Section 4: Incorrectness Logic

3 Incorrectness Logic (IL)

IL is a program logic tailored for *bug-catching*. Where classical verification aims to establish that programs satisfy desired properties on *all* executions, bug-finding and testing are inherently *existential*: one wants evidence that there exists an execution leading to a problematic outcome. Traditional Hoare Logic (HL) is extremely successful for *absence-of-bugs* reasoning, but its semantics are universal, and therefore it does not directly support the kind of "reach a bad state" claims that arise in debugging and test generation.

The modern motivation for IL comes from the observation that many automated analysis tools (symbolic execution, bounded exploration, fuzzing-guided search, abstract interpretation tuned for bug-finding) naturally produce *under-approximations*: they explore some executions and return concrete reachable states, but they do not claim completeness. IL provides a principled logical framework for this style of reasoning: instead of proving that *all* reachable outcomes satisfy a specification, it proves that *some* specified outcomes are genuinely reachable. This makes IL particularly well-suited for certifying discovered counterexamples and for composing bug-finding results across program structure.

Historically, IL was introduced in the context of *under-approximate* reasoning and compositional bug-catching. It puts reachability on an equal footing with correctness: both are interpreted using the same underlying program semantics, but they reverse the direction of inclusion. As a consequence, IL supports "no false positives" reasoning: if an IL triple is derivable, then there is a real terminating execution witnessing the claimed outcome set.

3.1 Extension of Cmd and its semantics

In Hoare Logic, a triple $\{p\}\, c\, \{q\}$ is an *over-approximate* claim: starting from any state in p, every terminating run ends in a state in q. IL is the dual viewpoint: IL asserts the *reachability* of a described set of outcomes, i.e. an *under-approximation* of what the program can do.

Let Σ be the set of program states. To align with later quantum incorrectness logics (which track outcomes/exits explicitly), we model commands with at least two exits:

$$\epsilon \in \{\mathsf{ok}, \mathsf{er}\}.$$

For each command c, its denotation is given by two relations

$$\llbracket c \rrbracket_{\mathsf{ok}} \subseteq \Sigma \times \Sigma, \qquad \llbracket c \rrbracket_{\mathsf{er}} \subseteq \Sigma \times \Sigma,$$

or sometimes written as:

$$\llbracket c \rrbracket \subseteq \Sigma \times (\Sigma \times \{\mathsf{ok}, \mathsf{er}\}),$$

where $(\sigma, \sigma') \in \llbracket c \rrbracket_{\mathsf{ok}}$ means: *there is a terminating execution of c from σ to σ' that exits normally,* and similarly for er (abnormal/error exit).

To state the proof system of Incorrectness Logic in a compositional way, we extend the Cmd grammar with the following primitive constructs: an explicit error command, a path-filtering assume, nondeterministic assignment, local scoping, nondeterministic choice, and Kleene iteration. These primitives are convenient because standard constructs such as `if`, `while`, and assert can be encoded using them.

Definition 3.1 (Extended command language for IL)**.** Let $x \in \mathsf{Var}$, $a \in \mathsf{AExp}$, and $B \in \mathsf{BExp}$. The (extended) commands $C \in \mathsf{Cmd}$ are generated by:

$$\begin{aligned} C ::= \quad & \mathsf{skip} \mid x := a \mid x := \mathsf{nondet}() \\ & \mid C_1; C_2 \mid C_1 + C_2 \mid C^{\star} \\ & \mid \mathsf{assume}\, B \mid \mathsf{error}() \mid \mathsf{local}\ x.\, C. \end{aligned}$$

Here $C_1 + C_2$ is nondeterministic choice, $C^\star$ denotes Kleene iteration (zero or more repetitions), $\mathsf{assume}\, B$ filters executions to those satisfying B, $\mathsf{error}()$ forces abnormal exit, and $\mathsf{local}\ x.\, C$ introduces a fresh local variable scope for x within C.

$$\begin{aligned}
\mathsf{while}\ B\ \mathsf{do}\ C &\triangleq (\mathsf{assume}(B); C)^\star; \mathsf{assume}(\neg B),\\
\mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 &\triangleq (\mathsf{assume}(B); C_1) + (\mathsf{assume}(\neg B); C_2),\\
\mathsf{assert}(B) &\triangleq \mathsf{assume}(B) + (\mathsf{assume}(\neg B); \mathsf{error}()).
\end{aligned}$$

Semantics of IL commands. Fix a set of states Σ, and let $\epsilon \in \{\mathsf{ok}, \mathsf{er}\}$ range over exits. Each command C denotes two relations $[\![C]\!]_\epsilon \subseteq \Sigma \times \Sigma$. Boolean expressions are interpreted by a total function $[\![B]\!] : \Sigma \to \{\mathsf{tt}, \mathsf{ff}\}$.

We use relational composition: for $R, S \subseteq \Sigma \times \Sigma$,

$$R\, ;S \;\triangleq\; \{(\sigma, \sigma'') \mid \exists \sigma'.\, (\sigma, \sigma') \in R \wedge (\sigma', \sigma'') \in S\}.$$

$$\begin{aligned}
[\![\mathsf{skip}]\!]_{\mathsf{ok}} &\triangleq \{(\sigma, \sigma) \mid \sigma \in \Sigma\}, & [\![\mathsf{skip}]\!]_{\mathsf{er}} &\triangleq \varnothing,\\
[\![\mathsf{error}()]\!]_{\mathsf{ok}} &\triangleq \varnothing, & [\![\mathsf{error}()]\!]_{\mathsf{er}} &\triangleq \{(\sigma, \sigma) \mid \sigma \in \Sigma\},\\
[\![\mathsf{assume}\, B]\!]_{\mathsf{ok}} &\triangleq \{(\sigma, \sigma) \mid [\![B]\!](\sigma) = \mathsf{tt}\}, & [\![\mathsf{assume}\, B]\!]_{\mathsf{er}} &\triangleq \varnothing,\\
[\![C_1 + C_2]\!]_{\mathsf{ok}} &\triangleq [\![C_1]\!]_{\mathsf{ok}} \cup [\![C_2]\!]_{\mathsf{ok}}, & [\![C_1 + C_2]\!]_{\mathsf{er}} &\triangleq [\![C_1]\!]_{\mathsf{er}} \cup [\![C_2]\!]_{\mathsf{er}},\\
[\![C_1; C_2]\!]_{\mathsf{ok}} &\triangleq [\![C_1]\!]_{\mathsf{ok}}\, ; [\![C_2]\!]_{\mathsf{ok}}, & [\![C_1; C_2]\!]_{\mathsf{er}} &\triangleq \big([\![C_1]\!]_{\mathsf{ok}}\, ; [\![C_2]\!]_{\mathsf{er}}\big) \;\cup\; [\![C_1]\!]_{\mathsf{er}},\\
[\![C^\star]\!]_{\mathsf{ok}} &\triangleq \bigcup_{i \in \mathbb{N}} [\![C^i]\!]_{\mathsf{ok}}, & [\![C^\star]\!]_{\mathsf{er}} &\triangleq \bigcup_{i \in \mathbb{N}} [\![C^i]\!]_{\mathsf{er}}.
\end{aligned}$$

where the iterates are defined by $C^0 \triangleq \mathsf{skip}, \quad C^{i+1} \triangleq C; C^i$.

For $\sigma \in \Sigma$ and $v \in \mathsf{Val}$, write $\sigma[x \mapsto v]$ for the store that agrees with σ everywhere except at x.

$$\begin{aligned}
[\![x := e]\!]_{\mathsf{ok}} &\triangleq \{(\sigma,\ \sigma[x \mapsto [\![e]\!](\sigma)]) \mid \sigma \in \Sigma\},\\
[\![x := e]\!]_{\mathsf{er}} &\triangleq \varnothing,
\end{aligned}$$

$$\begin{aligned}
[\![x := \mathsf{nondet}()]\!]_{\mathsf{ok}} &\triangleq \{(\sigma,\ \sigma[x \mapsto v]) \mid \sigma \in \Sigma,\ v \in \mathsf{Val}\},\\
[\![x := \mathsf{nondet}()]\!]_{\mathsf{er}} &\triangleq \varnothing,
\end{aligned}$$

$$[\![\mathsf{local}\ x.\, C]\!]_\epsilon \triangleq \{(\sigma[x \mapsto v],\ \sigma'[x \mapsto v]) \mid (\sigma, \sigma') \in [\![C]\!]_\epsilon,\ v \in \mathsf{Val}\}.$$

Post-image operator. For any relation $R \subseteq \Sigma \times \Sigma$ and predicate (set of states) $p \subseteq \Sigma$, the post-image is:

$$\mathrm{post}(R)(p) \;\triangleq\; \{\sigma' \in \Sigma \mid \exists \sigma \in p.\ (\sigma, \sigma') \in R\}.$$

We abbreviate $\mathrm{post}_\epsilon(c)(p) \triangleq \mathrm{post}([\![c]\!]_\epsilon)(p)$, and may use them interchangably.

IL triples (under-approximate). An IL triple has the form $[p]\,c\,[\epsilon : q]$, and is *semantically valid*

$$\models [p]\,c\,[\epsilon : q] \quad\Longleftrightarrow\quad q \subseteq \mathrm{post}_\epsilon(c)(p). \tag{1}$$

Intuitively:

> "Every state described by q is actually reachable (via exit ϵ) from *some* initial state in p."

When both exits matter, we also write a *quadruple*:

$$[p]\,c\,[\mathsf{ok} : q]\,[\mathsf{er} : r] \quad \text{as shorthand for} \quad [p]\,c\,[\mathsf{ok} : q] \ \wedge\ [p]\,c\,[\mathsf{er} : r].$$

Reachability characterization. Equation (1) is equivalent to an explicit reachability property:

$$\models [p]\,c\,[\epsilon : q] \quad\Longleftrightarrow\quad \forall \sigma_q \in q.\ \exists \sigma_p \in p.\ (\sigma_p, \sigma_q) \in \llbracket c \rrbracket_\epsilon. \tag{2}$$

This makes the existential nature of IL explicit: IL is about *witnessing* reachable outcomes.

No false positivity. Because IL requires q to be a subset of actually reachable final states, any state included in q comes with a semantic guarantee that it can happen in a real execution (from some p-state). This is the logical core of the "no false positives" philosophy for bug-catching: IL can miss bugs (false negatives) if the reasoning is too weak, but it does not invent unreachable outcomes.

3.2 Duality

Hoare Logic and Incorrectness Logic can be presented as two inequalities around the same semantic transformer post. For a fixed command c and exit ϵ, the post-image $\mathrm{post}_\epsilon(c)$ is a predicate transformer:

$$\mathrm{post}_\epsilon(c) : \mathscr{P}(\Sigma) \to \mathscr{P}(\Sigma).$$

Both HL and IL speak about how a candidate postcondition compares to the exact post-image:

$$\begin{aligned} \models \{p\}\,c\,\{\epsilon : q\} &\iff \mathrm{post}_\epsilon(c)(p) \subseteq q, \\ \models [p]\,c\,[\epsilon : q] &\iff q \subseteq \mathrm{post}_\epsilon(c)(p). \end{aligned}$$

Thus, HL and IL are dual directions of approximation:

$$\begin{tikzcd} \mathscr{P}(\Sigma) \arrow[r, "\mathrm{post}_\epsilon(c)"] \arrow[r, dashed, bend left, "\{-\}\,c\,\{\epsilon{:}-\}"] \arrow[r, dashed, bend right, "{[-]\,c\,[\epsilon{:}-]}"'] & \mathscr{P}(\Sigma) \arrow[r, shift left, "\subseteq"] \arrow[r, shift right, "\supseteq"'] & \mathscr{P}(\Sigma) \end{tikzcd}$$

One immediate consequences worth recording is that the exact $\mathrm{post}_\epsilon(c)(p)$ is simultaneously: the *strongest* over-approximate postcondition (smallest q with post $\subseteq q$), and the *weakest* under-approximate postcondition (largest q with $q \subseteq$ post).

Symmetry principles: ($\vee/\wedge$) and consequence Because IL talks about subset *into* the exact post-image, conjunction of IL triples corresponds to *disjunction* of result predicates:

$$\models [p]\,c\,[\epsilon : q_1] \wedge\ \models [p]\,c\,[\epsilon : q_2] \quad\Longleftrightarrow\quad \models [p]\,c\,[\epsilon : q_1 \vee q_2]. \tag{3}$$

In contrast, HL combines postconditions by conjunction:

$$\models \{p\}\,c\,\{\epsilon : q_1\} \wedge\ \models \{p\}\,c\,\{\epsilon : q_2\} \quad\Longleftrightarrow\quad \models \{p\}\,c\,\{\epsilon : q_1 \wedge q_2\}.$$

For IL, enlarging the precondition and shrinking (strengthening) the result is sound:

$$\frac{p \subseteq p' \qquad \models [p]\, c\, [\epsilon : q] \qquad q' \subseteq q}{\models [p']\, c\, [\epsilon : q']}. \tag{4}$$

Read: we may *weaken* the presumption (allow more starts) and *strengthen* the result (commit to fewer, more precise reachable outcomes).

For HL the directions flip:

$$\frac{p' \subseteq p \qquad \models \{p\}\, c\, \{\epsilon : q\} \qquad q \subseteq q'}{\models \{p'\}\, c\, \{\epsilon : q'\}}.$$

From (4), if $q_1 \subseteq q_1 \vee q_2$ then

$$\models [p]\, c\, [\epsilon : q_1 \vee q_2] \implies \models [p]\, c\, [\epsilon : q_1].$$

This is a hallmark of under-approximation: one may focus on fewer than all paths, as long as one remains truthful about reachability.

3.3 Proof system

We now give an axiomatic proof system for IL triples. As in the semantic presentation above, assertions $p, q, \ldots$ are treated extensionally as subsets of Σ; hence we freely use set-theoretic reasoning (subset inclusion) as an "oracle" side condition in rules. We write ff for the empty predicate $\varnothing$, and use $\wedge, \vee, \neg$ for intersection, union, and complement (relative to Σ). A rule is *sound* if it preserves semantic validity in (1). Since assertions are treated extensionally as sets of states, $\exists x.\, p$ means *forget the value of* x (project it away):

$$\exists x.\, p \;\triangleq\; \{\sigma \in \Sigma \mid \exists v \in \mathsf{Val}.\ \sigma[x \mapsto v] \in p\}.$$

So $\sigma \models \exists x.\, p$ iff there exists some choice of x that makes p true.

$$\frac{}{[p]\, C\, [\epsilon : \mathsf{ff}]}\ \textsc{Empty} \qquad \frac{p \subseteq p' \qquad [p]\, C\, [\epsilon : q] \qquad q' \subseteq q}{[p']\, C\, [\epsilon : q']}\ \textsc{Consequence}$$

$$\frac{[p_1]\, C\, [\epsilon : q_1] \qquad [p_2]\, C\, [\epsilon : q_2]}{[p_1 \vee p_2]\, C\, [\epsilon : q_1 \vee q_2]}\ \textsc{Disjunction}$$

$$\frac{}{[p]\, \mathsf{skip}\, [\mathsf{ok} : p]\, [\mathsf{er} : \mathsf{ff}]}\ \textsc{Skip} \qquad \frac{}{[p]\, \mathsf{error}()\, [\mathsf{ok} : \mathsf{ff}]\, [\mathsf{er} : p]}\ \textsc{Error}$$

$$\frac{}{[p]\, \mathsf{assume}\, B\, [\mathsf{ok} : p \wedge B]\, [\mathsf{er} : \mathsf{ff}]}\ \textsc{Assume} \qquad \frac{[p]\, C_i\, [\epsilon : q]}{[p]\, (C_1 + C_2)\, [\epsilon : q]}\ \textsc{Choice}\ (i \in \{1, 2\})$$

$$\frac{[p]\, C_1\, [\mathsf{er} : r]}{[p]\, (C_1; C_2)\, [\mathsf{er} : r]}\ \textsc{Seq-Er} \qquad \frac{[p]\, C_1\, [\mathsf{ok} : q] \qquad [q]\, C_2\, [\epsilon : r]}{[p]\, (C_1; C_2)\, [\epsilon : r]}\ \textsc{Seq-Ok}$$

$$\frac{}{[p]\, C^\star\, [\mathsf{ok} : p]}\ \textsc{Iter-Zero} \qquad \frac{[p]\, (C^\star; C)\, [\epsilon : q]}{[p]\, C^\star\, [\epsilon : q]}\ \textsc{Iter-NonZero}$$

$$\frac{[\, p(n) \wedge \mathsf{nat}(n)\,]\, C\, [\mathsf{ok} :\ p(n+1) \wedge \mathsf{nat}(n)\,]}{[\, p(0)\,]\, C^\star\, [\mathsf{ok} : \exists n.\, p(n) \wedge \mathsf{nat}(n)]}\ \textsc{Back-Var}\ (n \text{ fresh})$$

Iteration. Kleene iteration $C^\star$ can always do *zero* iterations and exit normally, hence Iterate zero says every predicate p is an under-approximate "invariant": starting in p, one way to execute $C^\star$ is to do nothing and end in p.

If you can prove a triple for $C^\star; C$, you have described executions with *at least one* final iteration. The rule then lets you conclude the same result for $C^\star$ itself (because $C^\star$ includes all non-zero iteration counts as well).

A typical reasoning pattern is:

$$\underbrace{[p]\,C^\star\,[\mathsf{ok} : \mathit{frontier}]}_{\text{reachable by some number of ok-iterations}} \qquad \underbrace{[\mathit{frontier}]\,C\,[\mathsf{er} : q]}_{\text{error on the next iteration}} \quad\Rightarrow\quad [p]\,C^\star\,[\mathsf{er} : q],$$

using Sequencing (normal) to get $[p]\,C^\star; C\,[\mathsf{er} : q]$ and then Iterate non-zero.

Backwards Variant. The rule Backwards Variant is IL's main principle for loop reasoning beyond bounded unrolling. It should be compared to classic termination arguments based on a *variant* that decreases on each iteration, but here the "decrease" is seen when reading the reachability claim *backwards.*

The conclusion $[p(0)]\,C^\star\,[\mathsf{ok} : \exists n.\, p(n) \wedge \mathsf{nat}(n)]$ means every final state that satisfies $p(n)$ for some natural n is indeed reachable by running $C^\star$ from *some* start state in $p(0)$.

The premise $[\,p(n) \wedge \mathsf{nat}(n)\,]\,C\,[\mathsf{ok} :\ p(n+1) \wedge \mathsf{nat}(n)\,]$ says every state satisfying $p(n+1)$ has some predecessor in $p(n)$ that reaches it by one C-step.

So, forward execution *increments* the index:

$$p(0) \xrightarrow{C} p(1) \xrightarrow{C} p(2) \xrightarrow{C} \cdots$$

and therefore $C^\star$ can produce $p(n)$ for any finite n.

Why "backwards". The under-approximate reading of IL triples is easier to visualize by thinking in reverse: pick any *final* state σ_f satisfying $p(k)$. To witness the IL triple, we must exhibit a *starting* state σ_0 in $p(0)$ such that $\sigma_0 \xrightarrow{C^\star} \sigma_f$.

The premise allows us to go backwards one step at a time: since $\sigma_f \in p(k)$, the premise guarantees there exists some predecessor $\sigma_{k-1} \in p(k-1)$ that can reach σ_f by one C-step. Repeating this k times yields a chain back to some $\sigma_0 \in p(0)$. Thus the numeric parameter decreases when searching for predecessors:

$$p(k) \rightsquigarrow p(k-1) \rightsquigarrow \cdots \rightsquigarrow p(0).$$

The side condition "n fresh" (not free in C nor in the schema $p(\cdot)$) ensures this index is a logical counter, not something the program can modify or depend on.

Why this matches IL (and why it does not force total termination). The conclusion does *not* say that all runs terminate. It only says that enough terminating runs exist to cover every state in the claimed result set. Other runs may diverge; IL remains sound because it never claims reachability of states that cannot be reached by a terminating run.

Example. Let C be $x := x + 1$, and let $p(n)$ be the predicate $x = n$. Then the premise states:

$$[x = n \wedge \mathsf{nat}(n)]\ x := x + 1\ [\mathsf{ok} :\ x = n + 1 \wedge \mathsf{nat}(n)],$$

so Backwards Variant yields:

$$[x = 0]\ (x := x + 1)^\star\ [\mathsf{ok} : \exists n.\ x = n \wedge \mathsf{nat}(n)],$$

i.e. from $x = 0$ we can reach any natural value of x by iterating the increment some number of times.

We write $p[e/x]$ for substitution of expression e for x in an assertion, and Free(r) for the (semantic) set of free variables of an assertion r (those it depends on). We write Mod(C) for the set of program variables that C may assign.

$$\frac{}{[p]\ x := e\ [\mathsf{ok}:\ \exists x'.\ p[x'/x]\ \wedge\ x = e[x'/x]]\ [\mathsf{er}:\mathsf{ff}]}\ \textsc{Assignment}$$

$$\frac{}{[p]\ x := \mathsf{nondet}()\ [\mathsf{ok}:\ \exists x'.\ p\ ]\ [\mathsf{er}:\mathsf{ff}]}\ \textsc{Nondet Assignment}$$

$$\frac{[p]\,C\,[\epsilon:q] \qquad \mathrm{Mod}(C)\cap\mathrm{Free}(f)=\varnothing}{[p\wedge f]\,C\,[\epsilon:q\wedge f]}\ \textsc{Constancy}$$

$$\frac{[p]\ C(y/x)\ [\epsilon:q]}{[p]\ \mathsf{local}\ x.\,C\ [\epsilon:\exists y.\ q]}\ \textsc{Local Variable}\ (y\ \text{fresh})$$

$$\frac{[p]\,C\,[\epsilon:q] \qquad (\mathrm{Free}(e)\cup\{x\})\cap\mathrm{Free}(C)=\varnothing}{([p]\,C\,[\epsilon:q])(e/x)}\ \textsc{Substitution I}$$

$$\frac{[p]\,C\,[\epsilon:q] \qquad y\notin\mathrm{Free}(p,C,q)}{([p]\,C\,[\epsilon:q])(y/x)}\ \textsc{Substitution II}\ (y\ \text{fresh})$$

In Hoare Logic, the assignment axiom is typically backward:

$$\{\,q[e/x]\,\}\ x := e\ \{\,q\,\}.$$

In IL this form is unsound, because it can introduce unreachable states into the *result* (assertion on the right of an IL triple must be a subset of *actually reachable* states). The IL assignment axiom therefore uses Floyd's forward style: it *remembers the old value* of x using an existentially quantified logical variable x'.

Concretely, in

$$[p]\ x := e\ [\mathsf{ok}:\ \exists x'.\ p[x'/x]\ \wedge\ x = e[x'/x]],$$

the conjunct $p[x'/x]$ asserts that the precondition held in the *old* state (with x renamed to x'), while $x = e[x'/x]$ ties the *new* value of x to the evaluation of e in that old state. This enforces that every state described by the postcondition is genuinely realizable by a single assignment step.

For $x :=$ nondet(), the postcondition cannot constrain the new value of x at all. The rule

$$[p]\ x := \mathsf{nondet}()\ [\mathsf{ok}:\exists x'.\,p]$$

says: a final store is reachable iff it agrees with some pre-store satisfying p on all variables except possibly x (the existential x' forgets the old value of x so that constraints on x in p do not incorrectly survive).

If f mentions only variables that C does not modify, then f is invariant along executions of C. Hence we can conjoin f to both sides:

$$[p]C[\epsilon:q]\ \Rightarrow\ [p\wedge f]C[\epsilon:q\wedge f].$$

This rule is essential when adapting a proven summary to a richer calling context: you carry along facts about variables that the command cannot change.

The construct local $x.\,C$ allocates a fresh local x for the duration of C and restores the old value of x afterward. The rule works in two conceptual steps:

(i) *Alpha-rename* the bound program variable x to a fresh variable y inside the body, written $C(y/x)$, to avoid clashes with variables mentioned in the surrounding assertion.

(ii) *Hide* the final value of the local by existentially quantifying it in the post: the outside world cannot observe which y was used internally, only that some value existed.

The existential $\exists y$ is not cosmetic: in IL you cannot freely weaken a postcondition, so you cannot in general derive $\exists y.\, q$ from q by Consequence. The rule therefore builds $\exists y$ into the conclusion.

These are meta-rules for reusing a proved triple in a new context.
Substitution II is safe alpha-renaming of a logical variable: if y is fresh, replacing x by y in the assertions does not change meaning.
Substitution I instantiates a logical variable x with an expression e, but only when C does not mention x nor any variables occurring in e. Intuitively: the program cannot interfere with the meaning of e, so the instantiation is stable.

3.4 Soundness of Incorrectness Logic

In this subsection we prove that the proof rules of Incorrectness Logic are *sound* with respect to the relational semantics introduced earlier. The proof follows the standard rule-by-rule style: each axiom is semantically valid, and each inference rule preserves semantic validity.

Semantic validity of IL triples. Recall that each command C is interpreted by two relations $[\![C]\!]_{\mathsf{ok}}, [\![C]\!]_{\mathsf{er}} \subseteq \Sigma \times \Sigma$, where Σ is the set of classical program states. For a predicate (set of states) $p \subseteq \Sigma$ and a relation $r \subseteq \Sigma \times \Sigma$, define the (forward) post-image operator

$$\mathrm{post}(r)(p) \triangleq \{\sigma' \in \Sigma \mid \exists \sigma \in p.\ (\sigma, \sigma') \in r\}.$$

An IL triple $[p]\, C\, [\mathsf{ok} : q][\mathsf{er} : r]$ is *semantically valid*, written $\models [p]\, C\, [\mathsf{ok} : q][\mathsf{er} : r]$, iff

$$q \subseteq \mathrm{post}([\![C]\!]_{\mathsf{ok}})(p) \qquad \text{and} \qquad r \subseteq \mathrm{post}([\![C]\!]_{\mathsf{er}})(p).$$

(Equivalently, for each exit condition $\epsilon \in \{\mathsf{ok}, \mathsf{er}\}$, $\models [p]\, C\, [\epsilon : q]$ iff $q \subseteq \mathrm{post}([\![C]\!]_{\epsilon})(p)$.)

A characterization lemma. We will use the following equivalent, "witness-style" formulation of under-approximation.

Lemma 3.2 (Characterization of under-approximation)**.** *For any relation $r \subseteq \Sigma \times \Sigma$ and predicates $p, q \subseteq \Sigma$,*

$$q \subseteq \mathrm{post}(r)(p) \quad \Longleftrightarrow \quad \forall \sigma' \in q.\ \exists \sigma \in p.\ (\sigma, \sigma') \in r.$$

Proof. ($\Rightarrow$) If $\sigma' \in q \subseteq \mathrm{post}(r)(p)$, then by definition of post there exists $\sigma \in p$ such that $(\sigma, \sigma') \in r$.

($\Leftarrow$) Suppose $\forall \sigma' \in q. \exists \sigma \in p. (\sigma, \sigma') \in r$. Then every $\sigma' \in q$ belongs to $\mathrm{post}(r)(p)$ by the definition of post, hence $q \subseteq \mathrm{post}(r)(p)$. □

Algebraic facts about post**.** The proofs below repeatedly use two elementary properties.

Lemma 3.3 (post distributes over union and composes over relational composition)**.** *Let $r_1, r_2 \subseteq \Sigma \times \Sigma$ be relations and $p \subseteq \Sigma$ a predicate. Then:*

$$\mathrm{post}(r_1 \cup r_2)(p) = \mathrm{post}(r_1)(p) \cup \mathrm{post}(r_2)(p), \qquad \mathrm{post}(r_1; r_2)(p) = \mathrm{post}(r_2)\big(\mathrm{post}(r_1)(p)\big),$$

Proof. Both equalities follow directly by unfolding the definitions of post, relational union, and relational composition, and rearranging existential quantifiers. □

Theorem 3.4 (Soundness of IL). *Every IL derivation is semantically valid. More precisely, if* $\vdash [p]\, C\, [\mathsf{ok} : q][\mathsf{er} : r]$ *is derivable using the proof rules of IL, then* $\models [p]\, C\, [\mathsf{ok} : q][\mathsf{er} : r]$*. Equivalently, each IL axiom is semantically valid, and each IL inference rule preserves semantic validity.*

Proof. We proceed rule-by-rule, showing that each axiom is valid and each inference rule preserves validity.

empty. The axiom asserts $\vdash [p]\, C\, [\epsilon : \mathsf{ff}]$. Semantically, $\emptyset \subseteq X$ holds for all sets X, hence $\models [p]\, C\, [\epsilon : \mathsf{ff}]$.

$\mathsf{consequence}$. Assume $\models [p]\, C\, [\epsilon : q]$, $p \subseteq p'$, and $q' \subseteq q$. Since $\mathrm{post}([\![C]\!]_\epsilon)$ is monotone in its argument, $\mathrm{post}([\![C]\!]_\epsilon)(p) \subseteq \mathrm{post}([\![C]\!]_\epsilon)(p')$. Thus $q' \subseteq q \subseteq \mathrm{post}([\![C]\!]_\epsilon)(p) \subseteq \mathrm{post}([\![C]\!]_\epsilon)(p')$, i.e., $\models [p']\, C\, [\epsilon : q']$.

$\mathsf{disjunction}$. Assume $\models [p_1]\, C\, [\epsilon : q_1]$ and $\models [p_2]\, C\, [\epsilon : q_2]$. Then $q_1 \subseteq \mathrm{post}([\![C]\!]_\epsilon)(p_1)$ and $q_2 \subseteq \mathrm{post}([\![C]\!]_\epsilon)(p_2)$, hence $q_1 \cup q_2 \subseteq \mathrm{post}([\![C]\!]_\epsilon)(p_1) \cup \mathrm{post}([\![C]\!]_\epsilon)(p_2) = \mathrm{post}([\![C]\!]_\epsilon)(p_1 \cup p_2)$ by Lemma 3.3. Therefore $\models [p_1 \vee p_2]\, C\, [\epsilon : q_1 \vee q_2]$.

skip, error, assume. These are immediate from the semantics:

$$[\![\mathsf{skip}]\!]_{\mathsf{ok}} = \{(\sigma, \sigma) \mid \sigma \in \Sigma\}, \quad [\![\mathsf{skip}]\!]_{\mathsf{er}} = \emptyset,$$

$$[\![\mathsf{error}]\!]_{\mathsf{ok}} = \emptyset, \quad [\![\mathsf{error}]\!]_{\mathsf{er}} = \{(\sigma, \sigma) \mid \sigma \in \Sigma\},$$

$$[\![\mathsf{assume}\ b]\!]_{\mathsf{ok}} = \{(\sigma, \sigma) \mid \sigma \models b\}, \quad [\![\mathsf{assume}\ b]\!]_{\mathsf{er}} = \emptyset.$$

From these, one calculates $\mathrm{post}([\![\cdot]\!]_\epsilon)(p)$ and checks that each corresponding axiom postcondition is a subset of the computed post-image.

$\mathsf{assignment}$. For deterministic assignment $x := e$, the semantic ok-relation is $[\![x := e]\!]_{\mathsf{ok}} = \{(\sigma, \sigma[x \mapsto [\![e]\!](\sigma)]) \mid \sigma \in \Sigma\}$, and $[\![x := e]\!]_{\mathsf{er}} = \emptyset$. The IL assignment axiom produces a postcondition describing the set of all states obtained by updating x according to e from some pre-state in p. By unfolding post and the above relation, the axiom's postcondition is exactly $\mathrm{post}([\![x := e]\!]_{\mathsf{ok}})(p)$, hence is certainly a subset of it.

choice. Let $C = C_1 + C_2$ (nondeterministic choice), with semantics $[\![C]\!]_\epsilon = [\![C_1]\!]_\epsilon \cup [\![C_2]\!]_\epsilon$. Assuming $\models [p]\, C_1\, [\epsilon : q]$, we have $q \subseteq \mathrm{post}([\![C_1]\!]_\epsilon)(p)$. Thus

$$q \subseteq \mathrm{post}([\![C_1]\!]_\epsilon)(p) \cup \mathrm{post}([\![C_2]\!]_\epsilon)(p) = \mathrm{post}([\![C_1]\!]_\epsilon \cup [\![C_2]\!]_\epsilon)(p) = \mathrm{post}([\![C]\!]_\epsilon)(p),$$

so $\models [p]\, (C_1 + C_2)\, [\epsilon : q]$. Similar argument if $\models [p]\, C_2\, [\epsilon : q]$.

$\mathsf{sequencing}$. Let $C = C_1; C_2$. In the relational semantics, normal termination composes, while error short-circuits:

$$[\![C_1; C_2]\!]_{\mathsf{ok}} = [\![C_1]\!]_{\mathsf{ok}}; [\![C_2]\!]_{\mathsf{ok}}, \qquad [\![C_1; C_2]\!]_{\mathsf{er}} = [\![C_1]\!]_{\mathsf{er}} \ \cup \ \big([\![C_1]\!]_{\mathsf{ok}}; [\![C_2]\!]_{\mathsf{er}}\big).$$

Normal sequencing rule. Assume $\models [p]\, C_1\, [\mathsf{ok} : q]$ and $\models [q]\, C_2\, [\epsilon : r]$. Then $q \subseteq \mathrm{post}([\![C_1]\!]_{\mathsf{ok}})(p)$ and $r \subseteq \mathrm{post}([\![C_2]\!]_\epsilon)(q)$. Therefore,

$$r \subseteq \mathrm{post}([\![C_2]\!]_\epsilon)(q) \subseteq \mathrm{post}([\![C_2]\!]_\epsilon)\big(\mathrm{post}([\![C_1]\!]_{\mathsf{ok}})(p)\big) = \mathrm{post}([\![C_1]\!]_\epsilon; [\![C_2]\!]_{\mathsf{ok}})(p) \subseteq \mathrm{post}([\![C_1; C_2]\!]_\epsilon)(p),$$

where we used Lemma 3.3 and the definition of $[\![C_1; C_2]\!]_{\mathsf{ok}}$, and in the $\epsilon = \mathsf{er}$ case we use that $[\![C_1]\!]_{\mathsf{er}}; [\![C_2]\!]_{\mathsf{ok}} \subseteq [\![C_1; C_2]\!]_{\mathsf{er}}$. Hence $\models [p]\, (C_1; C_2)\, [\epsilon : r]$ for this rule's conclusion.

Short-circuit (error-first) sequencing rule. Assume $\models [p]\, C_1\, [\mathsf{er} : r]$. Then $r \subseteq \mathrm{post}([\![C_1]\!]_{\mathsf{er}})(p)$. Since $[\![C_1]\!]_{\mathsf{er}} \subseteq [\![C_1; C_2]\!]_{\mathsf{er}}$, we have $\mathrm{post}([\![C_1]\!]_{\mathsf{er}})(p) \subseteq \mathrm{post}([\![C_1; C_2]\!]_{\mathsf{er}})(p)$. Thus $r \subseteq \mathrm{post}([\![C_1; C_2]\!]_{\mathsf{er}})(p)$, i.e., $\models [p]\, (C_1; C_2)\, [\mathsf{er} : r]$.

$\textsf{iteration}$. Let $C^\star$ denote Kleene iteration. One convenient presentation is via finite unrollings $C^0 \triangleq \textsf{skip}$, $C^{n+1} \triangleq C^n; C$, and:

$$[\![C^\star]\!]_{\textsf{ok}} = \bigcup_{n\geq 0} [\![C^n]\!]_{\textsf{ok}}, \qquad [\![C^\star]\!]_{\textsf{er}} = \bigcup_{n\geq 0} \big([\![C^n]\!]_{\textsf{ok}}; [\![C]\!]_{\textsf{er}}\big).$$

A key semantic equivalence is the "unfolding law":

$$C^\star \;\equiv\; \textsf{skip} \;+\; C^\star; C,$$

meaning equality of both ok and er relations.

Iterate-zero. The rule concludes $[p]\, C^\star\, [\textsf{ok} : p]$. This is sound because $C^0 = \textsf{skip}$ is included among the unrollings of $C^\star$, so $p = \mathrm{post}([\![\textsf{skip}]\!]_{\textsf{ok}})(p) \subseteq \mathrm{post}([\![C^\star]\!]_{\textsf{ok}})(p)$.

Iterate-nonzero. The rule (in the ϵ-generic form) derives $[p]\, C^\star\, [\epsilon : q]$ from $[p]\, (C^\star; C)\, [\epsilon : q]$. This is sound because by the unfolding law, $[\![C^\star]\!]_\epsilon = [\![\textsf{skip}]\!]_\epsilon \cup [\![C^\star; C]\!]_\epsilon$, hence $[\![C^\star; C]\!]_\epsilon \subseteq [\![C^\star]\!]_\epsilon$, so $\mathrm{post}([\![C^\star; C]\!]_\epsilon)(p) \subseteq \mathrm{post}([\![C^\star]\!]_\epsilon)(p)$, and therefore any $q \subseteq \mathrm{post}([\![C^\star; C]\!]_\epsilon)(p)$ is also a subset of $\mathrm{post}([\![C^\star]\!]_\epsilon)(p)$.

Backwards-Variant. This rule is specific to under-approximation: from a premise

$$\vdash\; [p(n) \wedge \textsf{nat}(n)]\, C\, [\textsf{ok} : p(n{+}1) \wedge \textsf{nat}(n)]$$

it concludes

$$\vdash\; [p(0)]\, C^\star\, [\textsf{ok} : \exists n.\, p(n) \wedge \textsf{nat}(n)].$$

To show soundness, let σ be an arbitrary state satisfying $\exists n.\, p(n) \wedge \textsf{nat}(n)$. We must prove that $\sigma \in \mathrm{post}([\![C^\star]\!]_{\textsf{ok}})(p(0))$. Choose a witness $j \in \mathbb{N}$ such that $\sigma \models p(j)$. If $j = 0$, then $\sigma \in p(0)$, so σ is reachable by taking 0 iterations of C, i.e., via the $C^0 = \textsf{skip}$ unrolling of $C^\star$.

If $j > 0$, we use the premise in the *under-approximate direction*: $p(j) \subseteq \mathrm{post}([\![C]\!]_{\textsf{ok}})(p(j{-}1))$. By Lemma 3.2, from $\sigma \in p(j)$ we obtain some $\sigma_{j-1} \in p(j-1)$ with $(\sigma_{j-1}, \sigma) \in [\![C]\!]_{\textsf{ok}}$. If $j - 1 > 0$, we apply the same reasoning to $\sigma_{j-1} \in p(j-1)$ to obtain $\sigma_{j-2} \in p(j-2)$ and $(\sigma_{j-2}, \sigma_{j-1}) \in [\![C]\!]_{\textsf{ok}}$, and so on. After finitely many steps, we obtain a chain

$$\sigma_0 \in p(0), \quad (\sigma_0, \sigma_1) \in [\![C]\!]_{\textsf{ok}}, \quad \ldots, \quad (\sigma_{j-1}, \sigma_j) \in [\![C]\!]_{\textsf{ok}}, \quad \sigma_j = \sigma.$$

This exactly means $(\sigma_0, \sigma) \in [\![C^j]\!]_{\textsf{ok}} \subseteq [\![C^\star]\!]_{\textsf{ok}}$, hence $\sigma \in \mathrm{post}([\![C^\star]\!]_{\textsf{ok}})(p(0))$. Since σ was arbitrary, we conclude $\exists n.\, p(n) \wedge \textsf{nat}(n) \subseteq \mathrm{post}([\![C^\star]\!]_{\textsf{ok}})(p(0))$, which is the desired soundness statement.

It remains to prove soundness of the rules (Constancy, Local Variable, Substitution I/II). We will use two standard semantic facts about commands that *do not mention* certain variables.

Two semantic invariance lemmas. Recall that for a store σ and value $v \in \textsf{Val}$, $\sigma[x \mapsto v]$ is the store agreeing with σ everywhere except at x.

Lemma 3.5 (Untouched variables are preserved). *If $x \notin \mathrm{Mod}(C)$ and $(\sigma, \sigma') \in [\![C]\!]_\epsilon$, then $\sigma'(x) = \sigma(x)$. More generally, if $X \cap \mathrm{Mod}(C) = \emptyset$, then σ and σ' agree on every $x \in X$.*

Lemma 3.6 (Parametricity in fresh variables). *Let $z \notin \mathrm{Free}(C)$, where $\mathrm{Free}(C)$ denotes the set of program variables whose values C may read or write. If $(\sigma, \sigma') \in [\![C]\!]_\epsilon$, then for every $v \in \textsf{Val}$,*

$$(\sigma[z \mapsto v],\; \sigma'[z \mapsto v]) \in [\![C]\!]_\epsilon.$$

$\textsf{constancy}$. Assume $\models [p]\, C\, [\epsilon : q]$ and $\mathrm{Mod}(C) \cap \mathrm{Free}(f) = \emptyset$. We must show $\models [p \wedge f]\, C\, [\epsilon : q \wedge f]$, i.e. $q \cap f \subseteq \mathrm{post}([\![C]\!]_\epsilon)(p \cap f)$.

Let $\sigma' \in q \cap f$. Since $\sigma' \in q$ and $\models [p]\, C\, [\epsilon : q]$, Lemma 3.2 gives a witness $\sigma \in p$ with $(\sigma, \sigma') \in [\![C]\!]_\epsilon$. Because $\mathrm{Free}(f)$ is disjoint from $\mathrm{Mod}(C)$, Lemma 3.5 implies that σ and σ' agree on all variables on which f depends; hence $\sigma' \models f$ implies $\sigma \models f$. Therefore $\sigma \in p \cap f$, and so $\sigma' \in \mathrm{post}(\llbracket C \rrbracket_\epsilon)(p \cap f)$. Since σ' was arbitrary in $q \cap f$, we conclude $q \cap f \subseteq \mathrm{post}(\llbracket C \rrbracket_\epsilon)(p \cap f)$.

local variable. Assume $\models [p]\, C(y/x)\, [\epsilon : q]$ with y fresh and $y \notin \mathrm{Free}(p, C)$. We must show $\models [p]\, (\mathsf{local}\ x.\, C)\, [\epsilon : \exists y.\, q]$.

Let $\sigma_f \in \exists y.\, q$. By the semantics of $\exists y$, pick a value $v_2 \in \mathsf{Val}$ such that $\sigma_f[y \mapsto v_2] \in q$.

Since $\sigma_f[y \mapsto v_2] \in q$ and $\models [p]\, C(y/x)\, [\epsilon : q]$, Lemma 3.2 yields a store $\sigma_i \in p$ such that

$$(\sigma_i,\ \sigma_f[y \mapsto v_2]) \in \llbracket C(y/x) \rrbracket_\epsilon.$$

Write $v := \sigma_f(x)$ (the externally-visible value of x in the final store). Because x does not occur in $C(y/x)$ (it has been replaced by y), we have $x \notin \mathrm{Mod}(C(y/x))$, so by Lemma 3.5 the execution above preserves x; hence $\sigma_i(x) = \sigma_f(x) = v$. Also write $v_y := \sigma_f(y)$ (the actual, externally-visible value of the fresh variable y in σ_f).

Now define a swapping permutation π on stores that exchanges x and y and leaves all other variables unchanged: $\pi(\sigma)(x) = \sigma(y)$, $\pi(\sigma)(y) = \sigma(x)$. Because y is fresh (in particular $y \notin \mathrm{Free}(C)$), the command $C(y/x)$ is exactly the result of renaming x to y in C, and one checks (by a routine induction on C) that semantic execution is equivariant under π:

$$(\sigma, \sigma') \in \llbracket C(y/x) \rrbracket_\epsilon \quad \Longrightarrow \quad (\pi(\sigma), \pi(\sigma')) \in \llbracket C \rrbracket_\epsilon.$$

Applying this to the pair above gives

$$(\pi(\sigma_i),\ \pi(\sigma_f[y \mapsto v_2])) \in \llbracket C \rrbracket_\epsilon.$$

Unfolding π, $\pi(\sigma_i)$ has $x = \sigma_i(y)$ (some value v_1) and $y = \sigma_i(x) = v$, while $\pi(\sigma_f[y \mapsto v_2])$ has $x = v_2$ and $y = \sigma_f(x) = v$. Since $y \notin \mathrm{Free}(C)$, we may change the value of y in both endpoints to the external value v_y without affecting membership in the relation (Lemma 3.6 with $z = y$). Hence

$$\Big(\ \pi(\sigma_i)[y \mapsto v_y],\ \pi(\sigma_f[y \mapsto v_2])[y \mapsto v_y]\ \Big) \in \llbracket C \rrbracket_\epsilon.$$

Finally, apply the semantics of $\mathsf{local}\ x.\, C$:

$$\llbracket \mathsf{local}\ x.\, C \rrbracket_\epsilon = \{(\tau[x \mapsto v],\ \tau'[x \mapsto v]) \mid (\tau, \tau') \in \llbracket C \rrbracket_\epsilon,\ v \in \mathsf{Val}\}.$$

Take $\tau := \pi(\sigma_i)[y \mapsto v_y]$ and $\tau' := \pi(\sigma_f[y \mapsto v_2])[y \mapsto v_y]$. Then $(\tau, \tau') \in \llbracket C \rrbracket_\epsilon$, and overwriting x with the external value v yields:

$$\tau[x \mapsto v] \in p \quad \text{(because } \tau \text{ differs from } \sigma_i \text{ only in } y\text{, and } y \notin \mathrm{Free}(p)\text{)},$$

$$\tau'[x \mapsto v] = \sigma_f \quad \text{(since } \tau' \text{ has the same non-}x \text{ components as } \sigma_f \text{ and } x \text{ is overwritten to } v = \sigma_f(x)\text{)}.$$

Thus $(\tau[x \mapsto v],\ \tau'[x \mapsto v]) \in \llbracket \mathsf{local}\ x.\, C \rrbracket_\epsilon$ with $\tau[x \mapsto v] \in p$ and $\tau'[x \mapsto v] = \sigma_f$, i.e. $\sigma_f \in \mathrm{post}(\llbracket \mathsf{local}\ x.\, C \rrbracket_\epsilon)(p)$. Since σ_f was arbitrary in $\exists y.\, q$, we conclude $\exists y.\, q \subseteq \mathrm{post}(\llbracket \mathsf{local}\ x.\, C \rrbracket_\epsilon)(p)$ as required.

substitution II. Assume $\models [p]\, C\, [\epsilon : q]$ and let y be fresh with $y \notin \mathrm{Free}(p, C, q)$. We must show $\models ([p]\, C\, [\epsilon : q])(y/x)$.

Unfolding the notation, this conclusion is $\models [p(y/x)]\, C(y/x)\, [\epsilon : q(y/x)]$. Let π be the swap of x and y as above. Since y is fresh, we have the standard semantic equalities:

$$\sigma \in p(y/x) \iff \pi(\sigma) \in p, \qquad \sigma \in q(y/x) \iff \pi(\sigma) \in q,$$

and (by the same equivariance argument used above)

$$(\sigma, \sigma') \in \llbracket C(y/x) \rrbracket_\epsilon \iff (\pi(\sigma), \pi(\sigma')) \in \llbracket C \rrbracket_\epsilon.$$

Now take any $\sigma' \in q(y/x)$. Then $\pi(\sigma') \in q$, and since $\models [p]\,C\,[\epsilon : q]$, Lemma 3.2 yields $\rho \in p$ with $(\rho, \pi(\sigma')) \in [\![C]\!]_\epsilon$. Let $\sigma := \pi(\rho)$. Then $\pi(\sigma) = \rho \in p$, so $\sigma \in p(y/x)$, and $(\sigma, \sigma') \in [\![C(y/x)]\!]_\epsilon$ by equivariance. Hence $\sigma' \in \mathrm{post}([\![C(y/x)]\!]_\epsilon)(p(y/x))$. Since σ' was arbitrary in $q(y/x)$, we obtain $q(y/x) \subseteq \mathrm{post}([\![C(y/x)]\!]_\epsilon)(p(y/x))$, i.e. $\models [p(y/x)]\,C(y/x)\,[\epsilon : q(y/x)]$.

substitution I. Assume $\models [p]\,C\,[\epsilon : q]$ and $(\mathrm{Free}(e) \cup \{x\}) \cap \mathrm{Free}(C) = \emptyset$. Because $x \notin \mathrm{Free}(C)$, substituting e for x does not change the command, so the conclusion $([p]\,C\,[\epsilon : q])(e/x)$ amounts to $\models [p[e/x]]\,C\,[\epsilon : q[e/x]]$.

Let $\sigma' \in q[e/x]$. By the usual meaning of assertion substitution,

$$\sigma' \in q[e/x] \quad \iff \quad \sigma'[x \mapsto [\![e]\!](\sigma')] \in q.$$

Define $\bar{\sigma}' := \sigma'[x \mapsto [\![e]\!](\sigma')]$, so $\bar{\sigma}' \in q$. From $\models [p]\,C\,[\epsilon : q]$ and Lemma 3.2, pick $\bar{\sigma} \in p$ with

$$(\bar{\sigma}, \bar{\sigma}') \in [\![C]\!]_\epsilon.$$

Since $(\mathrm{Free}(e) \cup \{x\}) \cap \mathrm{Free}(C) = \emptyset$, the command C neither reads nor writes any variable in $\mathrm{Free}(e) \cup \{x\}$. In particular:

- $x \notin \mathrm{Free}(C)$, so by Lemma 3.6 we may change x simultaneously in both endpoints of a C-execution without leaving the relation;
- every variable in $\mathrm{Free}(e) \setminus \{x\}$ is untouched by C, so by Lemma 3.5 its value is preserved along the execution.

Now set $\sigma := \bar{\sigma}[x \mapsto \sigma'(x)]$ (i.e. change only x in the initial store to match σ'). By Lemma 3.6 with $z = x$, from $(\bar{\sigma}, \bar{\sigma}') \in [\![C]\!]_\epsilon$ we obtain

$$(\sigma,\ \bar{\sigma}'[x \mapsto \sigma'(x)]) \in [\![C]\!]_\epsilon.$$

But $\bar{\sigma}'$ differs from σ' only at x, and $\bar{\sigma}'[x \mapsto \sigma'(x)] = \sigma'$, so we have $(\sigma, \sigma') \in [\![C]\!]_\epsilon$.

It remains to show $\sigma \in p[e/x]$, i.e. $\sigma[x \mapsto [\![e]\!](\sigma)] \in p$. Because C does not touch any variable in $\mathrm{Free}(e) \setminus \{x\}$ and σ and σ' agree on all those variables (they already agreed in $\bar{\sigma}$ and $\bar{\sigma}'$, and we only changed x), we have $[\![e]\!](\sigma) = [\![e]\!](\sigma')$. Also, by Lemma 3.5 (since $x \notin \mathrm{Mod}(C)$), the execution $(\bar{\sigma}, \bar{\sigma}') \in [\![C]\!]_\epsilon$ preserves x, hence

$$\bar{\sigma}(x) = \bar{\sigma}'(x) = [\![e]\!](\sigma').$$

Therefore,

$$\sigma[x \mapsto [\![e]\!](\sigma)] = \sigma[x \mapsto [\![e]\!](\sigma')] = \bar{\sigma} \in p,$$

so indeed $\sigma \in p[e/x]$. We have shown: for arbitrary $\sigma' \in q[e/x]$, there exists $\sigma \in p[e/x]$ with $(\sigma, \sigma') \in [\![C]\!]_\epsilon$, hence $q[e/x] \subseteq \mathrm{post}([\![C]\!]_\epsilon)(p[e/x])$. This is exactly $\models [p[e/x]]\,C\,[\epsilon : q[e/x]]$, completing soundness for Substitution I.

□

3.5 Completeness of Incorrectness Logic

From HL relative completeness to IL completeness. In Section 2.5 we proved Cook-style *relative completeness* for Hoare Logic: assuming the assertion logic can express weakest liberal preconditions and can prove all valid implications between assertions, every semantically valid Hoare triple is derivable. The need for these assumptions comes from the fact that HL uses a *syntactic* assertion language whose expressiveness may be limited.

IL admits a closely related completeness story, but with an important twist: instead of syntactic assertions, we treat assertions *semantically* as sets of states, and we treat implications/inclusions as an *oracle* side-condition. With semantic predicates, expressiveness is built in, so completeness reduces to whether the proof rules are missing anything essential. We'll show they are now: every true IL triple (with mild side conditions) is derivable. The remaining assumptions are: (i) an oracle for inclusions needed by CONSEQUENCE, and (ii) enough variables and naturals to support the BACK-VAR loop rule.

Definition 3.7 (Support and finite support)**.** Let $p \subseteq \Sigma$ be a predicate and let $x \in \mathsf{Var}$. Say that p is *independent of* x if for all states σ and all values $v, v' \in \mathsf{Val}$,

$$\sigma[x \mapsto v] \in p \iff \sigma[x \mapsto v'] \in p.$$

A predicate p is *finitely-supported* if there exists a finite set $S \subseteq \mathsf{Var}$ such that p is independent of every $x \notin S$.

Intuitively, finitely-supported predicates depend only on finitely many variables; this matches the fact that typical syntactic assertions mention finitely many variables.

Theorem 3.8 (Completeness of IL (semantic/relative completeness))**.** *Let C be any IL command and let $p, q \subseteq \Sigma$ be finitely-supported predicates. If $\models [p]\, C\, [\epsilon : q]$, then $\vdash [p]\, C\, [\epsilon : q]$. Equivalently, every true IL triple involving finitely-supported predicates is provable.*

Remark 3.9. This theorem is the IL analogue of HL relative completeness: it is "relative" to the implication oracle, and it uses semantic predicates (so expressiveness is not a limiting factor). It does not contradict undecidability: the oracle can hide non-computable reasoning, just as the HL completeness theorem hides it in wlp-expressiveness and implication-proving.

Lemma 3.10 (Derivable exact post-image)**.** *For every IL command C, every exit $\epsilon \in \{\mathsf{ok}, \mathsf{er}\}$, and every finitely-supported predicate p,*

$$\vdash [p]\, C\, [\epsilon : \mathrm{post}_\epsilon(C)(p)].$$

Proof. We prove the lemma by structural induction on C. Throughout, we repeatedly use Lemma 3.3 and Lemma 3.2 **skip.** By SKIP we have $[p]\, \mathsf{skip}\, [\mathsf{ok} : p]\, [\mathsf{er} : \mathsf{ff}]$. But $\mathrm{post}_{\mathsf{ok}}(\mathsf{skip})(p) = p$ and $\mathrm{post}_{\mathsf{er}}(\mathsf{skip})(p) = \varnothing$ by semantics, so the rule is exactly the desired statement (no extra use of CONSEQUENCE needed).

error. By ERROR, $[p]\, \mathsf{error}()\, [\mathsf{ok} : \mathsf{ff}]\, [\mathsf{er} : p]$. Semantically $\mathrm{post}_{\mathsf{ok}}(\mathsf{error}())(p) = \varnothing$ and $\mathrm{post}_{\mathsf{er}}(\mathsf{error}())(p) = p$, so again this is exactly the desired statement.

assume. By ASSUME, $[p]\, \mathsf{assume}\, B\, [\mathsf{ok} : p \wedge B]\, [\mathsf{er} : \mathsf{ff}]$. Semantically $\mathrm{post}_{\mathsf{ok}}(\mathsf{assume}\, B)(p) = p \cap \mathrm{Guard}(B) = p \wedge B$ and the error post-image is empty.

(assign) and (nondet). For deterministic assignment, ASSIGNMENT gives

$$[p]\ x := e\ [\mathsf{ok} : \exists x'.\ p[x'/x]\ \wedge\ x = e[x'/x]]\ [\mathsf{er} : \mathsf{ff}].$$

Unfolding $\mathrm{post}_{\mathsf{ok}}(x := e)(p)$ from the relational semantics shows that this postcondition describes *exactly* the set of states reachable by executing $x := e$ from some p-state, hence it is equal to $\mathrm{post}_{\mathsf{ok}}(x := e)(p)$. Therefore, by CONSEQUENCE using equality (both inclusions), we obtain $\vdash [p]\ x := e\ [\mathsf{ok} : \mathrm{post}_{\mathsf{ok}}(x := e)(p)]$ and also the er-case (which is ff).

The nondeterministic assignment case is analogous using NONDET ASSIGNMENT.

choice. We prove the ϵ-case (uniformly for ok or er). By induction hypothesis,

$$\vdash [p]\, C_1\, [\epsilon : \mathrm{post}_\epsilon(C_1)(p)] \qquad \text{and} \qquad \vdash [p]\, C_2\, [\epsilon : \mathrm{post}_\epsilon(C_2)(p)].$$

Apply CHOICE twice (once for each branch) to lift these to $C_1 + C_2$:

$$\vdash [p]\, (C_1 + C_2)\, [\epsilon : \mathrm{post}_\epsilon(C_1)(p)] \quad \text{and} \quad \vdash [p]\, (C_1 + C_2)\, [\epsilon : \mathrm{post}_\epsilon(C_2)(p)].$$

Now apply DISJUNCTION with $p_1 = p_2 = p$ to combine the postconditions:

$$\vdash [p]\,(C_1 + C_2)\,[\epsilon : \mathrm{post}_\epsilon(C_1)(p) \;\vee\; \mathrm{post}_\epsilon(C_2)(p)].$$

Finally, by semantics of choice, $\mathrm{post}_\epsilon(C_1 + C_2)(p) = \mathrm{post}_\epsilon(C_1)(p) \cup \mathrm{post}_\epsilon(C_2)(p)$, so the right-hand side is exactly $\mathrm{post}_\epsilon(C_1 + C_2)(p)$. A final CONSEQUENCE (again using equality) yields the goal.

sequencing. We do ok and er separately.

$\epsilon = \mathsf{ok}$. Let $q \triangleq \mathrm{post}_{\mathsf{ok}}(C_1)(p)$. By IH, $\vdash [p]\,C_1\,[\mathsf{ok} : q]$. By IH again, $\vdash [q]\,C_2\,[\mathsf{ok} : \mathrm{post}_{\mathsf{ok}}(C_2)(q)]$. Apply SEQ-OK with $\epsilon = \mathsf{ok}$ to obtain

$$\vdash [p]\,(C_1; C_2)\,[\mathsf{ok} : \mathrm{post}_{\mathsf{ok}}(C_2)(q)].$$

By post-composition (Lemma 3.3), $\mathrm{post}_{\mathsf{ok}}(C_1; C_2)(p) = \mathrm{post}_{\mathsf{ok}}(C_2)(\mathrm{post}_{\mathsf{ok}}(C_1)(p)) = \mathrm{post}_{\mathsf{ok}}(C_2)(q)$, so this is exactly the desired triple.

$\epsilon = \mathsf{er}$. The semantics gives the decomposition

$$\mathrm{post}_{\mathsf{er}}(C_1; C_2)(p) = \mathrm{post}_{\mathsf{er}}(C_1)(p) \;\vee\; \mathrm{post}_{\mathsf{er}}(C_2)\big(\,\mathrm{post}_{\mathsf{ok}}(C_1)(p)\big).$$

Let $q \triangleq \mathrm{post}_{\mathsf{ok}}(C_1)(p)$ as above. By IH we have $\vdash [p]\,C_1\,[\mathsf{er} : \mathrm{post}_{\mathsf{er}}(C_1)(p)]$. Apply SEQ-ER to get

$$\vdash [p]\,(C_1; C_2)\,[\mathsf{er} : \mathrm{post}_{\mathsf{er}}(C_1)(p)].$$

Also, by IH we have $\vdash [p]\,C_1\,[\mathsf{ok} : q]$ and $\vdash [q]\,C_2\,[\mathsf{er} : \mathrm{post}_{\mathsf{er}}(C_2)(q)]$, so by SEQ-OK (with $\epsilon = \mathsf{er}$) we get

$$\vdash [p]\,(C_1; C_2)\,[\mathsf{er} : \mathrm{post}_{\mathsf{er}}(C_2)(q)].$$

Combine these two error triples using DISJUNCTION (again with the same pre p) to obtain

$$\vdash [p]\,(C_1; C_2)\,[\mathsf{er} : \mathrm{post}_{\mathsf{er}}(C_1)(p) \;\vee\; \mathrm{post}_{\mathsf{er}}(C_2)(q)].$$

The right-hand side is exactly $\mathrm{post}_{\mathsf{er}}(C_1; C_2)(p)$, so we are done.

iteration. Again we handle ok and er separately.

$\epsilon = \mathsf{ok}$. We must derive $[p]\,C^\star\,[\mathsf{ok} : \mathrm{post}_{\mathsf{ok}}(C^\star)(p)]$.

Define a family of predicates $(p_n)_{n \in \mathbb{N}}$ by:

$$p_0 \triangleq p, \qquad p_{n+1} \triangleq \mathrm{post}_{\mathsf{ok}}(C)(p_n).$$

Intuitively, p_n is the set of states reachable by n successful iterations.

Let n be fresh for p and C (as required by BACK-VAR), and define the *variant schema* $p(n)$ by the usual "arithmetical interpretation"

$$p(n) \;\triangleq\; \bigvee_{k \in \mathbb{N}} \big((n{=}k) \wedge p_k\big).$$

(So a state satisfying $p(n)$ has some numeric value k stored in n and is in p_k.)

show the BACK-VAR premise is true, hence derivable. We claim that for each n,

$$\models [\,p(n) \wedge \mathsf{nat}(n)\,]\,C\,[\mathsf{ok} :\, p(n{+}1) \wedge \mathsf{nat}(n)\,].$$

Unfold semantic validity: we must show $p(n+1) \wedge \mathsf{nat}(n) \subseteq \mathrm{post}_{\mathsf{ok}}(C)(p(n) \wedge \mathsf{nat}(n))$. Take any $\sigma \in p(n{+}1) \wedge \mathsf{nat}(n)$ and write $k \triangleq \sigma(n) \in \mathbb{N}$. By the definition of $p(n{+}1)$, we have $\sigma \in p_{k+1}$, hence $\sigma \in \mathrm{post}_{\mathsf{ok}}(C)(p_k)$, so there exists $\sigma' \in p_k$ with $(\sigma', \sigma) \in [\![C]\!]_{\mathsf{ok}}$. Since n is fresh and C does not modify n, we have $\sigma'(n) = \sigma(n) = k$, hence $\sigma' \in p(n) \wedge \mathsf{nat}(n)$. Therefore $\sigma \in \mathrm{post}_{\mathsf{ok}}(C)(p(n) \wedge \mathsf{nat}(n))$, as required.

Now apply the induction hypothesis to the strict subcommand C with precondition $p(n) \wedge \mathsf{nat}(n)$ to obtain the exact-post triple

$$\vdash [\, p(n) \wedge \mathsf{nat}(n) \,]\, C \,[\mathsf{ok} :\ \mathrm{post}_{\mathsf{ok}}(C)(p(n) \wedge \mathsf{nat}(n)) \,].$$

Using the semantic inclusion proved above and Consequence, we derive the needed premise

$$\vdash [\, p(n) \wedge \mathsf{nat}(n) \,]\, C \,[\mathsf{ok} :\ p(n{+}1) \wedge \mathsf{nat}(n) \,].$$

apply Back-Var. By Back-Var we obtain

$$\vdash [p(0)]\, C^\star\, [\mathsf{ok} : \exists n.\, p(n) \wedge \mathsf{nat}(n)].$$

Since $p(0) = p_0 = p$, this gives

$$\vdash [p]\, C^\star\, [\mathsf{ok} : \exists n.\, p(n) \wedge \mathsf{nat}(n)].$$

identify $\exists n.\, p(n) \wedge \mathsf{nat}(n)$ with the exact post-image. We show

$$\mathrm{post}_{\mathsf{ok}}(C^\star)(p) \;=\; \bigvee_{n \in \mathbb{N}} p_n \qquad \text{and hence} \qquad \mathrm{post}_{\mathsf{ok}}(C^\star)(p) \;=\; \exists n.\, p(n) \wedge \mathsf{nat}(n).$$

By semantics, $[\![C^\star]\!]_{\mathsf{ok}} = \bigcup_{n \in \mathbb{N}} [\![C^n]\!]_{\mathsf{ok}}$, hence by distribution of post over unions (Lemma 3.3),

$$\mathrm{post}_{\mathsf{ok}}(C^\star)(p) = \bigcup_{n \in \mathbb{N}} \mathrm{post}([\![C^n]\!]_{\mathsf{ok}})(p).$$

But $\mathrm{post}([\![C^n]\!]_{\mathsf{ok}})(p)$ is exactly p_n by repeated unfolding of the recursion $p_{n+1} = \mathrm{post}_{\mathsf{ok}}(C)(p_n)$. Finally, by the definition of $p(n)$ above, the existential $\exists n$ (together with $\mathsf{nat}(n)$) expresses precisely the countable union $\bigvee_{n \in \mathbb{N}} p_n$. Therefore $\exists n.\, p(n) \wedge \mathsf{nat}(n)$ is *equal* to $\mathrm{post}_{\mathsf{ok}}(C^\star)(p)$.

conclude the desired triple by Consequence. From the derived triple with post $\exists n.\, p(n) \wedge \mathsf{nat}(n)$ and the equality above, a final Consequence yields $\vdash [p]\, C^\star\, [\mathsf{ok} : \mathrm{post}_{\mathsf{ok}}(C^\star)(p)]$.

$\epsilon = \mathsf{er}$. We show $\vdash [p]\, C^\star\, [\mathsf{er} : \mathrm{post}_{\mathsf{er}}(C^\star)(p)]$. Let

$$\mathit{frontier} \triangleq \mathrm{post}_{\mathsf{ok}}(C^\star)(p),$$

the states reachable after some number of successful iterations. By the already-proven ok-case,

$$\vdash [p]\, C^\star\, [\mathsf{ok} : \mathit{frontier}].$$

By the semantics of iteration, an er-exit from $C^\star$ consists of some number of ok-iterations reaching a frontier state, followed by one execution of C that exits with er; in particular,

$$\mathrm{post}_{\mathsf{er}}(C^\star)(p) \subseteq \mathrm{post}_{\mathsf{er}}(C)(\mathit{frontier}).$$

Equivalently, $\models [\mathit{frontier}]\, C\, [\mathsf{er} : \mathrm{post}_{\mathsf{er}}(C^\star)(p)]$.

Apply the induction hypothesis to the strict subcommand C with precondition *frontier* to get

$$\vdash [\mathit{frontier}]\, C\, [\mathsf{er} : \mathrm{post}_{\mathsf{er}}(C)(\mathit{frontier})].$$

Using the inclusion $\mathrm{post}_{\mathsf{er}}(C^\star)(p) \subseteq \mathrm{post}_{\mathsf{er}}(C)(\mathit{frontier})$ and Consequence, derive

$$\vdash [\mathit{frontier}]\, C\, [\mathsf{er} : \mathrm{post}_{\mathsf{er}}(C^\star)(p)].$$

Apply Seq-Ok with $\epsilon = \mathsf{er}$ to obtain

$$\vdash [p]\, (C^\star; C)\, [\mathsf{er} : \mathrm{post}_{\mathsf{er}}(C^\star)(p)].$$

Finally apply Iter-NonZero to conclude

$$\vdash [p]\, C^\star\, [\mathsf{er} : \mathrm{post}_{\mathsf{er}}(C^\star)(p)].$$

$\mathsf{local\ var}$**.** Let $\epsilon \in \{\mathsf{ok}, \mathsf{er}\}$. We must derive $[p]\, \mathsf{local}\ x.\, C_0\, [\epsilon : \mathrm{post}_\epsilon(\mathsf{local}\ x.\, C_0)(p)]$.

Pick a fresh variable y such that: (i) p is independent of y, and (ii) y is not free in C_0 (so renaming does not capture anything). Consider the renamed body $C_0(y/x)$. (We take the induction to be on the size of commands; renaming preserves size, so IH applies.)

By induction hypothesis applied to $C_0(y/x)$, we have:

$$\vdash [p]\, C_0(y/x)\, [\epsilon : \mathrm{post}_\epsilon(C_0(y/x))(p)].$$

Apply the Local Variable rule to obtain

$$\vdash [p]\ \mathsf{local}\ x.\, C_0\ [\epsilon : \exists y.\ \mathrm{post}_\epsilon(C_0(y/x))(p)].$$

It remains to relate $\exists y.\, \mathrm{post}_\epsilon(C_0(y/x))(p)$ to the semantic post-image of $\mathsf{local}\ x.\, C_0$.

Claim. $\mathrm{post}_\epsilon(\mathsf{local}\ x.\, C_0)(p) \subseteq \exists y.\, \mathrm{post}_\epsilon(C_0(y/x))(p)$.

Proof of the claim. Take any $\sigma_q \in \mathrm{post}_\epsilon(\mathsf{local}\ x.\, C_0)(p)$. By definition of post, there exists $\sigma_p \in p$ such that $(\sigma_p, \sigma_q) \in [\![\mathsf{local}\ x.\, C_0]\!]_\epsilon$. Unfolding the semantics of local, this means: there exist values $v_0, v_1, v_2 \in \mathsf{Val}$ and intermediate states such that the transition is witnessed by

$$\sigma_p[x \mapsto v_0] \xrightarrow{\text{init}} \sigma_p[x \mapsto v_1] \xrightarrow{C_0} \sigma_1[x \mapsto v_2] \xrightarrow{\text{final}} \sigma_q[x \mapsto v_0],$$

where the outer value v_0 is restored after executing C_0. Now consider the renamed program $C_0(y/x)$. Because p is independent of y, from $\sigma_p \in p$ we also have $\sigma_p[y \mapsto v_1] \in p$. Executing $C_0(y/x)$ from $\sigma_p[y \mapsto v_1]$ mimics the execution of C_0 from $\sigma_p[x \mapsto v_1]$, producing a final state $\sigma_q[y \mapsto v_2]$. Therefore $\sigma_q[y \mapsto v_2] \in \mathrm{post}_\epsilon(C_0(y/x))(p)$, and by the semantics of $\exists y$, $\sigma_q \in \exists y.\, \mathrm{post}_\epsilon(C_0(y/x))(p)$. This proves the inclusion.

Using the claim and Consequence, we can strengthen the derived triple for $\mathsf{local}\ x.\, C_0$ to obtain exactly

$$\vdash [p]\ \mathsf{local}\ x.\, C_0\ [\epsilon : \mathrm{post}_\epsilon(\mathsf{local}\ x.\, C_0)(p)].$$

This completes all cases of the induction, proving the lemma. □

Proof of Theorem 3.8. Assume $\models [p]\, C\, [\epsilon : q]$ with p, q finitely-supported. By validity, $q \subseteq \mathrm{post}_\epsilon(C)(p)$. By Lemma 3.10, $\vdash [p]\, C\, [\epsilon : \mathrm{post}_\epsilon(C)(p)]$. Apply Consequence with $q \subseteq \mathrm{post}_\epsilon(C)(p)$ to obtain $\vdash [p]\, C\, [\epsilon : q]$, as required. □

Remark 3.11. In the above proof, the core rules for completeness are those for skip, assume, error, assignment, choice, sequencing, iteration, Back-Var, and local variables, together with Consequence and Disjunction. The variable-adaptation rules Constancy and Substitution are useful for reusing summaries in richer contexts, but are not required for the completeness argument itself.