) we strengthen preconditions to $b \wedge \mathrm{wlp}(c_1, Q)$ and $\neg b \wedge \mathrm{wlp}(c_2, Q)$, respectively, and then apply (If) to obtain

$$\vdash \{\mathrm{wlp}(\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, Q)\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{Q\}.$$

- $c = \texttt{while}\ b\ \texttt{do}\ d$. Let

$$I \triangleq \mathrm{wlp}(\texttt{while}\ b\ \texttt{do}\ d, Q).$$

  For partial correctness, wlp of a while-loop is characterized as the *greatest fixed point* (why?) of the monotone functional

$$F(X) \triangleq (\neg b \Rightarrow Q) \ \wedge\ (b \Rightarrow \mathrm{wlp}(d, X)).$$

  (See the fixed-point characterization of wlp for `while` in Section 1.) In particular, I satisfies the unfolding law

$$I \ \Rightarrow\ F(I),$$

  so we may extract the two consequences:

$$I \wedge \neg b \Rightarrow Q, \tag{1}$$
$$I \wedge b \Rightarrow \mathrm{wlp}(d, I). \tag{2}$$

  Now we derive the Hoare triple for the loop.

  By the induction hypothesis applied to the body d with postcondition I, we have

$$\vdash \{\mathrm{wlp}(d, I)\}\ d\ \{I\}.$$

  Using (Conseq) together with (2), we strengthen the precondition to obtain

$$\vdash \{I \wedge b\}\ d\ \{I\}.$$

  From the previous line, (While) yields

$$\vdash \{I\}\ \texttt{while}\ b\ \texttt{do}\ d\ \{I \wedge \neg b\}.$$

  Finally, by (Conseq) and (1), we weaken the postcondition:

$$\vdash \{I\}\ \texttt{while}\ b\ \texttt{do}\ d\ \{Q\}.$$

  Since $I \equiv \mathrm{wlp}(\texttt{while}\ b\ \texttt{do}\ d, Q)$ by definition, this is exactly the required instance of the key lemma for the while-case.

Therefore, in all cases, $\vdash \{\mathrm{wlp}(c, Q)\}\, c\, \{Q\}$ holds, completing the key lemma and hence the theorem. □

Completeness via strongest postconditions. Using the post-image transformer $\mathrm{post}(c)(P)$, partial correctness validity is $\mathrm{post}(c)(P) \subseteq Q$. If the assertion language can express $\mathrm{post}(c)(P)$ (as a strongest postcondition) and reason about set inclusions, then completeness can alternatively be presented as: whenever $\mathrm{post}(c)(P) \subseteq Q$ holds semantically, one can derive $\vdash \{P\}\, c\, \{Q\}$ by expressing the intermediate strongest postconditions of subcommands and applying (Seq) and (Conseq) accordingly. This is essentially dual to the wlp-based proof above: wlp supports backward reasoning, while post supports forward reasoning.

2.6 Why Hoare Logic does not directly support bug-catching

Hoare Logic is a logic of *universal* guarantees: it proves that *all* terminating executions from P end in states satisfying Q. Bug-catching, however, is often an *existential* task: show that there exists an execution (or there exists a reachable bad state) witnessing a bug.

The mismatch: negating a Hoare triple. It is tempting to think that to show a program can reach a "bad" state $\neg Q$ from a precondition P, one could try to negate a correctness claim. However,

$$\neg\big(\models \{P\}\, c\, \{Q\}\big) \quad \not\equiv \quad \models \{P\}\, c\, \{\neg Q\}.$$

To see why, expand the semantics:

$$\models \{P\}\, c\, \{Q\} \iff \forall \sigma, \sigma'.\, \big(\sigma \models P \wedge (\sigma, \sigma') \in [\![c]\!]\big) \Rightarrow \sigma' \models Q.$$

Negating this statement yields:

$$\neg(\models \{P\}\, c\, \{Q\}) \iff \exists \sigma, \sigma'.\, \big(\sigma \models P \wedge (\sigma, \sigma') \in [\![c]\!] \wedge \sigma' \not\models Q\big).$$

Thus, $\neg(\models \{P\}\, c\, \{Q\})$ means *there exists* a terminating run from some P-state reaching a state that violates Q. This is precisely an existential reachability statement.

But $\models \{P\}\, c\, \{\neg Q\}$ means:

$$\forall \sigma, \sigma'.\, \big(\sigma \models P \wedge (\sigma, \sigma') \in [\![c]\!]\big) \Rightarrow \sigma' \models \neg Q,$$

i.e. *all* terminating executions from P end in states violating Q. This is a much stronger statement and is rarely what one wants when looking for bugs.

Set-theoretic view. Using the post-image operator:

$$\models \{P\}\, c\, \{Q\} \Longleftrightarrow \mathrm{post}(c)(P) \subseteq Q.$$

Negating it gives

$$\neg(\mathrm{post}(c)(P) \subseteq Q) \Longleftrightarrow \exists \sigma' \in \mathrm{post}(c)(P).\, \sigma' \notin Q \Longleftrightarrow \mathrm{post}(c)(P) \cap (\Sigma \setminus Q) \neq \emptyset,$$

which again expresses existence of a reachable bad state. In contrast,

$$\models \{P\}\, c\, \{\neg Q\} \Longleftrightarrow \mathrm{post}(c)(P) \subseteq (\Sigma \setminus Q),$$

which says all reachable states are bad.

Incorrectness Logic is designed to capture existential reachability *directly* in the logic, without encoding existential claims indirectly via negation of universal ones. In particular, the core incorrectness judgment will validate statements of the form

$$Q \subseteq \mathrm{post}(c)(P),$$

meaning every state in Q is genuinely reachable from some state in P. This under-approximation style supports bug-catching with "no false positives": deriving such a judgment certifies the existence of concrete executions that reach the asserted outcomes.

In the quantum setting, a similar mismatch becomes even more pronounced because negation of a correctness statement interacts subtly with probabilistic branching and mixed states. This motivates a dedicated quantum incorrectness logic rather than relying on negated quantum Hoare-style specifications.