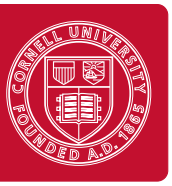
CS 4110 – Programming Languages and Logics
Lecture #9: Relative Completeness



Based on material by Stephen Chong, Greg Morrisett, Andrew Myers, George Necula, and Radu Rugina

1 Example: Factorial

As an example illustrating how we can use Hoare logic to verify the correctness of a program, consider a program that computes the factorial of a number n:

$$\{\mathsf{x} = n \wedge n > 0\}$$
$$\begin{array}{l} \mathsf{y} := 1; \\ \textbf{while } \mathsf{x} > 0 \textbf{ do } \{ \\ \quad \mathsf{y} := \mathsf{y} * \mathsf{x}; \\ \quad \mathsf{x} := \mathsf{x} - 1 \\ \} \end{array}$$
$$\{\mathsf{y} = n!\}$$

Because the derivation for this proof is somewhat large, we will go through the reasoning used to construct it step by step.

At the top level, the program is a sequence of an assignment and a loop. To use the SEQ rule, we need to find an assertion that holds after the assignment and before the loop. Examining the rule for while loops, we see that the assertion before the loop must be an invariant for the loop. Inspecting the loop we see that it builds the factorial up in y starting with n, then multiplying it by $n-1$, then $n-2$, etc. At each iteration, x contains the next value multiplied into y, that is:

$$\mathsf{y} = n * (n-1) * \ldots * (\mathsf{x}+1)$$

If we multiply both sides of this equality by x! and re-write the equality we get $\mathsf{x}! * \mathsf{y} = n!$, which is an invariant for the loop. However, to make the proof go through, we need a slightly stronger invariant:

$$I = \mathsf{x}! * \mathsf{y} = n! \wedge \mathsf{x} \geq 0$$

Having identified a suitable loop invariant, let us take a step back and review where we are. We want to prove that our overall partial correctness specification is valid. To do this, we need to show two facts:

$$\{\mathsf{x} = n \wedge n > 0\}\ \mathsf{y} := 1\ \{I\} \tag{1}$$
$$\{I\}\ \textbf{while } \mathsf{x} > 0 \textbf{ do } \{\mathsf{y} := \mathsf{y} * \mathsf{x}; \mathsf{x} := \mathsf{x} - 1\}\ \{\mathsf{y} = n!\} \tag{2}$$

After showing that both (1) and (2) hold, we can use the rule SEQ to obtain the desired result.

To show (1), we use the ASSIGN axiom and obtain the following: $\{I[1/\mathsf{y}]\}\ \mathsf{y} := 1\ \{I\}$. Expanding this out, we obtain:

$$\{\mathsf{x}! * 1 = n! \wedge \mathsf{x} \geq 0\}\ y := 1\ \{\mathsf{x}! * \mathsf{y} = n! \wedge \mathsf{x} \geq 0\}$$

With the following implication,

$$\mathsf{x} = n \wedge n > 0 \implies \mathsf{x}! * 1 = n! \wedge \mathsf{x} \geq 0,$$

(which can be shown by an easy calculation) we obtain (1) using the rule CONSEQUENCE.

Now let us prove (2). To use the WHILE rule, we need to show that I is an invariant for the loop:

$$\{I \wedge \mathsf{x} > 0\}\ \mathsf{y} := \mathsf{y} * \mathsf{x}; \mathsf{x} := \mathsf{x} - 1\ \{I\} \tag{3}$$

We will show this by going backwards through the sequence of assignments:

$$\{(\mathsf{x}-1)! * \mathsf{y} = n! \wedge (\mathsf{x}-1) \geq 0\}\ \mathsf{x} := \mathsf{x} - 1\ \{I\} \tag{4}$$

$$\{(\mathsf{x}-1)! * \mathsf{y} * \mathsf{x} = n! \wedge (\mathsf{x}-1) \geq 0\}\ \mathsf{y} := \mathsf{y} * \mathsf{x}\ \{(\mathsf{x}-1)! * \mathsf{y} = n! \wedge (\mathsf{x}-1) \geq 0\} \tag{5}$$

Then, using the following implication:

$$I \wedge \mathsf{x} > 0 \implies (\mathsf{x}-1)! * \mathsf{y} * \mathsf{x} = n! \wedge (\mathsf{x}-1) \geq 0$$

we obtain (3) using CONSEQUENCE, (4), and (5). Thus, I is an invariant for the loop and so by WHILE we obtain,

$$\{I\}\ \textbf{while}\ \mathsf{x} > 0\ \textbf{do}\ \{\mathsf{y} := \mathsf{y} * \mathsf{x}; \mathsf{x} := \mathsf{x} - 1\}\ \{I \wedge \mathsf{x} \leq 0\}$$

To finish the proof, we just have to show

$$\begin{array}{rl} & I \wedge \mathsf{x} \leq 0 \implies \mathsf{y} = n! \\ \textit{i.e.}, & \mathsf{x}! * \mathsf{y} = n! \wedge \mathsf{x} \geq 0 \wedge \mathsf{x} \leq 0 \implies \mathsf{y} = n! \end{array}$$

which holds as $\mathsf{x} \geq 0$ and $\mathsf{x} \leq 0$ implies $\mathsf{x} = 0$ and so $\mathsf{x}! = 1$. The result follows by CONSEQUENCE.

2 Relative Completeness

In the last lecture, we discussed the issue of completeness—i.e., whether it is possible to derive every valid partial correctness specification using the axioms and rules of Hoare logic. Sadly, Hoare logic is not complete. To see why, consider the following partial correctness specifications:

$$\{\textbf{true}\}\ \textbf{skip}\ \{P\} \qquad\qquad \{\textbf{true}\}\ c\ \{\textbf{false}\}$$

The first specification is valid if and only if the assertion P is valid while the second is valid if and only if the command c halts.

It turns out that the main culprit is the CONSEQUENCE rule:

$$\textsc{Consequence}\ \frac{\vDash (P \Rightarrow P') \qquad \{P'\}\ c\ \{Q'\} \qquad \vDash (Q' \Rightarrow Q)}{\{P\}\ c\ \{Q\}}$$

Although we cannot decide validity, Hoare logic does enjoy the property stated in the following theorem:

Theorem. $\forall P, Q \in \mathbf{Assn}, c \in \mathbf{Com}.\ \vDash \{P\}\ c\ \{Q\} \quad \textit{implies} \quad \vdash \{P\}\ c\ \{Q\}.$

This result, due to Cook (1974), is known as the *relative completeness* of Hoare logic. It says that Hoare logic is no more incomplete than our language of assertions—i.e., if we assume an oracle to decide the validity of assertions, then we can decide the validity of partial correctness specifications.

3 Weakest Liberal Preconditions

Cook's proof of relative completeness depends on the notion of *weakest liberal preconditions*. Given a command c and a postcondition Q the weakest liberal precondition is the weakest assertion P such that $\{P\}\ c\ \{Q\}$ is a valid triple. Here, "weakest" means that any other valid precondition implies P—i.e., P most accurately describes input states for which c either does not terminate or ends up in a state satisfying Q.

Formally, an assertion P is a weakest liberal precondition of c and Q if:

$$\forall \sigma, I.\ \sigma \vDash_I P \iff (\mathcal{C}[\![c]\!]\ \sigma) \text{ undefined } \vee (\mathcal{C}[\![c]\!]\sigma) \vDash_I Q$$

We will write $wlp(c, Q)$ for the weakest liberal precondition of command c and postcondition Q. From left-to-right, the formula above states that $wlp(c, Q)$ is a valid precondition: $\vDash \{P\}\ c\ \{Q\}$. The right-to-left implication says it is the weakest valid precondition: if another assertion R satisfies $\vDash \{R\}\ c\ \{Q\}$, then R implies P. It can be shown that weakest liberal preconditions are unique modulo equivalence.

We can calculate the weakest liberal precondition of a command as follows:

$$\begin{array}{rcl} wlp(\textbf{skip}, P) & = & P \\ wlp((\mathsf{x} := a, P) & = & P[a/\mathsf{x}] \\ wlp((c_1; c_2), P) & = & wlp(c_1, wlp(c_2, P)) \\ wlp(\textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2, P) & = & (b \implies wlp(c_1, P)) \wedge (\neg b \implies wlp(c_2, P)) \end{array}$$

The definition of $wlp(\textbf{while } b \textbf{ do } c, P)$ is a bit more complicated—it encodes the weakest liberal precondition for every possible iteration of the loop. The details are not important for this course (if you're curious, see Winskel Chapter 7).

To check that our definition is correct, we can prove (how?) that it yields a valid partial correctness specification:

Lemma 1.

$$\begin{array}{l} \forall c \in \mathbf{Com}, Q \in \mathbf{Assn}. \\ \quad \vDash \{wlp(c, Q)\}\ c\ \{Q\} \textit{ and } \forall R \in \mathbf{Assn}.\ \vDash \{R\}\ c\ \{Q\} \textit{ implies } (R \implies wlp(c, Q)) \end{array}$$

It is not hard to prove that it also yields a provable specification:

Lemma 2.

$$\forall c \in \mathbf{Com}, Q \in \mathbf{Assn}.\ \vdash \{wlp(c, Q)\}\ c\ \{Q\}$$

Relative completeness follows by a simple argument:

Proof Sketch. Let c be a command and let P and Q be assertions such that the partial correctness specification $\{P\}\ c\ \{Q\}$ is valid. By Lemma 1 we have $\vDash P \implies wlp(c, Q)$. By Lemma 2 we have $\vdash \{wlp(c, Q)\}\ c\ \{Q\}$. We conclude $\vdash \{P\}\ c\ \{Q\}$ using the CONSEQUENCE rule. □