CS611 Lecture 16 **Relative Completeness of Hoare Logic** 3 October 2003

Scribe: Maya Haridasan, Mohamed ElHawary Lecturer: Radu Rugina

---

In the last lecture we proved the concept of soundness, as being that every theorem in Hoare Logic is a valid assertion. We want now to find out if Hoare Logic is complete, that is, if every valid assertion is also a theorem in Hoare Logic:

$$\models \{A\}\ c\ \{B\} \quad \Longrightarrow \ \vdash \{A\}\ c\ \{B\}$$

The proof of completeness for Hoare logic uses the concept of *weakest preconditions.*

1 Weakest preconditions

Given c{B}, a command c and an assertion B, the weakest precondition $wp(c, B)$ is an assertion such that:

$$\forall \sigma, I \ \ \sigma \ \models^I wp(c, B) \quad \Longleftrightarrow \quad \mathcal{C}[\![c]\!]\sigma \models^I \{B\}$$

This definition shows two facts:

- $\Longrightarrow$ implication: $wp(c, B)$ is a precondition:

$$\models \{wp(c, B)\}\ c\ \{B\}$$

- $\Longleftarrow$ implication: $wp(c, B)$ is the weakest precondition: For any precondition A:

$$\models \{A\}\ c\ \{B\} \quad \Longrightarrow \quad (\models \{A\} \ \Longrightarrow \ wp(c, B))$$

1.1 Questions on weakest preconditions

- Existence and uniqueness: Does a weakest precondition exist for any command? and if so, is it unique?
- Expressiveness: If the precondition exists, can it be expressed by assertions?
- Can we prove using Hoare Logic that a weakest precondition is a precondition?

$$(\vdash \{wp(c, B)\}\ c\ \{B\})$$

The answers to these questions are yes. First, we will use these facts to prove the completeness result for Hoare logic. Next, we will go into more details about how to prove these facts.

2 Cook's Theorem(1974), Relative Completeness

It turns out that there is no proof system whose theorems are precisely all of the valid assertions $A \in \mathbf{Assn}$. Recall that the consequence rule has $\models A \Rightarrow A'$ and $\models B' \Rightarrow B$ in its premises. It means that there is no complete proof system which can derive these facts. As a result, Hoare Logic is not complete. However, if we assume we can get such proofs from an oracle, then the rest of Hoare logic is complete. This is known as the *partial completeness* of Hoare Logic.

Cook's Theorem . Hoare Logic is complete, modulo the derivations for assertions.

Proof. We want to prove that:

$$\models \{A\}\ c\ \{B\} \quad \Longrightarrow \ \vdash \{A\}\ c\ \{B\}$$

We know that the following is a theorem in Hoare Logic:

$$(\vdash \{wp(c, B)\}\ c\ \{B\})$$

From the definition of weakest precondition:

$$\models \{A\} \implies wp(c,B)$$

We can construct this proof tree:

$$\dfrac{A \Rightarrow wp(c,B) \qquad \dfrac{\vdots}{\{wp(c,B)\}\ c\ \{B\}} \qquad B \Rightarrow B}{\{A\}\ c\ \{B\}}$$

Here we assume that we get the proof for $A \Rightarrow wp(c,B)$ from an oracle. The rest of the proof, i.e., the derivation for $\{wp(c,B)\}\ c\ \{B\}$, we get it from Hoare Logic. Hence, we showed that $\vdash \{A\}\ c\ \{B\}$, which completes the proof.

Now we go back to the properties of weakest preconditions.

3 Properties of Weakest Precondition

3.1 Existence

We show the existence of weakest preconditions by construction. We define $wp(c,B)$ based on the structure of c:

- **skip**

$$wp(\mathbf{skip}, B) = B$$

- *Assignment*

$$wp(x{:=}\, a, B) = B[X \to a]$$

- *Sequencing*

$$wp(c_0; c_1, B) = wp(c_0, wp(c_1, B))$$

- *Conditionals*

$$wp(\mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1, B) = (b \wedge wp(c_0, B)) \vee (\neg\, b \wedge wp(c_1, B))$$

  An equivalent way to write this is:

$$(b \Rightarrow wp(c_0, B)) \wedge (\neg\, b \Rightarrow wp(c_1, B))$$

- *While loops*

$$wp(\mathbf{while}\, b\, \mathbf{do}\, c, B) = (\bigwedge P_i, i \geq 0) \;\; where \;\; P_0 = \mathbf{true}, P_{k+1} = (b \wedge wp(c_1, P_k)) \vee (\neg\, b \wedge B)$$

  Define a sequence of preconditions and apply the $\wedge$ operator. If we think of the states in which these conditions hold, the set of states decreases in this sequence. We can think of this as an ω-chain in the domain of sets of states, with the subset inclusion as the partial order. The states for which the infinite disjunction is valid is precisely the least upper bound of this chain.

  If $W \equiv wp(\mathbf{if}\, b\, \mathbf{then}\, c_0\, \mathbf{else}\, c_1$,B), one can prove the following properties:

$$W \wedge \neg\, b \Longrightarrow B$$

$$W \wedge b \Longrightarrow wp(c, W)$$

  The proof for these properties is left as an exercise.

With the above definitions, one can show that $wp(c,B)$ is indeed a weakest precondition for c and B. This involves showing that: a) it is a precondition; and b) it is the weakest. The proof is non-trivial and we leave it as an exercise.

3.2 Uniqueness

Suppose there are two weakest preconditions wp_1, wp_2

$$\sigma \models^I wp_1 \Longleftrightarrow \mathcal{C}[\![c]\!]\sigma \models^I B$$

$$\sigma \models^I wp_2 \Longleftrightarrow \mathcal{C}[\![c]\!]\sigma \models^I B$$

$$\Longrightarrow \sigma \models^I (wp_1 \Longleftrightarrow wp_2)$$

So, weakest preconditions are unique up to assertion equivalence.

3.3 Expressiveness

All the expressions $wp(c, B)$ defined above are in our assertion language **Assn**, except for the weakest precondition of the while loop, which is expressed as an infinite conjunction. It turns out that this infinite conjunction can be expressed as a finite formula in **Assn**, using Godel's β function. Proving this fact is not easy and details about it can be found in the textbook.

3.4 $(\vdash \{wp(c, B)\}\ c\ \{B\})$

We will prove that using structure induction on the command c. We only show the proof in the case of sequencing and while loops.

- *Sequencing* ($c \equiv c_0; c_1$) By Induction Hypothesis, we have $(\vdash \{wp(c_0, B)\}\ c_0\ \{B\})$ and $(\vdash \{wp(c_1, B)\}\ c_1\ \{B\})$. We can build the following derivation:

$$\dfrac{\dfrac{\vdots}{\{wp(c_0, wp(c_1, B))\}c_0\{wp(c_1, B)\}} \qquad \dfrac{\vdots}{\{wp(c_1, B)\}c_1\{B\}}}{\{wp(c_0; c_1, B)\}c_0; c_1\{B\}}$$

- *While loops* ($c \equiv$ **while** b **do** c): By Induction Hypothesis, we have $(\vdash \{wp(c, B)\}\ c\ \{B\})$ for any assertion B. Let W $\equiv wp(\textbf{while}\ b\ \textbf{do}\ c)$. We already know that:

$$W \wedge \neg\, b \Longleftrightarrow \neg\, b \wedge B$$

$$W \wedge b \Longleftrightarrow b \wedge wp(c, W)$$

  We can then build the following derivation for $\{W\}\textbf{while}\ b\ \textbf{do}\ c\{B\}$:

$$\dfrac{\dfrac{\dfrac{W \wedge b \Longrightarrow wp(c, W) \qquad \dfrac{\vdots}{\{wp(c, W)\}c_0\{W\}}}{\{W \wedge b\}c\{W\}}}{\{W\}\textbf{while}\ b\ \textbf{do}\ c\{W \wedge \neg\, b\}} \qquad W \wedge \neg\, b \Longrightarrow B}{\{W\}\textbf{while}\ b\ \textbf{do}\ c\{B\}}$$

The other rules are left as an exercise.

4 Verification Conditions and Proving Correctness

How can we build a tool to verify that Partial Correctness Assertions are valid? For instance, take $\{A\}c\{B\}$. How can we automate the process of proving whether or not this p.c.a. is valid for all states and all interpretations?

One approach would be to build a program which computes $wp(c, B)$ for any given c and B. Then, we have:

$$\models \{A\}c\{B\} \iff (\models A \Rightarrow wp(c, B))$$

Hence, proving the validity of $\{A\}c\{B\}$ only requires to prove that the formula $A \Rightarrow wp(c, B)$ is valid. Hence, we obtained a logic formula whose validity implies the validity of $\{A\}c\{B\}$. Such formulas are called *verification conditions*. One can then feed this formula into a theorem proved. If the prover succeeds in verifying it, then $\{A\}c\{B\}$. However, the problem is that for certain constructs, such as while loops, it may be difficult or impossible to automatically construct the weakest precondition.

Another alternative is to try to build a proof in Hoare Logic for $\{A\}c\{B\}$. Since Hoare Logic is sound, it means that would be enough to prove its validity. But if we look at the Hoare rules, there are two rules which may cause problems when trying to automatically construct a proof. First, in the rule for sequences $\{A\}c_0\,;\,c_1\{B\}$, the tool must guess what is the assertion $\{C\}$ which is a postcondition for c_0 and a precondition for c_1. Second, for while loops, it must guess what the loop invariant is.

A solution to these problems is to build language extensions and to require the user to provide such invariants and assertions. Suppose we extend IMP as follows:

$$c ::= \textbf{skip} \;\mid\; c_0; \{D\}c_1 \;\mid\; X := a \;\mid\; \textbf{if}\, b\, \textbf{then}\, c_0\, \textbf{else}\, c_1 \;\mid\; \textbf{while}\, b\, \textbf{do}\, \{D\}c$$

We can now easily construct verification conditions using the annotations provided by the user. We denote by $VC(\{A\}c\{B\})$ the verification condition for the p.c.a. $\{A\}c\{B\}$. We can easily construct $VC(\{A\}c\{B\})$ as follows:

$$VC(\{A\}\textbf{skip}\{B\}) = A \Longrightarrow B$$

$$VC(\{A\}X := a\{B\}) = A \Longrightarrow B[X \to a]$$

$$VC(\{A\}c_0; \{D\}; c_1\{B\}) = VC(\{A\}c_0\{D\}) \wedge VC(\{D\}c_1\{B\})$$

$$VC(\{A\}\textbf{if}\, b\, \textbf{then}\, c_0\, \textbf{else}\, c_1\{B\}) = VC(\{A \wedge b\}c_0\{B\}) \wedge VC(\{A \wedge \neg\, b\}c_1\{B\})$$

$$VC(\{A\}\ \textbf{while}\, b\, \textbf{do}\, \{D\}c_0\ \ \{B\}) = VC(\{D \wedge b\}c_0\{D\}) \wedge (A \Longrightarrow D) \wedge (D \wedge \neg\, b \Longrightarrow B)$$

One can prove that if the verification condition $VC(\{A\}c\{B\})$ holds, then the p.c.a. $\{A\}c_u\{B\}$ is valid, where c_u is the un-annotated version of c. (we leave this proof as an exercise)

Again, these verification conditions can be fed into a theorem prover and automatically checked. If the prover is able to prove them, then $\{A\}c_u\{B\}$ is guaranteed to be valid; otherwise, it may or may not be valid.