Soundness and Completeness of Hoare Logic

(Based on [Apt, de Boer, and Olderog 2009])

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:

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

  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

$$\frac{}{\{q[t/u]\}\ u := t\ \{q\}} \quad \text{(Assignment)}$$

$$\frac{}{\{p\}\ \textbf{skip}\ \{p\}} \quad \text{(Skip)}$$

$$\frac{\{p\}\ S_1\ \{q\} \qquad \{q\}\ S_2\ \{r\}}{\{p\}\ S_1; S_2\ \{r\}} \quad \text{(Sequence)}$$

$$\frac{\{p \wedge B\}\ S_1\ \{q\} \qquad \{p \wedge \neg B\}\ S_2\ \{q\}}{\{p\}\ \textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi}\ \{q\}} \quad \text{(Conditional)}$$

Proof Rules of Hoare Logic (cont.)

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

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

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 (which essentially is a value assignment).
- A transition
$$\langle S, \sigma \rangle \to \langle R, \tau \rangle$$
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 $\to$ 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. $\langle \textbf{skip}, \sigma\rangle \rightarrow \langle E, \sigma\rangle$
2. $\langle u := t, \sigma\rangle \rightarrow \langle E, \sigma[u := \sigma(t)]\rangle$
3. $\dfrac{\langle S_1, \sigma\rangle \rightarrow \langle S_2, \tau\rangle}{\langle S_1; S, \sigma\rangle \rightarrow \langle S_2; S, \tau\rangle}$
4. $\langle \textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}, \sigma\rangle \rightarrow \langle S_1, \sigma\rangle$, when $\sigma \models B$
5. $\langle \textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}, \sigma\rangle \rightarrow \langle S_2, \sigma\rangle$, when $\sigma \models \neg B$
6. $\langle \textbf{while } B \textbf{ do } S \textbf{ od}, \sigma\rangle \rightarrow \langle S; \textbf{while } B \textbf{ do } S \textbf{ od}, \sigma\rangle$, when $\sigma \models B$
7. $\langle \textbf{while } B \textbf{ do } S \textbf{ od}, \sigma\rangle \rightarrow \langle E, \sigma\rangle$, when $\sigma \models \neg B$

Transition Systems

- 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

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

- 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:

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

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
  1. $\langle R, \tau \rangle = \langle S, \sigma \rangle$ or
  2. $\langle S_0, \sigma_0 \rangle (= \langle S, \sigma \rangle) \rightarrow \langle S_1, \sigma_1 \rangle \rightarrow \cdots \rightarrow \langle S_n, \sigma_n \rangle (= \langle R, \tau \rangle)$ is a finite transition sequence.

Input/Output Semantics

- Let Σ be the set of all proper states.
- The *partial correctness semantics* is a mapping
  $\mathcal{M}[\![S]\!] : \Sigma \to \mathcal{P}(\Sigma)$

  with
  $\mathcal{M}[\![S]\!](\sigma) = \{\tau \mid \langle S, \sigma\rangle \to^* \langle E, \tau\rangle\}$.
- Extensions of $\mathcal{M}[\![S]\!]$
  - $\mathcal{M}[\![S]\!](\bot) = \emptyset$.
  - For $X \subseteq \Sigma \cup \{\bot\}$, $\mathcal{M}[\![S]\!](X) = \bigcup_{\sigma \in X} \mathcal{M}[\![S]\!](\sigma)$.

Validity of a Hoare Triple

- Let $[\![p]\!]$ 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

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

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:

$$\begin{array}{lcl} (\textbf{while } B \textbf{ do } S \textbf{ od})^0 & = & \Omega \\ (\textbf{while } B \textbf{ do } S \textbf{ od})^{k+1} & = & \textbf{if } B \textbf{ then } S; (\textbf{while } B \textbf{ do } S \textbf{ od})^k \\ & & \textbf{else skip fi} \end{array}$$

- For example, $(\textbf{while } B \textbf{ do } S \textbf{ od})^2$

$$\begin{array}{cll} = & \textbf{if } B \textbf{ then } S; & (\textbf{while } B \textbf{ do } S \textbf{ od})^1 \\ & \textbf{else skip fi} & \\ = & \textbf{if } B \textbf{ then } S; & \textbf{if } B \textbf{ then } S; (\textbf{while } B \textbf{ do } S \textbf{ od})^0 \\ & & \textbf{else skip fi} \\ & \textbf{else skip fi} & \\ = & \textbf{if } B \textbf{ then } S; & \textbf{if } B \textbf{ then } S; \Omega \\ & & \textbf{else skip fi} \\ & \textbf{else skip fi} & \end{array}$$

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

1. $\mathcal{M}[\![S]\!]$ is monotonic, i.e., $X \subseteq Y \subseteq \Sigma \cup \{\bot\}$ 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. $\mathcal{M}[\![\textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}]\!](X) = \mathcal{M}[\![S_1]\!](X \cap [\![B]\!]) \cup \mathcal{M}[\![S_2]\!](X \cap [\![\neg B]\!])$.
5. $\mathcal{M}[\![\textbf{while } B \textbf{ do } S \textbf{ od}]\!] = \bigcup_{k=0}^{\infty} \mathcal{M}[\![(\textbf{while } B \textbf{ do } S \textbf{ od})^k]\!]$.

Soundness

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

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

The proof is by induction, i.e., by proving 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**: $\mathcal{M}[\![\textbf{skip}]\!]([\![p]\!]) \subseteq [\![p]\!]$

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

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

  It can be shown that (1) $\sigma(s[t/u]) = \sigma[u := \sigma(t)](s)$ 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
  $\mathcal{M}[\![u := t]\!](\sigma) \subseteq [\![p]\!]$ and hence $\mathcal{M}[\![u := t]\!]([\![p[t/u]]\!]) \subseteq [\![p]\!]$.

Soundness (cont.)

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

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

  By an earlier lemma, $\mathcal{M}[\![S_2]\!](\mathcal{M}[\![S_1]\!]([\![p]\!])) = \mathcal{M}[\![S_1; S_2]\!]([\![p]\!])$.
- Conditional: $\mathcal{M}[\![S_1]\!]([\![p \wedge B]\!]) \subseteq [\![q]\!]$ and $\mathcal{M}[\![S_2]\!]([\![p \wedge \neg B]\!]) \subseteq [\![q]\!]$ imply $\mathcal{M}[\![\textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}]\!]([\![p]\!]) \subseteq [\![q]\!]$.

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

Soundness (cont.)

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

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

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

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

  The base case $k = 0$ is clear.

Soundness (cont.)

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

Soundness (cont.)

- Consequence: $p \to p'$, $\mathcal{M}[\![S]\!]([\![p']\!]) \subseteq [\![q']\!]$, and $q' \to q$ imply $\mathcal{M}[\![S]\!]([\![p]\!]) \subseteq [\![q]\!]$.

  First of all, $[\![p]\!] \subseteq [\![p']\!]$ and $[\![q']\!] \subseteq [\![q]\!]$.

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

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 $[\![p]\!] = \Phi$.
- Assuming that the assertion language includes addition and multiplication of natural numbers,

  there is an assertion p defining $wlp(S, [\![q]\!])$, i.e., with $[\![p]\!] = wlp(S, [\![q]\!])$.

- 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 $[\![p]\!] = wlp(S, [\![q]\!])$.

Lemmas for *wlp*

1. $wlp(\textbf{skip}, q) \leftrightarrow q$.
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. $wlp(\textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}, q) \leftrightarrow$
   $(B \wedge wlp(S_1, q)) \vee (\neg B \wedge wlp(S_2, q))$.
5. $wlp(\textbf{while } B \textbf{ do } S_1 \textbf{ od}, q) \wedge B \rightarrow$
   $wlp(S_1, wlp(\textbf{while } B \textbf{ do } S_1 \textbf{ od}, q))$.
6. $wlp(\textbf{while } B \textbf{ do } S_1 \textbf{ od}, q) \wedge \neg B \rightarrow q$.
7. $\models \{p\}\ S\ \{q\}$ (i.e., $\mathcal{M}[\![S]\!]([\![p]\!]) \subseteq [\![q]\!]$) 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 a special case of the theorem: for all S and q, $\models \{wlp(S,q)\}\ S\ \{q\}$ implies $\vdash_{PD} \{wlp(S,q)\}\ S\ \{q\}$.
- As $\models \{wlp(S,q)\}\ S\ \{q\}$ (i.e., $\mathcal{M}[\![S]\!]([\![wlp(S,q)]\!]) \subseteq [\![q]\!]$) always holds, the case simplifies to $\vdash_{PD} \{wlp(S,q)\}\ S\ \{q\}$, for all S and q.
- This is done by induction.
- The base cases (**skip** and assignment) are trivial.

Completeness (cont.)

- Conditional: $S \equiv$ **if** B **then** S_1 **else** S_2 **fi**.

  To prove $\vdash_{PD} \{wlp(S, q)\}\ S\ \{q\}$ via the conditional rule, we need
  (1) $\vdash_{PD} \{wlp(S, q) \wedge B\}\ S_1\ \{q\}$ and
  (2) $\vdash_{PD} \{wlp(S, q) \wedge \neg B\}\ S_2\ \{q\}$.

  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, we are done if
  (5) $wlp(S, q) \wedge B \rightarrow wlp(S_1, q)$ and
  (6) $wlp(S, q) \wedge \neg B \rightarrow wlp(S_2, q)$.

  (5) and (6) follows from Lemma 4 for *wlp*.

Completeness (cont.)

A proof of (5) $wlp(S,q) \wedge B \rightarrow wlp(S_1,q)$:

$$
\begin{array}{ll}
 & wlp(S,q) \wedge B \\
\leftrightarrow & \{\ \text{definition of } S\ \} \\
 & wlp(\textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}, q) \wedge B \\
\leftrightarrow & \{\ \text{Lemma 4 for } wlp\ \} \\
 & ((B \wedge wlp(S_1,q)) \vee (\neg B \wedge wlp(S_2,q))) \wedge B \\
\leftrightarrow & \{\ \text{distribute } \wedge \text{ over } \vee\ \} \\
 & ((B \wedge wlp(S_1,q)) \wedge B) \vee ((\neg B \wedge wlp(S_2,q)) \wedge B) \\
\leftrightarrow & \{\ \text{commutitivity of } \wedge,\ B \wedge B \leftrightarrow B, \text{ and } \neg B \wedge B \leftrightarrow \mathit{false}\ \} \\
 & (B \wedge wlp(S_1,q)) \vee \mathit{false} \\
\leftrightarrow & \{\ A \vee \mathit{false} \leftrightarrow A\ \} \\
 & B \wedge wlp(S_1,q) \\
\rightarrow & \{\ B \wedge A \rightarrow A\ \} \\
 & wlp(S_1,q)
\end{array}
$$

Completeness (cont.)

- While: $S \equiv \textbf{while } B \textbf{ do } S_1 \textbf{ od}$.

  To prove $\vdash_{PD} \{wlp(S,q)\}\ S\ \{q\}$, we apply Lemma 6 for *wlp* and the consequence rule to reduce the goal to $\vdash_{PD} \{wlp(S,q)\}\ S\ \{wlp(S,q) \wedge \neg B\}$.

  Using the while rule, the goal is further reduced to $\vdash_{PD} \{wlp(S,q) \wedge B\}\ S_1\ \{wlp(S,q)\}$.

  From Lemma 5 for *wlp* and the consequence rule, we need $\vdash_{PD} \{wlp(S_1, wlp(S,q))\}\ S_1\ \{wlp(S,q)\}$.

  This follows from the induction hypothesis, which states that $\vdash_{PD} \{wlp(S_1,q')\}\ S_1\ \{q'\}$, for all 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, $\vdash_{PD} \{p\}\ S\ \{q\}$.

References

- K.R. Apt, F.S. de Boer, and E.-R. Olderog. *Verification of Sequential and Concurrent Programs, Third Edition*, Springer, 2009.