Introduction

In this lecture, we continue the discussions on the completeness of the proof rules for language IMP. In the last lecture, we found three apparent difficulties in proving a statement correct. Namely,

1. Occurence of the intermediate partial correctness assertion $\{C\}$ in the Hoare rule for a sequence,
2. The logical implication in the assertion $A \Rightarrow A'$ of the rule of consequence,
3. The logical implication in the assertion $B' \Rightarrow B$ of the rule of consequence.

Form our discussions in last lecture, we concluded that the first one does not pose any problem. We also proposed definitions of weakest pre-condition and strongest post-condition to tackle the second and third difficulties.

Definition : The *weakest pre-condition*, $wp(c, B)$, of a command c with post condition $\{B\}$ is defined as follows,

$$wp(c, B) \quad = \quad \{\sigma : \mathcal{C}[\![c]\!]\sigma \models^I B\} \tag{1}$$

From this definition, we could show the following,

$$\models \{A\}c\{B\} \Leftrightarrow \sigma \models^I A \Rightarrow \mathcal{C}[\![c]\!]\sigma \models^I B \tag{2}$$

$$\{\sigma : \sigma \models^I A\} \subseteq \{\sigma : \mathcal{C}[\![c]\!]\sigma \models^I B\} \tag{3}$$

Define A' as $wp(c, B) = \{\sigma \models^I A'\}$, then

$$A \Rightarrow A' \quad \Leftrightarrow \quad \{\sigma \models^I A\} \subseteq \{\sigma : \mathcal{C}[\![c]\!]\sigma \models^I B\} \tag{4}$$

$$\{A\}c\{B\} \quad \Leftrightarrow \quad A \Rightarrow wp(c, B) \tag{5}$$

In this lecture, we will give constructions for weakest pre-condition and use it to show that our proof system is relatively complete. We will follow this with a discussion on verification conditions, which is a technique to prove correctness of programs by inserting appropriate assertions in the code without leaving them to the choice of the prover. Finally, we will end the discussion on proof systems with a note on total correctness.

Weakest Pre-condition

We will give construction for the weakest pre-condition of commands of IMP as follows.

1. $wp(skip, B) = \{\sigma : \mathcal{C}[\![skip]\!]\sigma \models^I B\} = \{\sigma : \sigma \models^I B\} = B$
2. $wp(X := a, B) = \{\sigma : \mathcal{C}[\![X := a]\!]\sigma \models^I B\} = \{\sigma : \sigma[X \mapsto \mathcal{A}[\![a]\!]\sigma] \models^I B\}$
   $\Leftrightarrow \sigma[X \mapsto \mathcal{A}[\![a]\!]\sigma] \models^I B = \{\sigma : \sigma \models^I B[a/X]\} = B[a/X]$
   This follows from the lemma we stated in last lecture.
3. $wp(c_1; c_2, B) = wp(c_1, wp(c_2, B))$
4. $wp(\text{if b then } c_1 \text{ else } c_2, B) = (b \wedge wp(c_1, B)) \vee (\neg b \wedge wp(c_2, B))$
5. $wp(\text{while b do c}, B) = \{\sigma : \forall k \in \omega . \forall \sigma_0, \sigma_1, \cdots, \sigma_k$
   $[\sigma_0 = \sigma \wedge \forall i : 0 \le i < k . \sigma_i \models^I b \wedge \sigma_{i+1} = \mathcal{C}[\![c]\!]\sigma_i] \Rightarrow \sigma_k \models^I b \vee B\}$
   The actual expressiveness of the weakest precondition of a while statement involves godel numbers. For a more detailed discussion on this please refer to the textbook.

Lemma : We can always construct the proof for the fact

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

Proof : We will prove this using structural induction. The proof for the skip and assignment rules are trivial. We will consider the non-trivial case of a sequence. The proofs for the if statement and while statement are discussed in detail in the text book. In the case of $c = c_0; c_1$,

$$\begin{aligned} & \vdash \{wp(c_1, wp(c_2, B))\}c_1\{wp(c_2, B)\}\text{and} && (6)\\ & \vdash \{wp(c_2, B)\}c_2\{B\}, \text{by inductive hypothesis} && (7)\\ \Rightarrow \quad & \vdash \{wp(c_1, wp(c_2, B))\}c_1; c_2\{B\}, \text{by proof rule for sequence} && (8)\\ \Rightarrow \quad & \vdash \{wp(c_1; c_2, B)\}c_1; c_2\{B\}, \text{by wp construction for sequence} && (9) \end{aligned}$$

Theorem : The proof system for partial correctness is relatively complete. i.e., $\vdash \{A\}c\{B\}$ if $\models \{A\}c\{B\}$.
Proof : Since $\vdash \{wp(c, B)\}c\{B\}$ is true and further by consequence rule $\models (A \Rightarrow wp(c, B))$ is true, it follows that $\vdash \{A\}c\{B\}$.

We know that by Godel's incompleteness theorem, we cannot prove the total completeness of our proof system. In fact considering the weakest pre-condition $wp(c, false)$ where c does not terminate, we realise by proving this, we could solve the halting problem. This further emphasizes the sanctity of the incompleteness theorem.

Verification Conditions

Suppose we want to build a system which will automatically take a program c, pre-condition $\{A\}$ and a post-condition $\{B\}$ and will derive the proof of correctness for the program, we know that constructing weakest pre-conditions as defined above will not be practical. Especially in the case of while statements, constructing weakest pre-conditions is very complicated. So we allow code to be written with annotated partial correctness assertions. We will use these partial correctness assertions, to convert our prover into a proof verifier.

The rules for language IMP with annotated partial correctness assertions is as follows.

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

where D stands for a partial correctness assertion.

Now we can build a proof verifier by defining its semantics. A command c can be converted into a predicate as follows

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

The semantics for different structures of commands are as follows

1. $VC[\![\{A\}skip\{B\}]\!] = A \Rightarrow B$
2. $VC[\![\{A\}X := a\{B\}]\!] = A \Rightarrow B[a/X]$
3. $VC[\![\{A\}c_0; \{D\}c_1\{B\}]\!] = VC[\![\{A\}c_0\{D\}]\!] \wedge VC[\![\{D\}c_1\{B\}]\!]$
4. $VC[\![\{A\}\text{if b then } c_0 \text{ else } c_1\{B\}]\!] = VC[\![\{b \wedge A\}c_0\{B\}]\!] \wedge VC[\![\{\neg b \wedge A\}c_1\{B\}]\!]$
5. $VC[\![\{A\}\text{while b do } \{D\} \text{ c}\{B\}]\!]$
   $= VC[\![\{D \wedge b\}c\{D\}]\!] \wedge (A \Rightarrow D) \wedge (D \wedge \neg b \Rightarrow B)$

Note that the annotation for the while statement is the predicate which would represent a loop invariant. The appropriate choice of this by the prover is important for the verifier to approve the proof suggested by the prover.

Total Correctness

So far, we have discussed the proof verification of partial correctness only. When proving total correctness, we should also prove that the command terminates. Since only the while statement has a potential to not terminate, we try to solve this problem for while statement alone. The usual way of doing this is by defining a function d which decrements atleast by 1 with every iteration of the loop. Further, the loop terminates when this function goes below 0. When the control first enters the while loop, this function is verified for a positive value to check the termination of the loop. This can be expressed as follows

$$\frac{[A \wedge b \wedge v = d]c[A \wedge v > d] \quad \models d \leq 0 \Rightarrow \neg b,}{[A \wedge (d \geq 0)] \text{ while b do c } [A \wedge \neg b]}$$