1 Axiomatic Semantics

So far we have focused on *operational semantics*, which are natural for modeling computation or talking about how state changes from one step of the computation to the next. In operational semantics, there is a well-defined notion of *state*. We take great pains to say exactly what a state is and how it is manipulated by a program.

In *axiomatic semantics*, on the other hand, we do not so much care what the states actually are, but only the properties that we can observe about them. This approach emphasizes the relationship between the properties of the input (preconditions) and properties of the output (postconditions). This approach is useful for specifying what a program is supposed to do and talk about a program's correctness with respect to that specification.

2 Preconditions and Postconditions

The *preconditions* and *postconditions* of a program say what is true before and after the program executes, respectively. Often the correctness of the program is specified in these terms. Typically this is expressed as a contract: as long as the caller guarantees that the initial state satisfies some set of preconditions, then the program will guarantee that the final state will satisfy some desired set of postconditions. Axiomatic semantics attempts to say exactly what preconditions are necessary for ensuring a given set of postconditions.

3 An Example

Consider the following program to compute x^p:

```
$y := 1;$
$q := 0;$
while $q < p$ {
    $y := y \cdot x;$
    $q := q + 1;$
}
```

The desired postcondition we would like to ensure is $y = x^p$; that is, the final value of the program variable y is the pth power of x. We would also like to ensure that the program halts. One essential precondition needed to ensure halting is $p \geq 0$, because the program will only halt and compute x^p correctly if that holds. Note that $p > 0$ will also guarantee that the program halts and produces the correct output, but this is a stronger condition (is satisfied by fewer states, has more logical consequences).

$$\underbrace{p > 0}_{\text{stronger}} \quad \Rightarrow \quad \underbrace{p \geq 0}_{\text{weaker}}$$

The weaker precondition is better because it is less restrictive of the possible starting values of p that ensure correctness. Typically, given a postcondition expressing a desired property of the output state, we would like to know the *weakest precondition* that guarantees that the program halts and satisfies that postcondition upon termination.

4 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, 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.

5 Hoare Logic

Hoare logic is named for its inventor, Sir Charles Antony Richard Hoare (1934–), who also invented quicksort. It is a logic for reasoning about the relationship between pre- and postconditions.

To define the syntax Hoare logic, we need to define the well-formed formulas. 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

$$\{\varphi\}\, c\, \{\psi\}.$$

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

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

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

$$\text{(while)} \qquad \frac{\{b \wedge \varphi\}\, c\, \{\varphi\}}{\{\varphi\}\, \mathsf{while}\ b\ \mathsf{do}\ c\, \{\varphi \wedge \neg b\}}$$

$$\text{(weakening)} \qquad \frac{\varphi \Rightarrow \varphi' \quad \{\varphi'\}\, c\, \{\psi'\} \quad \psi' \Rightarrow \psi}{\{\varphi\}\, c\, \{\psi\}}.$$

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 $\Rightarrow$ is implication in the underlying logic. Note the parallels between these rules and the definitions of wlp.

6 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 Stephen Cook (1939–), the discoverer NP-completeness. Although first-order logic is not expressive enough to express weakest preconditions over arbitrary domains of computation, it is expressive enough over $\mathbb{N}$ or $\mathbb{Z}$. Therefore Hoare logic is relatively complete for IMP programs over the integers.

7 Semantics of IMP Revisited

Recall the big-step operational rules of IMP and their characterization in terms of binary relations on states $\sigma : \mathit{Var} \to \mathbb{Z}$. The big-step rules are

$$\langle \mathsf{skip},\, \sigma\rangle \Downarrow_c \sigma \qquad \frac{\langle a,\, \sigma\rangle \Downarrow_a n}{\langle x := a,\, \sigma\rangle \Downarrow_c \sigma[n/x]} \qquad \frac{\langle c_0,\, \sigma\rangle \Downarrow_c \tau \qquad \langle c_1,\, \tau\rangle \Downarrow_c \rho}{\langle c_0\ ;\ c_1,\, \sigma\rangle \Downarrow_c \rho}$$

$$\frac{\langle b,\, \sigma\rangle \Downarrow_b \mathsf{true} \qquad \langle c_1,\, \sigma\rangle \Downarrow_c \tau}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2,\, \sigma\rangle \Downarrow_c \tau} \qquad \frac{\langle b,\, \sigma\rangle \Downarrow_b \mathsf{false} \qquad \langle c_2,\, \sigma\rangle \Downarrow_c \tau}{\langle \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2,\, \sigma\rangle \Downarrow_c \tau}$$

$$\frac{\langle b,\, \sigma\rangle \Downarrow_b \mathsf{false}}{\langle \mathsf{while}\ b\ \mathsf{do}\ c,\, \sigma\rangle \Downarrow_c \sigma} \qquad \frac{\langle b,\, \sigma\rangle \Downarrow_b \mathsf{true} \qquad \langle c,\, \sigma\rangle \Downarrow_c \tau \qquad \langle \mathsf{while}\ b\ \mathsf{do}\ c,\, \tau\rangle \Downarrow_c \rho}{\langle \mathsf{while}\ b\ \mathsf{do}\ c,\, \sigma\rangle \Downarrow_c \rho}$$

Let Env be the set of all states $\sigma : \mathit{Var} \to \mathbb{Z}$. For each program c, the big-step rules determine a binary input/output relation on Env, namely

$$[\![c]\!] \triangleq \{(\sigma, \tau) \mid \langle c,\, \sigma\rangle \Downarrow_c \tau\} \quad \subseteq \quad \mathit{Env} \times \mathit{Env}.$$

With this notation, we can express the big-step rules in terms of some basic operations on binary relations, namely *relational composition* (;) and *reflexive transitive closure* (*):

$$R\ ;\ S \triangleq \{(\sigma,\rho) \mid \exists\tau\ (\sigma,\tau)\in R,\ (\tau,\rho)\in S\}$$

$$R^* \triangleq \bigcup_{n\geq 0} R^n = \{(\sigma,\tau) \mid \exists\sigma_0,\ldots,\sigma_n\ \sigma=\sigma_0,\ \tau=\sigma_n,\ \text{and } (\sigma_i,\sigma_{i+1})\in R,\ 0\leq i\leq n-1\},$$

where $R^0 \triangleq \{(\sigma,\sigma) \mid \sigma\in \mathit{Env}\}$ and $R^{n+1} \triangleq R^n\ ;\ R$. The big-step rules are equivalent to the following:

$$
\begin{aligned}
[\![\mathsf{skip}]\!] &= \{(\sigma,\sigma) \mid \sigma\in \mathit{Env}\} && (\mathsf{skip})\\
[\![x := a]\!] &= \{(\sigma,\sigma[n/x]) \mid \langle a,\,\sigma\rangle \Downarrow_a n\} && \text{(assignment)}\\
[\![c_0\ ;\ c_1]\!] &= [\![c_0]\!]\ ;\ [\![c_1]\!] && \text{(sequential composition)}\\
[\![\mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2]\!] &= [\![b]\!]\ ;\ [\![c_1]\!]\ \cup\ [\![\neg b]\!]\ ;\ [\![c_2]\!] && \text{(conditional)}\\
[\![\mathsf{while}\ b\ \mathsf{do}\ c]\!] &= ([\![b]\!]\ ;\ [\![c]\!])^*\ ;\ [\![\neg b]\!] && \text{(while loop)},
\end{aligned}
$$

where in the conditional and while loop,

$$
\begin{aligned}
[\![b]\!] &\triangleq \{(\sigma,\sigma) \mid \langle b,\,\sigma\rangle \Downarrow_b \mathsf{true}\}\\
[\![\neg b]\!] &\triangleq \{(\sigma,\sigma) \mid \langle b,\,\sigma\rangle \Downarrow_b \mathsf{false}\} = [\![\mathsf{skip}]\!] - [\![b]\!].
\end{aligned}
$$

In fact, this would have been a much more compact way to define them originally.

7.1 Semantics of Weakest Liberal Preconditions and Partial Correctness Assertions

We can now give a formal semantics for weakest liberal preconditions and Hoare partial correctness assertions. Let L denote the underlying logic (typically first-order logic). Write $\sigma \vDash \varphi$ if the formula φ of L is true in state σ, and write $\vDash \varphi$ if φ is true in all states. We wish to define what it means for a weakest liberal precondition assertion $\mathsf{wlp}\ c\ \psi$ to be true in a state σ, written $\sigma \vDash \mathsf{wlp}\ c\ \psi$, and for a partial correctness assertion $\{\varphi\}\,c\,\{\psi\}$ to be true, written $\vDash \{\varphi\}\,c\,\{\psi\}$.

$$
\begin{aligned}
\sigma \vDash \mathsf{wlp}\ c\ \psi &\overset{\triangle}{\Leftrightarrow} \forall\tau\quad (\sigma,\tau)\in[\![c]\!]\ \Rightarrow\ \tau\vDash\psi\\
\vDash \{\varphi\}\,c\,\{\psi\} &\overset{\triangle}{\Leftrightarrow} \forall\sigma\quad \sigma\vDash\varphi\ \Rightarrow\ \sigma\vDash \mathsf{wlp}\ c\ \psi\\
&\Leftrightarrow\ \forall\sigma,\tau\quad \sigma\vDash\varphi\wedge(\sigma,\tau)\in[\![c]\!]\ \Rightarrow\ \tau\vDash\psi.
\end{aligned}
$$

7.2 Soundness and Relative Completeness of Hoare Logic

Let us write $\vdash \{\varphi\}\,c\,\{\psi\}$ to assert that $\{\varphi\}\,c\,\{\psi\}$ is provable in Hoare logic. Then soundness and relative completeness of Hoare logic are captured in the following theorems.

Theorem 16.1 (Soundness). $\vdash \{\varphi\}\,c\,\{\psi\}\ \Rightarrow\ \vDash \{\varphi\}\,c\,\{\psi\}$.

Theorem 16.2 (Relative Completeness). *Assume that the underlying logic L is* expressive *in the sense that all weakest liberal preconditions are expressible in L; that is, for each program c and formula ψ of L, there is a formula ψ' of L such that for all σ, $\sigma\vDash\psi'$ iff $\sigma\vDash \mathsf{wlp}\ c\ \psi$. Then $\vDash \{\varphi\}\,c\,\{\psi\}\ \Rightarrow\ \vdash \{\varphi\}\,c\,\{\psi\}$, provided we are allowed to assume all true formulas of L as axioms.*

Proof sketch. The proof is by structural induction on c. We will just sketch the proof for two cases, assignments and the while loop.

For an assignment $x := a$, suppose $\models \{\varphi\} x := a \{\psi\}$. Then for all σ, if $\sigma \models \varphi$, then $\sigma \models \mathsf{wlp}\ (x := a)\ \psi$. But $\mathsf{wlp}\ (x := a)\ \psi = \psi\{a/x\}$, so for all σ, if $\sigma \models \varphi$, then $\sigma \models \psi\{a/x\}$, therefore $\models \varphi \Rightarrow \psi\{a/x\}$. We can thus assume $\vdash \varphi \Rightarrow \psi\{a/x\}$, since we are allowed to take true formulas of L as axioms. Then $\vdash \{\psi\{a/x\}\} x := a \{\psi\}$ by the assignment rule of Hoare logic, thus $\vdash \{\varphi\} x := a \{\psi\}$ by the weakening rule of Hoare logic.

Now for the while loop. Suppose $\models \{\varphi\} \mathsf{while}\ b\ \mathsf{do}\ c \{\psi\}$. Then for all σ, if $\sigma \models \varphi$, then $\sigma \models \mathsf{wlp}\ (\mathsf{while}\ b\ \mathsf{do}\ c)\ \psi$. Since L is expressive, $\mathsf{wlp}\ (\mathsf{while}\ b\ \mathsf{do}\ c)\ \psi$ is equivalent to a formula ρ of L, and $\models \varphi \Rightarrow \rho$. Since the programs

$$\mathsf{while}\ b\ \mathsf{do}\ c \qquad\qquad \mathsf{if}\ b\ \mathsf{then}\ (c\ ;\ \mathsf{while}\ b\ \mathsf{do}\ c)\ \mathsf{else\ skip}$$

are semantically equivalent, we have

$$\begin{aligned} \rho &\Leftrightarrow \mathsf{wlp}\ (\mathsf{while}\ b\ \mathsf{do}\ c)\ \psi \\ &\Leftrightarrow \mathsf{wlp}\ (\mathsf{if}\ b\ \mathsf{then}\ (c\ ;\ \mathsf{while}\ b\ \mathsf{do}\ c)\ \mathsf{else\ skip})\ \psi \\ &\Leftrightarrow (b \ \Rightarrow\ \mathsf{wlp}\ c\ (\mathsf{wlp}\ (\mathsf{while}\ b\ \mathsf{do}\ c)\ \psi)) \wedge (\neg b \ \Rightarrow\ \mathsf{wlp\ skip}\ \psi) \\ &\Leftrightarrow (b \ \Rightarrow\ \mathsf{wlp}\ c\ \rho) \wedge (\neg b \ \Rightarrow\ \psi), \end{aligned}$$

thus $\models \rho \wedge \neg b \Rightarrow \psi$ and $\models \rho \wedge b \Rightarrow \mathsf{wlp}\ c\ \rho$. The latter says exactly that $\models \{\rho \wedge b\} c \{\rho\}$. By the induction hypothesis, $\vdash \{\rho \wedge b\} c \{\rho\}$, and by the fact that we may assume all true formulas of L as axioms, $\vdash \varphi \Rightarrow \rho$ and $\vdash \rho \wedge \neg b \Rightarrow \psi$. Therefore

$$\begin{aligned} \vdash \{\rho \wedge b\} c \{\rho\} \ &\Rightarrow\ \vdash \{\rho\} \mathsf{while}\ b\ \mathsf{do}\ c \{\rho \wedge \neg b\} && \text{by the Hoare while rule} \\ &\Rightarrow\ \vdash \{\varphi\} \mathsf{while}\ b\ \mathsf{do}\ c \{\psi\} && \text{by weakening.} \end{aligned}$$

□