

Soundness and Completeness of Hoare Logic

(Based on [Apt and Olderog 1997])

Yih-Kuen Tsay

Department of Information Management
National Taiwan University

Overview

- Given an adequate semantics for the programming language under consideration, the **validity** of a Hoare triple $\{p\} S \{q\}$ can be precisely defined.
- A Hoare Logic for a programming language is **sound** if every *Hoare triple proven by the logic is valid*.
- A Hoare Logic for a programming language is **complete** if every *valid Hoare triple can be proven by the logic*.
- We shall develop these results for a very simple **deterministic** programming language.

A Simple Programming Language

- 🌐 We will consider a Hoare Logic for the following simple (deterministic) programming language:

$$S ::= \begin{array}{l} \mathbf{skip} \\ | \quad u := t \\ | \quad S_1; S_2 \\ | \quad \mathbf{if } B \mathbf{ then } S_1 \mathbf{ else } S_2 \mathbf{ fi} \\ | \quad \mathbf{while } B \mathbf{ do } S \mathbf{ od} \end{array}$$

Note: here t is an expression (first-order term) of the same type as variable u ; B is a boolean expression.

- 🌐 We consider only programs that are free of syntactical or typing errors.

Proof Rules of Hoare Logic

$$\frac{}{\{q[t/u]\} u := t \{q\}}$$

(Assignment)

$$\frac{}{\{p\} \text{skip} \{p\}}$$

(Skip)

$$\frac{\{p\} S_1 \{q\} \quad \{q\} S_2 \{r\}}{\{p\} S_1; S_2 \{r\}}$$

(Sequence)

$$\frac{\{p \wedge B\} S_1 \{q\} \quad \{p \wedge \neg B\} S_2 \{q\}}{\{p\} \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi} \{q\}}$$

(Conditional)

Proof Rules of Hoare Logic (cont.)

$$\frac{\{p \wedge B\} S \{p\}}{\{p\} \mathbf{while} B \mathbf{do} S \mathbf{od} \{p \wedge \neg B\}} \quad \text{(While)}$$

$$\frac{p \rightarrow p' \quad \{p'\} S \{q'\} \quad q' \rightarrow q}{\{p\} S \{q\}} \quad \text{(Consequence)}$$

We will refer to this proof system as *System PD*.

Operational Semantics

- 🌐 A program/statement with a start state is seen as an **abstract machine**.
- 🌐 (1) The **part of program that remains** to be executed and (2) the **current state** constitute the **configuration** of the abstract machine.
- 🌐 By executing the program step by step, the machine transforms from one configuration to another.
- 🌐 A **transition relation** naturally arises between configurations.
- 🌐 The (input/output) semantics $\mathcal{M}[S]$ of a program S can then be defined with the help of the above transition relation.

Operational Semantics (cont.)

- At a high level, a configuration is a pair $\langle S, \sigma \rangle$ where S is a program and σ is a “proper” state.
- A transition

$$\langle S, \sigma \rangle \rightarrow \langle R, \tau \rangle$$

means “executing S one step in state σ leads to state τ with R as the remainder of S to be executed.”

- Let E denote the empty program. When the remainder R equals E , it means that S has terminated.
- The transition relation \rightarrow can be defined inductively (in the form of axioms and rules) over the structure of a program.

Semantics of the Simple Language

To give an **operational semantics** of the simple language, we postulate the following **transition axioms** and **rules**:

1. $\langle \mathbf{skip}, \sigma \rangle \rightarrow \langle E, \sigma \rangle$
2. $\langle u := t, \sigma \rangle \rightarrow \langle E, \sigma[u := \sigma(t)] \rangle$
3.
$$\frac{\langle S_1, \sigma \rangle \rightarrow \langle S_2, \tau \rangle}{\langle S_1; S, \sigma \rangle \rightarrow \langle S_2; S, \tau \rangle}$$
4. $\langle \mathbf{if } B \mathbf{ then } S_1 \mathbf{ else } S_2 \mathbf{ fi}, \sigma \rangle \rightarrow \langle S_1, \sigma \rangle$, when $\sigma \models B$
5. $\langle \mathbf{if } B \mathbf{ then } S_1 \mathbf{ else } S_2 \mathbf{ fi}, \sigma \rangle \rightarrow \langle S_2, \sigma \rangle$, when $\sigma \models \neg B$
6. $\langle \mathbf{while } B \mathbf{ do } S \mathbf{ od}, \sigma \rangle \rightarrow \langle S; \mathbf{while } B \mathbf{ do } S \mathbf{ od}, \sigma \rangle$, when $\sigma \models B$
7. $\langle \mathbf{while } B \mathbf{ do } S \mathbf{ od}, \sigma \rangle \rightarrow \langle E, \sigma \rangle$, when $\sigma \models \neg B$

- 🌐 The preceding set of transition axioms and rules can be seen as a formal proof system, called a **transition system**.
- 🌐 A transition $\langle S, \sigma \rangle \rightarrow \langle R, \tau \rangle$ is possible if it can be **deduced** in the transition system.
- 🌐 This semantic is “high level”, as assignments and evaluations of Boolean expressions are done in one step.

Transition Sequences and Computations

- A *transition sequence* of S starting in σ is a finite or infinite sequence of configurations

$$\langle S_0, \sigma_0 \rangle (= \langle S, \sigma \rangle) \rightarrow \langle S_1, \sigma_1 \rangle \rightarrow \cdots \rightarrow \langle S_i, \sigma_i \rangle \rightarrow \cdots$$

- A *computation* of S starting in σ is a transition sequence of S starting in σ that cannot be extended.
- A computation of S *terminates* in τ if it is finite and its last configuration is $\langle E, \tau \rangle$.
- A computation of S *diverges* if it is infinite.

An Example

- Consider the following program

$$S \equiv a[0] := 1; a[1] := 0; \mathbf{while} \ a[x] \neq 0 \ \mathbf{do} \ x := x + 1 \ \mathbf{od}$$

- Let σ be a state in which x is 0.
- Let σ' stand for $\sigma[a[0] := 1][a[1] := 0]$.
- The following is the computation of S starting in σ :

$$\begin{aligned} & \langle S, \sigma \rangle \\ \rightarrow & \langle a[1] := 0; \mathbf{while} \ a[x] \neq 0 \ \mathbf{do} \ x := x + 1 \ \mathbf{od}, \sigma[a[0] := 1] \rangle \\ \rightarrow & \langle \mathbf{while} \ a[x] \neq 0 \ \mathbf{do} \ x := x + 1 \ \mathbf{od}, \sigma' \rangle \\ \rightarrow & \langle x := x + 1; \mathbf{while} \ a[x] \neq 0 \ \mathbf{do} \ x := x + 1 \ \mathbf{od}, \sigma' \rangle \\ \rightarrow & \langle \mathbf{while} \ a[x] \neq 0 \ \mathbf{do} \ x := x + 1 \ \mathbf{od}, \sigma'[x := 1] \rangle \\ \rightarrow & \langle E, \sigma'[x := 1] \rangle \end{aligned}$$

Finite Transition Sequences

- For partial correctness of sequential programs, we will need only to talk about **finite** transition sequences.
- To that end, we take the **reflexive transitive closure** \rightarrow^* of \rightarrow .
- So, $\langle S, \sigma \rangle \rightarrow^* \langle R, \tau \rangle$ holds when
 - $\langle R, \tau \rangle = \langle S, \sigma \rangle$ or
 - $\langle S_0, \sigma_0 \rangle (= \langle S, \sigma \rangle) \rightarrow \langle S_1, \sigma_1 \rangle \rightarrow \cdots \rightarrow \langle S_n, \sigma_n \rangle (= \langle R, \tau \rangle)$ is a finite transition sequence.

Input/Output Semantics

- Let Σ be the set of all “proper” states.
- The *partial correctness semantics* is a mapping

$$\mathcal{M}[[S]] : \Sigma \rightarrow \mathcal{P}(\Sigma)$$

with

$$\mathcal{M}[[S]](\sigma) = \{\tau \mid \langle S, \sigma \rangle \rightarrow^* \langle E, \tau \rangle\}.$$

- Extensions of $\mathcal{M}[[S]]$
 - $\mathcal{M}[[S]](\perp) = \emptyset.$
 - For $X \subseteq \Sigma \cup \{\perp\}$, $\mathcal{M}[[S]](X) = \bigcup_{\sigma \in X} \mathcal{M}[[S]](\sigma).$

Validity of a Hoare Triple

- Let $\llbracket p \rrbracket$ denote $\{\sigma \in \Sigma \mid \sigma \models p\}$, i.e., the set of states where p holds.
- The Hoare triple $\{p\} S \{q\}$ is **valid** in the sense of partial correctness, written $\models \{p\} S \{q\}$, if

$$\mathcal{M}[S](\llbracket p \rrbracket) \subseteq \llbracket q \rrbracket.$$

About the While Loop

- Let Ω be a program such that $\mathcal{M}[\Omega](\sigma) = \emptyset$, for any σ .
- Define the following sequence of deterministic programs:

$$\begin{aligned}(\mathbf{while } B \mathbf{ do } S \mathbf{ od})^0 &= \Omega \\(\mathbf{while } B \mathbf{ do } S \mathbf{ od})^{k+1} &= \mathbf{if } B \mathbf{ then } S; (\mathbf{while } B \mathbf{ do } S \mathbf{ od})^k \\ &\quad \mathbf{else skip fi.}\end{aligned}$$

Lemmas for $\mathcal{M}[[S]]$

1. $\mathcal{M}[[S]]$ is **monotonic**, i.e.,
 $X \subseteq Y \subseteq \Sigma \cup \{\perp\}$ implies $\mathcal{M}[[S]](X) \subseteq \mathcal{M}[[S]](Y)$.
2. $\mathcal{M}[[S_1; S_2]](X) = \mathcal{M}[[S_2]](\mathcal{M}[[S_1]](X))$.
3. $\mathcal{M}[[S_1; S_2]; S_3]](X) = \mathcal{M}[[S_1; (S_2; S_3)]](X)$.
4. $\mathcal{M}[[\text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}]](X) =$
 $\mathcal{M}[[S_1]](X \cap [[B]]) \cup \mathcal{M}[[S_2]](X \cap [[\neg B]])$.
5. $\mathcal{M}[[\text{while } B \text{ do } S \text{ od}]] = \bigcup_{k=0}^{\infty} \mathcal{M}[[\text{(while } B \text{ do } S \text{ od)}^k]]$.

Theorem (Soundness): The proof system PD is sound for partial correctness of programs in the simple programming language, i.e.,

$$\vdash_{PD} \{p\} S \{q\} \text{ implies } \models \{p\} S \{q\}.$$

It suffices to prove that (1) the Hoare triples in all axioms of PD are valid and (2) all proof rules of PD are sound.

Note: a proof rule is sound if the validity of the Hoare triples in the premises implies the validity of the Hoare triple in the conclusion.

Soundness (cont.)

🌐 **skip:** $\mathcal{M}[\mathbf{skip}]([\![p]\!]) \subseteq [\![p]\!]$

$$\begin{aligned} \mathcal{M}[\mathbf{skip}]([\![p]\!]) &= \bigcup_{\sigma \in [\![p]\!]} \{ \tau \mid \langle \mathbf{skip}, \sigma \rangle \rightarrow^* \langle E, \tau \rangle \} \\ &= \bigcup_{\sigma \in [\![p]\!]} \{ \sigma \} = [\![p]\!] \subseteq [\![p]\!]. \end{aligned}$$

🌐 **Assignment:** $\mathcal{M}[u := t]([\![p[t/u]]\!]) \subseteq [\![p]\!]$

It can be shown that (1) $\sigma(s[u := t]) = \sigma[u := \sigma(t)](s)$ and (2) $\sigma \models p[t/u]$ iff $\sigma[u := \sigma(t)] \models p$.

Let $\sigma \in [\![p[t/u]]\!]$.

From the transition axiom for assignment,

$$\mathcal{M}[u := t](\sigma) = \{ \sigma[u := \sigma(t)] \}.$$

Since $\sigma \models p[t/u]$ iff $\sigma[u := \sigma(t)] \models p$, we have

$$\mathcal{M}[u := t](\sigma) \subseteq [\![p]\!] \text{ and hence } \mathcal{M}[u := t]([\![p[t/u]]\!]) \subseteq [\![p]\!].$$

Soundness (cont.)

- Composition: $\mathcal{M}[S_1](\llbracket p \rrbracket) \subseteq \llbracket r \rrbracket$ and $\mathcal{M}[S_2](\llbracket r \rrbracket) \subseteq \llbracket q \rrbracket$ imply $\mathcal{M}[S_1; S_2](\llbracket p \rrbracket) \subseteq \llbracket q \rrbracket$.

From the monotonicity of $\mathcal{M}[S_2]$,
 $\mathcal{M}[S_2](\mathcal{M}[S_1](\llbracket p \rrbracket)) \subseteq \mathcal{M}[S_2](\llbracket r \rrbracket) \subseteq \llbracket q \rrbracket$.

By an earlier lemma, $\mathcal{M}[S_2](\mathcal{M}[S_1](\llbracket p \rrbracket)) = \mathcal{M}[S_1; S_2](\llbracket p \rrbracket)$.

- Conditional: $\mathcal{M}[S_1](\llbracket p \wedge B \rrbracket) \subseteq \llbracket q \rrbracket$ and $\mathcal{M}[S_2](\llbracket p \wedge \neg B \rrbracket) \subseteq \llbracket q \rrbracket$ imply $\mathcal{M}[\mathbf{if } B \mathbf{ then } S_1 \mathbf{ else } S_2 \mathbf{ fi}](\llbracket p \rrbracket) \subseteq \llbracket q \rrbracket$.

This follows from an earlier lemma,
 $\mathcal{M}[\mathbf{if } B \mathbf{ then } S_1 \mathbf{ else } S_2 \mathbf{ fi}](X) =$
 $\mathcal{M}[S_1](X \cap \llbracket B \rrbracket) \cup \mathcal{M}[S_2](X \cap \llbracket \neg B \rrbracket)$.

Soundness (cont.)

- While: $\mathcal{M}[[S]](\llbracket p \wedge B \rrbracket) \subseteq \llbracket p \rrbracket$ implies $\mathcal{M}[\mathbf{while} B \mathbf{do} S \mathbf{od}](\llbracket p \rrbracket) \subseteq \llbracket p \wedge \neg B \rrbracket$.

From Lemma 5 for $\mathcal{M}[\cdot]$, it boils down to show that $\bigcup_{k=0}^{\infty} \mathcal{M}[(\mathbf{while} B \mathbf{do} S \mathbf{od})^k](\llbracket p \rrbracket) \subseteq \llbracket p \wedge \neg B \rrbracket$.

We prove by induction that, for all $k \geq 0$,

$$\mathcal{M}[(\mathbf{while} B \mathbf{do} S \mathbf{od})^k](\llbracket p \rrbracket) \subseteq \llbracket p \wedge \neg B \rrbracket.$$

The base case $k = 0$ is clear.

Soundness (cont.)

$$\begin{aligned}
 & \mathcal{M}[\langle\langle \mathbf{while} \ B \ \mathbf{do} \ S \ \mathbf{od} \rangle\rangle^{k+1}\rangle(\llbracket p \rrbracket) \\
 = & \quad \{ \text{definition of } \langle\langle \mathbf{while} \ B \ \mathbf{do} \ S \ \mathbf{od} \rangle\rangle^{k+1} \} \\
 & \mathcal{M}[\langle\langle \mathbf{if} \ B \ \mathbf{then} \ S; \langle\langle \mathbf{while} \ B \ \mathbf{do} \ S \ \mathbf{od} \rangle\rangle^k \ \mathbf{else} \ \mathbf{skip} \ \mathbf{fi} \rangle\rangle(\llbracket p \rrbracket) \\
 = & \quad \{ \text{Lemma 4 for } \mathcal{M}[\cdot] \} \\
 & \mathcal{M}[\langle\langle S; \langle\langle \mathbf{while} \ B \ \mathbf{do} \ S \ \mathbf{od} \rangle\rangle^k \rangle\rangle(\llbracket p \wedge B \rrbracket) \cup \mathcal{M}[\langle\langle \mathbf{skip} \rangle\rangle(\llbracket p \wedge \neg B \rrbracket)] \\
 = & \quad \{ \text{Lemma 2 for } \mathcal{M}[\cdot] \text{ and semantics of } \mathbf{skip} \} \\
 & \mathcal{M}[\langle\langle \mathbf{while} \ B \ \mathbf{do} \ S \ \mathbf{od} \rangle\rangle^k](\mathcal{M}[S](\llbracket p \wedge B \rrbracket) \cup \llbracket p \wedge \neg B \rrbracket) \\
 \subseteq & \quad \{ \text{the premise and monotonicity of } \mathcal{M}[\cdot] \} \\
 & \mathcal{M}[\langle\langle \mathbf{while} \ B \ \mathbf{do} \ S \ \mathbf{od} \rangle\rangle^k](\llbracket p \rrbracket) \cup \llbracket p \wedge \neg B \rrbracket \\
 \subseteq & \quad \{ \text{induction hypothesis} \} \\
 & \llbracket p \wedge \neg B \rrbracket \cup \llbracket p \wedge \neg B \rrbracket
 \end{aligned}$$

Soundness (cont.)

🌐 Consequence: $p \rightarrow p'$, $\mathcal{M}[S](\llbracket p' \rrbracket) \subseteq \llbracket q' \rrbracket$, and $q' \rightarrow q$ imply $\mathcal{M}[S](\llbracket p \rrbracket) \subseteq \llbracket q \rrbracket$.

First of all, $\llbracket p \rrbracket \subseteq \llbracket p' \rrbracket$ and $\llbracket q' \rrbracket \subseteq \llbracket q \rrbracket$.

From the monotonicity of $\mathcal{M}[S]$,
 $\mathcal{M}[S](\llbracket p \rrbracket) \subseteq \mathcal{M}[S](\llbracket p' \rrbracket) \subseteq \llbracket q' \rrbracket \subseteq \llbracket q \rrbracket$.

About Completeness

- 🌐 Assertions that we use for a programming language often involve numbers/integers.
- 🌐 According to **Gödel's First Incompleteness Theorem**, there is no complete proof system (that is consistent/sound) for the first-order theory of arithmetic.
- 🌐 We therefore assume that all true assertions are given (as axioms).
- 🌐 The completeness of Hoare Logic then is actually **relative to the truth of all assertions**.

Weakest Liberal Precondition

- Let S be a program in the simple programming language.
- For a set Φ of states, we define

$$wlp(S, \Phi) = \{\sigma \mid \mathcal{M}[[S]](\sigma) \subseteq \Phi\}.$$

- $wlp(S, \Phi)$ is called the *weakest liberal precondition* of S with respect to Φ .
- Informally, $wlp(S, \Phi)$ is the set of all states σ such that whenever S is activated in σ and properly terminates, the output state is in Φ .

Definability of $wlp(S, \Phi)$

- 🌍 An assertion p defines a set Φ of states if $\llbracket p \rrbracket = \Phi$.
- 🌍 Assuming that the assertion language includes addition and multiplication of natural numbers,
there is an assertion p defining $wlp(S, \llbracket q \rrbracket)$, i.e., with $\llbracket p \rrbracket = wlp(S, \llbracket q \rrbracket)$.
- 🌍 Proof of the above statement requires a technique called *Gödelization* and will not be given here.
- 🌍 We will write $wlp(S, q)$ to denote the assertion p such that $\llbracket p \rrbracket = wlp(S, \llbracket q \rrbracket)$.

Lemmas for wlp

1. $wlp(\mathbf{skip}, q) \leftrightarrow q$.
2. $wlp(u := t, q) \leftrightarrow q[t/u]$.
3. $wlp(S_1; S_2, q) \leftrightarrow wlp(S_1, wlp(S_2, q))$.
4. $wlp(\mathbf{if } B \mathbf{ then } S_1 \mathbf{ else } S_2 \mathbf{ fi}, q) \leftrightarrow (B \wedge wlp(S_1, q)) \vee (\neg B \wedge wlp(S_2, q))$.
5. $wlp(\mathbf{while } B \mathbf{ do } S_1 \mathbf{ od}, q) \wedge B \rightarrow wlp(S_1, wlp(\mathbf{while } B \mathbf{ do } S_1 \mathbf{ od}, q))$.
6. $wlp(\mathbf{while } B \mathbf{ do } S_1 \mathbf{ od}, q) \wedge \neg B \rightarrow q$.
7. $\models \{p\} S \{q\}$ iff $p \rightarrow wlp(S, q)$.

Completeness

Theorem (Completeness): The proof system PD is complete for partial correctness of programs in the simple programming language, i.e.,

$$\models \{p\} S \{q\} \text{ implies } \vdash_{PD} \{p\} S \{q\}.$$

We first prove $\vdash_{PD} \{wlp(S, q)\} S \{q\}$, for all S and q . This is done by induction.

The base cases (**skip** and assignment) are trivial.

Completeness (cont.)

🌐 Conditional: $S \equiv \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi.}$

From Lemma 4 for *wlp*, we have

- (1) $wlp(S, q) \wedge B \rightarrow wlp(S_1, q)$ and
- (2) $wlp(S, q) \wedge \neg B \rightarrow wlp(S_2, q)$.

From the induction hypothesis, we have

- (3) $\vdash_{PD} \{wlp(S_1, q)\} S_1 \{q\}$ and
- (4) $\vdash_{PD} \{wlp(S_2, q)\} S_2 \{q\}$.

Applying the consequence rule to (1) and (3) and to (2) and (4), we have $\vdash_{PD} \{wlp(S, q) \wedge B\} S_1 \{q\}$ and $\vdash_{PD} \{wlp(S, q) \wedge \neg B\} S_2 \{q\}$.

From the conditional rule, we have $\vdash_{PD} \{wlp(S, q)\} S \{q\}$.

Completeness (cont.)

🌐 While: $S \equiv \mathbf{while\ } B \mathbf{ do\ } S_1 \mathbf{ od.}$

The induction hypothesis states that
 $\vdash_{PD} \{wlp(S_1, wlp(S, q))\} S_1 \{wlp(S, q)\}.$

Then, from Lemma 5 for wlp and the consequence rule,
 $\vdash_{PD} \{wlp(S, q) \wedge B\} S_1 \{wlp(S, q)\}.$

So, from the while rule, $\vdash_{PD} \{wlp(S, q)\} S \{wlp(S, q) \wedge \neg B\}.$

From Lemma 6 for wlp and the consequence rule,
 $\vdash_{PD} \{wlp(S, q)\} S \{q\}.$

Completeness (cont.)

- Now suppose $\models \{p\} S \{q\}$.
- From Lemma 7 for wlp , $p \rightarrow wlp(S, q)$.
- From $\vdash_{PD} \{wlp(S, q)\} S \{q\}$ and the consequence rule, $\vdash_{PD} \{p\} S \{q\}$.