# Inference rules for Hoare triples

Farzaneh Derakhshan
based on material by Stefan Muller, Jim Sasaki, and Adrian Sampson

CS 536: Science of Programming, Fall 2023
Lecture 9

In the last two lectures, we introduced two types of triples:

- Partial correctness triple $\{P\}S\{Q\}$, and
- Total correctness triple $[P]S[Q]$.

We defined the validity of partial and total correctness triples as *for all states $\sigma$, we have $\sigma \Vdash \{P\}S\{Q\}$, and $\sigma \Vdash [P]S[Q]$, respectively.* We also saw that to show a triple is valid, we have to consider states $\sigma$ and the execution of statement $S$ using our operational semantics.

It turns out that there is an elegant way of deriving valid partial correctness triples without having to reason about states and program executions. We can use a set of axioms and inference rules called Hoare rules, originally designed by Hoare. These rules allow us to derive provable partial correctness triples directly, forming a proof system known as Hoare logic.

We write $\vdash \{P\}S\{Q\}$ to denote that the triple is provable, meaning that we can derive a proof for $\{P\}S\{Q\}$ in Hoare logic. Provable and valid Hoare triples are closely related, which we will discuss later relation in Section 3.

## 1 A review to inference rules

A proof system consists of axioms and inference rules. The axioms are rules that have no premises, i.e.,

$$\frac{}{axiom}\ (\textsc{Ax})$$

The inference rules are of the form

$$\frac{premise\ 1 \qquad \cdots \qquad premise\ n}{conclusion}\ (\textsc{Rule})$$

It states that the conclusion can be deduced from the premises numbered 1 to n. Premises are also called the hypotheses of the rule.

A formal proof of a triple is a tree-shaped structure. The triple we want to prove is the root (conclusion), and axioms are the leaves (premises). We use the rules of inference to derive the root (conclusion) from the leaves (premises):

$$\dfrac{\dfrac{\dfrac{}{\vdots}\ (\textsc{Ax})}{premise\ 1}\ (\textsc{Rule 1}) \qquad \cdots \qquad \dfrac{\dfrac{}{\vdots}\ (\textsc{Ax})}{premise\ n}\ (\textsc{Rule n})}{conclusion}\ (\textsc{Rule})$$

A line is either an instance of an axiom or follows from previous lines by a rule of inference.

# 2 Hoare logic

We define an axiom/rule for each statement built using our grammar:

$$S := \mathsf{skip} \mid S; S \mid x := e \mid a[e] := e \mid \mathsf{if}\ e\ \mathsf{then}\ S\ \mathsf{else}\ S\ \mathsf{fi} \mid \mathsf{while}\ e\ \mathsf{do}\ S\ \mathsf{od}$$

## 2.1 Skip axiom

The skip axiom states that any property that is true before executing the statement skip is also true after:

$$\frac{}{\{P\}\,\mathsf{skip}\,\{P\}}\ (\textsc{Skip})$$

**Example 1.** The follwoing is a proof for $\vdash \{x = 2 \wedge y = 3\}\ \mathsf{skip}\ \{x = 2 \wedge y = 3\}$:

$$\frac{}{\{x = 2 \wedge y = 3\}\ \mathsf{skip}\ \{x = 2 \wedge y = 3\}}\ (\textsc{Skip})$$

## 2.2 Assign axiom

Let's start with an example. Is the triple $\{y = 2\}\ x := 2\ \{y = x\}$ correct? If initially $y$ is equal to 2, and we assign the value of $x$ to 2, then finally both $x$ and $y$ have the same value if 2. So, yes, the triple is correct. We design a general rule based on this example:

$$\frac{}{\{[e/x]P\}\,x := e\,\{P\}}\ (\textsc{Assign})$$

We get the (pre)condition $[e/x]P$ from the (post)condition $P$ by substituting the expression $e$ for all (free) occurences of the variable $x$. In the triple , $P$ is $y{=}2$ and $[e/x]P$ is $[2/x]y{=}x\ :=\ y{=}2$.
**Example 2.** The follwoing is a proof for $\vdash \{x - 1 > 0\}\ x := x - 1\ \{x > 0\}$

$$\frac{}{\vdash \{x - 1 > 0\}\ x := x - 1\ \{x > 0\}}\ (\textsc{Assign})$$

given that $[x - 1/x]\,x - 1{>}0\ :=\ x{>}0$.

We need to define substitution $[e/x]P$ more formally. But first let's look at one simple example:

$$[2 + y/x](x = 2 \wedge y \leq x)\ :=\ (2 + y = 2 \wedge y \leq 2 + y).$$

Next, we define $[e/x]P$ recursively on $P$:

| | | | | |
|---|---|---|---|---|
| | $[e/x]c$ | $:=$ | $c$ | $c \in \{T, F\}$ |
| | $[e/x](P \circ Q)$ | $:=$ | $[e/x]P \circ [e/x]Q$ | $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ |
| | $[e/x]\neg P$ | $:=$ | $\neg[e/x]P$ | |
| | $[e/x]P(e_1, \cdots, e_n)$ | $:=$ | $P([e/x]e_1, \cdots, [e/x]e_n)$ | |
| | $[e/x]Qx.P$ | $:=$ | $Qx.P$ | $Q \in \{\forall, \exists\}$ |
| ★ | $[e/x]Qy.P$ | $:=$ | $Qy.[e/x]P$ | $Q \in \{\forall, \exists\}, x \neq y,$ |

In Section 2.2.1 we will revisit the last row of the above definition to ensure that substituting a variable in a predicate with a quantifier is handled correctly. The definition above also uses a substitution for expressions, $[e/x]e_1$, which we define as

| | | | |
|---|---|---|---|
| $[e/x]x$ | $:=$ | $e$ | |
| $[e/x]y$ | $:=$ | $y$ | $x \neq y$ |
| $[e/x](e_1\ op\ e_2)$ | $:=$ | $[e/x]e_1\ op\ [e/x]e_2$ | |
| $[e/x](a[e_1])$ | $:=$ | $a[[e/x]e_1]$ | |
| $[e/x](size(a))$ | $:=$ | $size(a)$ | |
| $[e/x](if\ e'\ then\ e_1\ else\ e_2)$ | $:=$ | $if\ [e/x]e'\ then\ [e/x]e_1\ else\ [e/x]e_2$ | |

**Example 3.** $[2/x]y = y$. In general, if $x$ does not appear in $e_1$, we have $[e/x]e_1 := e_1$. Similarly, if $x$ does not appear in $P$, we have $[e/x]P := P$.
**Example 4.** $[2/x]\forall x.x > y = \forall x.x > y$. In general, if $x$ is not a free variable $P$, then $[e/x]P := P$.
**Example 5.** $[e/x]((\forall y.y > x \to (\exists x.y > x)) \land x \neq 0) := (\forall y.y > e \to (\exists x.y > x)) \land e \neq 0$.

### 2.2.1 Substitution and quantifiers

Substituting a variable in a predicate with quantifiers is subtle. For example, consider the predicate $\exists y.x < y$, which is true for any value given to $x$. Can we substitute $y + 2$ for $x$? the answer is no! If we do the substitution, we get $[y + 2/x]\exists y.x < y := \exists y.y + 2 < y$, leading to a contradictory formula. In other words, the free variable $y$ in the expression $(y + 2)$ got "*captured*" by the quantifier $\exists y$. To avoid these form of substitutions, we define the notion of $e$ beging substitutable for $x$ in $P$ as follows:

- $P$ is atomic or $\bot$ then $e$ is substitutable for $x$ in $P$.
- $e$ is substitutable for $x$ in $P \circ Q$ if $e$ is substitutable for $x$ in $P$ and $e$ is substitutable for $x$ in $Q$.
- $e$ is substituable for $x$ in $\forall y.P$, $\exists y.P$ iff either
  - $x$ does not occur in $\forall y.\, P$, or
  - $y$ does not occur in $e$ and $e$ is substituable for $x$ in $P$.

We can only write $[e/x]P$ if $e$ in indeed substitutable for the variable $x$ in $P$. For example, with this definition, $y + 2$ is not substitutable for $x$ in $\exists y.x < y$, preventing us from writing the bogus substitution $[y + 2/x]\exists y.x < y$. What we can do instead is to rename the variable $y$ in $\exists y.x < y$ first, and then apply the substitution: we first rewrite $\exists y.x < y$ as an equivalent formula $\exists z.x < z$, and next apply the substitution $[y + 2/x]$ to it, i.e., $[y + 2/x]\exists z.x < z := \exists z.y + 2 < z$.

## 2.3 Sequence rule

The Sequence rule enables proving a partial correctness triple for a sequence $S_1; S_2$ given triples for $S_1$ and $S_2$. If the postcondition of $S_1$ matches the precondition of $S_2$, we can derive a specification for their sequential composition.

$$\frac{\{P\}\, S_1\, \{R\} \qquad \{R\}\, S_2\, \{Q\}}{\{P\}\, S_1; S_2\, \{Q\}}\ (\textsc{Seq})$$

**Example 6.** Here is a proof for $\vdash \{x = a \land y = b\}\, z := x; x := y; y := z\, \{x = b \land y = a\}$:

$$\frac{\dfrac{}{\{x = a \land y = b\}\, z := x\, \{z = a \land y = b\}}\ (\textsc{Assign}) \qquad \begin{array}{c}\vdots\\ \{z = a \land y = b\}\, x := y; y := z\, \{x = b \land y = a\}\end{array}}{\{x = a \land y = b\}\, z := x; x := y; y := z\, \{x = b \land y = a\}}\ (\textsc{Seq})$$

The above proof relies on a proof for the premise $\{z = a \land y = b\}\, x := y; y := z\, \{x = b \land y = b\}$, which we build below:

$$\frac{\dfrac{}{\{z = a \land y = b\}\, x := y\, \{z = a \land x = b\}}\ (\textsc{Assign}) \qquad \dfrac{}{\{z = a \land x = b\}\, y := z\, \{x = b \land y = a\}}\ (\textsc{Assign})}{\{z = a \land y = b\}\, x := y; y := z\, \{x = b \land y = a\}}\ (\textsc{Seq})$$
$$\vdots$$

These two can be merged into one single (complete) proof tree. But we've broken them down into two parts due to limited space. This is the main disadvantage of using proof trees. We will see some alternatives in the next lecture note.

## 2.4 Conditional rule

The rule for conditional is based on the observation that if $e$ is true, then $S_1$ is executed, and if $\neg e$ is true, then $S_2$ is executed:

$$\frac{\{P \wedge e\}\, S_1\, \{Q\} \qquad \{P \wedge \neg\, e\}\, S_2\, \{Q\}}{\{P\}\, \mathsf{if}\ e\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ \mathsf{fi}\, \{Q\}}\ (\textsc{If})$$

**Example 7.** Here is a proof for $\vdash \{T\}\, \mathsf{if}\ x \geq y\ \mathsf{then}\ z := x\ \mathsf{else}\ z := y\ \mathsf{fi}\{z = \max(x, y)\}$:

$$\frac{\{T \wedge x \geq y\}\, z := x\, \{z = \max(x, y)\} \qquad \{T \wedge \neg\, x \geq y\}\, z := y\, \{z = \max(x, y)\}}{\{P\}\, \mathsf{if}\ e\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ \mathsf{fi}\, \{Q\}}\ (\textsc{If})$$

This proof relies on the proofs for the premises $\{T \wedge x \geq y\}\, z := x\, \{z = \max(x, y)\}$ and $\{T \wedge \neg\, x \geq y\}\, z := y\, \{z = \max(x, y)\}$. But How can we prove these two premises? They are not immediate applications of the assign rule (**Exercise 1.** Why not?). It turns out we need another rule called the consequence rule to prove these two premises. We will look at this rule next.

## 2.5 Consequence rule

The rule of consequence allows us to strengthen the precondition and weaken the postcondition:

$$\frac{P \Rightarrow P' \qquad \{P'\}\, S\, \{Q'\} \qquad Q' \Rightarrow Q}{\{P\}\, S\, \{Q\}}\ (\textsc{Consequence})$$

Here, premises $P \Rightarrow P'$ and $Q' \Rightarrow Q$ are different kind of judgments – they are logical implications. The judgment $P \Rightarrow P'$ says the precondition can always get stronger and $Q' \Rightarrow Q$ says the postcondition can always get weaker.

**Example 8.** We cannot prove $\vdash \{x = 2\}\, z := x\, \{z \geq 2\}$ using the assign rule, but we can prove $\vdash \{x = 2\}\, z := x\, \{z = 2\}$. By applying the consequence rule, we can prove the former from the latter. For example, we can weaken the postcondition:

$$\frac{x = 2 \Rightarrow x = 2 \qquad \dfrac{}{\vdash \{x = 2\}\, z := x\, \{z = 2\}}\ (\textsc{Assign}) \qquad z = 2 \Rightarrow z \geq 2}{\{x = 2\}\, z := x\, \{z \geq 2\}}\ (\textsc{Consequence})$$

or alternatively strengthen the precondition:

$$\frac{x = 2 \Rightarrow x \geq 2 \qquad \dfrac{}{\vdash \{x \geq 2\}\, z := x\, \{z \geq 2\}}\ (\textsc{Assign}) \qquad z \geq 2 \Rightarrow z \geq 2}{\{x = 2\}\, z := x\, \{z \geq 2\}}\ (\textsc{Consequence})$$

**Example 9.** Having the consequence rule, we can complete the proof of Example 7, by providing a proof for the premise $\{T \wedge x \geq y\}\, z := x\, \{z = \max(x, y)\}$:

$$\frac{\begin{array}{c} T \wedge x \geq y \Rightarrow x \geq x \wedge x \geq y \wedge (x = x \vee x = y) \\ \{x \geq x \wedge x \geq y \wedge (x = x \vee x = y)\}\, z := x\, \{z \geq x \wedge z \geq y \wedge (z = x \vee z = y)\} \\ z \geq x \wedge z \geq y \wedge (z = x \vee z = y) \Rightarrow z = \max(x, y) \end{array}}{\{T \wedge x \geq y\}\, z := x\, \{z = \max(x, y)\}}\ (\textsc{Consequence})$$

**Exercise 2.** Prove the second premise from Example 7, i.e., provide a derivation for $\vdash \{T \wedge \neg\, x \geq y\}\, z := y\, \{z = \max(x, y)\}$.

**Exercise 3. An alternative rule for the conditional.** Show that the following rule is equivalent to the If rule introduced in Section 2.4. (Hint: use the consequence rule and the fact that $Q_1 \Rightarrow Q_1 \vee Q_2$.)

$$\frac{\{P \wedge e\}\, S_1\, \{Q_1\} \qquad \{P \wedge \neg\, e\}\, S_2\, \{Q_2\}}{\{P\}\, \mathsf{if}\ e\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ \mathsf{fi}\, \{Q_1 \vee Q_2\}}\ (\textsc{If}^*)$$

### 2.6 Summary of the Hoare logic rules – so far

Here is the summary of the Hoare logic rules we have seen so far. We haven't discussed the while rule yet. We will talk about it later in the course.

$$\frac{}{\{P\}\,\mathsf{skip}\,\{P\}}\ (\textsc{Skip}) \qquad \frac{}{\{P[e/x]\}\,x := e\,\{P\}}\ (\textsc{Assign}) \qquad \frac{\{P\}\,S_1\,\{R\} \qquad \{R\}\,S_2\,\{Q\}}{\{P\}\,S_1; S_2\,\{Q\}}\ (\textsc{Seq})$$

$$\frac{\{P \wedge e\}\,S_1\,\{Q\} \qquad \{P \wedge \neg\, e\}\,S_2\,\{Q\}}{\{P\}\,\mathsf{if}\ e\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ \mathsf{fi}\,\{Q\}}\ (\textsc{If}) \qquad \frac{P \Rightarrow P' \qquad \{P'\}\,S\,\{Q'\} \qquad Q' \Rightarrow Q}{\{P\}\,S\,\{Q\}}\ (\textsc{Consequence})$$

## 3 Soundness and completeness

At this point we have two kinds of partial correctness assertions:

1. valid partial correctness triples $\Vdash \{P\}S\{Q\}$, which hold for all states and according to the operational semantics of S, and
2. provable partial correctness triples $\vdash \{P\}S\{Q\}$, which can be derived using the axioms and rules of Hoare logic.

The question is how do these two assertions relate to each other? More precisely, we have to answer two questions. First, are provable triples guaranteed to be valid? In other words,

$$\text{does } \vdash \{P\}S\{Q\} \text{ imply } \Vdash \{P\}S\{Q\} \text{ ?}$$

The answer is yes, and it shows that Hoare logic is *sound*. Soundness is important because it says that Hoare logic doesn't allow us to derive partial correctness assertions that actually are not valid. We won't cover the proof in this course, but for those of you who are interested, the proof is by induction on the derivation of $\vdash \{P\}S\{Q\}$.

The second question refers to the expressiveness and power of Hoare rules: can we always build a Hoare logic proof for each valid assertion? In other words,

$$\text{does } \Vdash \{P\}S\{Q\} \text{ imply } \vdash \{P\}S\{Q\} \text{ ?}$$

This is a bit more complicated. But the answer is a qualified yes: if $\Vdash \{P\}S\{Q\}$, then there is a proof of the triple, i.e., $\vdash \{P\}S\{Q\}$, using the rules of Hoare logic, provided there are proofs for the logical implication of predicates that occur in the rule of consequence, i.e., $P \Rightarrow P$ and $Q' \Rightarrow Q$. This result is known as the *relative completeness* of Hoare logic and is due to Cook [1]. Cook also showed that we cannot prove a standalone completeness result for Hoare logic as the underlying predicate logic with which we reason about preconditions and postconditions might be incomplete (By Gödel's incompleteness theorem).

[1] Cook, Stephen A. "Soundness and completeness of an axiom system for program verification." SIAM Journal on Computing 7.1 (1978): 70-90.