CS611 Lecture 15 Axiomatics Semantics II and Hoare Logic 30 September, 2005
Scribe: Bryant Adams, Mia Minnes Lecturer: Michael Clarkson

1 Axiomatic Semantics II

1.1 Refinement

Definition. For programs S and T, we say that S **refines** T iff the set of possible outcomes of S is a (not necessarily strict) subset of the set of possible outcomes of T.

Consider, for example

$$\begin{aligned} \mathsf{S}: &\quad \mathsf{if}\ x = 1 \rightarrow y := 1 \,\Box\, x \neq 1 \rightarrow \mathsf{skip\ fi} \\ \mathsf{T}: &\quad \mathsf{if}\ x = 1 \rightarrow y := 1 \,\Box\, x = 1 \rightarrow y := 2 \,\Box\, x \neq 1 \rightarrow \mathsf{skip\ fi} \end{aligned}$$

Then the possible outcomes of S are $\{(x = 1, y = 1), (x \neq 1, y = ?)\}$, and the possible outcomes of T are $\{(x = 1, y = 1), (x = 1, y = 2), (x \neq 1, y = ?)\}$. Hence, S refines T.
Note that the definition of refinement is non-trivial only in the case of languages with non-deterministic choice. Otherwise, each program has exactly one possible outcome.

1.2 Weakest Liberal Preconditions

Recall that the weakest precondition for DO was not computationally tractable. But, we're often interested in computing preconditions in order to verify program correctness.

Definition. Q is a **liberal precondition** for program S with postcondition R if when Q is satisfied before execution, S terminating implies that R holds after execution.

Definition. The **weakest liberal precondition** (wlp) is the weakest condition which guarantees that the postcondition is met if the program terminates (so the program is allowed to not terminate). More formally, $\mathsf{Q} = wlp.\mathsf{S}.\mathsf{R}$ if and only if Q is a liberal precondition and for all liberal preconditions Q', $\mathsf{Q}' \implies \mathsf{Q}$.

In fact, we won't be able to give an easily computable definition for the wlp of DO. But, we can find some liberal precondition (that won't necessarily be the weakest). In other words, we will find Q such that $\mathsf{Q} \implies wlp.\mathsf{S}.\mathsf{R}$.
Recall that DO loops are defined as

$$\begin{aligned} D &::= \mathsf{DO} B_1 \rightarrow S_1 \Box \cdots \Box B_n \rightarrow S_n \mathsf{OD} \\ \mathsf{BB} &\triangleq \exists i : B_i \\ \mathsf{IF} &\triangleq \mathsf{IF} B_1 \rightarrow S_1 \Box \cdots \Box B_n \rightarrow S_n \mathsf{FI} \qquad \text{corresponding to DO} \end{aligned}$$

We say that $wlp.\mathsf{DO}.\mathsf{R} \leftarrow \mathsf{Q}$ if the following conditions are met for any P:

1. $\mathsf{Q} \implies \mathsf{P}$
2. $\mathsf{P} \wedge \mathsf{BB} \implies wlp.\mathsf{IF}.\mathsf{P}$
3. $\mathsf{P} \wedge \neg\mathsf{BB} \implies \mathsf{R}$.

The P in the definition above is a **loop invariant**: it holds before and after the execution of the loop. This is the definition of loop invariants. Note that this definition allows the invariant not to hold somewhere inside iterations of the loop, so long as once the loop iteration is complete, the invariant holds again.
Let's look at an example of an invariant. Consider the program which for some function $f : \mathsf{Int} \rightarrow \mathsf{Bool}$ finds the least x such that $f.x$ (assume that $\exists x : f.x$ so program will terminate). Define the program as:

$$\begin{aligned} x :=&\ 0; \\ \mathsf{DO}&\ \neg f.x \rightarrow x := x + 1; \mathsf{OD} \end{aligned}$$

An appropriate postcondition to capture what we want the program to do is

$$\mathsf{R}: \qquad f.x \wedge (\forall i | 0 \leq i < x : \neg f.i)$$

(read as "f holds of x and for all numbers smaller than x, f doesn't hold of them"). One method of finding a good precondition is by looking at ways of weakening the postcondition (i.e. allowing more states to satisfy the predicate). For example, in this case, we can eliminate the conjunct asserting that we've already found a good x. This yields the precondition

$$\mathsf{P}: \qquad \forall i | 0 \leq i < x : \neg f.i$$

Using the definition of *wlp* we can check that P itself is a liberal precondition for this program.
We can also define *wlp*'s for the rest of the GCL syntax:

- As before, $wlp.\mathsf{skip}.\mathsf{R} = \mathsf{R}$, $wlp.\mathsf{x} := \mathsf{e}.\mathsf{R} = \mathsf{R}^x_\mathsf{e}$, $wlp.\mathsf{S}_1; \mathsf{S}_2.\mathsf{R} = wlp.\mathsf{S}_1.(wlp.\mathsf{S}_2.\mathsf{R})$
- $wlp.\mathsf{IF}.\mathsf{R} = (\forall i : B_i \implies wlp.\mathsf{S_i}.\mathsf{R})$ (Note that we no longer require BB.)
- *wlp* for DO is above.

2 Hoare Logic

As we saw in the section above, calculating preconditions by hand is hard and not always tractable. We will now define a logic which allows us to reason about when assertions hold and therefore (hopefully) bypass most of these kinds of computations.
There are two kinds of approaches to program verification:

- Partial correctness: check if program is correct when it terminates (for example *wlp* and the Hoare logic we'll define shortly)
- Total correctness: ensure both that the program terminates and that it is correct (for example, *wp*).

2.1 Language and Semantics

To define Hoare logic, we need to define the language of the formula, i.e what are well-formed formulas in the logic. The formulas will be **Hoare triples**, and will look like:

$$\{\mathsf{Q}\}\mathsf{S}\{\mathsf{R}\}.$$

Informally, when we write a Hoare triple we will want to mean "if Q holds before execution of S then if S terminates, R holds after execution". Formally, we introduce the validity symbol $\vDash$ and define:

$$\vDash \{\mathsf{Q}\}\mathsf{S}\{\mathsf{R}\} \qquad \Longleftrightarrow \qquad \mathsf{Q} \implies wlp.\mathsf{S}.\mathsf{R}$$

Comment: The definition above of the meaning of Hoare triples is only one of many possible definitions. The definition of Hoare triples depends on the semantics of the programming language being analyzed.
We have yet to formalize what predicates can be. Assume that there is some set *Pred* such that $\mathsf{Q}, \mathsf{R} \in Pred$ and that we also have a semantics on this set, i.e. a way of determining whether Q, R hold. More notation:

$$I[\![\mathsf{Q}]\!] \qquad \equiv \qquad \vDash_I \mathsf{Q}.$$

which (in words) says that "Q has meaning $[\![\mathsf{Q}]\!]$ according to I is equivalent to Q being valid in interpretation I ". One catch - whether a predicate holds or not may depend on the value of variables (i.e. on the store). So, we say that $\vDash_I \mathsf{Q}$ if $\forall \sigma . \sigma \vDash_I \mathsf{Q}$, or $\forall \sigma . I[\![\mathsf{Q}]\!]\sigma$. Now we can write the semantics of Hoare triples more formally:

$$\vDash_I \{\mathsf{Q}\}\mathsf{S}\{\mathsf{R}\} \quad \triangleq \quad \vDash_I \mathsf{Q} \implies wlp.\mathsf{S}.\mathsf{R} \quad \equiv \quad \forall \sigma . (\sigma \vDash_I \mathsf{Q} \implies \sigma \vDash_I wlp.\mathsf{S}.\mathsf{R})$$

and in words: "a Hoare triple condition is valid if for every state, if the state satisfies Q then it also satisfies the weakest liberal precondition of S and R ".

2.2 Proof rules

Let $\vdash$ be the symbol for **provability** and we'd like to ask when we can write $\vdash \{\mathsf{Q}\}\mathsf{S}\{\mathsf{R}\}$, i.e. when we can say that the Hoare triple is provable. We need to start with a set of axioms and a set of rules of inferences. Note that these axioms and rules of inference depend on which language we are verifying.
For IMP: We give rule schema (recall that these are valid for any instantiation of the meta-variables).

1. (Axiom) $\dfrac{}{\vdash \{\mathsf{Q}\}\mathsf{skip}\{\mathsf{Q}\}}$

2. (Axiom) $\dfrac{}{\vdash \{\mathsf{Q}^x_\mathsf{e}\}\, x := \mathsf{e}\,\{\mathsf{Q}\}}$

3. (Rule - for sequences) $\dfrac{\vdash \{\mathsf{Q}\}\mathsf{S}_1\{\mathsf{T}\} \qquad \vdash \{\mathsf{T}\}\mathsf{S}_2\{\mathsf{R}\}}{\vdash \{\mathsf{Q}\}\mathsf{S}_1;\mathsf{S}_2\{\mathsf{R}\}}$

4. (Rule - for if statements) $\dfrac{\vdash \{\mathsf{Q} \wedge \mathsf{B}\}\mathsf{S}_1\{\mathsf{R}\} \qquad \vdash \{\mathsf{Q} \wedge \neg\mathsf{B}\}\mathsf{S}_2\{\mathsf{R}\}}{\vdash \{\mathsf{Q}\}\mathsf{if}\ \mathsf{B}\ \mathsf{then}\ \mathsf{S}_1\ \mathsf{else}\ \mathsf{S}_2\{\mathsf{R}\}}$

5. (Rule - establishes loop invariants) $\dfrac{\vdash \{\mathsf{P} \wedge \mathsf{B}\}\mathsf{S}\{\mathsf{P}\}}{\vdash \{\mathsf{P}\}\mathsf{while}\ \mathsf{B}\ \mathsf{do}\ \mathsf{S}\{\mathsf{P} \wedge \neg\mathsf{B}\}}$

6. (Rule of consequence - uses semantics of predicates) $\dfrac{\models_I \mathsf{Q} \implies \mathsf{Q}' \qquad \vdash \{\mathsf{Q}'\}\mathsf{S}\{\mathsf{R}'\} \qquad \models_I \mathsf{R}' \implies \mathsf{R}}{\vdash \{\mathsf{Q}\}\mathsf{S}\{\mathsf{R}\}}$

Note the parallels between the above proof rules and the definitions of *wlp*. This will be the reason for the following observations.
Two fundamental properties of a logic are

1. Soundness: The proof system never lets you derive formulas that are false.

$$\vdash \{\mathsf{Q}\}\mathsf{S}\{\mathsf{R}\} \implies \models \{\mathsf{Q}\}\mathsf{S}\{\mathsf{R}\}$$

2. Completeness: There are no true formulas which are not provable.

$$\models \{\mathsf{Q}\}\mathsf{S}\{\mathsf{R}\} \implies \vdash \{\mathsf{Q}\}\mathsf{S}\{\mathsf{R}\}$$

Hoare logic as presented is sound but not complete. However, it is relatively complete given an oracle for the semantics of predicates (in particular, the oracle answers the question "Does P imply Q "for predicates), if the predicate language is expressive (i.e. if it is possible to write any precondition as a predicate). For more on this, see Winskel.