1 Axiomatic Semantics II

1.1 Refinement

For nondeterministic programs S and T, we say that S *refines* T iff for any starting state, the set of possible final states of S is a (not necessarily strict) subset of the set of possible final states of T.

Consider for example

$$\begin{aligned} \mathsf{S} &\triangleq \mathsf{if}\ x = 1 \to y := 1 \ [\!]\ x \neq 1 \to \mathsf{skip}\ \mathsf{fi} \\ \mathsf{T} &\triangleq \mathsf{if}\ x = 1 \to y := 1 \ [\!]\ x = 1 \to y := 2 \ [\!]\ x \neq 1 \to \mathsf{skip}\ \mathsf{fi} \end{aligned}$$

For input state $(x = a, y = b)$, the only possible final states of S are $(x = 1, y = 1)$ if $a = 1$ and $(x = a, y = b)$ if $a \neq 1$, whereas the possible final states of T are $\{(x = 1, y = 1), (x = 1, y = 2)\}$ if $a = 1$ and just $(x = a, y = b)$ if $a \neq 1$, therefore S refines T.

The refinement relation is usually used only with programming languages with some form of nondeterministic choice as a relative measure of how nondeterministic a program is. A correctness specification might be written using nondeterministic choice, since often we may be happy with any one of a range of outcomes. Any deterministic program that refines the specification is considered correct.

1.2 Weakest Liberal Preconditions

Recall that the weakest precondition of a program S and a postcondition φ is the weakest precondition that guarantees that S halts and satisfies φ upon halting.

The *weakest liberal precondition* (wlp) of a program S and a postcondition φ is the weakest precondition that guarantees that if S halts, then it satisties φ upon halting. The weakest liberal precondition of S and φ is denoted $\mathsf{wlp}\ S\ \varphi$.

The difference between $\mathsf{wp}\ S\ \varphi$ and $\mathsf{wlp}\ S\ \varphi$ is that $\mathsf{wp}\ S\ \varphi$ implies that S terminates, whereas $\mathsf{wlp}\ S\ \varphi$ does not. Since $\mathsf{wlp}\ S\ \varphi$ is weaker than $\mathsf{wp}\ S\ \varphi$, it is presumably easier to establish.

Recall that weakest preconditions of the do construct of GCL are not necessarily expressible in first-order logic. In fact, the same will be true of the weakest liberal preconditions. However, we may be able to find *some* precondition that will be sufficient to establish correctness, even though that precondition may not necessarily be the weakest. In other words, we will find ψ such that $\psi \Rightarrow \mathsf{wlp}\ S\ \varphi$.

Recall that if and do statements look like

$$\begin{aligned} &\mathsf{if}\ B_1 \to S_1 \ [\!]\ B_2 \to S_2 \ [\!] \cdots [\!]\ B_n \to S_n\ \mathsf{fi} \\ &\mathsf{do}\ B_1 \to S_1 \ [\!]\ B_2 \to S_2 \ [\!] \cdots [\!]\ B_n \to S_n\ \mathsf{od} \end{aligned}$$

Define

$$B \triangleq \bigvee_{i=1}^{n} B_i.$$

We will look for a property ψ such that

(i) $\psi \wedge B \Rightarrow \mathsf{wlp}\ (\mathsf{if}\ B_1 \to S_1 \ [\!]\ B_2 \to S_2 \ [\!] \cdots [\!]\ B_n \to S_n\ \mathsf{fi})\ \psi$

(ii) $\psi \wedge \neg B \Rightarrow \varphi$.

Property (i) says that ψ is a *loop invariant*: if it holds before execution of the body of the do loop, and if at least one clause in the body is enabled, then it holds after one execution of the body. It follows by induction

that if ψ holds before execution of the do loop, then it will hold after any number of iterations of the loop. Property (ii) says that if ψ holds and no clause of the loop is enabled (so that the loop will fall through), then the postcondition φ is satisfied.

These observations say that the following proof rule is valid:

$$\frac{\psi \wedge B \;\Rightarrow\; \mathsf{wlp}\ (\mathsf{if}\ B_1 \to S_1 \,[]\, B_2 \to S_2 \,[]\, \cdots \,[]\, B_n \to S_n\ \mathsf{fi})\ \psi \qquad \psi \wedge \neg B \;\Rightarrow\; \varphi}{\psi \;\Rightarrow\; \mathsf{wlp}\ (\mathsf{do}\ B_1 \to S_1 \,[]\, B_2 \to S_2 \,[]\, \cdots \,[]\, B_n \to S_n\ \mathsf{do})\ \varphi}.$$

Note that a loop invariant need not hold continuously throughout the execution of the body of the loop. It is enough that it holds when the loop iteration is complete.

For example, consider the following program, which for some function $f : \mathsf{Int} \to \mathsf{Bool}$ finds the least x such that $f(x)$ (assume that such an x exists, so that the program will terminate).

$$\begin{array}{l} x := 0; \\ \mathsf{do}\ \neg f(x) \to x := x + 1\ \mathsf{od} \end{array}$$

An appropriate postcondition that specifies what it means for the program to be correct is

$$\varphi \quad \stackrel{\triangle}{\Longleftrightarrow} \quad f(x) \;\wedge\; \forall y \;\; 0 \le y < x \;\Rightarrow\; \neg f(y).$$

(read as "f holds of x and does not hold of any number smaller than x"). One method of finding a good loop invariant is to look at ways of weakening the postcondition, thereby allowing more states to satisfy the predicate. In this case, we can eliminate the conjunct asserting that we have already found a good x. This yields the invariant

$$\psi \quad \stackrel{\triangle}{\Longleftrightarrow} \quad \forall y \;\; 0 \le y < x \;\Rightarrow\; \neg f(y).$$

One can show that this is indeed an invariant and satisfies the two premises of the proof rule above with $B = \neg f(x)$, therefore

$$\psi \quad \Rightarrow \quad \mathsf{wlp}\ (\mathsf{do}\ \neg f(x) \to x := x + 1\ \mathsf{od})\ \varphi.$$

The definitions of $\mathsf{wlp}\ S\ \varphi$ for the other basic constructs of GLC are the same as $\mathsf{wp}\ S\ \varphi$ except for if, which is

$$\mathsf{wlp}\ (\mathsf{if}\ B_1 \to S_1 \,[]\, \cdots \,[]\, B_n \to S_n\ \mathsf{fi})\ \varphi \quad \stackrel{\triangle}{=} \quad \bigwedge_{i=1}^{n} (B_i \Rightarrow \mathsf{wlp}\ S_i\ \varphi).$$

Note that we no longer require B as with wp, since we do not have to ensure halting.

2 Hoare Logic

As we have seen, 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.

2.1 Partial vs. Total Correctness

Two approaches to program verification are:

- *Partial correctness*: check if program is correct when it terminates. This is characterized by wlp and the Hoare logic we will define shortly. The termination issue is handled separately.
- *Total correctness*: ensure both that the program terminates and that it is correct. This is characterized by wp.

Partial correctness is the more common approach nowadays, since it separates the two issues of correctness and termination. These two verification tasks use very different methods, and it is helpful to separate them. Often partial correctness is easier to establish, and once this is done, the correctness conditions can be used in conjunction with a well-founded relation to establish termination.

2.2 Syntax of Hoare Logic

To define Hoare logic, we need to define the well-formed formulas in the logic. Hoare logic is built on top of another conventional logic, such as first-order logic. For now, let us take first-order logic as our base logic. Let $\varphi, \psi, \ldots$ denote first-order formulas. The formulas of Hoare logic are *partial correctness assertions* (PCA's), also known as *Hoare triples*. They look like

$$\{\varphi\}\,\mathsf{S}\,\{\psi\}.$$

Informally, this means, "if φ holds before execution of S, and if S terminates, then ψ will hold upon termination." This is equivalent to

$$\varphi \quad \Rightarrow \quad \mathsf{wlp}\ S\ \psi.$$

2.3 Proof Rules

We will discuss the semantics of Hoare logic later. For now, we just give the deduction rules for the language IMP with programs

$$c \quad ::= \quad \mathsf{skip} \ \mid\ x := a \ \mid\ c_0; c_1 \ \mid\ \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 \ \mid\ \mathsf{while}\ b\ \mathsf{do}\ c.$$

The rules are

(skip) $\quad \{\varphi\}\,\mathsf{skip}\,\{\varphi\}$

(assignment) $\quad \{\varphi\{a/x\}\}\,x := a\,\{\varphi\}$

(sequential composition) $\quad \dfrac{\{\varphi\}\,c_1\,\{\psi\} \quad \{\psi\}\,c_2\,\{\sigma\}}{\{\varphi\}\,c_1; c_2\,\{\sigma\}}$

(conditional) $\quad \dfrac{\{b \wedge \varphi\}\,c_1\,\{\psi\} \quad \{\neg b \wedge \varphi\}\,c_2\,\{\psi\}}{\{\varphi\}\,\mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2\,\{\psi\}}$

(while) $\quad \dfrac{\{b \wedge \varphi\}\,c\,\{\varphi\}}{\{\varphi\}\,\mathsf{while}\ b\ \mathsf{do}\ c\,\{\varphi \wedge \neg b\}}$

(weakening) $\quad \dfrac{\varphi \to \varphi' \quad \{\varphi'\}\,c\,\{\psi'\} \quad \psi' \to \psi}{\{\varphi\}\,c\,\{\psi\}}.$

In the assignment rule, $\varphi\{a/x\}$ denotes the safe substitution of the arithmetic expression a for the variable x in φ. As with the λ-calculus, there may be bound variables in φ bound by quantifiers $\forall$ and $\exists$, and these may have to be renamed to avoid capturing the free variables of a. In the weakening rule, the operator $\to$ is implication in the underlying logic. Note the parallels between these rules and the definitions of wlp.

2.4 Soundness and Completeness

A deduction system defines what it means for a formula to be *provable*, whereas a semantics defines what it means for a formula to be *true*. Given a logic with a semantics and a deduction system, two desirable properties are

- *Soundness*: Every provable formula is true.
- *Completeness*: Every true formula is provable.

Soundness is a basic requirement of any logical system. A logic would not be good for much if its theorems were false! With respect to the small-step or big-step semantics of IMP, Hoare logic is sound.

Completeness, on the other hand, is a much more difficult issue. Hoare logic, as presented, is not complete in general. However, it is *relatively complete* given an oracle for truth in the underlying logic, provided that logic is expressive enough to express weakest preconditions. This is a famous result of Cook. Although first-order logic is not expressive enough to express weakest preconditions over arbitrary domains of computation, it is expressive enough over the natural numbers. Therefore Hoare logic is relatively complete for IMP programs over the integers.