# 7 Completeness of the Hoare rules

In this chapter it is discussed what it means for the Hoare rules to be complete. Gödel's Incompleteness Theorem implies there is no complete proof system for establishing precisely the valid assertions. The Hoare rules inherit this incompleteness. However by separating incompleteness of the assertion language from incompleteness due to inadequacies in the axioms and rules for the programming language constructs, we can obtain relative completeness in the sense of Cook. The proof that the Hoare rules are relatively complete relies on the idea of weakest liberal precondition, and leads into a discussion of verification-condition generators.

## 7.1 Gödel's Incompleteness Theorem

Look again at the proof rules for partial correctness assertions, and in particular at the consequence rule. Knowing we have a rule instance of the consequence rule requires that we determine that certain assertions in **Assn** are valid. Ideally, of course, we would like a proof system of axioms and rules for assertions which enabled us to prove all the assertions of **Assn** which are valid, and none which are invalid. Naturally we would like the proof system to be *effective* in the sense that it is a routine matter to check that something proposed as a rule instance really is one. It should be routine in the sense that there is a computable method in the form of a program which, with input a real rule instance, returns a confirmation that it is, and returns no confirmation on inputs which are not rule instances, without necessarily even terminating. Lacking such a computable method we might well have a proof derivation without knowing it because it uses a step we cannot check is a rule instance. We cannot claim that the proof system of Hoare rules is effective because we do not have a computable method for checking instances of the consequence rule. Having such depends on having a computable method to check that assertions of **Assn** are valid. But here we meet an absolute limit. The great Austrian logician Kurt Gödel showed that it is logically impossible to have an effective proof system in which one can prove precisely the valid assertions of **Assn**. This remarkable result, called Gödel's Incompleteness Theorem[1] is not so hard to prove nowadays, if one goes about it via results from the theory of computability. Indeed a proof of the theorem, stated now, will be given in Section 7.3 based on some results from computability. Any gaps or shortcomings there can be made up for by consulting the Appendix on computability and undecidability based on the language of while programs, **IMP**.

[1] The Incompleteness Theorem is not to be confused with Gödel's Completeness Theorem which says that the proof system for predicate calculus generates precisely those assertions which are valid for *all* interpretations.

**Theorem 7.1** *Gödel's Incompleteness Theorem (1931):*
*There is no effective proof system for* **Assn** *such that the theorems coincide with the valid assertions of* **Assn**.

This theorem means we cannot have an effective proof system for partial correctness assertions. As $\models B$ iff $\models \{\mathbf{true}\}\mathbf{skip}\{B\}$, if we had an effective proof system for partial correctness it would reduce to an effective proof system for assertions in **Assn**, which is impossible by Gödel's Incompleteness Theorem. In fact we can show there is no effective proof system for partial correctness assertions more directly.

**Proposition 7.2** *There is no effective proof system for partial correctness assertions such that its theorems are precisely the valid partial correctness assertions.*

**Proof:** Observe that $\models \{\mathbf{true}\}c\{\mathbf{false}\}$ iff the command $c$ diverges on all states. If we had an effective proof system for partial correction assertions it would yield a computable method of confirming that a command $c$ diverges on all states. But this is known to be impossible—see Exercise A.13 of the Appendix. □

Faced with this unsurmountable fact, we settle for the proof system of Hoare rules in Section 6.4 even though we know it to be not effective because of the nature of the consequence rule; determining that we have an instance of the consequence rule is dependent on certain assertions being valid. Still, we can inquire as to the completeness of this system. That it is complete was established by S. Cook in [33]. If a partial correctness assertion is valid then there is a proof of it using the Hoare rules, *i.e.* for any partial correctness assertion $\{A\}c\{B\}$,

$$\models \{A\}c\{B\} \text{ implies } \vdash \{A\}c\{B\},$$

though the fact that it is a proof can rest on certain assertions in **Assn** being valid. It is as if in building proofs one could consult an oracle at any stage one needs to know if an assertion in **Assn** is valid. For this reason Cook's result is said to establish the *relative completeness* of the Hoare rules for partial correctness—their completeness is relative to being able to draw from the set of valid assertions about arithmetic. In this way one tries to separate concerns about programs and reasoning about them from concerns to do with arithmetic and the incompleteness of any proof system for it.

## 7.2 Weakest preconditions and expressiveness

The proof of relative completeness relies on another concept. Consider trying to prove

$$\{A\}c_0; c_1\{B\}.$$

In order to use the rule for composition one requires an intermediate assertion $C$ so that

$$\{A\}c_0\{C\} \text{ and } \{C\}c_1\{B\}$$

are provable. How do we know such an intermediate assertion $C$ can be found? A sufficient condition is that for every command $c$ and postconditions $B$ we can express their weakest precondition[2] in **Assn**.

Let $c \in \mathbf{Com}$ and $B \in \mathbf{Assn}$. Let $I$ be an interpretation. The weakest precondition $wp^I[\![c, B]\!]$ of $B$ with respect to $c$ in $I$ is defined by:

$$wp^I[\![c, B]\!] = \{\sigma \in \Sigma_\perp \mid C[\![c]\!]\sigma \models^I B\}.$$

It's all those states from which the execution of $c$ either diverges or ends up in a final state satisfying $B$. Thus if $\models^I \{A\}c\{B\}$ we know

$$A^I \subseteq wp^I[\![c, B]\!]$$

and *vice versa*. Thus $\models^I \{A\}c\{B\}$ iff $A^I \subseteq wp^I[\![c, B]\!]$.

Suppose there is an assertion $A_0$ such that in all interpretations $I$,

$$A_0^I = wp^I[\![c, B]\!].$$

Then

$$\models^I \{A\}c\{B\} \text{ iff } \models^I (A \Rightarrow A_0),$$

for any interpretation $I$ *i.e.*

$$\models \{A\}c\{B\} \text{ iff } \models (A \Rightarrow A_0).$$

So we see why it is called the weakest precondition, it is implied by any precondition which makes the partial correctness assertion valid. However it's not obvious that a particular language of assertions has an assertion $A_0$ such that $A_0^I = wp^I[\![c, B]\!]$.

**Definition:** Say **Assn** is *expressive* iff for every command $c$ and assertion $B$ there is an assertion $A_0$ such that $A_0^I = wp^I[\![c, B]\!]$ for any interpretation $I$.

In showing expressiveness we will use Gödel's $\beta$ predicate to encode facts about sequences of states as assertions in **Assn**. The $\beta$ predicate involves the operation $a$ **mod** $b$ which gives the remainder of $a$ when divided by $b$. We can express this notion as an assertion in **Assn**. For $x = a$ **mod** $b$ we write

[2]What we shall call weakest precondition is generally called weakest liberal precondition, the term weakest precondition referring to a related notion but for total correctness.

$$a \geq 0 \quad \wedge \quad b \geq 0 \quad \wedge$$
$$\exists k.[k \geq 0 \ \wedge \ k \times b \leq a \ \wedge \ (k+1) \times b > a \ \wedge \ x = a - (k \times b)].$$

**Lemma 7.3** *Let $\beta(a, b, i, x)$ be the predicate over natural numbers defined by*

$$\beta(a, b, i, x) \Leftrightarrow_{def} x = a \ \mathbf{mod}(1 + (1 + i) \times b).$$

*For any sequence $n_0, \ldots, n_k$ of natural numbers there are natural numbers $n, m$ such that for all $j$, $0 \leq j \leq k$, and all $x$ we have*

$$\beta(n, m, j, x) \Leftrightarrow x = n_j.$$

**Proof:** The proof of this arithmetical fact is left to the reader as a small series of exercises at the end of this section. □

The $\beta$ predicate is important because with it we can encode a sequence of $k$ natural numbers $n_0, \cdots, n_k$ as a pair $n, m$. Given $n, m$, for any length $k$, we can extract a sequence, *viz.*that sequence of numbers $n_0, \cdots, n_k$ such that

$$\beta(n, m, j, n_j)$$

for $0 \leq j \leq k$. Notice that the definition of $\beta$ shows that the list $n_0, \cdots, n_k$ is uniquely determined by the choice of $n, m$. The lemma above asserts that any sequence $n_0, \cdots, n_k$ can be encoded in this way.

We must now face a slight irritation. Our states and our language of assertions can involve negative as well as positive numbers. We are obliged to extend Gödel's $\beta$ predicate so as to encode sequences of positive and negative numbers. Fortunately, this is easily done by encoding positive numbers as the even and negative numbers as the odd natural numbers.

**Lemma 7.4** *Let $F(x, y)$ be the predicate over natural numbers $x$ and positive and negative numbers $y$ given by*

$$\begin{array}{lll} F(x,y) & \equiv & x \geq 0 \ \ \& \\ \exists z \geq 0. & & [(x = 2 \times z \Rightarrow y = z) \ \ \& \\ & & (x = 2 \times z + 1 \Rightarrow y = -z)] \end{array}$$

*Define*

$$\beta^{\pm}(n, m, j, y) \Leftrightarrow_{def} \exists x.(\beta(n, m, j, x) \wedge F(x, y)).$$

*Then for any sequence $n_0, \ldots, n_k$ of positive or negative numbers there are natural numbers $n, m$ such that for all $j$, $0 \leq j \leq k$, and all $x$ we have*

$$\beta^{\pm}(n, m, j, x) \Leftrightarrow x = n_j.$$

**Proof:** Clearly $F(n, m)$ expresses the 1-1 correspondence between natural numbers $m \in \omega$ and $n \in \mathbf{N}$ in which even $m$ stand for non-negative and odd $m$ for negative numbers. The lemma follows from Lemma 7.3. □

The predicate $\beta^{\pm}$ is expressible in **Assn** because $\beta$ and $F$ are. To avoid introducing a further symbol, let us write $\beta^{\pm}$ for the assertion in **Assn** expressing this predicate. This assertion in **Assn** will have free integer variables, say $n, m, j, x$, understood in the same way as above, *i.e.* $n, m$ encodes a sequence with $j$th element $x$. We will want to use other integer variables besides $n, m, j, x$, so we write $\beta^{\pm}(n', m', j', x')$ as an abbreviation for $\beta^{\pm}[n'/n, m'/m, j'/j, x'/x]$, got by substituting the the integer variable $n'$ for $n$, $m'$ for $m$, and so on. We have not give a formal definition of what it means to substitute integer variables in an assertion. The definition of substitution in Section 6.2.2 only defines substitutions $A[a/i]$ of arithmetic expressions $a$ without integer variables, for an integer variable $i$ in an assertion $A$. However, as long as the variables $n', m', j', x'$ are "fresh" in the sense of their being distinct and not occurring (free or bound) in $\beta^{\pm}$, the same definition applies equally well to the substitution of integer variables; the assertion $\beta^{\pm}[n'/n, m'/m, j'/j, x'/x]$ is that given by $\beta^{\pm}[n'/n][m'/m][j'/j][x'/x]$ using the definition of Section 6.2.2.[3]

Now we can show:

**Theorem 7.5 Assn** *is expressive.*

**Proof:** We show by structural induction on commands $c$ that for all assertions $B$ there is an assertion $w[\![c, B]\!]$ such that for all interpretations $I$

$$wp^I[\![c, B]\!] = w[\![c, B]\!]^I,$$

for all commands $c$.

Note that by the definition of weakest precondition that, for $I$ an interpretation, the equality $wp^I[\![c, B]\!] = w[\![c, B]\!]^I$ amounts to

$$\sigma \models^I w[\![c, B]\!] \text{ iff } \mathcal{C}[\![c]\!]\sigma \models^I B,$$

[3] To illustrate the technical problem with substitution of integer variables which are not fresh, consider the assertion $A \equiv (\exists i'.\ 2 \times i' = i)$ which means "$i$ is even." The naive definition of $A[i'/i]$ yields the assertion $(\exists i'.\ 2 \times i' = i')$ which happens to be valid, and so certainly does not mean "$i$ is even."

holding for all states $\sigma$, a fact we shall use occasionally in the proof.

$c \equiv \textbf{skip}$ : In this case, take $w[\![\textbf{skip}, B]\!] \equiv B$. Clearly, for all states $\sigma$ and interpretations $I$,

$$\begin{aligned}\sigma \in wp^I[\![\textbf{skip}, B]\!] \text{ iff } & \mathcal{C}[\![\textbf{skip}]\!]\sigma \models^I B \\ \text{iff } & \sigma \models^I B \\ \text{iff } & \sigma \models^I w[\![\textbf{skip}, B]\!].\end{aligned}$$

$c \equiv (X := a)$ : In this case, define $w[\![X := a, B]\!] \equiv B[a/X]$. Then

$$\begin{aligned}\sigma \in wp^I[\![X := a, B]\!] \text{ iff } & \sigma[\mathcal{A}[\![a]\!]\sigma/X] \models^I B \\ \text{iff } & \sigma \models^I B[a/X] \qquad \text{by Lemma 6.9} \\ \text{iff } & \sigma \models^I w[\![X := a, B]\!].\end{aligned}$$

$c \equiv c_0; c_1$ : Inductively define $w[\![c_0; c_1, B]\!] \equiv w[\![c_0, w[\![c_1, B]\!]]\!]$. Then, for $\sigma \in \Sigma$ and interpretation $I$,

$$\begin{aligned}\sigma \in wp^I[\![c_0; c_1, B]\!] \text{ iff } & \mathcal{C}[\![c_0; c_1]\!]\sigma \models^I B \\ \text{iff } & \mathcal{C}[\![c_1]\!](\mathcal{C}[\![c_0]\!]\sigma) \models^I B \\ \text{iff } & \mathcal{C}[\![c_0]\!]\sigma \models^I w[\![c_1, B]\!], \qquad \text{by induction,} \\ \text{iff } & \sigma \models^I w[\![c_0, w[\![c_1, B]\!]]\!], \qquad \text{by induction,} \\ \text{iff } & \sigma \models^I w[\![c_0; c_1, B]\!].\end{aligned}$$

$c \equiv \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1$ : Define

$$w[\![\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, B]\!] \equiv [(b \wedge w[\![c_0, B]\!]) \vee (\neg b \wedge w[\![c_1, B]\!])].$$

Then, for $\sigma \in \Sigma$ and interpretation $I$,

$$\begin{aligned}\sigma \in wp^I[\![c, B]\!] \text{ iff } & \mathcal{C}[\![c]\!]\sigma \models^I B \\ \text{iff } & ([\mathcal{B}[\![b]\!]\sigma = \textbf{true} \ \&\ \mathcal{C}[\![c_0]\!]\sigma \models^I B] \text{ or} \\ & \quad [\mathcal{B}[\![b]\!]\sigma = \textbf{false} \ \&\ \mathcal{C}[\![c_1]\!]\sigma \models^I B]) \\ \text{iff } & ([\sigma \models^I b \ \&\ \sigma \models^I w[\![c_0, B]\!]] \text{ or} \\ & \quad [\sigma \models^I \neg b \ \&\ \sigma \models^I w[\![c_1, B]\!]]), \quad \text{by induction,} \\ \text{iff } & \sigma \models^I [(b \wedge w[\![c_0, B]\!]) \vee (\neg b \wedge w[\![c_1, B]\!])] \\ \text{iff } & \sigma \models^I w[\![c, B]\!].\end{aligned}$$

$c \equiv$ **while** $b$ **do** $c_0$: This is the one difficult case. For a state $\sigma$ and interpretation $I$, we have (from Exercise 5.8) that $\sigma \in wp^I[\![c, B]\!]$ iff

$$\begin{array}{l} \forall k\, \forall \sigma_0, \ldots, \sigma_k \in \Sigma. \\ [\sigma = \sigma_0\ \& \\ \\ \forall i (0 \leq i < k).\ (\ \sigma_i \models^I b\ \& \\ \\ \qquad\qquad \mathcal{C}[\![c_0]\!]\sigma_i = \sigma_{i+1})] \\ \\ \Rightarrow (\sigma_k \models^I b \vee B). \end{array} \tag{1}$$

As it stands the mathematical characterisation of states $\sigma$ in $wp^I[\![c, B]\!]$ is not an assertion in **Assn**; in particular it refers directly to states $\sigma_0, \cdots, \sigma_k$. However we show how to replace it by an equivalent description which is. The first step is to replace all references to the states $\sigma_0, \ldots, \sigma_k$ by references to the values they contain at the locations mentioned in $c$ and $B$. Suppose $\bar{X} = X_1, \ldots, X_l$ are the locations mentioned in $c$ and $B$—the values at the remaining locations are irrelevant to the computation. We make use of the following fact:
Suppose $A$ is an assertion in **Assn** which mentions only locations from $\bar{X} = X_1, \ldots, X_l$. For a state $\sigma$, let $s_i = \sigma(X_i)$, for $1 \leq i \leq l$, and write $\bar{s} = s_1, \cdots, s_l$. Then

$$\sigma \models^I A \text{ iff } \models^I A[\bar{s}/\bar{X}] \tag{$*$}$$

for any interpretation $I$. The assertion $A[\bar{s}/\bar{X}]$ is that obtained by the simultaneous substitution of $\bar{s}$ for $\bar{X}$ in $A$. This fact can be proved by structural induction (Exercise!).

Using the fact $(*)$ we can convert (1) into an equivalent assertion about sequences. For $i \geq 0$, let $\bar{s_i}$ abbreviate $s_{i1}, \ldots, s_{il}$, a sequence in $\mathbf{N}$. We claim: $\sigma \in wp^I[\![c, B]\!]$ iff

$$\begin{array}{l} \forall k\, \forall \bar{s}_0, \ldots, \bar{s}_k \in \mathbf{N}. \\ [\sigma \models^I \bar{X} = \bar{s}_0\ \& \\ \\ \forall i\ (0 \leq i < k).\ (\models^I b[\bar{s}_i/\bar{X}]\ \& \\ \\ \qquad\qquad \models^I (w[\![c_0, \bar{X} = \bar{s}_{i+1}]\!] \wedge \neg w[\![c_0, \mathbf{false}]\!])[\bar{s}_i/\bar{X}]) \\ \\ \Rightarrow \models^I (b \vee B)[\bar{s}_k/\bar{X}] \end{array} \tag{2}$$

We have used $\bar{X} = \bar{s}_0$ to abbreviate $X_1 = s_{01} \wedge \ldots \wedge X_l = s_{0l}$.

To prove the claim we argue that (1) and (2) are equivalent. Parts of the argument are straightforward. For example, it follows directly from (*) that, assuming state $\sigma_i$ has values $\bar{s}_i$ at $\bar{X}$,

$$\sigma_i \models^I b \text{ iff } \models^I b[\bar{s}_i/\bar{X}],$$

for an interpretation $I$. The hard part hinges on showing that assuming $\sigma_i$ and $\sigma_{i+1}$ have values $\bar{s}_i$ and $\bar{s}_{i+1}$, respectively, at $\bar{X}$ and agree elsewhere, we have

$$\mathcal{C}[\![c_0]\!]\sigma_i = \sigma_{i+1} \text{ iff } \models^I (w[\![c_0, \bar{X} = \bar{s}_{i+1}]\!] \wedge \neg w[\![c_0, \mathbf{false}]\!])[\bar{s}_i/\bar{X}],$$

for an interpretation $I$. To see this we first observe that

$$\mathcal{C}[\![c_0]\!]\sigma_i = \sigma_{i+1} \text{ iff } \sigma_i \in wp^I[\![c_0, \bar{X} = \bar{s}_{i+1}]\!] \;\&\; \mathcal{C}[\![c_0]\!]\sigma_i \text{ is defined.}$$

(Why?) From the induction hypothesis we obtain

$$\begin{array}{lll} \sigma_i \in wp^I[\![c_0, \bar{X} = \bar{s}_{i+1}]\!] & \text{iff} & \sigma_i \models^I (w[\![c_0, \bar{X} = \bar{s}_{i+1}]\!], \text{ and} \\ \mathcal{C}[\![c_0]\!]\sigma_i \text{ is defined} & \text{iff} & \sigma_i \models^I \neg w[\![c_0, \mathbf{false}]\!] \end{array}$$

—recall that $\sigma_i \in wp^I[\![c_0, \mathbf{false}]\!]$ iff $c_0$ diverges on $\sigma_i$. Consequently,

$$\begin{array}{lll} \mathcal{C}[\![c_0]\!]\sigma_i = \sigma_{i+1} \text{ iff } & \sigma_i \models^I (w[\![c_0, \bar{X} = \bar{s}_{i+1}]\!] \wedge \neg w[\![c_0, \mathbf{false}]\!]) & \\ \text{iff } & \models^I (w[\![c_0, \bar{X} = \bar{s}_{i+1}]\!] \wedge \neg w[\![c_0, \mathbf{false}]\!])[\bar{s}_i/\bar{X}] & \text{by (*).} \end{array}$$

This covers the difficulties in showing (1) and (2) equivalent.

Finally, notice how (2) can be expressed in **Assn**, using the Gödel predicate $\beta^{\pm}$. For simplicity assume $l = 1$ with $\bar{X} = X$. Then we can rephrase (2) to get: $\sigma \in wp^I[\![c, B]\!]$ iff

$$\begin{array}{l} \sigma \models^I \forall k\, \forall m, n \geq 0. \\ \quad [\beta^{\pm}(n, m, 0, X) \wedge \\ \quad \forall i\, (0 \leq i < k).\, (\forall x.\, \beta^{\pm}(n, m, i, x) \Rightarrow b[x/X]) \wedge \\ \qquad \forall x, y.\, (\beta^{\pm}(n, m, i, x) \wedge \beta^{\pm}(n, m, i+1, y) \Rightarrow \\ \qquad\qquad (w[\![c_0, X = y]\!] \wedge \neg w[\![c_0, \mathbf{false}]\!])[x/X])] \\ \qquad \Rightarrow (\beta^{\pm}(n, m, k, x) \Rightarrow (b \vee B)[x/X]) \end{array}$$

This is the assertion we take as $w[\![c, B]\!]$ in this case. (In understanding this assertion compare it line-for-line with (2), bearing in mind that $\beta^{\pm}(n, m, i, x)$ means that $x$ is the

$i$th element of the sequence encoded by the pair $n, m$.) The form of the assertion in the general case, for arbitrary $l$, is similar, though more clumsy, and left to the reader.

This completes the proof by structural induction. □

As **Assn** is expressive for any command $c$ and assertion $B$ there is an assertion $w[\![c, B]\!]$ with the property that

$$w[\![c, B]\!]^I = wp^I[\![c, B]\!]$$

for any interpretation $I$. Of course, the assertion $w[\![c, B]\!]$ constructed in the proof of expressiveness above, is not the unique assertion with this property (Why not?). However suppose $A_0$ is another assertion such that $A_0^I = wp^I[\![c, B]\!]$ for all $I$. Then

$$\models (w[\![c, B]\!] \iff A_0).$$

So the assertion expressing a weakest precondition is unique to within logical equivalence. The useful key fact about such an assertion $w[\![c, B]\!]$ is that, from the definition of weakest precondition, it is characterised by:

$$\sigma \models^I w[\![c, B]\!] \text{ iff } \mathcal{C}[\![c]\!]\sigma \models^I B,$$

for all states $\sigma$ and interpretations $I$.

From the expressiveness of **Assn** we shall prove relative completeness. First an important lemma.

**Lemma 7.6** *For $c \in$ **Com** and $B \in$ **Assn**, let $w[\![c, B]\!]$ be an assertion expressing the weakest precondition i.e. $w[\![c, B]\!]^I = wp^I[\![c, B]\!]$ (the assertion $w[\![c, B]\!]$ need not be necessarily that constructed by Theorem 7.5 above). Then*

$$\vdash \{w[\![c, B]\!]\}c\{B\}.$$

**Proof:** Let $w[\![c, B]\!]$ be an assertion which expresses the weakest precondition of a command $c$ and postcondition $B$. We show by structural induction on $c$ that

$$\vdash \{w[\![c, B]\!]\}c\{B\} \quad \text{for all } B \in \mathbf{Assn},$$

for all commands $c$.
(In all but the last case, the proof overlaps with that of Theorem 7.5.)

$c \equiv \mathbf{skip}$ : In this case $\models w[\![\mathbf{skip}, B]\!] \iff B$, so $\vdash \{w[\![\mathbf{skip}, B]\!]\}\mathbf{skip}\{B\}$ by the consequence rule.

$c \equiv (X := a)$ : In this case

$$\begin{aligned}\sigma \in wp^I[\![c, B]\!] \text{ iff } & \sigma[\mathcal{A}[\![a]\!]\sigma/X] \models^I B \\ \text{iff } & \sigma \models^I B[a/X].\end{aligned}$$

Thus $\models (w[\![c, B]\!] \iff B[a/X])$. Hence by the rule for assignment with the consequence rule we see $\vdash \{w[\![c, B]\!]\}c\{B\}$ in this case.

$c \equiv c_0; c_1$ : In this case, for $\sigma \in \Sigma$ and interpretation $I$,

$$\begin{aligned}\sigma \models^I w[\![c_0; c_1, B]\!] \text{ iff } & \mathcal{C}[\![c_0; c_1]\!]\sigma \models^I B \\ \text{iff } & \mathcal{C}[\![c_1]\!](\mathcal{C}[\![c_0]\!]\sigma) \models^I B \\ \text{iff } & \mathcal{C}[\![c_0]\!]\sigma \models^I w[\![c_1, B]\!] \\ \text{iff } & \sigma \models^I w[\![c_0, w[\![c_1, B]\!]]\!].\end{aligned}$$

Thus $\models w[\![c_0; c_1, B]\!] \iff w[\![c_0, w[\![c_1, B]\!]]\!]$. By the induction hypothesis

$$\begin{aligned}&\vdash \{w[\![c_0, w[\![c_1, B]\!]]\!]\}c_0\{w[\![c_1, B]\!]\} \quad \text{and} \\ &\vdash \{w[\![c_1, B]\!]\}c_1\{B\}.\end{aligned}$$

Hence, by the rule for sequencing, we deduce

$$\vdash \{w[\![c_0, w[\![c_1, B]\!]]\!]\}c_0; c_1\{B\}$$

By the consequence rule we get

$$\vdash \{w[\![c_0; c_1, B]\!]\}c_0; c_1\{B\}.$$

$c \equiv \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1$ : In this case, for $\sigma \in \Sigma$ and interpretation $I$,

$$\begin{aligned}\sigma \models^I w[\![c, B]\!] \text{ iff } & \mathcal{C}[\![c]\!]\sigma \models^I B \\ \text{iff } & ([\mathcal{B}[\![b]\!]\sigma = \textbf{true} \ \&\ \mathcal{C}[\![c_0]\!]\sigma \models^I B] \text{ or} \\ & \quad [\mathcal{B}[\![b]\!]\sigma = \textbf{false} \ \&\ \mathcal{C}[\![c_1]\!]\sigma \models^I B]) \\ \text{iff } & ([\sigma \models^I b \ \&\ \sigma \models^I w[\![c_0, B]\!]] \text{ or} \\ & \quad [\sigma \models^I \neg b \ \&\ \sigma \models^I w[\![c_1, B]\!]]) \\ \text{iff } & \sigma \models^I [(b \wedge w[\![c_0, B]\!]) \vee (\neg b \wedge w[\![c_1, B]\!])].\end{aligned}$$

Hence

$$\models w[\![c, B]\!] \iff [(b \wedge w[\![c_0, B]\!]) \vee (\neg b \wedge w[\![c_1, B]\!])].$$

Now by the induction hypothesis

$$\vdash \{w[\![c_0, B]\!]\}c_0\{B\} \text{ and } \vdash \{w[\![c_1, B]\!]\}c_1\{B\}.$$

But

$$\models (w[\![c, B]\!] \wedge b) \iff w[\![c_0, B]\!] \quad \text{and}$$
$$\models (w[\![c, B]\!] \wedge \neg b) \iff w[\![c_1, B]\!].$$

So by the consequence rule

$$\vdash \{w[\![c, B]\!] \wedge b\}c_0\{B\} \text{ and } \vdash \{w[\![c, B]\!] \wedge \neg b\}c_1\{B\}.$$

By the rule for conditionals we obtain $\vdash \{w[\![c, B]\!]\}c\{B\}$ in this case.

Finally we consider the case:
$c \equiv$ **while** $b$ **do** $c_0$ : Take $A \equiv w[\![c, B]\!]$. We show

(1) $\models \{A \wedge b\}c_0\{A\}$,
(2) $\models (A \wedge \neg b) \Rightarrow B$.

Then, from (1), by the induction hypothesis we obtain $\vdash \{A \wedge b\}c_0\{A\}$, so that by the while-rule $\vdash \{A\}c\{A \wedge \neg b\}$. Continuing, by (2), using the consequence rule, we obtain $\vdash \{A\}c\{B\}$. Now we prove (1) and (2).

(1) Let $\sigma \models^I A \wedge b$, for an interpretation $I$. Then $\sigma \models^I w[\![c, B]\!]$ and $\sigma \models^I b$, *i.e.* $\mathcal{C}[\![c]\!]\sigma \models^I B$ and $\sigma \models^I b$. But $\mathcal{C}[\![c]\!]$ is defined so

$$\mathcal{C}[\![c]\!] = \mathcal{C}[\![\textbf{if } b \textbf{ then } c_0; c \textbf{ else skip}]\!],$$

which makes $\mathcal{C}[\![c_0; c]\!]\sigma \models^I B$, *i.e.* $\mathcal{C}[\![c]\!](\mathcal{C}[\![c_0]\!]\sigma) \models^I B$. Therefore $\mathcal{C}[\![c_0]\!]\sigma \models^I w[\![c, B]\!]$, *i.e.* $\mathcal{C}[\![c_0]\!]\sigma \models^I A$. Thus $\models \{A \wedge b\}c_0\{A\}$.

(2) Let $\sigma \models^I A \wedge \neg b$, for an interpretation $I$. Then $\mathcal{C}[\![c]\!]\sigma \models^I B$ and $\sigma \models^I \neg b$. Again note $\mathcal{C}[\![c]\!] = \mathcal{C}[\![\textbf{if } b \textbf{ then } c_0; c \textbf{ else skip}]\!]$, so $\mathcal{C}[\![c]\!]\sigma = \sigma$. Therefore $\sigma \models^I B$. It follows that $\models^I A \wedge \neg b \Rightarrow B$. Thus $\models A \wedge \neg b \Rightarrow B$, proving (2).

This completes all the cases. Hence, by structural induction, the lemma is proved. □

**Theorem 7.7** *The proof system for partial correctness is relatively complete, i.e. for any partial correctness assertion* $\{A\}c\{B\}$,

$$\vdash \{A\}c\{B\} \textit{ if } \models \{A\}c\{B\}.$$

**Proof:** Suppose $\models \{A\}c\{B\}$. Then by the above lemma $\vdash \{w[\![c, B]\!]\}c\{B\}$ where $w[\![c, B]\!]^I = wp^I[\![c, B]\!]$ for any interpretation $I$. Thus as $\models (A \Rightarrow w[\![c, B]\!])$, by the consequence rule, we obtain $\vdash \{A\}c\{B\}$. □

**Exercise 7.8** (The Gödel $\beta$ predicate)

(a) Let $n_0, \ldots, n_k$ be a sequence of natural numbers and let

$$m = (\max\{k, n_0, \ldots, n_k\})!$$

Show that the numbers

$$p_i = 1 + (1 + i) \times m, \text{ for } 0 \leq i \leq k$$

are coprime (*i.e.*, $\gcd(p_i, p_j) = 1$ for $i \neq j$) and that $n_i < p_i$.

(b) Further, define

$$c_i = p_0 \times \ldots \times p_k / p_i, \text{ for } 0 \leq i \leq k.$$

Show that for all $i$, $0 \leq i \leq k$, there is a unique $d_i$, $0 \leq d_i < p_i$, such that

$$(c_i \times d_i) \textbf{ mod } p_i = 1$$

(c) In addition, define

$$n = \sum_{i=0}^{k} c_i \times d_i \times n_i.$$

Show that

$$n_i = n \textbf{ mod } p_i$$

when $0 \leq i \leq k$.

(d) Finally prove lemma 3.

□

## 7.3 Proof of Gödel's Theorem

Gödel's Incompleteness Theorem amounts to the fact that the subset of valid assertions in **Assn** is not recursively enumerable (*i.e.*, there is no program which given assertions as input returns a confirmation precisely on the valid assertions—see the Appendix on computability for a precise definition and a more detailed treatment).

**Theorem 7.9** *The subset of assertions* $\{A \in \textbf{Assn} \mid \models A\}$ *is not recursively enumerable.*

**Proof:** Suppose on the contrary that the set $\{A \in \mathbf{Assn} \mid \models A\}$ is recursively enumerable. Then there is a computable method to confirm that an assertion is valid. This provides a computable method to confirm that a command $c$ diverges on the zero-state $\sigma_0$, in which each location $X$ has contents 0:
Construct the assertion $w[\![c, \mathbf{false}]\!]$ as in the proof of Theorem 7.5. Let $\vec{X}$ consist of all the locations mentioned in $w[\![c, \mathbf{false}]\!]$. Let $A$ be the assertion $w[\![c, \mathbf{false}]\!][\vec{0}/\vec{X}]$, obtained by replacing the locations by zeros. Then the divergence of $c$ on the zero-state can be confirmed by checking the validity of $A$, for which there is assumed to be a computable method.
But it is known that the commands $c$ which diverge on the zero-state do not form a recursively enumerable set—see Theorem A.12 in the Appendix. This contradiction shows $\{A \in \mathbf{Assn} \mid \models A\}$ to not be recursively enumerable. □

As a corollary we obtain Gödel's Incompleteness Theorem:

**Theorem 7.10** *(Theorem 7.1 restated) (Gödel's Incompleteness Theorem):*
*There is no effective proof system for* **Assn** *such that its theorems coincide with the valid assertions of* **Assn**.

**Proof:** Assume there were an effective proof system such that for an assertion $A$, we have $A$ is provable iff $A$ is valid. The proof system being effective implies that there is a computable method to confirm precisely when something is a proof. Searching through all proofs systematically till a proof of an assertion $A$ is found provides a computable method of confirming precisely when an assertion $A$ is valid. Thus there cannot be an effective proof system. □

Although we have stated Gödel's Theorem for assertions **Assn** the presence of locations plays no essential role in the results. Gödel's Theorem is generally stated for the smaller language of assertions without locations—the language of arithmetic. The fact that the valid assertions in this language do not form a recursively enumerable set means that the axiomatisation of arithmetic is never finished—there will always be some fact about arithmetic which remains unprovable. Nor can we hope to have a program which generates an infinite list of axioms and effective proof rules so that all valid assertions about arithmetic follow. If there were such a program there would be an effective proof system for arithmetical assertions, contradicting Gödel's Incompleteness Theorem.

Gödel's result had tremendous historical significance. Gödel did not have the concepts of computability available to him. Rather his result stimulated logicians to research different formulations of what it meant to be computable. The original proof worked by expressing the concept of provability of a formal system for assertions as an assertion

itself, and constructing an assertion which was valid iff it was not provable. It should be admitted that we have only considered Gödel's First Incompleteness Theorem; there is also a second which says that a formal system for arithmetic cannot be proved free of contradiction in the system itself. It was clear to Gödel that his proofs of incompleteness hinged on being able to express a certain set of functions on the natural numbers by assertions—the set has come to be called the *primitive recursive functions*. The realisation that a simple extension led to a stable notion of computable function took some years longer, culminating in the Church-Turing thesis. The incompleteness theorem devastated the programme set up by Hilbert. As a reaction to paradoxes like Russell's in mathematical foundations, Hilbert had advocated a study of the finitistic methods employed when reasoning within some formal system, hoping that this would lead to proofs of consistency and completeness of important proof systems, like one for arithmetic. Gödel's Theorem established an absolute limit on the power of finitistic reasoning.

## 7.4 Verification conditions

In principle, the fact that **Assn** is expressive provides a method to reduce the demonstration that a partial correctness assertion is valid to showing the validity of an assertion in **Assn**; the validity of a partial correctness assertion of the form $\{A\}c\{B\}$ is equivalent to the validity of the assertion $A \Rightarrow w[\![c, B]\!]$, from which the command has been eliminated. In this way, given a theorem prover for predicate calculus we might hope to derive a theorem prover for **IMP** programs. Unfortunately, the method we used to obtain $w[\![c, B]\!]$ was convoluted and inefficient, and definitely not practical.

However, useful automated tools for establishing the validity of partial correctness assertions can be obtained along similar lines once we allow a little human guidance. Let us annotate programs by assertions. Define the syntactic set of annotated commands by:

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

where $X$ is a location, $a$ an arithmetic expression, $b$ is a boolean expression, $c, c_0, c_1$ are annotated commands and $D$ is an assertion such that in $c_0; \{D\}c_1$, the annotated command $c_1$, is *not* an assignment. The idea is that an assertion at a point in an annotated command is true whenever flow of control reaches that point. Thus we only annotate a command of the form $c_0; c_1$ at the point where control shifts from $c_0$ to $c_1$. It is unnecessary to do this when $c_1$ is an assignment $X := a$ because in that case an annotation can be derived simply from a postcondition. An annotated while-loop

$$\textbf{while } b \textbf{ do } \{D\}c$$

contains an assertion $D$ which is intended to be an invariant.

An *annotated partial correctness assertion* has the form

$$\{A\}c\{B\}$$

where $c$ is an annotated command. Annotated commands are associated with ordinary commands, got by ignoring the annotations. It is sometimes convenient to treat annotated commands as their associated commands. In this spirit, we say an annotated partial correctness assertion is valid when its associated (unannotated) partial correctness assertion is.

An annotated while-loop

$$\{A\}\textbf{while } b \textbf{ do } \{D\}c\{B\}$$

contains an assertion $D$, which we hope has been chosen judiciously so $D$ is an invariant. Being an invariant means that

$$\{D \wedge b\}c\{D\}$$

is valid. In order to ensure

$$\{A\}\ \textbf{while } b \textbf{ do } \{D\}c\{B\}$$

is valid, once it is known that $D$ is an invariant, it suffices to show that both assertions

$$A \Rightarrow D, \quad D \wedge \neg b \Rightarrow B$$

are valid. A quick way to see this is to notice that we can derive $\{A\}\textbf{while } b \textbf{ do } c\{B\}$ from $\{D \wedge b\}c\{D\}$ using the Hoare rules which we know to be sound. As is clear, not all annotated partial correctness assertions are valid. To be so it is sufficient to establish the validity of certain assertions, called *verification conditions* for which all mention of commands is eliminated. Define the verification conditions (abbreviated to $vc$) of an annotated partial correctness assertion by structural induction on annotated commands:

$$\begin{aligned}
vc(\{A\}\textbf{skip}\{B\}) &= \{A \Rightarrow B\} \\
vc(\{A\}X := a\{B\}) &= \{A \Rightarrow B[a/X]\} \\
vc(\{A\}c_0; X := a\{B\}) &= vc(\{A\}c_0\{B[a/X]\}) \\
vc(\{A\}c_0; \{D\}c_1\{B\}) &= vc(\{A\}c_0\{D\}) \cup vc(\{D\}c_1\{B\}) \\
&\quad \text{where } c_1 \text{is not an assignment} \\
vc(\{A\}\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1\{B\}) &= vc(\{A \wedge b\}c_0\{B\}) \cup vc(\{A \wedge \neg b\}c_1\{B\}) \\
vc(\{A\}\textbf{while } b \textbf{ do } \{D\}c\{B\}) &= vc(\{D \wedge b\}c\{D\}) \cup \{A \Rightarrow D\} \\
&\qquad \cup \{D \wedge \neg b \Rightarrow B\}
\end{aligned}$$

**Exercise 7.11** Prove by structural induction on annotated commands that for all annotated partial correctness assertions $\{A\}c\{B\}$ if all assertions in $vc(\{A\}c\{B\})$ are valid then $\{A\}c\{B\}$ is valid. (The proof follows the general line of Lemma 7.6. A proof can be found in [42], Section 3.5.) □

Thus to show the validity of an annotated partial correctness assertion it is sufficient to show its verification conditions are valid. In this way the task of program verification can be passed to a theorem prover for predicate calculus. Some commercial program-verification systems, like Gypsy [41], work in this way.

Note, that while the validity of its verification conditions is sufficient to guarantee the validity of an annotated partial correctness assertion, it is not necessary. This can occur because the invariant chosen is inappropriate for the pre and post conditions. For example, although

$$\{\mathbf{true}\}\mathbf{while\ false\ do}\ \{\mathbf{false}\}\mathbf{skip}\{\mathbf{true}\}$$

is certainly valid with **false** as an invariant, its verification conditions contain

$$\mathbf{true} \Rightarrow \mathbf{false},$$

which is certainly not a valid assertion.

We conclude this section by pointing out a peculiarity in our treatment of annotated commands. Two commands, built up as $(c; X := a_1); X := a_2$ and $c; (X := a_1; X := a_2)$, are understood in essentially the same way; indeed in many imperative languages they would both be written as:

$$\begin{aligned} &c;\\ &X := a_1;\\ &X := a_2 \end{aligned}$$

However the two commands support different annotations according to our syntax of annotated commands. The first would only allow possible annotations to appear in $c$ whereas the second would be annotated as $c; \{D\}(X := a_1; X := a_2)$. The rules for annotations do not put annotations before a single assignment but would put an annotation in before any other chain of assignments. This is even though it is still easily possible to derive the annotation from the postcondition, this time through a series of substitutions.

**Exercise 7.12** Suggest a way to modify the syntax of annotated commands and the definition of their verification conditions to address this peculiarity, so that any chain of assignments or **skip** is treated in the same way as a single assignment is presently. □

**Exercise 7.13** A larger project is to program a verification-condition generator (*e.g.*in standard ML or prolog) which, given an annotated partial correctness assertion as input, outputs a set, or list, of its verification conditions. (See Gordon's book [42] for a program in lisp.) □

## 7.5 Predicate transformers

This section is optional and presents an abstract, rather more mathematical view of assertions and weakest preconditions. Abstractly a command is a function $f : \Sigma \to \Sigma_\perp$ from states to states together with an element $\perp$, standing for undefined; such functions are sometimes called *state transformers*. They form a cpo, isomorphic to that of the partial functions on states, when ordered pointwise. Abstractly, an assertion for partial correctness is a subset of states which contains $\perp$, so we define the set of *partial correctness predicates* to be

$$\mathrm{Pred}(\Sigma) = \{Q \mid Q \subseteq \Sigma_\perp \ \& \ \perp \in Q\}.$$

We can make predicates into a cpo by ordering them by reverse inclusion. The cpo of predicates for partial correctness is

$$(\mathrm{Pred}(\Sigma), \supseteq).$$

Here, more information about the final state delivered by a command configuration corresponds to having bounded it to lie within a smaller set provided its execution halts. In particular the very least information corresponds to the element $\perp_{Pred} = \Sigma \cup \{\perp\}$. We shall use simply $\mathrm{Pred}(\Sigma)$ for the cpo of partial-correctness predicates.

The weakest precondition construction determines a continuous function on the cpo of predicates—a predicate transformer.[4]

**Definition:** Let $f : \Sigma \to \Sigma_\perp$ be a partial function on states. Define

$$\begin{aligned} &Wf : \mathrm{Pred}(\Sigma) \to \mathrm{Pred}(\Sigma); \\ &(Wf)(Q) = (f^{-1}Q) \cup \{\perp\} \\ &\textit{i.e.}, (Wf)(Q) = \{\sigma \in \Sigma_\perp \mid f(\sigma) \in Q\} \cup \{\perp\}. \end{aligned}$$

A command $c$ can be taken to denote a state transformer $\mathcal{C}[\![c]\!] : \Sigma \to \Sigma_\perp$ with the convention that undefined is represented by $\perp$. Let $B$ be an assertion. According to this understanding, with respect to an interpretation $I$,

$$(W(\mathcal{C}[\![c]\!]))(B^I) = wp^I[\![c, B]\!].$$

[4]This term is generally used for the corresponding notion when considering total correctness.

**Exercise 7.14** Write $ST$ for the cpo of state transformers $[\Sigma_\perp \to_\perp \Sigma_\perp]$ and $PT$ for the cpo of predicate transformers $[\mathrm{Pred}(\Sigma) \to \mathrm{Pred}(\Sigma)]$.
Show $W : ST \to_\perp PT$ and $W$ is continuous (Care! there are lots of things to check here).
Show $W(Id_{\Sigma_\perp}) = Id_{Pred(\Sigma)}$ *i.e.*, $W$ takes the identity function on the cpo of states to the identity function on predicates $\mathrm{Pred}(\Sigma)$.
Show $W(f \circ g) = (Wg) \circ (Wf)$. □

In the context of total correctness Dijkstra has argued that one can specify the meaning of a command as a predicate transformer [36]. He argued that to understand a command amounts to knowing the weakest precondition which ensures a given postcondition. We do this for partial correctness. As we now have a cpo of predicates we also have the cpo

$$[\mathrm{Pred}(\Sigma) \to \mathrm{Pred}(\Sigma)]$$

of predicate transformers. Thus we can give a denotational semantics of commands in **IMP** as predicate transformers, instead of as state transformers. We can define a semantic function

$$\mathcal{P}t : \mathbf{Com} \to [\mathrm{Pred}(\Sigma) \to \mathrm{Pred}(\Sigma)]$$

from commands to predicate transformers. Although this denotational semantics, in which the denotation of a command is a predicate transformer is clearly a different denotational semantics to that using partial functions, if done correctly it should be equivalent in the sense that two commands denote the same predicate transformer iff they denote the same partial function. You may like to do this as the exercise below.

**Exercise 7.15** (Denotations as predicate transformers)
Define a semantic function

$$\mathcal{P}t : \mathbf{Com} \to PT$$

by

$$\mathcal{P}t[\![X := a]\!]Q = \{\sigma \in \Sigma_\perp \mid \sigma[\mathcal{A}[\![a]\!]\sigma/X] \in Q\}$$
$$\mathcal{P}t[\![\mathbf{skip}]\!]Q = Q$$
$$\mathcal{P}t[\![c_0; c_1]\!]Q = \mathcal{P}t[\![c_0]\!](\mathcal{P}t[\![c_1]\!]Q)$$
$$\mathcal{P}t[\![\mathbf{if}\ b\ \mathbf{then}\ c_0\ \mathbf{else}\ c_1]\!]Q = \mathcal{P}t[\![c_0]\!](\overline{b} \cap Q) \cup \mathcal{P}t[\![c_1]\!](\overline{\neg b} \cap Q)$$
where $\overline{b} = \{\sigma \mid \sigma = \perp \text{ or } \mathcal{B}[\![b]\!]\sigma = \mathbf{true}\}$ for any boolean $b$
$$\mathcal{P}t[\![\mathbf{while}\ b\ \mathbf{do}\ c]\!]Q = \mathit{fix}(G)$$

where $G : PT \to PT$ is given by $G(p)(Q) = (\overline{b} \cap \mathcal{P}t[\![c_0]\!](p(Q)) \cup (\overline{\neg b} \cap Q)$.
Show $G$ is continuous.

Show $W(\mathcal{C}[\![c]\!]_\perp) = Pt[\![c]\!]$ for any command $c$. Observe

$$Wf = Wf' \Rightarrow f = f'$$

for two strict continuous functions $f, f'$ on $\Sigma_\perp$. Deduce

$$\mathcal{C}[\![c]\!] = \mathcal{C}[\![c']\!] \text{ iff } Pt[\![c]\!] = Pt[\![c']\!]$$

for any commands $c, c'$.

Recall the ordering on predicates. Because it is reverse inclusion:

$$fix(G) = \bigcap_{n\in\omega} G^n(\perp_{Pred}).$$

This suggests that if we were to allow infinite conjunctions in our language of assertions, and did not have quantifiers, we could express weakest preconditions directly. Indeed this is so, and you might like to extend **Bexp** by infinite conjunctions, to form another set of assertions to replace **Assn**, and modify the above semantics to give an assertion, of the new kind, which expresses the weakest precondition for each command. Once we have expressiveness a proof of relative completeness follows for this new kind of assertion, in the same way as earlier in Section 7.2. □

## 7.6 Further reading

The book "What is mathematical logic?" by Crossley *et al* [34] has an excellent explanation of Gödel's Incompleteness Theorem, though with the details missing. The logic texts by Kleene [54], Mendelson [61] and Enderton [38] have full treatments. A treatment aimed at Computer Science students is presented in the book [11] by Kfoury, Moll and Arbib. Cook's original proof of relative completeness in [33] used "strongest postconditions" instead of weakest preconditions; the latter are used instead by Clarke in [23] and his earlier work. The paper by Clarke has, in addition, some negative results showing the impossibility of having sound and relatively complete proof systems for programming languages richer than the one here. Apt's paper [8] provides good orientation. Alternative presentations of the material of this chapter can be found in [58], [13]. Gordon's book [42] contains a more elementary and detailed treatment of verification conditions.