Gradual Exact Logic

Unifying Hoare Logic and Incorrectness Logic via Gradual Verification

Conrad Zimmerman
zimmerman.co@northeastern.edu
Northeastern University
Boston, MA, USA

Jenna DiVincenzo
jennad@purdue.edu
Purdue University
West Lafayette, IN, USA

Abstract

Previously, gradual verification has been developed using overapproximating logics such as Hoare logic. We show that the static verification component of gradual verification is also connected to underapproximating logics like incorrectness logic. To do this, we use a novel definition of gradual verification and a novel gradualization of exact logic [Maksimovic et al. 2023] which we call *gradual exact logic*. Further, we show that Hoare logic, incorrectness logic, and gradual verification can be defined in terms of gradual exact logic.

We hope that this connection can be used to develop tools and techniques that apply to both gradual verification and bug-finding. For example, we envision that techniques defined in terms of exact logic can be directly applied to verification, bug-finding, and gradual verification, using the principles of gradual typing [Garcia et al. 2016].

1 Overview

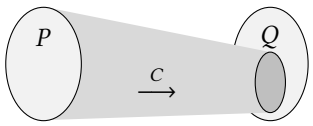
Incorrectness logic [O'Hearn 2020] has been recently developed as a formal basis for "true bug-finding" and has been applied in industrial-strength tools [Le et al. 2022]. Deductions in this logic prove reachability, which enables bug-finding tools to prove the existence of an invalid state while selectively exploring the possible paths.

At the same time, gradual verification (GV) [Bader et al. 2018] addresses the complexity of traditional static verification. Gradually verified programs may contain *imprecise specifications*—logical formulas annotated to indicate that they contain only a partial specification of behavior. A gradual verifier checks the imprecise specifications using static verification where it can and run-time checks (i.e. dynamic verification) elsewhere. These run-time checks can be exercised, e.g. with testing, giving the programmer confidence that their code will not enter a state that violates their partial specifications. Using gradual verification, programmers can incrementally verify a program, incrementally learn verification constructs, and safely guard unverified components.

More recent work [Lööw et al. 2024; Maksimovic et al. 2023; Zilberstein et al. 2023] has produced logics and tools that unify over-approximating (OX) logic, often used in verification, and under-approximating (UX) logics including IL. In this paper, we propose another unification of OX and UX logics that we derive using the principles of gradual verification. The resulting characterization of IL demonstrates a previously unexplored connection between GV and IL.

1.1 Hoare logic

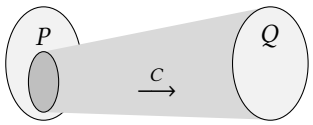
A triple in Hoare logic (HL) [Hoare 1969] is denoted $\{P\}\ C\ \{Q\}$. (P, Q refer to logical specifications; C refers to a program statement.) Semantically, the triple is valid if, for every state $\sigma \in P$ (i.e., P is true of σ), when $\sigma \xrightarrow{C} \sigma'$ (i.e., executing C results in σ'), then $\sigma' \in Q$:



Thus HL is an *overapproximating* (OX) logic—the postcondition Q overapproximates the states that are reachable from P; precisely, $Q \supseteq \{\sigma' \mid \exists \sigma \in P : \sigma \xrightarrow{C} \sigma'\}$. HL is a formal foundation for program verification precisely because the postcondition is true in *all* ending states.

1.2 Incorrectness logic

A triple in incorrectness logic (IL) [O'Hearn 2020] is denoted $[P]\ C\ [Q]$[1]. Semantically, the triple is valid if, for every $\sigma' \in Q$, there is some $\sigma \in P$ such that $\sigma \xrightarrow{C} \sigma'$:



Thus IL is an *underapproximating* (UX) logic—the postcondition Q underapproximates the states that are reachable from P; precisely, $Q \subseteq \{\sigma' \mid \exists \sigma \in P : \sigma \xrightarrow{C} \sigma'\}$. Interpreting Q as a specification of a bug, IL is a logic for finding bugs since a valid triple indicates that the bug is reachable.

1.3 Gradual verification

Gradual verification (GV) [Bader et al. 2018] reduces the burden of static verification by allowing incomplete (*imprecise*) specifications. A gradual verifier may make optimistic assumptions when verifying imprecisely-specified code. Typically, the final program is elaborated to check these assumptions at run-time. However, in this work we focus solely on the static verification component of GV so that we can compare its logic with HL and IL.

We denote imprecise triples by $\{? \wedge P\}\ C\ \{Q\}$. Intuitively, this triple is valid if there is some $P' \Rightarrow P$ such that $\{P'\}\ C\ \{Q\}$

[1] IL triples often include a specification for error states, but we omit error handling to simplify the comparison with other logics.

is valid in HL. One can think of ? as representing the additional assumptions introduced by P'.

For example, the following imprecise triple is valid:

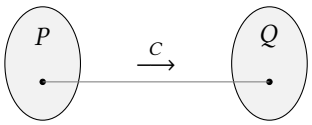
$$\{? \land \top\}\ x := x + 1\ \{x > 0\}$$

This follows from the validity of

$$\{x \geq 0 \land \top\}\ x := x + 1\ \{x > 0\}.$$

That is, the postcondition is ensured when assuming $x \geq 0$.

However, the assumptions must be plausible; formally, P' (and thus P) must be satisfiable (i.e., $P' \not\equiv \bot$). Otherwise, all imprecise triples would be vacuously valid by taking $P' \equiv \bot$. With this in mind, GV can be stated as a *reachability* problem—semantically, $\{? \land P\}\ C\ \{Q\}$ is valid if there exists some states σ and σ' such that $\sigma \in P$, $\sigma \xrightarrow{C} \sigma'$, and $\sigma' \in Q$:



By contrapositive, the triple is invalid (i.e., static verification will error) if it is never possible for C to ensure Q, given P.

By comparing this diagram with those in §1.1 and §1.2, we can deduce that HL *and* IL triples are valid imprecise triples, except for vacuous cases where $P \equiv \bot$ in HL or $Q \equiv \bot$ in IL.

Our semantic definition of validity is equivalent to the previous definition which uses Hoare triples: Let P' be a formula that represents σ as specifically as possible, then (intuitively) $\{P'\}\ C\ \{Q\}$ is valid since $\sigma \xrightarrow{C} \sigma'$ and $\sigma' \in Q$.

But, we can also define valid imprecise triples in terms of IL—$\{? \land P\}\ C\ \{Q\}$ is valid if $[P]\ C\ [Q']$ is valid for some $Q' \Rightarrow Q$. In §2.5 we will prove these definitions equivalent.

2 Formal foundations

We will now sketch our formal definitions and results. See the appendices for the full statements and proofs.

1. We define *gradual exact logic*—a consistent lifting [Garcia et al. 2016] of exact logic [Maksimovic et al. 2023].
2. We show that HL, IL, and GV can be characterized by gradual exact logic.
3. We show that GV contains OX and UX deductions.
4. We show that, for imprecise specifications, GV can be equivalently defined using OX or UX logics.

2.1 Exact logic

Exact logic (EL) [Maksimovic et al. 2023] is the intersection of HL and IL: an EL triple $(P)\ C\ (Q)$ is valid if $\{P\}\ C\ \{Q\}$ and $[P]\ C\ [Q]$ are both valid. Deductions are thus *exact*—they can neither under- or overapproximate behavior (see Appendix E for rules).

2.2 Gradual exact logic

We further define *gradual exact logic* ($\widetilde{\text{EL}}$) as a consistent lifting [Garcia et al. 2016] of EL.

First, some definitions: Formula denotes all formulas in FOL with arithmetic; we call these *precise*. SatFormula denotes all satisfiable formulas. An *imprecise formula* is of the form $? \land P$ where $P \in$ Formula. A *gradual formula* $\widetilde{P} \in \widetilde{\text{Formula}}$ can be either precise or imprecise.

The *concretization* $\gamma : \widetilde{\text{Formula}} \to \mathcal{P}(\text{Formula})$ interprets a gradual formulas as sets of precise formulas:

$$\gamma(? \land P) = \{P' \in \text{SatFormula} \mid P' \Rightarrow P\}, \quad \gamma(P) = \{P\}$$

Let $\vdash (P)\ C\ (Q)$ denote a valid EL triple. Deductions in $\widetilde{\text{EL}}$, denoted $\widetilde{\vdash}\ (\widetilde{P})\ C\ (\widetilde{Q})$, are defined as a consistent lifting of EL deductions (Appendix E.2):

$$\widetilde{\vdash}\ (\widetilde{P})\ C\ (\widetilde{Q}) \overset{\text{def}}{\iff} \vdash (P)\ C\ (Q) \text{ for some } P \in \gamma(\widetilde{P}), Q \in \gamma(\widetilde{Q})$$

2.3 Strongest postconditions

$\text{sp}(P, C)$ denotes the strongest (WRT $\Rightarrow$) Q for which $\{P\}\ C\ \{Q\}$ is valid (calculated as usual; see Appendix B.2). Strongest postconditions are related to HL, IL, and EL as follows:

$$\vdash \{P\}\ C\ \{Q\} \iff \text{sp}(P, C) \Rightarrow Q \qquad \text{Theorem 2}$$
$$\vdash [P]\ C\ [Q] \iff \text{sp}(P, C) \Leftarrow Q \qquad \text{Theorem 3}$$
$$\vdash (P)\ C\ (Q) \iff \text{sp}(P, C) \equiv Q \qquad \text{Theorem 4}$$

2.4 HL and IL via $\widetilde{\text{EL}}$

We can characterize valid HL triples as $\widetilde{\text{EL}}$ triples where the postcondition is made imprecise. Assuming that $P \not\equiv \bot$ and C terminates (in either of these cases $\{P\}\ C\ \{Q\}$ is vacuous), we have $P, Q, \text{sp}(P, C) \in \text{SatFormula}$ and thus (Theorem 5)

$$\begin{aligned} \vdash \{P\}\ C\ \{Q\} &\iff \text{sp}(P, C) \Rightarrow Q \\ &\iff \text{sp}(P, C) \in \gamma(? \land Q) \\ &\iff \widetilde{\vdash}\ (P)\ C\ (? \land Q). \end{aligned}$$

Likewise, we can characterize valid IL triples as $\widetilde{\text{EL}}$ triples where the precondition is made imprecise. Weakest preconditions are not always defined for IL [O'Hearn 2020], however, we can reuse weakest preconditions for HL to witness the necessary formula (see Appendix B.3). Assuming $Q \not\equiv \bot$ (otherwise $[P]\ C\ [Q]$ is vacuous), we have (Theorem 6)

$$\begin{aligned} \vdash [P]\ C\ [Q] &\iff Q \Rightarrow \text{sp}(P, C) \\ &\iff Q \equiv \text{sp}(\text{wp}(Q, C) \land P, C) \\ &\iff \vdash (P \land \text{wp}(Q, C))\ C\ (Q) \\ &\iff \widetilde{\vdash}\ (? \land P)\ C\ (Q). \end{aligned}$$

2.5 GV via HL, $\widetilde{\text{EL}}$, and IL

We can now give a precise definition of GV in terms of HL, as sketched in §1.3[2]. We denote valid GV triples as $\widetilde{\vdash}\ \{\widetilde{P}\}\ C\ \{Q\}$.

$$\widetilde{\vdash}\ \{\widetilde{P}\}\ C\ \{Q\} \overset{\text{def}}{\iff} \vdash \{P\}\ C\ \{Q\} \text{ for some } P \in \gamma(\widetilde{P})$$

[2]Here we define GV more generally for all *gradual* preconditions, whereas in §1.3 we defined it for *imprecise* preconditions.

Using this definition and applying the characterization of HL from §2.4 to characterize GV in terms of EL:

$$\widetilde{\vdash} \{\widetilde{P}\}\ C\ \{Q\} \iff \widetilde{\vdash} (\widetilde{P})\ C\ (? \wedge Q)$$

For sake of comparison, we can define $\widetilde{\text{IL}}$ as a lifting of IL, the same way we have lifted HL to GV.

$$\widetilde{\vdash} [P]\ C\ [\widetilde{Q}] \overset{\text{def}}{\iff} \vdash [P]\ C\ [Q] \text{ for some } Q \in \gamma(\widetilde{Q})$$

But, in the case of imprecision this is equivalent to GV. We can see this using the characterization of IL given in §2.4:

$$\begin{aligned}\widetilde{\vdash} [P]\ C\ [? \wedge Q] &\iff \widetilde{\vdash} (? \wedge P)\ C\ (? \wedge Q)\\ &\iff \widetilde{\vdash} \{? \wedge P\}\ C\ \{Q\}.\end{aligned}$$

Note: GV and $\widetilde{\text{IL}}$ differ on precise formulas—$\widetilde{\vdash} [P]\ C\ [Q]$ is *not* equivalent to $\widetilde{\vdash} \{P\}\ C\ \{Q\}$. Also, we do not gradualize postconditions in HL or preconditions in IL because we can arbitrarily weaken these specifications already.

While verification of precise formulas is a key aspect of GV, this demonstrates that verification of imprecise formulas can be accomplished using IL, and moreover that GV, when verifying imprecisely-specified code, is proving an IL deduction. Finally, we can make precise our claim from §1.3 that both OX and UX deductions are valid in (the imprecise fragment of) GV. Assuming $P \not\equiv \bot$, we have (immediate from definition of GV)

$$\vdash \{P\}\ C\ \{Q\} \implies \widetilde{\vdash} \{? \wedge P\}\ C\ \{Q\}.$$

Also, assuming $Q \not\equiv \bot$, we have

$$\begin{aligned}\vdash [P]\ C\ [Q] &\implies \widetilde{\vdash} (? \wedge P)\ C\ (Q)\\ &\implies \widetilde{\vdash} (? \wedge P)\ C\ (? \wedge Q)\\ &\implies \widetilde{\vdash} \{? \wedge P\}\ C\ \{Q\}.\end{aligned}$$

Thus GV (and $\widetilde{\text{EL}}$) represents the union of HL and IL, while EL represents the intersection.

3 Applications

GV and IL verifiers in practice operate quite similarly; for example, compare the core *consume* operation of Zimmerman et al. [2024] for a gradual verifier and of Lööw et al. [2024] for an IL verifier. Both types of verifiers make assumptions, including pruning paths, when establishing a postcondition. This work formalizes the connection between these methods of verification; in particular, gradual verification of imprecisely-specified code is equivalent to IL deductions. We hope that this will allow verifiers that already incorporate both OX and UX logics, for example Lööw et al. [2024], to be easily extended to GV.

Similarly, we hope our approach can be used as a framework for unifying techniques across static verification, GV, and bug-finding. For example, bi-abduction has been developed in OX logics [Calcagno et al. 2011], applied to UX verification [Lööw et al. 2024], and is related to GV [DiVincenzo 2023]. We expect that techniques like this could be developed in the context of an exact logic, and then their applications to OX, UX, and GV logics could be derived using AGT-style techniques [Garcia et al. 2016].

4 Caveats and future work

While we have proven the results described, we have done so only for a very restrictive language, and thus our results should be considered preliminary. In particular, we do not consider heaps, method calls, or loops. We expect our results extend to these constructs, but showing this will require significant work.

Our definition of GV differs from previous definitions [Bader et al. 2018; Wise et al. 2020; Zimmerman et al. 2024]. Our novel definition more clearly demonstrates the connection with IL, and we believe it captures the essence of GV. However, this definition only considers static verification, and thus does not consider run-time assertions. We expect this could be added, and we have hopes that we may be able to model run-time assertions using an evidence-based calculus similar to that of gradual typing Garcia et al. [2016]. In addition, further work is necessary to elucidate how the "gradual guarantees" [Garcia et al. 2016] affect the relation of GV to IL. In particular, the static gradual guarantee seems to prohibit arbitrarily dropping paths, which IL can do.

Finally, our work is (to our knowledge) the first to explore a gradualization of exact logic. It remains to be seen whether this is useful its own right. For example, if library developers write exact specifications, a gradual exact logic could be used to aid development of these specifications, similar to how GV aids OX verification. But significantly more work would be necessary for this.

5 Conclusion

We have demonstrated the similarities and differences of GV and IL. Specifically, the relation between IL deductions and $\widetilde{\text{EL}}$ deductions with imprecise preconditions shows that the notion of assumptions used in GV is equivalent to the consequence rule (and path pruning) in IL. We also have shown that GV with imprecise specifications is equivalent to IL. Furthermore, we have defined gradual exact logic and used this to formally compare IL, HL, and GV. While much work remains before it is widely applicable, we hope that this framework can be used to develop techniques that uniformly target all three methods of verification.

Acknowledgments

We thank Devin Singh for his contributions to this work.

References

Johannes Bader, Jonathan Aldrich, and Éric Tanter. 2018. Gradual Program Verification. In *Verification, Model Checking, and Abstract Interpretation - 19th International Conference, VMCAI 2018, Los Angeles, CA, USA, January 7-9, 2018, Proceedings (Lecture Notes in Computer Science, Vol. 10747)*, Isil Dillig and Jens Palsberg (Eds.). Springer, 25–46. https://doi.org/10.1007/978-3-319-73721-8_2

Cristiano Calcagno, Dino Distefano, Peter W. O'Hearn, and Hongseok Yang. 2011. Compositional Shape Analysis by Means of Bi-Abduction. *J. ACM* 58, 6, Article 26 (Dec. 2011), 66 pages. https://doi.org/10.1145/2049697.2049700

Jenna Wise DiVincenzo. 2023. *Gradual Verification of Recursive Heap Data Structures*. Ph. D. Dissertation. Carnegie Mellon University.

Ronald Garcia, Alison M. Clark, and Éric Tanter. 2016. Abstracting gradual typing. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, Rastislav Bodík and Rupak Majumdar (Eds.). ACM, 429–442. https://doi.org/10.1145/2837614.2837670

C. A. R. Hoare. 1969. An Axiomatic Basis for Computer Programming. *Commun. ACM* 12, 10 (1969), 576–580. https://doi.org/10.1145/363235.363259

Quang Loc Le, Azalea Raad, Jules Villard, Josh Berdine, Derek Dreyer, and Peter W. O'Hearn. 2022. Finding real bugs in big programs with incorrectness logic. *Proc. ACM Program. Lang.* 6, OOPSLA1 (2022), 1–27. https://doi.org/10.1145/3527325

Andreas Lööw, Daniele Nantes-Sobrinho, Sacha-Élie Ayoun, Caroline Cronjäger, Petar Maksimovic, and Philippa Gardner. 2024. Compositional Symbolic Execution for Correctness and Incorrectness Reasoning. In *38th European Conference on Object-Oriented Programming, ECOOP 2024, September 16-20, 2024, Vienna, Austria (LIPIcs, Vol. 313)*, Jonathan Aldrich and Guido Salvaneschi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 25:1–25:28. https://doi.org/10.4230/LIPICS.ECOOP.2024.25

Petar Maksimovic, Caroline Cronjäger, Andreas Lööw, Julian Sutherland, and Philippa Gardner. 2023. Exact Separation Logic: Towards Bridging the Gap Between Verification and Bug-Finding. In *37th European Conference on Object-Oriented Programming, ECOOP 2023, July 17-21, 2023, Seattle, Washington, United States (LIPIcs, Vol. 263)*, Karim Ali and Guido Salvaneschi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 19:1–19:27. https://doi.org/10.4230/LIPICS.ECOOP.2023.19

Peter W. O'Hearn. 2020. Incorrectness logic. *Proc. ACM Program. Lang.* 4, POPL (2020), 10:1–10:32. https://doi.org/10.1145/3371078

Jenna Wise, Johannes Bader, Cameron Wong, Jonathan Aldrich, Éric Tanter, and Joshua Sunshine. 2020. Gradual verification of recursive heap data structures. *Proc. ACM Program. Lang.* 4, OOPSLA (2020), 228:1–228:28. https://doi.org/10.1145/3428296

Noam Zilberstein, Derek Dreyer, and Alexandra Silva. 2023. Outcome Logic: A Unifying Foundation for Correctness and Incorrectness Reasoning. *Proc. ACM Program. Lang.* 7, OOPSLA1 (2023), 522–550. https://doi.org/10.1145/3586045

Conrad Zimmerman, Jenna DiVincenzo, and Jonathan Aldrich. 2024. Sound Gradual Verification with Symbolic Execution. *Proc. ACM Program. Lang.* 8, POPL (2024), 2547–2576. https://doi.org/10.1145/3632927

A Grammar

We define a basic language that includes mutable variables and conditionals. Note that we do not include `while` loops or functions; these constructs are left for future work.

$$\begin{aligned}\text{Expr} \ni E &::= n \mid \bot \mid \top \mid x \mid E \vee E \mid E \wedge E \mid \\ &\quad E = E \mid E < E \mid \neg E \\ \text{Cmd} \ni C &::= \texttt{skip} \mid x \coloneqq E \mid \texttt{if } E \texttt{ then } C \texttt{ else } C \mid C; C \\ \text{Asrt} \ni P, Q, R &::= E \mid \neg P \mid P \wedge P \mid P \vee P \mid \exists x\, P\end{aligned}$$

where $n \in \mathbb{Z}$ and x a variable name.

We assume that programs are typed correctly; for example, expressions E in $\texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2$ will always evaluate to a boolean value.

Definition 1 (Implication)**.** $P \Rightarrow Q$ if all states that satisfy P also satisfy Q.

Note: we do not formalize states or the semantics of assertions; we assume that our logic is close enough to first-order logic and thus use FOL deductions to reason about assertions.

Note: We use $\cdot \Rightarrow \cdot$ to denote implication between formulas in first-order logic with booleans and arithmetic. $\cdot \implies \cdot$ denotes "if-then" in our metatheory.

Definition 2 (Equivalence of propositions)**.** $P \equiv Q$ denotes that P and Q are logically equivalent; that is,

$$P \equiv Q \overset{\text{def}}{\iff} P \Rightarrow Q \text{ and } Q \Rightarrow P.$$

Throughout this paper, we assume that identity of propositions coincides with equivalence; that is, when we write an individual proposition, technically we are denoting the equivalence class that contains that proposition. For example, $\{\top, \bot\} = \{1 = 1, 1 = 2\}$.

Definition 3 (Replacement)**.** $P[x/E]$ denotes P with all free occurrences of x replaced by E.

Definition 4. $\text{fv}(P)$ denotes the free variables in the assertion P.

Definition 5. $\text{fv}(C)$ denotes all variables referenced or assigned in C.

Definition 6. $\text{mod}(C)$ denotes all variables assigned in C. Explicitly,

$$\begin{aligned}\text{mod}(\texttt{skip}) &\coloneqq \emptyset \\ \text{mod}(x \coloneqq E) &\coloneqq x \\ \text{mod}(C_1; C_2) &\coloneqq \text{mod}(C_1) \cup \text{mod}(C_2) \\ \text{mod}(\texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) &\coloneqq \text{mod}(C_1) \cup \text{mod}(C_2)\end{aligned}$$

B Predicate transformers

B.1 Weakest preconditions

Definition 7. P is the *weakest precondition* for a statement C and postcondition Q, denoted $\text{wp}(C, Q)$, if it is the weakest predicate (WRT $\Rightarrow$) that ensures that if a state satisfies P, then after executing C, Q holds.

The following explicit calculations correspond to the previous definition (proved in Theorem 1):

$$\begin{aligned}&\text{wp}(\texttt{skip}, Q) = Q \\ &\text{wp}(x \coloneqq E, Q) = Q[x/E] \\ &\text{wp}(C_1; C_2, Q) = \text{wp}(C_1, \text{wp}(C_2, Q)) \\ &\text{wp}(\texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2, Q) = \\ &\quad (\text{wp}(C_1, Q) \wedge E) \vee (\text{wp}(C_2, Q) \wedge \neg E)\end{aligned}$$

Lemma 1 (Stronger postcondition)**.** *If $Q \Rightarrow Q'$ then* $\text{wp}(C, Q) \Rightarrow \text{wp}(C, Q')$.

Proof. By induction on C:

`skip`**:**

$$\text{wp}(\texttt{skip}, Q) \equiv Q \Rightarrow Q' \equiv \text{wp}(\texttt{skip}, Q')$$

$x \coloneqq E$**:**

$$\text{wp}(x \coloneqq E, Q) \equiv Q[E/x] \Rightarrow Q'[E/x] \equiv \text{wp}(x \coloneqq E, Q')$$

$C_1; C_2$**:**

(1) $\text{wp}(C_2, Q) \Rightarrow \text{wp}(C_2, Q')$ by induction

(2) $\text{wp}(C_1, \text{wp}(C_2, Q)) \Rightarrow \text{wp}(C_1, \text{wp}(C_2, Q'))$ by induction using (1)

(3) $\text{wp}(C_1; C_2, Q) \Rightarrow \text{wp}(C_1; C_2, Q')$ by (2) and definition of wp

$\texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2$**:**

(1) $\text{wp}(C_1, Q) \Rightarrow \text{wp}(C_1, Q')$ by induction

(2) $\text{wp}(C_2, Q) \Rightarrow \text{wp}(C_2, Q')$ by induction

(3) By (1) and (2)

$$(\text{wp}(C_1, Q) \wedge E) \vee (\text{wp}(C_2, Q) \wedge \neg E) \Rightarrow (\text{wp}(C_1, Q') \wedge E) \vee (\text{wp}(C_2, Q') \wedge \neg E)$$

(4) By (3) and definition of wp

$$\text{wp}(\texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2, Q) \Rightarrow \text{wp}(\texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2, Q')$$

□

B.2 Strongest postconditions

Definition 8. Q is the *strongest postcondition* for precondition P and statement C, denoted $\text{sp}(P, C)$, if it is the strongest predicate (WRT $\Rightarrow$) that ensures that if a state satisfies P, then after executing C, Q holds.

The following explicit calculations correspond to the previous definition (proved in Theorem 2):

$$\begin{aligned}&\text{sp}(P, \texttt{skip}) = P \\ &\text{sp}(P, x \coloneqq E) = \exists v (x = E[x/v] \wedge P[x/v]) \\ &\quad \text{where } v \notin \text{fv}(P) \\ &\text{sp}(P, C_1; C_2) = \text{sp}(\text{sp}(P, C_1), C_2) \\ &\text{sp}(P, \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) = \\ &\quad \text{sp}(P \wedge E, C_1) \vee \text{sp}(P \wedge \neg E, C_2)\end{aligned}$$

Lemma 2 (Stronger precondition)**.** *If $P \Rightarrow P'$ then* $\text{sp}(P, C) \Rightarrow \text{sp}(P', C)$.

Proof. By induction on C:

`skip`**:** $\text{sp}(P, \texttt{skip}) \equiv P \Rightarrow P' \equiv \text{sp}(P', \texttt{skip})$.

$x := E$: Note that $P[x/v] \Rightarrow P'[x/v]$ by logic. Then:

$$\begin{aligned}\mathrm{sp}(P, x := E) &\equiv \exists v(x = E[x/v] \wedge P[x/v]) && \text{defn sp}\\ &\Rightarrow \exists v(x = E[x/v] \wedge P'[x/v]) && \text{logic}\\ &\equiv \mathrm{sp}(P', x := E) && \text{defn sp}\end{aligned}$$

$C_1; C_2$:

(1) $\mathrm{sp}(P, C_1) \Rightarrow \mathrm{sp}(P', C_1)$ by induction

(2) $\mathrm{sp}(\mathrm{sp}(P, C_1), C_2) \Rightarrow \mathrm{sp}(\mathrm{sp}(P', C_1), C_2)$ by induction using (1)

(3) By (2) and definition of sp,

$$\begin{aligned}\mathrm{sp}(P, C_1; C_2) &\equiv \mathrm{sp}(\mathrm{sp}(P, C_1), C_2)\\ &\Rightarrow \mathrm{sp}(\mathrm{sp}(P', C_1), C_2)\\ &\equiv \mathrm{sp}(P', C_1; C_2)\end{aligned}$$

`if` E `then` C_1 `else` C_2:

(1) $P \wedge E \Rightarrow P' \wedge E$ by logic

(2) $\mathrm{sp}(P \wedge E, C_1) \Rightarrow \mathrm{sp}(P' \wedge E, C_1)$ by induction using (1)

(3) $P \wedge \neg E \Rightarrow P' \wedge \neg E$ by logic

(4) $\mathrm{sp}(P \wedge \neg E, C_2) \Rightarrow \mathrm{sp}(P' \wedge \neg E, C_2)$ by induction using (3)

(5)

$$\begin{aligned}&\mathrm{sp}(P, \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2)\\ &\quad\equiv \mathrm{sp}(P \wedge E, C_1) \vee \mathrm{sp}(P \wedge \neg E, C_2) && \text{defn sp}\\ &\quad\Rightarrow \mathrm{sp}(P' \wedge E, C_1) \vee \mathrm{sp}(P' \wedge \neg E, C_2) && \text{(2), (4), logic}\\ &\quad\equiv \mathrm{sp}(P', \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) && \text{defn sp}\ \square\end{aligned}$$

Lemma 3. $\mathrm{sp}(\bot, C) \equiv \bot$

Proof. By induction on C:

`skip`: $\mathrm{sp}(\bot, \texttt{skip}) \equiv \bot$ by definition.

$x := E$: $\mathrm{sp}(\bot, x := E) \equiv \exists v(x = E[x/v] \wedge \bot) \equiv \bot$ by logic.

`if` E `then` C_1 `else` C_2:

$$\begin{aligned}&\mathrm{sp}(\bot, \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2)\\ &\quad\equiv \mathrm{sp}(\bot \wedge E, C_1) \vee \mathrm{sp}(\bot \wedge \neg E, C_2) && \text{defn sp}\\ &\quad\equiv \mathrm{sp}(\bot, C_1) \vee \mathrm{sp}(\bot, C_2) && \text{logic}\\ &\quad\equiv \bot \vee \bot && \text{induction}\\ &\quad\equiv \bot && \text{logic}\end{aligned}$$

$C_1; C_2$:

$$\begin{aligned}\mathrm{sp}(\bot, C_1; C_2) &\equiv \mathrm{sp}(\mathrm{sp}(\bot, C_1), C_2) && \text{defn sp}\\ &\equiv \mathrm{sp}(\bot, C_2) && \text{induction}\\ &\equiv \bot && \text{induction}\ \square\end{aligned}$$

Lemma 4. $\mathrm{sp}(P, C) \equiv \bot \implies P \equiv \bot$

Note: for a more expressive language, we would also need termination for this to hold.

Proof. By induction on C:

`skip`: $\bot \equiv \mathrm{sp}(P, \texttt{skip}) \equiv P$ by definition.

$x := E$:

(1) $\bot \equiv \mathrm{sp}(P, x := E)$ by assumption

(2) $\mathrm{sp}(P, x := E) \equiv \exists v(x = E[x/v] \wedge P[x/v])$

(3) $\top \equiv \forall v(x \neq E[x/v] \vee \neg P[x/v])$ by logic using (1) and (2)

(4) $\top \equiv \forall v \neg P[x/v]$ by logic using (3)

We prove this using a semantic argument (assuming standard Kripke semantics for FOL):

Let $\mathcal{M}$ be a model and $\mathsf{y} \in |\mathcal{M}|$. By (3) we have $\mathcal{M} \vDash \forall v(x \neq E[x/v] \vee \neg P[x/v])$, and thus $\mathcal{M}[v \mapsto \mathsf{y}] \vDash x \neq E[x/v] \vee \neg P[x/v]$.

Let $\mathsf{z} = [\![E[x/v]]\!]_{\mathcal{M}[v \mapsto \mathsf{y}]}$. Then we have $\mathcal{M}[v \mapsto \mathsf{y}, x \mapsto \mathsf{z}] \nvDash x \neq E[x/v]$.

Thus we have $\mathcal{M}[v \mapsto \mathsf{y}, x \mapsto \mathsf{z}] \vDash \neg P[x/v]$. Since $x \notin \mathrm{fv}(P[x/v])$, we have $\mathcal{M}[v \mapsto \mathsf{y}] \vDash \neg P[x/v]$.

Thus we can conclude that $\mathcal{M} \vDash \forall v \neg P[x/v]$.

(5) $\top \equiv \neg P[x/v] \equiv \neg P$ by logic using (4) and since $v \notin \mathrm{fv}(P)$

(6) $\bot \equiv P$ by logic using (5)

$C_1; C_2$

(1) $\bot \equiv \mathrm{sp}(P, C_1; C_2)$ by assumption

(2) $\mathrm{sp}(P, C_1; C_2) \equiv \mathrm{sp}(\mathrm{sp}(P, C_1), C_2)$ by definition

(3) $\bot \equiv \mathrm{sp}(\mathrm{sp}(P, C_1), C_2)$ by logic using (1) and (2)

(4) $\bot \equiv \mathrm{sp}(P, C_1)$ by induction using (3)

(5) $\bot \equiv P$ by induction using (4)

[`if` E `then` C_1 `else` C_2]

(1) $\bot \equiv \mathrm{sp}(P, \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2)$ by assumption

(2) $\mathrm{sp}(P, \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) \equiv \mathrm{sp}(P \wedge E, C_1) \vee \mathrm{sp}(P \wedge \neg E, C_2)$ by definition

(3) $\bot \equiv \mathrm{sp}(P \wedge E, C_1) \vee \mathrm{sp}(P \wedge \neg E, C_2)$ by (1) and (2)

(4) $\bot \equiv \mathrm{sp}(P \wedge E, C_1)$ by logic using (3)

(5) $\bot \equiv \mathrm{sp}(P \wedge \neg E, C_2)$ by logic using (3)

(6) $\bot \equiv P \wedge E$ by induction using (4)

(7) $\bot \equiv P \wedge \neg E$ by induction using (5)

(8) $\bot \equiv (P \wedge E) \vee (P \wedge \neg E) \equiv P \wedge (E \vee \neg E) \equiv P$ by logic using (6) and (7) $\square$

Lemma 5. *If* $P \in$ *SatFormula then* $\mathrm{sp}(P, C) \in$ *SatFormula.*

Proof. We prove the contrapositive: assume $\mathrm{sp}(P, C) \notin$ SatFormula, then $\mathrm{sp}(P, C) \equiv \bot$, and thus by Lemma 4 $P \equiv \bot$. Therefore $P \notin$ SatFormula. $\square$

Lemma 6. *If* $y \notin \mathrm{fv}(C)$ *then* $\mathrm{sp}(\exists y\, P, C) \equiv \exists y\, \mathrm{sp}(P, C)$.

Proof. By induction on C:

`skip`: $\mathrm{sp}(\exists y\, P, \texttt{skip}) \equiv \exists y\, P \equiv \exists y\ \mathrm{sp}(P, \texttt{skip})$

$x := E$: Assuming $y \not\equiv v$,

$$\begin{aligned}&\mathrm{sp}(\exists y\, P, x := E)\\ &\quad\equiv \exists v(x = E[x/v] \wedge (\exists y\, P)[x/v]) && \text{defn sp}\\ &\quad\equiv \exists v(x = E[x/v] \wedge (\exists y\, P[x/v])) && y \not\equiv x \in \mathrm{fv}(x := E)\\ &\quad\equiv \exists y \exists v(x = E[x/v] \wedge P[x/v]) && y \notin \mathrm{fv}(E[x/v])\\ &\quad\equiv \exists y\ \mathrm{sp}(P, x := E) && \text{defn sp}\end{aligned}$$

$C_1; C_2$:

$$\begin{aligned}
\mathrm{sp}(\exists y\, P, C_1; C_2) &\equiv \mathrm{sp}(\mathrm{sp}(\exists y\, P, C_1), C_2) && \text{defn sp}\\
&\equiv \mathrm{sp}(\exists y\ \mathrm{sp}(P, C_1), C_2) && \text{induction}\\
&\equiv \exists y\ \mathrm{sp}(\mathrm{sp}(P, C_1), C_2) && \text{induction}\\
&\equiv \exists y\ \mathrm{sp}(P, C_1; C_2) && \text{defn sp}
\end{aligned}$$

$\texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2$:

$$\begin{aligned}
&\mathrm{sp}(\exists y\, P, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)\\
&\equiv \mathrm{sp}((\exists y\, P) \wedge E, C_1) \vee \mathrm{sp}((\exists y\, P) \wedge \neg E, C_2) && \text{defn sp}\\
&\equiv \mathrm{sp}(\exists y (P \wedge E), C_1) \vee \mathrm{sp}(\exists y (P \wedge \neg E), C_2) && y \notin \mathrm{fv}(E)\\
&\equiv (\exists y\ \mathrm{sp}(P \wedge E, C_1)) \vee (\exists y\ \mathrm{sp}(P \wedge \neg E, C_2)) && \text{induction}\\
&\equiv \exists y (\mathrm{sp}(P \wedge E, C_1) \vee \mathrm{sp}(P \wedge \neg E, C_2)) && \text{logic}\\
&\equiv \exists y\ \mathrm{sp}(P, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2) && \text{defn sp}\ \square
\end{aligned}$$

Lemma 7. $\mathrm{sp}(P_1 \vee P_2, C) \equiv \mathrm{sp}(P_1, C) \vee \mathrm{sp}(P_2, C)$

Proof. By induction on C:

$\texttt{skip}$: $\mathrm{sp}(P_1 \vee P_2, \texttt{skip}) \equiv P_1 \vee P_2 \equiv \mathrm{sp}(P_1, \texttt{skip}) \vee \mathrm{sp}(P_2, \texttt{skip})$.

$x := E$:

$$\begin{aligned}
&\mathrm{sp}(P_1 \vee P_2, x := E)\\
&\equiv \exists v (x = E[x/v] \wedge (P_1 \vee P_2)[x/v]) && \text{defn sp}\\
&\equiv \exists v (x = E[x/v] \wedge (P_1[x/v] \vee P_2[x/v])) && \text{subst.}\\
&\equiv \exists v ((x = E[x/v] \wedge P_1[x/v]) \vee (x = E[x/v] \wedge P_2[x/v])) && \text{logic}\\
&\equiv (\exists v (x = E[x/v] \wedge P_1[x/v])) \vee (\exists v (x = E[x/v] \wedge P_2[x/v])) && \text{logic}\\
&\equiv \mathrm{sp}(P_1, x := E) \vee \mathrm{sp}(P_2, x := E) && \text{defn sp}
\end{aligned}$$

$C_1; C_2$:

$$\begin{aligned}
&\mathrm{sp}(P_1 \vee P_2, C_1; C_2)\\
&\equiv \mathrm{sp}(\mathrm{sp}(P_1 \vee P_2, C_1), C_2) && \text{defn sp}\\
&\equiv \mathrm{sp}(\mathrm{sp}(P_1, C_1) \vee \mathrm{sp}(P_2, C_1), C_2) && \text{induction}\\
&\equiv \mathrm{sp}(\mathrm{sp}(P_1, C_1), C_2) \vee \mathrm{sp}(\mathrm{sp}(P_2, C_1), C_2) && \text{induction}\\
&\equiv \mathrm{sp}(P_1, C_1; C_2) \vee \mathrm{sp}(P_2, C_1; C_2) && \text{defn sp}
\end{aligned}$$

$\texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2$:

$$\begin{aligned}
&\mathrm{sp}(P_1 \vee P_2, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)\\
&\equiv \mathrm{sp}(E \wedge (P_1 \vee P_2), C_1) \vee \mathrm{sp}(\neg E \wedge (P_1 \vee P_2), C_2) && \text{defn sp}\\
&\equiv \mathrm{sp}((E \wedge P_1) \vee (E \wedge P_2), C_1) \vee \mathrm{sp}((\neg E \wedge P_1) \vee (\neg E \wedge P_2), C_2) && \text{logic}\\
&\equiv \mathrm{sp}(E \wedge P_1, C_1) \vee \mathrm{sp}(E \wedge P_2, C_1) \vee \mathrm{sp}(\neg E \wedge P_1, C_2) \vee \mathrm{sp}(\neg E \wedge P_2, C_2) && \text{Lemma 7}\\
&\equiv (\mathrm{sp}(E \wedge P_1, C_1) \vee \mathrm{sp}(\neg E \wedge P_1, C_2)) \vee (\mathrm{sp}(E \wedge P_2, C_1) \vee \mathrm{sp}(\neg E \wedge P_2, C_2)) && \text{logic}\\
&\equiv \mathrm{sp}(P_1, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2) \vee \mathrm{sp}(P_2, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2) && \text{defn sp}
\end{aligned}$$

$\square$

Lemma 8 (Frame rule). *If* $\mathrm{mod}(C) \cap \mathrm{fv}(P) = \emptyset$ *then* $\mathrm{sp}(P \wedge R, C) \equiv P \wedge \mathrm{sp}(R, C)$.

Proof. By induction on C:

$\texttt{skip}$: $\mathrm{sp}(P \wedge R, \texttt{skip}) \equiv P \wedge R \equiv P \wedge \mathrm{sp}(R, \texttt{skip})$

$x := E$: Let $v \notin \mathrm{fv}(P)$. Note that $x \in \mathrm{fv}(x := E)$, thus from our assumptions $x \notin \mathrm{fv}(P)$ Then,

$$\begin{aligned}
&\mathrm{sp}(P \wedge R, x := E)\\
&\equiv \exists v (x = E[x/v] \wedge P[x/v] \wedge R[x/v]) && \text{defn sp}\\
&\equiv \exists v (x = E[x/v] \wedge P \wedge R[x/v]) && x \notin \mathrm{fv}(P)\\
&\equiv P \wedge \exists v (x = E[x/v] \wedge R[x/v]) && v \notin \mathrm{fv}(P)\\
&\equiv P \wedge \mathrm{sp}(R, x := E) && \text{defn sp}
\end{aligned}$$

$C_1; C_2$: Note that $x \notin \mathrm{mod}(C_1; C_2)$ thus $x \notin \mathrm{mod}(C_1)$ and $x \, \mathrm{mod}(C_2)$. Then,

$$\begin{aligned}
\mathrm{sp}(P \wedge R, C_1; C_2) &\equiv \mathrm{sp}(\mathrm{sp}(P \wedge R, C_1), C_2) && \text{defn sp}\\
&\equiv \mathrm{sp}(P \wedge \mathrm{sp}(R, C_1), C_2) && \text{induction}\\
&\equiv P \wedge \mathrm{sp}(\mathrm{sp}(R, C_1), C_2) && \text{induction}\\
&\equiv P \wedge \mathrm{sp}(R, C_1; C_2) && \text{defn sp}
\end{aligned}$$

$\texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2$: Note that by assumption $x \notin \mathrm{mod}(\texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)$ thus $x \notin \mathrm{mod}(C_1)$ and $x \notin \mathrm{mod}(C_2)$. Then,

$$\begin{aligned}
&\mathrm{sp}(P \wedge R, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)\\
&\equiv \mathrm{sp}(P \wedge R \wedge E, C_1) \vee \mathrm{sp}(P \wedge R \wedge \neg E, C_2) && \text{defn sp}\\
&\equiv (P \wedge \mathrm{sp}(R \wedge E, C_1)) \vee (P \wedge \mathrm{sp}(R \wedge \neg E, C_2)) && \text{ind.}\\
&\equiv P \wedge (\mathrm{sp}(R \wedge E, C_1) \vee \mathrm{sp}(R \wedge \neg E, C_2)) && \text{logic}\\
&\equiv P \wedge \mathrm{sp}(R, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2) && \text{defn sp}\ \square
\end{aligned}$$

Lemma 9.

$$\mathrm{sp}(P \wedge E, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2) \equiv \mathrm{sp}(P \wedge E, C_1)$$
$$\mathrm{sp}(P \wedge \neg E, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2) \equiv \mathrm{sp}(P \wedge \neg E, C_2)$$

Proof.

$$\begin{aligned}
&\mathrm{sp}(P \wedge E, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)\\
&\equiv \mathrm{sp}(P \wedge E \wedge E, C_1) \vee \mathrm{sp}(P \wedge E \wedge \neg E, C_2) && \text{defn sp}\\
&\equiv \mathrm{sp}(P \wedge E, C_1) \vee \mathrm{sp}(\bot, C_2) && \text{logic}\\
&\equiv \mathrm{sp}(P \wedge E, C_1) \vee \bot && \text{Lemma 3}\\
&\equiv \mathrm{sp}(P \wedge E, C_1) && \text{logic}
\end{aligned}$$

$$\begin{aligned}
&\mathrm{sp}(P \wedge \neg E, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)\\
&\equiv \mathrm{sp}(P \wedge E \wedge \neg E, C_1) \vee \mathrm{sp}(P \wedge \neg E \wedge \neg E, C_2) && \text{defn sp}\\
&\equiv \mathrm{sp}(\bot, C_1) \vee \mathrm{sp}(P \wedge \neg E, C_2) && \text{logic}\\
&\equiv \bot \vee \mathrm{sp}(P \wedge \neg E, C_2) && \text{Lemma 3}\\
&\equiv \mathrm{sp}(P \wedge \neg E, C_2) && \text{logic}\ \square
\end{aligned}$$

B.3 Fixpoint

We can define a function $Q \mapsto \mathrm{sp}(\mathrm{wp}(C, Q), C)$. We demonstrate that this function reaches a fixpoint.

Lemma 10. $\mathrm{sp}(\mathrm{wp}(C, Q) \wedge P, C) \equiv Q \wedge \mathrm{sp}(P, C)$

Proof. By induction on C:

`skip`: $\mathrm{sp}(\mathrm{wp}(\texttt{skip}, Q) \wedge P, \texttt{skip}) \equiv Q \wedge P \equiv Q \wedge \mathrm{sp}(P, \texttt{skip})$

$x \coloneqq E$: Let $v \notin \mathrm{fv}(Q)$. Then,

$$\begin{aligned}
&\mathrm{sp}(\mathrm{wp}(x \coloneqq E, Q) \wedge P, x \coloneqq E) \\
&\quad\equiv \exists v (x = E[x/v] \wedge \\
&\qquad (\mathrm{wp}(x \coloneqq E, Q) \wedge P)[x/v]) && \text{defn sp} \\
&\quad\equiv \exists v (x = E[x/v] \wedge (Q[x/E] \wedge P)[x/v]) && \text{defn wp} \\
&\quad\equiv \exists v (x = E[x/v] \wedge Q[x/E[x/v]] \wedge P[x/v]) && \text{subst} \\
&\quad\equiv \exists v (x = E[x/v] \wedge Q[x/x] \wedge P[x/v]) && \text{subst} = \\
&\quad\equiv \exists v (x = E[x/v] \wedge Q \wedge P[x/v]) && \text{redundant} \\
&\quad\equiv Q \wedge (\exists v\, x = E[x/v] \wedge P[x/v]) && v \notin \mathrm{fv}(Q) \\
&\quad\equiv Q \wedge \mathrm{sp}(P, x \coloneqq E) && \text{defn sp}
\end{aligned}$$

`if` E `then` C_1 `else` C_2:

(1) $Q \wedge \mathrm{sp}(P \wedge E, C_1) \equiv \mathrm{sp}(\mathrm{wp}(C_1, Q) \wedge P \wedge E, C_1)$ by induction using C_1

(2) $Q \wedge \mathrm{sp}(P \wedge \neg E, C_2) \equiv \mathrm{sp}(\mathrm{wp}(C_2, Q) \wedge P \wedge \neg E, C_2)$ by induction using C_2

(3)

$$\begin{aligned}
&Q \wedge \mathrm{sp}(P, \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) \\
&\quad\equiv Q \wedge (\mathrm{sp}(P \wedge E, C_1) \vee \mathrm{sp}(P \wedge \neg E, C_2)) && \text{defn sp} \\
&\quad\equiv (Q \wedge \mathrm{sp}(P \wedge E, C_1)) \vee \\
&\qquad (Q \wedge \mathrm{sp}(P \wedge \neg E, C_2)) && \text{logic} \\
&\quad\equiv \mathrm{sp}(\mathrm{wp}(C_1, Q) \wedge P \wedge E, C_1) \vee \\
&\qquad \mathrm{sp}(\mathrm{wp}(C_2, Q) \wedge P \wedge \neg E, C_2) && \text{(1), (2)} \\
&\quad\equiv \mathrm{sp}(\mathrm{wp}(C_1, Q) \wedge P \wedge E, \\
&\qquad\quad \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) \vee \\
&\qquad \mathrm{sp}(\mathrm{wp}(C_2, Q) \wedge P \wedge \neg E, \\
&\qquad\quad \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) && \text{Lemma 9} \\
&\quad\equiv \mathrm{sp}(((\mathrm{wp}(C_1, Q) \wedge E) \vee \\
&\qquad\quad (\mathrm{wp}(C_2, Q) \wedge \neg E)) \wedge P, \\
&\qquad\quad \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) && \text{Lemma 7} \\
&\quad\equiv \mathrm{sp}(\mathrm{wp}(\texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2, Q) \wedge P, \\
&\qquad\quad \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) && \text{defn wp}
\end{aligned}$$

$C_1; C_2$:

(1) $\mathrm{wp}(C_2, Q) \wedge \mathrm{sp}(P, C_1) \equiv \mathrm{sp}(\mathrm{wp}(C_1, \mathrm{wp}(C_2, Q)) \wedge P, C_1)$ by induction using C_1

(2) $Q \wedge \mathrm{sp}(\mathrm{sp}(P, C_1), C_2) \equiv \mathrm{sp}(\mathrm{wp}(C_2, Q) \wedge \mathrm{sp}(P, C_1), C_2)$ by induction using C_2. Then,

$$\begin{aligned}
&Q \wedge \mathrm{sp}(P, C_1; C_2) \\
&\quad\equiv Q \wedge \mathrm{sp}(\mathrm{sp}(P, C_1), C_2) && \text{defn sp} \\
&\quad\equiv \mathrm{sp}(\mathrm{wp}(C_2, Q) \wedge \mathrm{sp}(P, C_1), C_2) && \text{(2)} \\
&\quad\equiv \mathrm{sp}(\mathrm{sp}(\mathrm{wp}(C_1, \mathrm{wp}(C_2, Q)) \wedge P, C_1), C_2) && \text{(1)} \\
&\quad\equiv \mathrm{sp}(\mathrm{sp}(\mathrm{wp}(C_1; C_2, Q) \wedge P, C_1), C_2) && \text{defn wp} \\
&\quad\equiv \mathrm{sp}(\mathrm{wp}(C_1; C_2, Q) \wedge P, C_1; C_2) && \text{defn sp}\ \square
\end{aligned}$$

Lemma 11 (Fixpoint property). *If* $Q \Rightarrow \mathrm{sp}(\top, C)$ *then* $Q \equiv \mathrm{sp}(\mathrm{wp}(C, Q), C)$.

Proof. Immediate from Lemma 10, noting that $Q \equiv Q \wedge \mathrm{sp}(\top, C)$ in this case. $\square$

Lemma 12. *If* $Q \Rightarrow \mathrm{sp}(P, C)$ *then* $Q \equiv \mathrm{sp}(P \wedge \mathrm{wp}(C, Q), C)$.

Proof.

$$\begin{aligned}
Q &\equiv Q \wedge \mathrm{sp}(P, C) && Q \Rightarrow \mathrm{sp}(P, C) \\
&\equiv \mathrm{sp}(P \wedge \mathrm{wp}(C, Q), C) && \text{Lemma 10}
\end{aligned}$$ $\square$

C Hoare logic

Hoare triples are denoted by $\vdash \{P\}\ C\ \{Q\}$. Deductions in Hoare logic are characterized by the following rules:

$$\frac{}{\vdash \{P\}\ \texttt{skip}\ \{P\}}\ \textsc{OX-Skip} \qquad \frac{}{\vdash \{P[x/E]\}\ x \coloneqq E\ \{P\}}\ \textsc{OX-Assign}$$

$$\frac{\vdash \{P\}\ C_1\ \{R\} \quad \vdash \{R\}\ C_2\ \{Q\}}{\vdash \{P\}\ C_1; C_2\ \{Q\}}\ \textsc{OX-Seq} \qquad \frac{\vdash \{E \wedge P\}\ C_1\ \{Q\} \quad \vdash \{\neg E \wedge P\}\ C_2\ \{Q\}}{\vdash \{P\}\ \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2\ \{Q\}}\ \textsc{OX-If}$$

$$\frac{P \Rightarrow P' \qquad \vdash \{P'\}\ C\ \{Q'\} \qquad Q' \Rightarrow Q}{\vdash \{P\}\ C\ \{Q\}}\ \textsc{OX-Cons}$$

C.1 Weakest preconditions

Lemma 13. *If* $\vdash \{P\}\ C\ \{Q\}$ *then* $P \Rightarrow \mathrm{wp}(C, Q)$.

Proof. By induction on the derivation of $\vdash \{P\}\ C\ \{Q\}$:

OX-Skip:

(1) $P \equiv Q$ by inversion

(2) $P \equiv Q \equiv \mathrm{wp}(\texttt{skip}, Q)$ by definition

OX-Assign:

(1) $P \equiv Q[E/x]$ by inversion

(2) $P \equiv Q[E/x] \equiv \mathrm{wp}(x \coloneqq E, Q)$ by definition

OX-Seq:

(1) $\vdash \{P\}\ C_1\ \{R\}$ for some R by inversion

(2) $P \Rightarrow \mathrm{sp}(C_1, R)$ by induction using (1)

(3) $\vdash \{R\}\ C_2\ \{Q\}$ by inversion

(4) $R \Rightarrow \mathrm{wp}(C_2, Q)$ by induction using (3)

(5) $\mathrm{wp}(C_1, R) \Rightarrow \mathrm{wp}(C_1, \mathrm{wp}(C_2, Q))$ by Lemma 1 using (4)

(6) By (4), (5), and definition of wp,

$$\begin{aligned}
P &\Rightarrow \mathrm{wp}(C_1, R) \\
&\Rightarrow \mathrm{wp}(C_1, \mathrm{wp}(C_2, Q)) \\
&\equiv \mathrm{wp}(C_1; C_2, Q)
\end{aligned}$$

OX-If:

(1) $\vdash \{E \wedge P\}\ C_1\ \{Q\}$ by inversion

(2) $E \wedge P \Rightarrow \mathrm{wp}(C_1, Q)$ by induction using (1)

(3) $\vdash \{\neg E \wedge P\}\ C_2\ \{Q\}$ by inversion

(4) $\neg E \wedge P \Rightarrow \mathrm{wp}(C_2, Q)$ by induction using (3)

(5)

$P \equiv (P \wedge E) \vee (P \wedge \neg E)$ logic
$\Rightarrow (\mathrm{wp}(C_1, Q) \wedge E) \vee (\mathrm{wp}(C_2, Q) \wedge \neg E)$ (2) and (4)
$\Rightarrow \mathrm{wp}(\texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2, Q)$ defn wp

OX-Cons:

(1) $P \Rightarrow P'$ for some P' by inversion
(2) $Q' \Rightarrow Q$ for some Q' by inversion
(3) $\vdash \{P'\}\ C\ \{Q'\}$ by inversion
(4) $P' \Rightarrow \mathrm{wp}(C, Q')$ by induction using (3)
(5) $\mathrm{wp}(C, Q') \Rightarrow \mathrm{wp}(C, Q)$ by Lemma 1 using (4)
(6) $P \Rightarrow P' \Rightarrow \mathrm{wp}(C, Q') \Rightarrow \mathrm{wp}(C, Q)$ by (1), (4), and (5) □

Lemma 14. $\vdash \{\mathrm{wp}(C, Q)\}\ C\ \{Q\}$

Proof. By induction on C:

`skip`: $\mathrm{wp}(\texttt{skip}, Q) \equiv Q$ by definition; $\vdash (Q)\ \texttt{skip}\ (Q)$ by OX-Skip.

$x := E$:

(1) $\mathrm{wp}(x := E, Q) \equiv Q[x/E]$ by definition
(2) $\vdash \{Q[x/E]\}\ x := E\ \{Q\}$ by OX-Assign using (1)

$C_1; C_2$:

(1) $\vdash \{\mathrm{wp}(C_1, \mathrm{wp}(C_2, Q))\}\ C_1\ \{\mathrm{wp}(C_2, Q)\}$ by induction
(2) $\vdash \{\mathrm{wp}(C_2, Q)\}\ C_2\ \{Q\}$ by induction
(3) $\vdash \{\mathrm{wp}(C_1, \mathrm{wp}(C_2, Q))\}\ C_1; C_2\ \{Q\}$ by OX-Seq using (1) and (2)
(4) $\mathrm{wp}(C_1; C_2, Q) \equiv \mathrm{wp}(C_1, \mathrm{wp}(C_2, Q))$ by definition
(5) $\vdash \{\mathrm{wp}(C_1; C_2, Q)\}\ C_1; C_2\ \{Q\}$ by (3) and (4)

$\texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2$:

(1) Let $P \equiv (\mathrm{wp}(C_1, Q) \wedge E) \vee (\mathrm{wp}(C_2, Q) \wedge \neg E)$
(2) $\vdash \{\mathrm{wp}(C_1, Q)\}\ C_1\ \{Q\}$ by induction
(3) $P \wedge E \Rightarrow \mathrm{wp}(C_1, Q)$ by logic
(4) $\vdash \{P \wedge E\}\ C_1\ \{Q\}$ by OX-Cons using (2) and (3)
(5) $\vdash \{\mathrm{wp}(C_2, Q)\}\ C_2\ \{Q\}$ by induction
(6) $P \wedge \neg E \Rightarrow \mathrm{wp}(C_2, Q)$ by logic
(7) $\vdash \{P \wedge \neg E\}\ C_2\ \{Q\}$ by OX-Cons using (5) and (6)
(8) $\vdash \{P\}\ \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2\ \{Q\}$ by OX-If using (4) and (7)
(9) $P \equiv \mathrm{wp}(\texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2, Q)$ by (1) and definition of wp
(10) $\vdash \{\mathrm{wp}(\texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2)\}\ \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2\ \{Q\}$ by (8) and (9) □

Lemma 15. *If* $P \Rightarrow \mathrm{wp}(C, Q)$ *then* $\vdash \{P\}\ C\ \{Q\}$.

Proof. By Lemma 14, $\vdash \{\mathrm{wp}(C, Q)\}\ C\ \{Q\}$, thus by OX-Cons $\vdash \{P\}\ C\ \{Q\}$. □

Theorem 1. $P \Rightarrow \mathrm{wp}(C, Q) \iff \vdash \{P\}\ C\ \{Q\}$.

Proof. $\Longrightarrow$: Lemma 15; $\Longleftarrow$: Lemma 13. □

C.2 Strongest postconditions

Lemma 16. *If* $\vdash \{P\}\ C\ \{Q\}$ *then* $\mathrm{sp}(P, C) \Rightarrow Q$.

Proof. By induction on $\vdash \{P\}\ C\ \{Q\}$:

OX-Skip: Then $P \equiv Q$ and $C \equiv \texttt{skip}$, thus $\mathrm{sp}(P, \texttt{skip}) \equiv P \equiv Q$.

OX-Assign: Then $P \equiv Q[x/E]$ and $C \equiv x := E$. Assuming $v \notin \mathrm{fv}(Q)$:

$\mathrm{sp}(Q[x/E], x := E)$
$\equiv \exists v(x = E[x/v] \wedge Q[x/E][x/v])$ defn
$\equiv \exists v(x = E[x/v] \wedge Q[x/E[x/v]])$ substitution
$\equiv \exists v(x = E[x/v] \wedge Q[x/x])$ subst. equality
$\equiv \exists v(x = E[x/v] \wedge Q)$ redundant
$\Rightarrow Q$ $v \notin \mathrm{fv}(Q)$

OX-Seq:

(1) $C \equiv C_1; C_2$ for some C_1, C_2 by inversion
(2) $\vdash \{P\}\ C_1\ \{R\}$ for some R by inversion
(3) $\vdash \{R\}\ C_2\ \{Q\}$ by inversion
(4) $\mathrm{sp}(P, C_1) \Rightarrow R$ by induction using (2)
(5) $\mathrm{sp}(\mathrm{sp}(P, C_1), C_2) \Rightarrow \mathrm{sp}(R, C_2)$ by Lemma 2 using (4)
(6) $\mathrm{sp}(R, C_2) \Rightarrow Q$ by induction using (3)
(7) $\mathrm{sp}(\mathrm{sp}(P, C_1), C_2) \Rightarrow Q$ by (5) and (6)
(8) $\mathrm{sp}(P, C) \equiv \mathrm{sp}(P, C_1; C_2) \Rightarrow Q$ by (1), (7), and definition of sp

OX-If:

(1) $C \equiv \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2$ for some E, C_1, C_2 by inversion
(2) $\vdash \{E \wedge P\}\ C_1\ \{Q\}$ by inversion
(3) $\vdash \{\neg E \wedge P\}\ C_2\ \{Q\}$ by inversion
(4) $\mathrm{sp}(E \wedge P, C_1) \Rightarrow Q$ by induction using (2)
(5) $\mathrm{sp}(\neg E \wedge P, C_2) \Rightarrow Q$ by induction using (3)
(6) $\mathrm{sp}(E \wedge P, C_1) \vee \mathrm{sp}(\neg E \wedge P, C_2) \Rightarrow Q$ by logic using (4) and (5)
(7) $\mathrm{sp}(P, \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) \Rightarrow Q$ by (6) and definition of sp

OX-Cons:

(1) $P \Rightarrow P'$ for some P' by inversion
(2) $Q' \Rightarrow Q$ for some Q' by inversion
(3) $\vdash \{P'\}\ C\ \{Q'\}$ by inversion
(4) $\mathrm{sp}(P, C) \Rightarrow \mathrm{sp}(P', C)$ by Lemma 2 and (1)
(5) $\mathrm{sp}(P', C) \Rightarrow Q'$ by induction using (3)
(6) $\mathrm{sp}(P, C) \Rightarrow Q$ by (4), (5), and (2) □

Lemma 17. $\vdash \{P\}\ C\ \{\mathrm{sp}(P, C)\}$

Proof. By induction on C:

`skip`:

(1) $\vdash \{P\}\ \texttt{skip}\ \{P\}$ by OX-Skip
(2) $P \equiv \mathrm{sp}(P, \texttt{skip})$ by OX-Skip
(3) $\vdash \{P\}\ \texttt{skip}\ \{\mathrm{sp}(P, \texttt{skip})\}$ by OX-Cons using (1) and (2)

$x := E$:

(1) Let $Q \equiv \exists v(x = E[x/v] \wedge P[x/v])$ where $v \notin \mathrm{fv}(P)$.
(2) $\vdash \{Q[E/x]\}\ x := E\ \{Q\}$ by OX-Assign

(3) $P \Rightarrow \exists v(E = E[x/v] \wedge P[x/v]) \equiv Q[x/E]$ by logic (witnessed by letting $v = x$).
(4) $\vdash \{P\}\ x := E\ \{Q\}$ by OX-Cons using (2) and (3)
(5) $\vdash \{P\}\ x := E\ \{\mathrm{sp}(P, x := E)\}$ by (1), (4), and definition of sp

$C_1; C_2$**:**
(1) $\vdash \{P\}\ C_1\ \{\mathrm{sp}(P, C_1)\}$ by induction
(2) $\vdash \{\mathrm{sp}(P, C_1)\}\ C_2\ \{\mathrm{sp}(\mathrm{sp}(P, C_1), C_2)\}$ by induction
(3) $\vdash \{P\}\ C_1; C_2\ \{\mathrm{sp}(\mathrm{sp}(P, C_1), C_2)\}$ by OX-Seq using (1) and (2)
(4) $\mathrm{sp}(P, C_1; C_2) \equiv \mathrm{sp}(\mathrm{sp}(P, C_1), C_2)$ by definition
(5) $\vdash \{P\}\ C_1; C_2\ \{\mathrm{sp}(P, C_1; C_2)\}$ by (3) and (4)

`if` E `then` C_1 `else` C_2**:**
(1) $\vdash \{P \wedge E\}\ C_1\ \{\mathrm{sp}(P \wedge E, C_1)\}$ by induction
(2) $\vdash \{P \wedge \neg E\}\ C_2\ \{\mathrm{sp}(P \wedge \neg E, C_2)\}$ by induction
(3) $\vdash \{P \wedge E\}\ C_1\ \{\mathrm{sp}(P \wedge E, C_1) \vee \mathrm{sp}(P \wedge \neg E, C_2)\}$ by OX-Cons using (1)
(4) $\vdash \{P \wedge \neg E\}\ C_2\ \{\mathrm{sp}(P \wedge E, C_1) \vee \mathrm{sp}(P \wedge \neg E, C_2)\}$ by OX-Cons using (2)
(5) $\vdash \{P\}$ `if` E `then` C_1 `else` $C_2\ \{\mathrm{sp}(P \wedge E, C_1) \vee \mathrm{sp}(P \wedge \neg E, C_2)\}$ by OX-If using (3) and (4)
(6) $\vdash \{P\}$ `if` E `then` C_1 `else` C_2 $\{\mathrm{sp}(P,$ `if` E `then` C_1 `else` $C_2)\}$ by (5) and definition of sp □

Lemma 18. *If* $\mathrm{sp}(P, C) \Rightarrow Q$ *then* $\vdash \{P\}\ C\ \{Q\}$.

Proof. Immediate from Lemma 17, applying OX-Cons. □

Theorem 2. $\mathrm{sp}(P, C) \Rightarrow Q \iff \vdash \{P\}\ C\ \{Q\}$

Proof. $\Longrightarrow$: Lemma 18; $\Longleftarrow$: Lemma 16 □

D Incorrectness logic

Incorrectness logic triples are denoted $\vdash [P]\ C\ [Q]$. Deductions in incorrectness logic are defined as follows:

$$\text{UX-Skip}\ \frac{}{\vdash [P]\ \texttt{skip}\ [P]} \qquad \text{UX-Seq}\ \frac{\vdash [P]\ C_1\ [Q] \qquad \vdash [Q]\ C_2\ [R]}{\vdash [P]\ C_1; C_2\ [R]}$$

$$\text{UX-Assign}\ \frac{}{\vdash [P]\ x := E\ [\exists v(x = E[x/v] \wedge P[x/v])]}$$

$$\text{UX-IfThen}\ \frac{\vdash [E \wedge P]\ C_1\ [Q]}{\vdash [E \wedge P]\ \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ [Q]} \qquad \text{UX-IfElse}\ \frac{\vdash [\neg E \wedge P]\ C_2\ [Q]}{\vdash [\neg E \wedge P]\ \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ [Q]}$$

$$\text{UX-Cons}\ \frac{P' \Rightarrow P \quad \vdash [P']\ C\ [Q'] \quad Q \Rightarrow Q'}{\vdash [P]\ C\ [Q]} \qquad \text{UX-Disj}\ \frac{\vdash [P_1]\ C\ [Q_1] \quad \vdash [P_2]\ C\ [Q_2]}{\vdash [P_1 \vee P_2]\ C\ [Q_1 \vee Q_2]}$$

D.1 Strongest postconditions

Lemma 19. $\vdash [P]\ C\ [\mathrm{sp}(P, C)]$

Proof. By induction on C:

`skip`**:** By UX-Skip $\vdash [P]$ `skip` $[P]$ and $P \equiv \mathrm{sp}(P,$ `skip`$)$ by definition.

$x := E$**:**
(1) $\vdash [P]\ x := E\ [\exists v(x = E[x/v] \wedge P[x/v])]$ by UX-Assign
(2) $\mathrm{sp}(P, x := E) \equiv \exists v(x = E[x/v] \wedge P[x/v])$ by definition of sp
(3) $\vdash [P]\ x := E\ [sp(P, x := E)]$ by (1) and (2)

$C_1; C_2$**:**
(1) $\vdash [P]\ C_1\ [\mathrm{sp}(P, C_1)]$ by induction
(2) $\vdash [\mathrm{sp}(P, C_1)]\ C_2\ [\mathrm{sp}(\mathrm{sp}(P, C_1), C_2)]$ by induction
(3) $\vdash [P]\ C_1; C_2\ [\mathrm{sp}(\mathrm{sp}(P, C_1), C_2)]$ by UX-Seq using (1), (2)
(4) $\mathrm{sp}(P, C_1; C_2) \equiv \mathrm{sp}(\mathrm{sp}(P, C_1), C_2)$ by definition
(5) $\vdash [P]\ C_1; C_2\ [\mathrm{sp}(P, C_1; C_2)]$ by (3), (4)

`if` E `then` C_1 `else` C_2**:**
(1) $\vdash [E \wedge P]\ C_1\ [\mathrm{sp}(E \wedge P, C_1)]$ by induction
(2) $\vdash [\neg E \wedge P]\ C_2\ [\mathrm{sp}(\neg E \wedge P, C_2)]$ by induction
(3) $\vdash [E \wedge P]$ `if` E `then` C_1 `else` $C_2\ [\mathrm{sp}(E \wedge P, C_1)]$ by UX-IfThen using (1)
(4) $\vdash [\neg E \wedge P]$ `if` E `then` C_1 `else` $C_2\ [\mathrm{sp}(\neg E \wedge P, C_2)]$ by UX-IfElse using (2)
(5) $\vdash [P]$ `if` E `then` C_1 `else` $C_2\ [\mathrm{sp}(E \wedge P, C_1) \vee \mathrm{sp}(\neg E \wedge P, C_2)]$ by UX-Disj using (3) and (4)
(6) $\mathrm{sp}(P,$ `if` E `then` C_1 `else` $C_2) \equiv \mathrm{sp}(E \wedge P, C_1) \vee \mathrm{sp}(\neg E \wedge P, C_2)$ by definition
(7) $\vdash [P]$ `if` E `then` C_1 `else` C_2 $[\mathrm{sp}(P,$ `if` E `then` C_1 `else` $C_2)]$ by (5) and (6) □

Lemma 20. *If* $Q \Rightarrow \mathrm{sp}(P, C)$ *then* $\vdash [P]\ C\ [Q]$.

Proof. Immediate from Lemma 19 and UX-Cons. □

Lemma 21. *If* $\vdash [P]\ C\ [Q]$ *then* $Q \Rightarrow \mathrm{sp}(P, C)$.

Proof. By induction on the derivation $\vdash [P]\ C\ [Q]$:

UX-Skip:
(1) $Q \equiv P$ by inversion
(2) $\mathrm{sp}(P,$ `skip`$) \equiv P$ by definition
(3) $Q \equiv \mathrm{sp}(P,$ `skip`$)$ by (1) and (2)

UX-Assign:
(1) $Q \equiv \exists v(x = E[x/v] \wedge P[x/v])$ by inversion
(2) $\mathrm{sp}(P, x := E) \equiv \exists v(x = E[x/v] \wedge P[x/v])$ by definition
(3) $Q \equiv \mathrm{sp}(P, x := E)$ by (1) and (2)

UX-Seq:
(1) $\vdash [P]\ C_1\ [R]$ for some R by inversion
(2) $\vdash [R]\ C_2\ [Q]$ by inversion
(3) $Q \Rightarrow \mathrm{sp}(R, C_2)$ by induction using (2)
(4) $R \Rightarrow \mathrm{sp}(P, C_1)$ by induction using (1)
(5) $\mathrm{sp}(R, C_2) \Rightarrow \mathrm{sp}(\mathrm{sp}(P, C_1), C_2)$ by Lemma 2 using (4)

(6)

$Q \Rightarrow \mathrm{sp}(R, C_2)$ (3)
$\Rightarrow \mathrm{sp}(\mathrm{sp}(P, C_1), C_2)$ (5)
$\equiv \mathrm{sp}(P, C_1; C_2)$ defn sp

UX-IfThen:

(1) $C \equiv \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2$ for some E, C_1, C_2 by inversion
(2) $P \equiv E \wedge P'$ for some P' by inversion
(3) $\vdash [E \wedge P']\ C_1\ [Q]$ by inversion
(4) $Q \Rightarrow \mathrm{sp}(E \wedge P', C_1)$ by induction using (3)
(5) $\mathrm{sp}(E \wedge P', C_1) \equiv \mathrm{sp}(E \wedge P', \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)$ by Lemma 9
(6)

$Q \Rightarrow \mathrm{sp}(E \wedge P', C_1)$ (4)
$\equiv \mathrm{sp}(E \wedge P', \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)$ (5)
$\equiv \mathrm{sp}(P, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)$ (2)

UX-IfElse:

(1) $C \equiv \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2$ for some E, C_1, C_2 by inversion
(2) $P \equiv \neg E \wedge P'$ for some P' by inversion
(3) $\vdash [\neg E \wedge P']\ C_2\ [Q]$ by inversion
(4) $Q \Rightarrow \mathrm{sp}(\neg E \wedge P', C_2)$ by induction using (3)
(5) $\mathrm{sp}(\neg E \wedge P', C_2) \equiv \mathrm{sp}(\neg E \wedge P', \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)$ by Lemma 9
(6)

$Q \Rightarrow \mathrm{sp}(\neg E \wedge P', C_2)$ (4)
$\equiv \mathrm{sp}(\neg E \wedge P', \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)$ (5)
$\equiv \mathrm{sp}(P, \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2)$ (2)

UX-Cons:

(1) $\vdash [P']\ C\ [Q']$ for some P', Q' by inversion
(2) $P' \Rightarrow P$ by inversion
(3) $Q \Rightarrow Q'$ by inversion
(4) $Q' \Rightarrow \mathrm{sp}(P', C)$ by induction using (1)
(5) $\mathrm{sp}(P', C) \Rightarrow \mathrm{sp}(P, C)$ by Lemma 2 using (2)
(6) $Q \Rightarrow Q' \Rightarrow \mathrm{sp}(P', C) \Rightarrow \mathrm{sp}(P, C)$ by (3), (4), and (5)

UX-Disj:

(1) $\vdash [P_1]\ C\ [Q_1]$ for some P_1, Q_1 by inversion
(2) $\vdash [P_2]\ C\ [Q_2]$ for some P_2, Q_2 by inversion
(3) $P \equiv P_1 \vee P_2$ by inversion
(4) $Q \equiv Q_1 \vee Q_2$ by inversion
(5) $Q_1 \Rightarrow \mathrm{sp}(P_1, C)$ by induction using (1)
(6) $Q_2 \Rightarrow \mathrm{sp}(P_2, C)$ by induction using (2)
(7) $Q_1 \vee Q_2 \Rightarrow \mathrm{sp}(P_1, C) \vee \mathrm{sp}(P_2, C)$ by logic using (5) and (6)
(8)

$Q \equiv Q_1 \vee Q_2$ (4)
$\Rightarrow \mathrm{sp}(P_1, C) \vee \mathrm{sp}(P_2, C)$ (7)
$\equiv \mathrm{sp}(P_1 \vee P_2, C)$ Lemma 7
$\equiv \mathrm{sp}(P, C)$ (3) □

Theorem 3. $\vdash [P]\ C\ [Q] \iff Q \Rightarrow \mathrm{sp}(P, C)$

Proof. $\Longrightarrow$: Lemma 21; $\Longleftarrow$: Lemma 20. □

D.2 Satisfiability

Lemma 22. *If* $\vdash [\bot]\ C\ [Q]$ *then* $Q \equiv \bot$.

Proof.

(1) $Q \Rightarrow \mathrm{sp}(\bot, C)$ by Theorem 2
(2) $\mathrm{sp}(\bot, C) \equiv \bot$ by Lemma 3
(3) $Q \Rightarrow \bot$ logic using (1) and (2)
(4) $Q \equiv \bot$ by logic using (3) □

Lemma 23. *If* $\vdash [P]\ C\ [Q]$ *and* $Q \in$ *SatFormula then* $P \in$ *SatFormula.*

Proof. Assume $\vdash [P]\ C\ [Q]$, then we prove the contrapositive; that is, $P \notin \textsc{SatFormula} \implies Q \notin \textsc{SatFormula}$.

Assuming $P \notin \textsc{SatFormula}$, since $P \in \textsc{Formula}$ we get $P \equiv \bot$, and thus $\vdash [\bot]\ C\ [Q]$ by assumption. Then by Lemma 22, $Q \equiv \bot$ thus $Q \notin \textsc{SatFormula}$. □

Lemma 24. *If* $\vdash [P]\ C\ [Q]$ *and* $Q \in$ *SatFormula then* $P \wedge \mathrm{wp}(P, C) \in$ *SatFormula.*

Proof.

(1) $\vdash [P]\ C\ [Q]$ by assumption
(2) $Q \in \textsc{SatFormula}$ by assumption
(3) $Q \Rightarrow \mathrm{sp}(P, C)$ by Lemma 21 using (1)
(4) $\mathrm{sp}(P \wedge \mathrm{wp}(P, C), C) \equiv Q$ by Lemma 12 using (3)
(5) $\vdash [P \wedge \mathrm{wp}(P, C)]\ C\ [Q]$ by Lemma 20 using (4)
(6) $P \wedge \mathrm{wp}(C, Q) \in \textsc{SatFormula}$ by Lemma 23 using (2) and (5) □

E Exact logic

Exact logic triples are denoted by $\vdash (P)\ C\ (Q)$. Deductions in exact logic are characterized by the following rules:

$$\textsc{Ex-Skip}\ \frac{}{\vdash (\top)\ \texttt{skip}\ (\top)} \qquad \textsc{Ex-Assign}\ \frac{x \notin \mathrm{fv}(E')}{\vdash (x = E')\ x := E\ (x = E[x/E'])}$$

$$\textsc{Ex-IfThen}\ \frac{\vdash (P \wedge E)\ C_1\ (Q)}{\vdash (P \wedge E)\ \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ (Q)} \qquad \textsc{Ex-Seq}\ \frac{\vdash (P)\ C_1\ (R) \quad \vdash (R)\ C_2\ (Q)}{\vdash (P)\ C_1; C_2\ (Q)}$$

$$\textsc{Ex-IfElse}\ \frac{\vdash (P \wedge \neg E)\ C_2\ (Q)}{\vdash (P \wedge \neg E)\ \texttt{if}\ E\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ (Q)} \qquad \textsc{Ex-Frame}\ \frac{\mathrm{mod}(C) \cap \mathrm{fv}(R) = \emptyset \quad \vdash (P)\ C\ (Q)}{\vdash (P \wedge R)\ C\ (Q \wedge R)}$$

$$\textsc{Ex-Exists}\ \frac{\vdash (P)\ C\ (Q) \quad x \notin \mathrm{fv}(C)}{\vdash (\exists x\, P)\ C\ (\exists x\, Q)} \qquad \textsc{Ex-Disj}\ \frac{\vdash (P_1)\ C\ (Q_1) \quad \vdash (P_2)\ C\ (Q_2)}{\vdash (P_1 \vee P_2)\ C\ (Q_1 \vee Q_2)}$$

Note: We drop the equivalence rule, since it is immediately valid by our characterization of formulas by equivalence classes.

E.1 Strongest postconditions

Lemma 25. $\vdash (P)\ C\ (\mathrm{sp}(P, C))$

Proof. By induction on C:

`skip`**:** By EX-Skip $\vdash (P)$ `skip` (P) and $P \equiv \mathrm{sp}(P, \texttt{skip})$ by definition.

$C \equiv x := E$**:**

(1) $P \equiv \exists y(x = y \wedge P')$ for some P' since $\exists y(x = y)$ is a tautology. WLOG assume $x \notin \mathrm{fv}(P')$ (instances of x can be replaced with y).
(2) $\vdash (x = y)\ x := E\ (x = E[x/y])$ by EX-Assign.
(3) $\vdash (x = y \wedge P')\ x := E\ (x = E[x/y] \wedge P')$ by EX-Frame and (2).
(4) $\vdash (\exists y(x = y \wedge P'))\ x := E\ (\exists y(x = E[x/y] \wedge P'))$ by EX-Exists and (3).
(5)

$$\begin{aligned}
\mathrm{sp}(P, x := E) &\equiv \exists v(x = E[x/v] \wedge P[x/v]) && \text{defn}\\
&\equiv \exists v(x = E[x/v] \wedge (\exists y(v = y \wedge P'))) && (1)\\
&\equiv \exists y(x = E[x/y] \wedge P') && \text{logic}
\end{aligned}$$

(6) $\vdash (P)\ x := E\ (\mathrm{sp}(P, x := E))$ by (1), (4), and (5).

$C_1; C_2$**:**

(1) $\vdash (P)\ C\ (\mathrm{sp}(P, C_1))$ by induction
(2) $\vdash (\mathrm{sp}(P, C_1))\ C_2\ (\mathrm{sp}(\mathrm{sp}(P, C_1), C_2))$ by induction
(3) $\vdash (P)\ C_1; C_2\ (\mathrm{sp}(\mathrm{sp}(P, C_1), C_2))$ by EX-Seq using (1), (2)
(4) $\mathrm{sp}(\mathrm{sp}(P, C_1), C_2) \equiv \mathrm{sp}(P, C_1; C_2)$ by definition
(5) $\vdash (P)\ C_1; C_2\ (\mathrm{sp}(P, C_1; C_2))$ by (3) and (4)

`if` E `then` C_1 `else` C_2**:**

(1) $\vdash (P \wedge E)\ C_1\ (\mathrm{sp}(P \wedge E, C_1))$ by induction
(2) $\vdash (P \wedge \neg E)\ C_2\ (\mathrm{sp}(P \wedge \neg E, C_2))$ by induction
(3) $\vdash (P \wedge E)$ `if` E `then` C_1 `else` $C_2\ (\mathrm{sp}(P \wedge E, C_1))$ by EX-IfThen using (1)
(4) $\vdash (P \wedge \neg E)$ `if` E `then` C_1 `else` $C_2\ (\mathrm{sp}(P \wedge \neg E, C_2))$ by EX-IfElse using (2)
(5) $\vdash ((P \wedge E) \vee (P \wedge \neg E))$ `if` E `then` C_1 `else` C_2 $(\mathrm{sp}(P \wedge E, C_1) \vee \mathrm{sp}(P \wedge \neg E, C_2))$ by EX-Disj using (3), (4)
(6) $(P \wedge E) \vee (P \wedge \neg E) \equiv P$ by logic
(7) $\mathrm{sp}(P \wedge E, C_1) \vee \mathrm{sp}(P \wedge \neg E, C_2) \equiv \mathrm{sp}(P,$ `if` E `then` C_1 `else` $C_2)$ by definition
(8) $\vdash (P)$ `if` E `then` C_1 `else` C_2 $(\mathrm{sp}(P,$ `if` E `then` C_1 `else` $C_2))$ by (5), (6), and (7)

□

Lemma 26. *If* $\vdash (P)\ C\ (Q)$ *then* $Q \equiv \mathrm{sp}(P, C)$.

Proof. By induction on the derivation $\vdash (P)\ C\ (Q)$:

EX-Skip: By inversion $C \equiv$ `skip` and $Q \equiv P$, and by definition $\mathrm{sp}(P,$ `skip`$) \equiv P$. Thus $\mathrm{sp}(P, C) \equiv \mathrm{sp}(P,$ `skip`$)$ $\equiv P \equiv Q$.

EX-Assign:

(1) $C \equiv x := E$ for some x, E by inversion
(2) $P \equiv x = E'$ for some E' by inversion
(3) $Q \equiv x = E[x/E']$ by inversion
(4) $x \notin \mathrm{fv}(E')$ by inversion
(5)

$$\begin{aligned}
&\mathrm{sp}(P, C)\\
&\quad\equiv \mathrm{sp}(x = E', x := E) && (1), (2)\\
&\quad\equiv \exists v(x = E[x/v] \wedge (x = E')[x/v]) && \text{defn sp}\\
&\quad\equiv \exists v(x = E[x/v] \wedge v = E') && (4)\\
&\quad\equiv \exists v(x = E[x/E'] \wedge v = E') && \text{logic}\\
&\quad\equiv x = E[x/E'] \wedge \exists v\, v = E' && (4)\\
&\quad\equiv x = E[x/E'] && \text{logic}\\
&\quad\equiv Q && (3)
\end{aligned}$$

EX-IfThen:

(1) $C \equiv$ `if` E `then` C_1 `else` C_2 by inversion
(2) $P \equiv P' \wedge E$ for some P' by inversion
(3) $\vdash (P' \wedge E)\ C_1\ (Q)$ by inversion
(4) $Q \equiv \mathrm{sp}(P' \wedge E, C_1)$ by induction using (3)
(5)

$$\begin{aligned}
&\mathrm{sp}(P, C)\\
&\quad\equiv \mathrm{sp}(P' \wedge E, \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) && (1), (2)\\
&\quad\equiv \mathrm{sp}(P' \wedge E, C_1) && \text{Lemma 9}\\
&\quad\equiv Q && (4)
\end{aligned}$$

EX-IfElse:

(1) $C \equiv$ `if` E `then` C_1 `else` C_2 by inversion
(2) $P \equiv P' \wedge \neg E$ for some P' by inversion
(3) $\vdash (P' \wedge \neg E)\ C_2\ (Q)$ by inversion
(4) $Q \equiv \mathrm{sp}(P' \wedge \neg E, C_2)$ by induction using (3)
(5)

$$\begin{aligned}
&\mathrm{sp}(P, C)\\
&\quad\equiv \mathrm{sp}(P' \wedge \neg E, \texttt{if } E \texttt{ then } C_1 \texttt{ else } C_2) && (1), (2)\\
&\quad\equiv \mathrm{sp}(P' \wedge \neg E, C_2) && \text{Lemma 9}\\
&\quad\equiv Q && (4)
\end{aligned}$$

EX-Seq:

(1) $C \equiv C_1; C_2$ for some C_1, C_2 by inversion
(2) $\vdash (P)\ C_1\ (R)$ for some R by inversion
(3) $\vdash (R)\ C_2\ (Q)$ by inversion
(4) $R \equiv \mathrm{sp}(P, C_1)$ by induction using (2)
(5) $Q \equiv \mathrm{sp}(R, C_2)$ by induction using (3)
(6)

$$\begin{aligned}
\mathrm{sp}(P, C) &\equiv \mathrm{sp}(P, C_1; C_2) && (1)\\
&\equiv \mathrm{sp}(\mathrm{sp}(P, C_1), C_2) && \text{defn sp}\\
&\equiv \mathrm{sp}(R, C_2) && (4)\\
&\equiv Q && (5)
\end{aligned}$$

EX-Exists:

(1) $P \equiv \exists x\, P'$ for some P' by inversion
(2) $Q \equiv \exists x\, Q'$ for some Q' by inversion
(3) $\vdash (P')\ C\ (Q')$ by inversion
(4) $x \notin \mathrm{fv}(C)$ by inversion
(5) $Q' \equiv \mathrm{sp}(P', C)$ by induction using (3)

(6)

$$\begin{aligned}\mathrm{sp}(P, C) &\equiv \mathrm{sp}(\exists x\, P', C) && (2)\\ &\equiv \exists x\ \mathrm{sp}(P', C) && \text{Lemma 6}\\ &\equiv \exists x\, Q' && (5)\\ &\equiv Q && (2)\end{aligned}$$

EX-Disj:

(1) $P \equiv P_1 \vee P_2$ for some P_1, P_2 by inversion
(2) $Q \equiv Q_1 \vee Q_2$ for some Q_1, Q_2 by inversion
(3) $\vdash (P_1)\ C\ (Q_1)$ by inversion
(4) $\vdash (P_2)\ C\ (Q_2)$ by inversion
(5) $Q_1 \equiv \mathrm{sp}(P_1, C)$ by induction using (3)
(6) $Q_2 \equiv \mathrm{sp}(P_2, C)$ by induction using (4)
(7)

$$\begin{aligned}\mathrm{sp}(P, C) &\equiv \mathrm{sp}(P_1 \vee P_2, C) && (1)\\ &\equiv \mathrm{sp}(P_1, C) \vee \mathrm{sp}(P_2, C) && \text{Lemma 7}\\ &\equiv Q_1 \vee Q_2 && (5), (6)\\ &\equiv Q && (2)\end{aligned}$$

EX-Frame:

(1) $P \equiv P' \wedge R$ for some P', R by inversion
(2) $Q \equiv Q' \wedge R$ for some Q' by inversion
(3) $\mathrm{mod}(C) \cap \mathrm{fv}(R) = \emptyset$ by inversion
(4) $\vdash (P')\ C\ (Q')$ by inversion
(5) $\mathrm{sp}(P', C) \equiv Q$ by induction using (4)
(6) $\mathrm{sp}(P' \wedge R, C) \equiv R \wedge \mathrm{sp}(P', C)$ by Lemma 8 using (3)
(7)

$$\begin{aligned}\mathrm{sp}(P, C) &\equiv \mathrm{sp}(P' \wedge R, C) && (1)\\ &\equiv R \wedge \mathrm{sp}(P', C) && (6)\\ &\equiv R \wedge Q' && (5)\\ &\equiv Q && (2)\end{aligned}$$

□

Theorem 4. $\vdash (P)\ C\ (Q) \iff Q \equiv \mathrm{sp}(P, C)$

Proof. $\Longrightarrow$: Lemma 26; $\Longleftarrow$: Lemma 25 □

E.2 Gradual exact logic

Valid triples in gradual exact logic are denoted $\widetilde{\vdash}\ (\widetilde{P})\ C\ (\widetilde{Q})$.

Definition 9. The *concretization* $\gamma : \widetilde{\textsc{Formula}} \to \mathcal{P}(\textsc{Formula})$ maps a gradual formula to the set of all formulas it can represent:

$$\gamma(P) \coloneqq \{P\}$$
$$\gamma(? \wedge P) \coloneqq \{P' \in \textsc{SatFormula} \mid P' \Rightarrow P\}$$

Definition 10. Deductions in gradual exact logic directly are lifted deductions in exact logic:

$$\widetilde{\vdash}\ (\widetilde{P})\ C\ (\widetilde{Q}) \overset{\text{def}}{\iff} \vdash (P)\ C\ (Q)$$
$$\text{for some } P \in \gamma(\widetilde{P}) \text{ and } Q \in \gamma(\widetilde{Q})$$

Theorem 5. *For $P \in$ SatFormula,*

$$\widetilde{\vdash}\ (P)\ C\ (? \wedge Q) \iff \vdash \{P\}\ C\ \{Q\}.$$

That is, except for the vacuous case where $P \equiv \bot$, gradualizing the postcondition exactly characterizes deductions in Hoare logic.

Proof. $\Longrightarrow$:

(1) $\widetilde{\vdash}\ (P)\ C\ (? \wedge Q)$ by assumptino
(2) $\vdash (P)\ C\ (Q')$ for some $Q' \in \gamma(? \wedge Q)$ by (1)
(3) $\mathrm{sp}(P, C) \equiv Q'$ by Lemma 26 using (2)
(4) $Q' \Rightarrow Q$ by definition of γ and (2)
(5) $\mathrm{sp}(P, C) \Rightarrow Q$ by (3) and (4)
(6) $\vdash \{P\}\ C\ \{Q\}$ by Lemma 18 using (5)

$\Longleftarrow$:

(1) $\vdash \{P\}\ C\ \{Q\}$ by assumption
(2) $P \in \textsc{SatFormula}$ by assumption
(3) $\mathrm{sp}(P, C) \Rightarrow Q$ by Lemma 16 using (1)
(4) $\mathrm{sp}(P, C) \in \textsc{SatFormula}$ by Lemma 5 using (2)
(5) $\mathrm{sp}(P, C) \in \gamma(? \wedge Q)$ by definition of γ using (3) and (4)
(6) $\vdash (P)\ C\ (\mathrm{sp}(P, C))$ by Lemma 25
(7) $\widetilde{\vdash}\ (P)\ C\ (? \wedge Q)$ by definition using (5) and (6) □

Theorem 6. *If $Q \in$ SatFormula,*

$$\widetilde{\vdash}\ (? \wedge P)\ C\ (Q) \iff \vdash [P]\ C\ [Q]$$

That is, except in the vacuous case where $Q \equiv \bot$, gradualizing the precondition exactly characterizes deductions in incorrectness logic.

Proof. $\Longrightarrow$:

(1) $\widetilde{\vdash}\ (? \wedge P)\ C\ (Q)$ by assumption
(2) $\vdash (P')\ C\ (Q)$ for some $P' \in \gamma(? \wedge P)$ by definition using (1)
(3) $P' \Rightarrow P$ by definition of γ using (2)
(4) $Q \equiv \mathrm{sp}(P', C)$ by Lemma 26 using (2)
(5) $\mathrm{sp}(P', C) \Rightarrow \mathrm{sp}(P, C)$ by Lemma 2 using (3)
(6) $Q \Rightarrow \mathrm{sp}(P, C)$ by (4) and (5)
(7) $\vdash [P]\ C\ [Q]$ by Theorem 2 using (6)

$\Longleftarrow$:

(1) $\vdash [P]\ C\ [Q]$ by assumption
(2) $Q \in \textsc{SatFormula}$ by assumption
(3) $P \wedge \mathrm{wp}(C, Q) \in \textsc{SatFormula}$ by Lemma 24 using (1) and (2)
(4) $P \wedge \mathrm{wp}(C, Q) \Rightarrow P$ by logic
(5) $P \wedge \mathrm{wp}(C, Q) \in \gamma(? \wedge P)$ by definition of γ using (3) and (4)
(6) $Q \Rightarrow \mathrm{sp}(P, C)$ by Theorem 2 using (1)
(7) $\mathrm{sp}(P \wedge \mathrm{wp}(C, Q), C) \equiv Q$ by Lemma 12 using (6)
(8) $\vdash (P \wedge \mathrm{wp}(C, Q))\ C\ (Q)$ by Theorem 4 using (7)
(9) $\widetilde{\vdash}\ (? \wedge P)\ C\ (Q)$ by (5) and (8) □