Chapter 6

Axiomatic Program Verification

The kinds of semantics we have seen so far specify the meaning of programs although they may also be used to prove that given programs possess certain properties. We may distinguish between several classes of properties: *partial correctness properties* are properties expressing that *if* a given program terminates *then* there will be a certain relationship between the initial and the final values of the variables. Thus a partial correctness property of a program need *not* ensure that it terminates. This is contrary to *total correctness properties* which express that the program *will* terminate *and* that there will be a certain relationship between the initial and the final values of the variables. Thus we have

partial correctness + termination = total correctness

Yet another class of properties is concerned with the *resources* used when executing the program. An example is the *time* used to execute the program on a particular machine.

6.1 Direct proofs of program correctness

In this section we shall give some examples that prove partial correctness of statements based directly on the operational and denotational semantics. We shall prove that the factorial statement

$\texttt{y} := \texttt{1};\ \texttt{while}\ \neg(\texttt{x}{=}\texttt{1})\ \texttt{do}\ (\texttt{y} := \texttt{y}{\star}\texttt{x};\ \texttt{x} := \texttt{x}{-}\texttt{1})$

is partially correct, that is *if* the statement terminates *then* the final value of `y` will be the factorial of the initial value of `x`.

Natural semantics

Using *natural semantics* the partial correctness of the factorial statement can be formalized as follows:

For all states s and s', if

$$\langle \mathtt{y} := 1;\ \mathtt{while}\ \neg(\mathtt{x}{=}1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1),\ s\rangle \to s'$$

then $s'\ \mathtt{y} = (s\ \mathtt{x})!$ and $s\ \mathtt{x} > \mathbf{0}$

This is indeed a partial correctness property because the statement does not terminate if the initial value $s\ \mathtt{x}$ of $\mathtt{x}$ is non-positive.

The proof proceeds in three stages:

Stage 1: We prove that the body of the `while` loop satisfies:

$$\begin{array}{l}\text{if } \langle \mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1,\ s\rangle \to s'' \text{ and } s''\ \mathtt{x} > \mathbf{0}\\ \text{then } (s\ \mathtt{y}) \star (s\ \mathtt{x})! = (s''\ \mathtt{y}) \star (s''\ \mathtt{x})! \text{ and } s\ \mathtt{x} > \mathbf{0}\end{array} \qquad (*)$$

Stage 2: We prove that the `while` loop satisfies:

$$\begin{array}{l}\text{if } \langle \mathtt{while}\ \neg(\mathtt{x}{=}1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1),\ s\rangle \to s''\\ \text{then } (s\ \mathtt{y}) \star (s\ \mathtt{x})! = s''\ \mathtt{y} \text{ and } s''\ \mathtt{x} = \mathbf{1} \text{ and } s\ \mathtt{x} > \mathbf{0}\end{array} \qquad (**)$$

Stage 3: We prove the partial correctness property for the complete program:

$$\begin{array}{l}\text{if } \langle \mathtt{y} := 1;\ \mathtt{while}\ \neg(\mathtt{x}{=}1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1),\ s\rangle \to s'\\ \text{then } s'\ \mathtt{y} = (s\ \mathtt{x})! \text{ and } s\ \mathtt{x} > \mathbf{0}\end{array} \qquad (***)$$

In each of the three stages the derivation tree of the given transition is inspected in order to prove the property.

In the *first stage* we consider the transition

$$\langle \mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1,\ s\rangle \to s''$$

According to $[\mathrm{comp}_{\mathrm{ns}}]$ there will be transitions

$$\langle \mathtt{y} := \mathtt{y}{\star}\mathtt{x},\ s\rangle \to s' \text{ and } \langle \mathtt{x} := \mathtt{x}{-}1,\ s'\rangle \to s''$$

for some s'. From the axiom $[\mathrm{ass}_{\mathrm{ns}}]$ we then get that $s' = s[\mathtt{y}{\mapsto}\mathcal{A}[\![\mathtt{y}{\star}\mathtt{x}]\!]s]$ and that $s'' = s'[\mathtt{x}{\mapsto}\mathcal{A}[\![\mathtt{x}{-}1]\!]s']$. Combining these results we have

$$s'' = s[\mathtt{y}{\mapsto}(s\ \mathtt{y}){\star}(s\ \mathtt{x})][\mathtt{x}{\mapsto}(s\ \mathtt{x}){-}1]$$

Assuming that $s''\ \mathtt{x} > \mathbf{0}$ we can then calculate

$$(s''\ \mathtt{y}) \star (s''\ \mathtt{x})! = ((s\ \mathtt{y}) \star (s\ \mathtt{x})) \star ((s\ \mathtt{x}){-}1)! = (s\ \mathtt{y}) \star (s\ \mathtt{x})!$$

and since $s\ \mathtt{x} = (s''\ \mathtt{x}) + \mathbf{1}$ this shows that (*) does indeed hold.

In the *second stage* we proceed by induction on the shape of the derivation tree for

$$\langle \mathtt{while}\ \neg(\mathtt{x}{=}1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1),\ s\rangle \to s'$$

One of two axioms and rules could have been used to construct this derivation. If $[\text{while}^{\text{ff}}_{\text{ns}}]$ has been used then $s' = s$ and $\mathcal{B}[\![\neg(\mathtt{x}{=}1)]\!]s = \mathbf{ff}$. This means that $s'\ \mathtt{x} = \mathbf{1}$ and since $\mathbf{1}! = \mathbf{1}$ we get the required $(s\ \mathtt{y}) \star (s\ \mathtt{x})! = s\ \mathtt{y}$ and $s\ \mathtt{x} > \mathbf{0}$. This proves (**).

Next assume that $[\text{while}^{\text{tt}}_{\text{ns}}]$ is used to construct the derivation. Then it must be the case that $\mathcal{B}[\![\neg(\mathtt{x}{=}1)]\!]s = \mathbf{tt}$ and

$$\langle \mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1,\ s\rangle \rightarrow s''$$

and

$$\langle \mathtt{while}\ \neg(\mathtt{x}{=}1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1),\ s''\rangle \rightarrow s'$$

for some state s''. The induction hypothesis applied to the latter derivation gives that

$$(s''\ \mathtt{y}) \star (s''\ \mathtt{x})! = s'\ \mathtt{y} \text{ and } s'\ \mathtt{x} = \mathbf{1} \text{ and } s''\ \mathtt{x} > \mathbf{0}$$

From (*) we get that

$$(s\ \mathtt{y}) \star (s\ \mathtt{x})! = (s''\ \mathtt{y}) \star (s''\ \mathtt{x})! \text{ and } s\ \mathtt{x} > \mathbf{0}$$

Putting these results together we get

$$(s\ \mathtt{y}) \star (s\ \mathtt{x})! = s'\ \mathtt{y} \text{ and } s'\ \mathtt{x} = \mathbf{1} \text{ and } s\ \mathtt{x} > \mathbf{0}$$

This proves (**) and thereby the second stage of the proof is completed.

Finally, consider the *third stage* of the proof and the derivation

$$\langle \mathtt{y} := 1;\ \mathtt{while}\ \neg(\mathtt{x}{=}1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1),\ s\rangle \rightarrow s'$$

According to $[\text{comp}_{\text{ns}}]$ there will be a state s'' such that

$$\langle \mathtt{y} := \mathbf{1},\ s\rangle \rightarrow s''$$

and

$$\langle \mathtt{while}\ \neg(\mathtt{x}{=}1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1),\ s''\rangle \rightarrow s'$$

From axiom $[\text{ass}_{\text{ns}}]$ we see that $s'' = s[\mathtt{y}{\mapsto}\mathbf{1}]$ and from (**) we get that $s''\ \mathtt{x} > \mathbf{0}$ and therefore $s\ \mathtt{x} > \mathbf{0}$. Hence $(s\ \mathtt{x})! = (s''\ \mathtt{y}) \star (s''\ \mathtt{x})!$ holds and using (**) we get

$$(s\ \mathtt{x})! = (s''\ \mathtt{y}) \star (s''\ \mathtt{x})! = s'\ \mathtt{y}$$

as required. This proves the partial correctness of the factorial statement.

Exercise 6.1 Use the natural semantics to prove the partial correctness of the statement

$\mathtt{z} := 0$; **while** $\mathtt{y} \leq \mathtt{x}$ **do** $(\mathtt{z} := \mathtt{z}+1;\ \mathtt{x} := \mathtt{x}-\mathtt{y})$

that is prove that *if* the statement terminates in s' when executed from a state s with $s\ \mathtt{x} > \mathbf{0}$ and $s\ \mathtt{y} > \mathbf{0}$, *then* $s'\ \mathtt{z} = (s\ \mathtt{x})$ **div** $(s\ \mathtt{y})$ and $s'\ \mathtt{x} = (s\ \mathtt{x})$ **mod** $(s\ \mathtt{y})$ where **div** is integer division and **mod** is the modulo operation. □

Exercise 6.2 Use the natural semantics to prove the following *total correctness* property for the factorial program: for all states s

if $s\ \mathtt{x} > \mathbf{0}$ then there exists a state s' such that

$$\langle \mathtt{y} := 1;\ \mathtt{while}\ \neg(\mathtt{x}=1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}\star\mathtt{x};\ \mathtt{x} := \mathtt{x}-1),\ s\rangle \to s'$$

and $s'\ \mathtt{y} = (s\ \mathtt{x})!$ □

Structural operational semantics

The partial correctness of the factorial statement can also be established using the *structural operational semantics*. The property is then reformulated as:

For all states s and s', if

$$\langle \mathtt{y} := 1;\ \mathtt{while}\ \neg(\mathtt{x}=1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}\star\mathtt{x};\ \mathtt{x} := \mathtt{x}-1),\ s\rangle \Rightarrow^* s'$$

then $s'\ \mathtt{y} = (s\ \mathtt{x})!$ and $s\ \mathtt{x} > \mathbf{0}$

Again it is worthwhile to approach the proof in stages:

Stage 1: We prove by induction on the length of derivation sequences that

if $\langle \mathtt{while}\ \neg(\mathtt{x}=1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}\star\mathtt{x};\ \mathtt{x} := \mathtt{x}-1),\ s\rangle \Rightarrow^k s'$

then $s'\ \mathtt{y} = (s\ \mathtt{y}) \star (s\ \mathtt{x})!$ and $s'\ \mathtt{x} = \mathbf{1}$ and $s\ \mathtt{x} > \mathbf{0}$

Stage 2: We prove that

if $\langle \mathtt{y} := 1;\ \mathtt{while}\ \neg(\mathtt{x}=1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}\star\mathtt{x};\ \mathtt{x} := \mathtt{x}-1),\ s\rangle \Rightarrow^* s'$

then $s'\ \mathtt{y} = (s\ \mathtt{x})!$ and $s\ \mathtt{x} > \mathbf{0}$

Exercise 6.3 Complete the proof of stages 1 and 2. □

Denotational semantics

We shall now use the denotational semantics to prove partial correctness properties of statements. The idea is to formulate the property as a *predicate* ψ on the ccpo $(\mathbf{State} \hookrightarrow \mathbf{State}, \sqsubseteq)$, that is

$$\psi\colon (\mathbf{State} \hookrightarrow \mathbf{State}) \to \mathbf{T}$$

As an example, the partial correctness of the factorial statement will be written as

$$\psi_{fac}(\mathcal{S}_{\mathrm{ds}}[\![\mathtt{y := 1;\ while\ \neg(x{=}1)\ do\ (y := y{\star}x;\ x := x{-}1)}]\!]) = \mathbf{tt}$$

where the predicate ψ_{fac} is defined by

$$\psi_{fac}(g) = \mathbf{tt}$$

if and only if

for all states s and s', if $g\ s = s'$ then $s'\ \mathtt{y} = (s\ \mathtt{x})!$ and $s\ \mathtt{x} > \mathbf{0}$

A predicate ψ: $D \to \mathbf{T}$ defined on a ccpo $(D,\sqsubseteq)$ is called an *admissible predicate* if and only if we have

$$\text{if } \psi\ d = \mathbf{tt} \text{ for all } d \in Y \text{ then } \psi(\bigsqcup Y) = \mathbf{tt}$$

for every chain Y in D. Thus if ψ holds on all the elements of the chain then it also holds on the least upper bound of the chain.

Example 6.4 Consider the predicate ψ'_{fac} defined on **State** $\hookrightarrow$ **State** by

$$\psi'_{fac}(g) = \mathbf{tt}$$

if and only if

for all states s and s', if $g\ s = s'$

then $s'\ \mathtt{y} = (s\ \mathtt{y}) \star (s\ \mathtt{x})!$ and $s\ \mathtt{x} > \mathbf{0}$

Then ψ'_{fac} is an admissible predicate. To see this assume that Y is a chain in **State** $\hookrightarrow$ **State** and assume that $\psi'_{fac}\ g = \mathbf{tt}$ for all $g \in Y$. We shall then prove that $\psi'_{fac}(\bigsqcup Y) = \mathbf{tt}$, that is

$$(\bigsqcup Y)\ s = s'$$

implies

$$s'\ \mathtt{y} = (s\ \mathtt{y}) \star (s\ \mathtt{x})! \text{ and } s\ \mathtt{x} > \mathbf{0}$$

From Lemma 4.25 we have $\mathrm{graph}(\bigsqcup Y) = \bigcup\{\ \mathrm{graph}(g) \mid g \in Y\ \}$. We have assumed that $(\bigsqcup Y)\ s = s'$ so Y cannot be empty and $\langle s,\ s'\rangle \in \mathrm{graph}(g)$ for some $g \in Y$. But then

$$s'\ \mathtt{y} = (s\ \mathtt{y}) \star (s\ \mathtt{x})! \text{ and } s\ \mathtt{x} > \mathbf{0}$$

as $\psi'_{fac}\ g = \mathbf{tt}$ for all $g \in Y$. This proves that ψ'_{fac} is an admissible predicate. □

For admissible predicates we have the following induction principle called *fixed point induction*:

Theorem 6.5 Let $(D,\sqsubseteq)$ be a ccpo and let $f\colon D \to D$ be a continuous function and let ψ be an admissible predicate on D. If for all $d \in D$

$$\psi\ d = \mathbf{tt} \text{ implies } \psi(f\ d) = \mathbf{tt}$$

then $\psi(\text{FIX } f) = \mathbf{tt}$.

Proof: We shall first note that

$$\psi\ \bot = \mathbf{tt}$$

holds by admissibility of ψ (applied to the chain $Y = \emptyset$). By induction on n we can then show that

$$\psi(f^{\mathrm{n}}\ \bot) = \mathbf{tt}$$

using the assumptions of the theorem. By admissibility of ψ (applied to the chain $Y = \{\ f^{\mathrm{n}}\ \bot \mid \mathrm{n} \geq 0\ \}$) we then have

$$\psi(\text{FIX } f) = \mathbf{tt}$$

This completes the proof. □

We are now in a position where we can prove the partial correctness of the factorial statement. The first observation is that

$$\mathcal{S}_{\mathrm{ds}}[\![\mathtt{y := 1;\ while\ \neg(x=1)\ do\ (y := y{\star}x;\ x := x{-}1)}]\!]s = s'$$

if and only if

$$\mathcal{S}_{\mathrm{ds}}[\![\mathtt{while\ \neg(x=1)\ do\ (y := y{\star}x;\ x := x{-}1)}]\!](s[\mathtt{y}{\mapsto}\mathbf{1}]) = s'$$

Thus it is sufficient to prove that

$$\psi'_{fac}(\mathcal{S}_{\mathrm{ds}}[\![\mathtt{while\ \neg(x=1)\ do\ (y := y{\star}x;\ x := x{-}1)}]\!]) = \mathbf{tt} \qquad (*)$$

(where ψ'_{fac} is defined in Example 6.4) as this will imply that

$$\psi_{fac}(\mathcal{S}_{\mathrm{ds}}[\![\mathtt{y := 1;\ while\ \neg(x=1)\ do\ (y := y{\star}x;\ x := x{-}1)}]\!]) = \mathbf{tt}$$

We shall now reformulate (*) slightly to bring ourselves in a position where we can use fixed point induction. Using the definition of $\mathcal{S}_{\mathrm{ds}}$ in Table 4.1 we have

$$\mathcal{S}_{\mathrm{ds}}[\![\mathtt{while\ \neg(x=1)\ do\ (y := y{\star}x;\ x := x{-}1)}]\!] = \text{FIX } F$$

where the functional F is defined by

$$F\ g = \text{cond}(\mathcal{B}[\![\mathtt{\neg(x=1)}]\!],\ g \circ \mathcal{S}_{\mathrm{ds}}[\![\mathtt{y := y{\star}x;\ x := x{-}1}]\!],\ \text{id})$$

Using the semantic equations defining $\mathcal{S}_{\text{ds}}$ we can rewrite this definition as

$$(F\ g)\ s = \begin{cases} s & \text{if } s\ x = \mathbf{1} \\ g(s[\mathtt{y} \mapsto (s\ \mathtt{y}) \star (s\ \mathtt{x})][\mathtt{x} \mapsto (s\ \mathtt{x}) - 1]) & \text{otherwise} \end{cases}$$

We have already seen that F is a continuous function (for example in the proof of Proposition 4.47) and from Example 6.4 we have that ψ'_{fac} is an admissible predicate. Thus we see from Theorem 6.5 that (*) follows if we show that

$$\psi'_{fac}\ g = \mathbf{tt} \text{ implies } \psi'_{fac}(F\ g) = \mathbf{tt}$$

To prove this implication assume that $\psi'_{fac}\ g = \mathbf{tt}$, that is for all states s and s'

$$\text{if } g\ s = s' \text{ then } s'\ \mathtt{y} = (s\ \mathtt{y}) \star (s\ \mathtt{x})! \text{ and } s\ \mathtt{x} > \mathbf{0}$$

We shall prove that $\psi'_{fac}(F\ g) = \mathbf{tt}$, that is for all states s and s'

$$\text{if } (F\ g)\ s = s' \text{ then } s'\ \mathtt{y} = (s\ \mathtt{y}) \star (s\ \mathtt{x})! \text{ and } s\ \mathtt{x} > \mathbf{0}$$

Inspecting the definition of F we see that there are two cases. First assume that $s\ \mathtt{x} = \mathbf{1}$. Then $(F\ g)\ s = s$ and clearly $s\ \mathtt{y} = (s\ \mathtt{y}) \star (s\ \mathtt{x})!$ and $s\ \mathtt{x} > \mathbf{0}$. Next assume that $s\ \mathtt{x} \neq \mathbf{1}$. Then

$$(F\ g)\ s = g(s[\mathtt{y} \mapsto (s\ \mathtt{y}) \star (s\ \mathtt{x})][\mathtt{x} \mapsto (s\ \mathtt{x}) - \mathbf{1}])$$

From the assumptions about g we then get that

$$s'\ \mathtt{y} = ((s\ \mathtt{y}) \star (s\ \mathtt{x})) \star ((s\ \mathtt{x}) - \mathbf{1})! \text{ and } (s\ \mathtt{x}) - \mathbf{1} > \mathbf{0}$$

so that the desired result

$$s'\ \mathtt{y} = (s\ \mathtt{y}) \star (s\ \mathtt{x})! \text{ and } s\ \mathtt{x} > \mathbf{0}$$

follows.

Exercise 6.6 Repeat Exercise 6.1 using the denotational semantics. □

6.2 Partial correctness assertions

One may argue that the above proofs are too detailed to be practically useful; the reason is that they are too closely connected with the semantics of the programming language. One may therefore want to capture the *essential properties* of the various constructs so that it would be less demanding to conduct proofs about given programs. Of course the choice of "essential properties" will determine the sort of properties that we may accomplish proving. In this section we shall be interested in partial correctness properties and therefore the "essential properties" of the various constructs will not include termination.

The idea is to specify properties of programs as *assertions*, or claims, about them. An assertion is a triple of the form

$$\{\ P\ \}\ S\ \{\ Q\ \}$$

where S is a statement and P and Q are predicates. Here P is called the *precondition* and Q is called the *postcondition*. Intuitively, the meaning of $\{\ P\ \}\ S\ \{\ Q\ \}$ is that

if P holds in the initial state, and

if the execution of S terminates when started in that state,

then Q will hold in the state in which S halts

Note that for $\{\ P\ \}\ S\ \{\ Q\ \}$ to hold we do *not* require that S halts when started in states satisfying P — merely that *if* it does halt *then* Q holds in the final state.

Logical variables

As an example we may write

$$\{\ \mathtt{x}{=}\mathtt{n}\ \}\ \mathtt{y} := 1;\ \mathtt{while}\ \neg(\mathtt{x}{=}1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{x}{\star}\mathtt{y};\ \mathtt{x} := \mathtt{x}{-}1)\ \{\ \mathtt{y}{=}\mathtt{n}! \wedge \mathtt{n}{>}0\ \}$$

to express that if the value of `x` is equal to the value of `n` *before* the factorial program is executed then the value of `y` will be equal to the factorial of the value of `n` *after* the execution of the program has terminated (if indeed it terminates). Here `n` is a special variable called a *logical* variable and these logical variables must not appear in any statement considered. The role of these variables is to "remember" the initial values of the program variables. Note that if we replace the postcondition $\mathtt{y}{=}\mathtt{n}! \wedge \mathtt{n}{>}0$ by the new postcondition $\mathtt{y}{=}\mathtt{x}! \wedge \mathtt{x}{>}0$ then the assertion above will express a relationship between the final value of `y` and the final value of `x` and this is not what we want. The use of logical variables solves the problem because it allows us to refer to initial values of variables.

We shall thus distinguish between two kinds of variables:

- program variables, and
- logical variables.

The states will determine the values of both kinds of variables and since logical variables do not occur in programs their values will always be the same. In case of the factorial program we know that the value of `n` is the same in the initial state and in the final state. The precondition $\mathtt{x} = \mathtt{n}$ expresses that `n` has the same value as `x` in the initial state. Since the program will not change the value of `n` the postcondition $\mathtt{y} = \mathtt{n}!$ will express that the final value of `y` is equal to the factorial of the initial value of `x`.

The assertion language

There are two approaches concerning how to specify the preconditions and postconditions of the assertions:

- the intensional approach, versus
- the extensional approach.

In the *intensional approach* the idea is to introduce an explicit language called an *assertion language* and then the conditions will be formulae of that language. This assertion language is in general much more powerful than the boolean expressions, **Bexp**, introduced in Chapter 1. In fact the assertion language has to be very powerful indeed in order to be able to express all the preconditions and postconditions we may be interested in; we shall return to this in the next section. The approach we shall follow is the *extensional approach* and it is a kind of shortcut. The idea is that the conditions are predicates, that is functions in **State** $\to$ **T**. Thus the meaning of $\{\ P\ \}\ S\ \{\ Q\ \}$ may be reformulated as saying that if P holds on a state s and if S executed from state s results in the state s' then Q holds on s'. We can write any predicates we like and therefore the expressiveness problem mentioned above does not arise.

Each boolean expression b defines a predicate $\mathcal{B}[\![b]\!]$. We shall feel free to let b include logical variables as well as program variables so the precondition `x = n` used above is an example of a boolean expression. To ease the readability, we introduce the following notation

$P_1 \wedge P_2$	for	P where $P\ s = (P_1\ s)$ and $(P_2\ s)$
$P_1 \vee P_2$	for	P where $P\ s = (P_1\ s)$ or $(P_2\ s)$
$\neg P$	for	P' where $P'\ s = \neg(P\ s)$
$P[x \mapsto \mathcal{A}[\![a]\!]]$	for	P' where $P'\ s = P\ (s[x \mapsto \mathcal{A}[\![a]\!]s])$
$P_1 \Rightarrow P_2$	for	$\forall s \in$ **State**: $P_1\ s$ implies $P_2\ s$

When it is convenient, but not when defining formal inference rules, we shall allow to dispense with $\mathcal{B}[\![\cdots]\!]$ and $\mathcal{A}[\![\cdots]\!]$ inside square brackets as well as within preconditions and postconditions.

Exercise 6.7 Show that

- $\mathcal{B}[\![b[x \mapsto a]]\!] = \mathcal{B}[\![b]\!][x \mapsto \mathcal{A}[\![a]\!]]$ for all b and a,
- $\mathcal{B}[\![b_1 \wedge b_2]\!] = \mathcal{B}[\![b_1]\!] \wedge \mathcal{B}[\![b_2]\!]$ for all b_1 and b_2, and
- $\mathcal{B}[\![\neg b]\!] = \neg\mathcal{B}[\![b]\!]$ for all b. □

$[\text{ass}_\text{p}]$	$\{\ P[x\mapsto\mathcal{A}[\![a]\!]]\ \}\ x := a\ \{\ P\ \}$
$[\text{skip}_\text{p}]$	$\{\ P\ \}\ \texttt{skip}\ \{\ P\ \}$
$[\text{comp}_\text{p}]$	$\dfrac{\{\ P\ \}\ S_1\ \{\ Q\ \},\quad \{\ Q\ \}\ S_2\ \{\ R\ \}}{\{\ P\ \}\ S_1; S_2\ \{\ R\ \}}$
$[\text{if}_\text{p}]$	$\dfrac{\{\ \mathcal{B}[\![b]\!]\wedge P\ \}\ S_1\ \{\ Q\ \},\quad \{\ \neg\mathcal{B}[\![b]\!]\wedge P\ \}\ S_2\ \{\ Q\ \}}{\{\ P\ \}\ \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \{\ Q\ \}}$
$[\text{while}_\text{p}]$	$\dfrac{\{\ \mathcal{B}[\![b]\!]\wedge P\ \}\ S\ \{\ P\ \}}{\{\ P\ \}\ \texttt{while}\ b\ \texttt{do}\ S\ \{\ \neg\mathcal{B}[\![b]\!]\wedge P\ \}}$
$[\text{cons}_\text{p}]$	$\dfrac{\{\ P'\ \}\ S\ \{\ Q'\ \}}{\{\ P\ \}\ S\ \{\ Q\ \}}$ if $P\Rightarrow P'$ and $Q'\Rightarrow Q$

Table 6.1: Axiomatic system for partial correctness

The inference system

The partial correctness assertions will be specified by an inference system consisting of a set of axioms and rules. The formulae of the inference system have the form

$$\{\ P\ \}\ S\ \{\ Q\ \}$$

where S is a statement in the language **While** and P and Q are predicates. The axioms and rules are summarized in Table 6.1 and will be explained below. The inference system specifies an *axiomatic semantics* for **While**.

The axiom for assignment statements is

$$\{\ P[x\mapsto\mathcal{A}[\![a]\!]]\ \}\ x := a\ \{\ P\ \}$$

This axiom assumes that the execution of $x := a$ starts in a state s that satisfies $P[x\mapsto\mathcal{A}[\![a]\!]]$, that is in a state s where $s[x\mapsto\mathcal{A}[\![a]\!]s]$ satisfies P. The axiom expresses that if the execution of $x := a$ terminates (which will always be the case) then the final state will satisfy P. From the earlier definitions of the semantics of **While** we know that the final state will be $s[x\mapsto\mathcal{A}[\![a]\!]s]$ so it is easy to see that the axiom is plausible.

For `skip` the axiom is

$$\{\ P\ \}\ \texttt{skip}\ \{\ P\ \}$$

Thus if P holds before `skip` is executed then it also holds afterwards. This is clearly plausible as `skip` does nothing.

Axioms [ass_p] and [skip_p] are really *axiom schemes* generating separate axioms for each choice of predicate P. The meaning of the remaining constructs are given by rules of inference rather than axiom schemes. Each such rule specifies a way of deducing an assertion about a compound construct from assertions about its constituents. For composition the rule is:

$$\frac{\{\ P\ \}\ S_1\ \{\ Q\ \},\quad \{\ Q\ \}\ S_2\ \{\ R\ \}}{\{\ P\ \}\ S_1;\ S_2\ \{\ R\ \}}$$

This says that if P holds prior to the execution of $S_1;\ S_2$ and if the execution terminates then we can conclude that R holds in the final state provided that there is a predicate Q for which we can deduce that

- if S_1 is executed from a state where P holds and if it terminates then Q will hold for the final state, and that
- if S_2 is executed from a state where Q holds and if it terminates then R will hold for the final state.

The rule for the conditional is

$$\frac{\{\ \mathcal{B}[\![b]\!] \wedge P\ \}\ S_1\ \{\ Q\ \},\quad \{\ \neg\mathcal{B}[\![b]\!] \wedge P\ \}\ S_2\ \{\ Q\ \}}{\{\ P\ \}\ \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \{\ Q\ \}}$$

The rule says that if `if` b `then` S_1 `else` S_2 is executed from a state where P holds and if it terminates, then Q will hold for the final state provided that we can deduce that

- if S_1 is executed from a state where P and b hold and if it terminates then Q holds on the final state, and that
- if S_2 is executed from a state where P and $\neg b$ hold and if it terminates then Q holds on the final state.

The rule for the iterative statement is

$$\frac{\{\ \mathcal{B}[\![b]\!] \wedge P\ \}\ S\ \{\ P\ \}}{\{\ P\ \}\ \texttt{while}\ b\ \texttt{do}\ S\ \{\ \neg\mathcal{B}[\![b]\!] \wedge P\ \}}$$

The predicate P is called an *invariant* for the `while`-loop and the idea is that it will hold *before* and *after* each execution of the body S of the loop. The rule says that if additionally b is true before each execution of the body of the loop then $\neg b$ will be true when the execution of the `while`-loop has terminated.

To complete the inference system we need one more rule of inference

$$\frac{\{\ P'\ \}\ S\ \{\ Q'\ \}}{\{\ P\ \}\ S\ \{\ Q\ \}}\quad \text{if } P \Rightarrow P' \text{ and } Q' \Rightarrow Q$$

This rule says that we can strengthen the precondition P' and weaken the post-condition Q'. This rule is often called the *rule of consequence*.

Note that Table 6.1 specifies a set of axioms and rules just as the tables defining the operational semantics in Chapter 2. The analogue of a derivation tree will now be called an *inference tree* since it shows how to infer that a certain property holds. Thus the leaves of an inference tree will be instances of axioms and the internal nodes will correspond to instances of rules. We shall say that the inference tree gives a *proof* of the property expressed by its root. We shall write

$$\vdash_{\mathrm{p}} \{\ P\ \}\ S\ \{\ Q\ \}$$

for the provability of the assertion $\{\ P\ \}\ S\ \{\ Q\ \}$. An inference tree is called *simple* if it is an instance of one of the axioms and otherwise it is called *composite*.

Example 6.8 Consider the statement `while true do skip`. From $[\mathrm{skip}_{\mathrm{p}}]$ we have (omitting the $\mathcal{B}[\![\cdots]\!]$)

$$\vdash_{\mathrm{p}} \{\ \mathtt{true}\ \}\ \mathtt{skip}\ \{\ \mathtt{true}\ \}$$

Since $(\mathtt{true} \wedge \mathtt{true}) \Rightarrow \mathtt{true}$ we can apply the rule of consequence $[\mathrm{cons}_{\mathrm{p}}]$ and get

$$\vdash_{\mathrm{p}} \{\ \mathtt{true} \wedge \mathtt{true}\ \}\ \mathtt{skip}\ \{\ \mathtt{true}\ \}$$

Hence by the rule $[\mathrm{while}_{\mathrm{p}}]$ we get

$$\vdash_{\mathrm{p}} \{\ \mathtt{true}\ \}\ \mathtt{while\ true\ do\ skip}\ \{\ \neg\mathtt{true} \wedge \mathtt{true}\ \}$$

We have that $\neg\mathtt{true} \wedge \mathtt{true} \Rightarrow \mathtt{true}$ so by applying $[\mathrm{cons}_{\mathrm{p}}]$ once more we get

$$\vdash_{\mathrm{p}} \{\ \mathtt{true}\ \}\ \mathtt{while\ true\ do\ skip}\ \{\ \mathtt{true}\ \}$$

The inference above can be summarized by the following inference tree:

$$\dfrac{\dfrac{\dfrac{}{\{\ \mathtt{true}\ \}\ \mathtt{skip}\ \{\ \mathtt{true}\ \}}}{\{\ \mathtt{true} \wedge \mathtt{true}\ \}\ \mathtt{skip}\ \{\ \mathtt{true}\ \}}}{\{\ \mathtt{true}\ \}\ \mathtt{while\ true\ do\ skip}\ \{\ \neg\mathtt{true} \wedge \mathtt{true}\ \}}$$

$$\{\ \mathtt{true}\ \}\ \mathtt{while\ true\ do\ skip}\ \{\ \mathtt{true}\ \}$$

It is now easy to see that we cannot claim that $\{\ P\ \}\ S\ \{\ Q\ \}$ means that S will terminate in a state satisfying Q when it is started in a state satisfying P. For the assertion $\{\ \mathtt{true}\ \}\ \mathtt{while\ true\ do\ skip}\ \{\ \mathtt{true}\ \}$ this reading would mean that the program would always terminate and clearly this is not the case. □

Example 6.9 To illustrate the use of the axiomatic semantics for verification we shall prove the assertion

$$\{\ \mathtt{x} = \mathtt{n}\ \}$$
$$\mathtt{y} := \mathtt{1};\ \mathtt{while}\ \neg(\mathtt{x}{=}\mathtt{1})\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}\mathtt{1})$$
$$\{\ \mathtt{y} = \mathtt{n}! \wedge \mathtt{n} > \mathtt{0}\ \}$$

where, for the sake of readability, we write $\mathtt{y} = \mathtt{n}! \wedge \mathtt{n} > \mathtt{0}$ for the predicate

$$P \text{ where } P\ s = (s\ \mathtt{y} = (s\ \mathtt{n})! \wedge s\ \mathtt{n} > \mathbf{0})$$

The inference of this assertion proceeds in a number of stages. First we define the predicate INV that is going to be the invariant of the `while`-loop:

$$INV\ s = (s\ \mathtt{x} > \mathbf{0} \text{ implies } ((s\ \mathtt{y}) \star (s\ \mathtt{x})! = (s\ \mathtt{n})! \text{ and } s\ \mathtt{n} \geq s\ \mathtt{x}))$$

We shall then consider the body of the loop. Using $[\mathrm{ass}_\mathrm{p}]$ we get

$$\vdash_\mathrm{p} \{\ INV[\mathtt{x}{\mapsto}\mathtt{x}{-}\mathtt{1}]\ \}\ \mathtt{x} := \mathtt{x}{-}\mathtt{1}\ \{\ INV\ \}$$

Similarly, we get

$$\vdash_\mathrm{p} \{\ (INV[\mathtt{x}{\mapsto}\mathtt{x}{-}\mathtt{1}])[\mathtt{y}{\mapsto}\mathtt{y}{\star}\mathtt{x}]\ \}\ \mathtt{y} := \mathtt{y} \star \mathtt{x}\ \{\ INV[\mathtt{x}{\mapsto}\mathtt{x}{-}\mathtt{1}]\ \}$$

We can now apply the rule $[\mathrm{comp}_\mathrm{p}]$ to the two assertions above and get

$$\vdash_\mathrm{p} \{\ (INV[\mathtt{x}{\mapsto}\mathtt{x}{-}\mathtt{1}])[\mathtt{y}{\mapsto}\mathtt{y}{\star}\mathtt{x}]\ \}\ \mathtt{y} := \mathtt{y} \star \mathtt{x};\ \mathtt{x} := \mathtt{x}{-}\mathtt{1}\ \{\ INV\ \}$$

It is easy to verify that

$$(\neg(\mathtt{x}{=}\mathtt{1}) \wedge INV) \Rightarrow (INV[\mathtt{x}{\mapsto}\mathtt{x}{-}\mathtt{1}])[\mathtt{y}{\mapsto}\mathtt{y}{\star}\mathtt{x}]$$

so using the rule $[\mathrm{cons}_\mathrm{p}]$ we get

$$\vdash_\mathrm{p} \{\ \neg(\mathtt{x} = \mathtt{1}) \wedge INV\ \}\ \mathtt{y} := \mathtt{y} \star \mathtt{x};\ \mathtt{x} := \mathtt{x}{-}\mathtt{1}\ \{\ INV\ \}$$

We are now in a position to use the rule $[\mathrm{while}_\mathrm{p}]$ and get

$$\vdash_\mathrm{p} \{\ INV\ \}$$
$$\quad \mathtt{while}\ \neg(\mathtt{x}{=}\mathtt{1})\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}\mathtt{1})$$
$$\quad \{\neg(\neg(\mathtt{x} = \mathtt{1})) \wedge INV\ \}$$

Clearly we have

$$\neg(\neg(\mathtt{x} = \mathtt{1})) \wedge INV \Rightarrow \mathtt{y} = \mathtt{n}! \wedge \mathtt{n} > \mathtt{0}$$

so applying rule [cons$_p$] we get

$$\vdash_p \{\ INV\ \}\ \texttt{while}\ \neg(\texttt{x=1})\ \texttt{do}\ (\texttt{y} := \texttt{y}\star\texttt{x};\ \texttt{x} := \texttt{x}-1)\ \{\ \texttt{y} = \texttt{n}!\ \wedge\ \texttt{n} > 0\ \}$$

We shall now apply the axiom [ass$_p$] to the statement $\texttt{y} := 1$ and get

$$\vdash_p \{\ INV[\texttt{y}\mapsto 1]\ \}\ \texttt{y} := 1\ \{\ INV\ \}$$

Using that

$$\texttt{x} = \texttt{n} \Rightarrow INV[\texttt{y}\mapsto 1]$$

together with [cons$_p$] we get

$$\vdash_p \{\ \texttt{x} = \texttt{n}\ \}\ \texttt{y} := 1\ \{\ INV\ \}$$

Finally, we can use the rule [comp$_p$] and get

$$\begin{array}{l}\vdash_p \{\ \texttt{x} = \texttt{n}\ \}\\ \quad \texttt{y} := 1;\ \texttt{while}\ \neg(\texttt{x=1})\ \texttt{do}\ (\texttt{y} := \texttt{y}\star\texttt{x};\ \texttt{x} := \texttt{x}-1)\\ \quad \{\ \texttt{y} = \texttt{n}!\ \wedge\ \texttt{n} > 0\ \}\end{array}$$

as required. □

Exercise 6.10 Specify a formula expressing the partial correctness property of the program of Exercise 6.1. Construct an inference tree giving a proof of this property using the inference system of Table 6.1. □

Exercise 6.11 Suggest an inference rule for `repeat` S `until` b. You are not allowed to rely on the existence of a `while`-construct in the language. □

Exercise 6.12 Suggest an inference rule for `for` $x := a_1$ `to` a_2 `do` S. You are not allowed to rely on the existence of a `while`-construct in the language. □

Properties of the semantics

In the operational and denotational semantics we defined a notion of two programs being semantically equivalent. We can define a similar notion for the axiomatic semantics: Two programs S_1 and S_2 are *provably equivalent* according to the axiomatic semantics of Table 6.1 if for all preconditions P and postconditions Q we have

$$\vdash_p \{\ P\ \}\ S_1\ \{\ Q\ \} \quad \text{if and only if} \quad \vdash_p \{\ P\ \}\ S_2\ \{\ Q\ \}$$

Exercise 6.13 Show that the following statements of **While** are provably equivalent in the above sense:

- S; `skip` and S
- S_1; (S_2; S_3) and (S_1; S_2); S_3 □

Proofs of properties of the axiomatic semantics will often proceed by *induction on the shape of the inference tree*:

Induction on the Shape of Inference Trees	
1:	Prove that the property holds for all the simple inference trees by showing that it holds for the *axioms* of the inference system.
2:	Prove that the property holds for all composite inference trees: For each *rule* assume that the property holds for its premises (this is called the *induction hypothesis*) and that the conditions of the rule are satisfied and then prove that it also holds for the conclusion of the rule.

Exercise 6.14 ** Using the inference rule for `repeat` S `until` b given in Exercise 6.11 show that `repeat` S `until` b is provably equivalent to S; `while` $\neg b$ `do` S. Hint: it is not too hard to show that what is provable about `repeat` S `until` b is also provable about S; `while` $\neg b$ `do` S. □

Exercise 6.15 Show that $\vdash_{\mathrm{p}} \{\ P\ \}\ S\ \{\ \mathtt{true}\ \}$ for all statements S and properties P. □

6.3 Soundness and completeness

We shall now address the relationship between the inference system of Table 6.1 and the operational and denotational semantics of the previous chapters. We shall prove that

- the inference system is *sound*: if some partial correctness property can be proved using the inference system then it does indeed hold according to the semantics, and
- the inference system is *complete*: if some partial correctness property does hold according to the semantics then we can also find a proof for it using the inference system.

The completeness result can only be proved because we use the extensional approach where preconditions and postconditions are arbitrary predicates. In the intensional approach we only have a weaker result; we shall return to this later in this section.

As the operational and denotational semantics are equivalent we only need to consider one of them here and we shall choose the natural semantics. The partial correctness assertion $\{ P \} S \{ Q \}$ is said to be *valid* if and only if

for all states s, if $P\ s = \mathbf{tt}$ and $\langle S,s\rangle \to s'$ for some s' then $Q\ s' = \mathbf{tt}$

and we shall write this as

$$\models_{\mathrm{p}} \{ P \} S \{ Q \}$$

The soundness property is then expressed by

$$\vdash_{\mathrm{p}} \{ P \} S \{ Q \} \text{ implies } \models_{\mathrm{p}} \{ P \} S \{ Q \}$$

and the completeness property is expressed by

$$\models_{\mathrm{p}} \{ P \} S \{ Q \} \text{ implies } \vdash_{\mathrm{p}} \{ P \} S \{ Q \}$$

We have

Theorem 6.16 For all partial correctness assertions $\{ P \} S \{ Q \}$ we have

$$\models_{\mathrm{p}} \{ P \} S \{ Q \} \text{ if and only if } \vdash_{\mathrm{p}} \{ P \} S \{ Q \}$$

It is customary to prove the soundness and completeness results separately.

Soundness

We shall first prove:

Lemma 6.17 The inference system of Table 6.1 is sound, that is for every partial correctness formula $\{ P \} S \{ Q \}$ we have

$$\vdash_{\mathrm{p}} \{ P \} S \{ Q \} \text{ implies } \models_{\mathrm{p}} \{ P \} S \{ Q \}$$

Proof: The proof is by induction on the shape of the inference tree used to infer $\vdash_{\mathrm{p}} \{ P \} S \{ Q \}$. This amounts to nothing but a formalization of the intuitions we gave when introducing the axioms and rules.

The case [$\mathrm{ass_p}$]: We shall prove that the axiom is valid, so suppose that

$$\langle x := a,\ s\rangle \to s'$$

and $(P[x \mapsto \mathcal{A}[\![a]\!]])\ s = \mathbf{tt}$. We shall then prove that $P\ s' = \mathbf{tt}$. From $[\text{ass}_{\text{ns}}]$ we get that $s' = s[x \mapsto \mathcal{A}[\![a]\!]s]$ and from $(P[x \mapsto \mathcal{A}[\![a]\!]])\ s = \mathbf{tt}$ we get that $P\ (s[x \mapsto \mathcal{A}[\![a]\!]s])$ $= \mathbf{tt}$. Thus $P\ s' = \mathbf{tt}$ as was to be shown.

The case $[\text{skip}_{\text{p}}]$: This case is immediate using the clause $[\text{skip}_{\text{ns}}]$.

The case $[\text{comp}_{\text{p}}]$: We assume that

$$\models_{\text{p}} \{\ P\ \}\ S_1\ \{\ Q\ \} \text{ and } \models_{\text{p}} \{\ Q\ \}\ S_2\ \{\ R\ \}$$

and we have to prove that $\models_{\text{p}} \{\ P\ \}\ S_1; S_2\ \{\ R\ \}$. So consider arbitrary states s and s'' such that $P\ s = \mathbf{tt}$ and

$$\langle S_1;S_2,\ s\rangle \to s''$$

From $[\text{comp}_{\text{ns}}]$ we get that there is a state s' such that

$$\langle S_1,\ s\rangle \to s' \quad \text{and} \quad \langle S_2,\ s'\rangle \to s''$$

From $\langle S_1,\ s\rangle \to s'$, $P\ s = \mathbf{tt}$ and $\models_{\text{p}} \{\ P\ \}\ S_1\ \{\ Q\ \}$ we get $Q\ s' = \mathbf{tt}$. From $\langle S_2,\ s'\rangle \to s''$, $Q\ s' = \mathbf{tt}$ and $\models_{\text{p}} \{\ Q\ \}\ S_2\ \{\ R\ \}$ it follows that $R\ s'' = \mathbf{tt}$ as was to be shown.

The case $[\text{if}_{\text{p}}]$: Assume that

$$\models_{\text{p}} \{\ \mathcal{B}[\![b]\!] \wedge P\ \}\ S_1\ \{\ Q\ \} \text{ and } \models_{\text{p}} \{\ \neg\mathcal{B}[\![b]\!] \wedge P\ \}\ S_2\ \{\ Q\ \}$$

To prove $\models_{\text{p}} \{\ P\ \}\ \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \{\ Q\ \}$ consider arbitrary states s and s' such that $P\ s = \mathbf{tt}$ and

$$\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \to s'$$

There are two cases. If $\mathcal{B}[\![b]\!]s = \mathbf{tt}$ then we get $(\mathcal{B}[\![b]\!] \wedge P)\ s = \mathbf{tt}$ and from $[\text{if}_{\text{ns}}]$ we have

$$\langle S_1,\ s\rangle \to s'$$

From the first assumption we therefore get $Q\ s' = \mathbf{tt}$. If $\mathcal{B}[\![b]\!]s = \mathbf{ff}$ the result follows in a similar way from the second assumption.

The case $[\text{while}_{\text{p}}]$: Assume that

$$\models_{\text{p}} \{\ \mathcal{B}[\![b]\!] \wedge P\ \}\ S\ \{\ P\ \}$$

To prove $\models_{\text{p}} \{\ P\ \}\ \texttt{while}\ b\ \texttt{do}\ S\ \{\ \neg\mathcal{B}[\![b]\!] \wedge P\ \}$ consider arbitrary states s and s'' such that $P\ s = \mathbf{tt}$ and

$$\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \to s''$$

and we shall show that $(\neg\mathcal{B}[\![b]\!]\wedge P)\ s'' = \mathbf{tt}$. We shall now proceed by induction on the shape of the derivation tree in the natural semantics. One of two cases apply. If $\mathcal{B}[\![b]\!]s = \mathbf{ff}$ then $s'' = s$ according to $[\text{while}^{\text{ff}}_{\text{ns}}]$ and clearly $(\neg\mathcal{B}[\![b]\!] \wedge P)\ s'' = \mathbf{tt}$ as required. Next consider the case where $\mathcal{B}[\![b]\!]s = \mathbf{tt}$ and

$$\langle S,\ s\rangle \to s' \quad \text{and} \quad \langle \texttt{while}\ b\ \texttt{do}\ S,\ s'\rangle \to s''$$

for some state s'. Thus $(\mathcal{B}[\![b]\!] \wedge P)\ s = \mathbf{tt}$ and we can then apply the assumption $\models_{\text{p}} \{\ \mathcal{B}[\![b]\!] \wedge P\ \}\ S\ \{\ P\ \}$ and get that $P\ s' = \mathbf{tt}$. The induction hypothesis can now be applied to the derivation $\langle \texttt{while}\ b\ \texttt{do}\ S,\ s'\rangle \to s''$ and gives that $(\neg\mathcal{B}[\![b]\!] \wedge P)\ s'' = \mathbf{tt}$. This completes the proof of this case.

The case $[\text{cons}_{\text{p}}]$: Suppose that

$$\models_{\text{p}} \{\ P'\ \}\ S\ \{\ Q'\ \} \text{ and } P \Rightarrow P' \text{ and } Q' \Rightarrow Q$$

To prove $\models_{\text{p}} \{\ P\ \}\ S\ \{\ Q\ \}$ consider states s and s' such that $P\ s = \mathbf{tt}$ and

$$\langle S,\ s\rangle \to s'$$

Since $P\ s = \mathbf{tt}$ and $P \Rightarrow P'$ we also have $P'\ s = \mathbf{tt}$ and the assumption then gives us that $Q'\ s' = \mathbf{tt}$. From $Q' \Rightarrow Q$ we therefore get $Q\ s' = \mathbf{tt}$ as required. □

Exercise 6.18 Show that the inference rule for `repeat` S `until` b suggested in Exercise 6.11 preserves validity. Argue that this means that the entire proof system consisting of the axioms and rules of Table 6.1 together with the rule of Exercise 6.11 is sound. □

Exercise 6.19 Define $\models' \{\ P\ \}\ S\ \{\ Q\ \}$ to mean that

> for all states s such that $P\ s = \mathbf{tt}$ there exists a state s' such that $Q\ s' = \mathbf{tt}$ and $\langle S,\ s\rangle \to s'$

Show that it is *not* the case that $\vdash_{\text{p}} \{\ P\ \}\ S\ \{\ Q\ \}$ implies $\models' \{\ P\ \}\ S\ \{\ Q\ \}$ and conclude that the proof system of Table 6.1 cannot be sound with respect to this definition of validity. □

Completeness (in the extensional approach)

Before turning to the proof of the completeness result we shall consider a special predicate $\text{wlp}(S,\ Q)$ defined for each statement S and predicate Q:

$$\text{wlp}(S,\ Q)\ s = \mathbf{tt}$$

if and only if for all states s',

if $\langle S,\ s\rangle \to s'$ then $Q\ s' = \mathbf{tt}$

The predicate is called the *weakest liberal precondition* for Q and it satisfies:

Fact 6.20 For every statement S and predicate Q we have

- $\models_{\mathrm{p}} \{\ \mathrm{wlp}(S,\ Q)\ \}\ S\ \{\ Q\ \}$ (*)
- if $\models_{\mathrm{p}} \{\ P\ \}\ S\ \{\ Q\ \}$ then $P \Rightarrow \mathrm{wlp}(S,\ Q)$ (**)

meaning that $\mathrm{wlp}(S,\ Q)$ is the weakest possible precondition for S and Q.

Proof: To verify that (*) holds let s and s' be states such that $\langle S,\ s\rangle \to s'$ and $\mathrm{wlp}(S,\ Q)\ s = \mathbf{tt}$. From the definition of $\mathrm{wlp}(S,\ Q)$ we get that $Q\ s' = \mathbf{tt}$ as required. To verify that (**) holds assume that $\models_{\mathrm{p}} \{\ P\ \}\ S\ \{\ Q\ \}$ and let $P\ s = \mathbf{tt}$. If $\langle S,\ s\rangle \to s'$ then $Q\ s' = \mathbf{tt}$ (because $\models_{\mathrm{p}} \{\ P\ \}\ S\ \{\ Q\ \}$) so clearly $\mathrm{wlp}(S,Q)\ s = \mathbf{tt}$. □

Exercise 6.21 Prove that the predicate INV of Example 6.9 satisfies

$$INV = \mathrm{wlp}(\texttt{while}\ \neg(\texttt{x=1})\ \texttt{do}\ (\texttt{y} := \texttt{y}\star\texttt{x};\ \texttt{x} := \texttt{x}-1),\ \texttt{y} = \texttt{n}!\ \wedge\ \texttt{n} > \texttt{0})$$ □

Exercise 6.22 Another interesting predicate called the *strongest postcondition* for S and P can be defined by

$\mathrm{sp}(P,\ S)\ s' = \mathbf{tt}$

if and only if

there exists s such that $\langle S,\ s\rangle \to s'$ and $P\ s = \mathbf{tt}$

Prove that

- $\models_{\mathrm{p}} \{\ P\ \}\ S\ \{\ \mathrm{sp}(P,\ S)\ \}$
- if $\models_{\mathrm{p}} \{\ P\ \}\ S\ \{\ Q\ \}$ then $\mathrm{sp}(P,\ S) \Rightarrow Q$

Thus $\mathrm{sp}(P,\ S)$ is the strongest possible postcondition for P and S. □

Lemma 6.23 The inference system of Table 6.1 is complete, that is for every partial correctness formula $\{\ P\ \}\ S\ \{\ Q\ \}$ we have

$\models_{\mathrm{p}} \{\ P\ \}\ S\ \{\ Q\ \}$ implies $\vdash_{\mathrm{p}} \{\ P\ \}\ S\ \{\ Q\ \}$

Proof: The completeness result follows if we can infer

$$\vdash_{\mathrm{p}} \{\ \mathrm{wlp}(S,\ Q)\ \}\ S\ \{\ Q\ \} \qquad (*)$$

for all statements S and predicates Q. To see this suppose that

$$\models_{\mathrm{p}} \{\ P\ \}\ S\ \{\ Q\ \}$$

Then Fact 6.20 gives that

$$P \Rightarrow \mathrm{wlp}(S,Q)$$

so that (*) and [$\mathrm{cons}_{\mathrm{p}}$] give

$$\vdash_{\mathrm{p}} \{\ P\ \}\ S\ \{\ Q\ \}$$

as required.

To prove (*) we proceed by structural induction on the statement S.

The case $x := a$: Based on the natural semantics it is easy to verify that

$$\mathrm{wlp}(x := a,\ Q) = Q[x \mapsto \mathcal{A}[\![a]\!]]$$

so the result follows directly from [$\mathrm{ass}_{\mathrm{p}}$].

The case `skip`: Since $\mathrm{wlp}(\texttt{skip},\ Q) = Q$ the result follows from [$\mathrm{skip}_{\mathrm{p}}$].

The case $S_1;S_2$: The induction hypothesis applied to S_1 and S_2 gives

$$\vdash_{\mathrm{p}} \{\ \mathrm{wlp}(S_2,\ Q)\ \}\ S_2\ \{\ Q\ \}$$

and

$$\vdash_{\mathrm{p}} \{\ \mathrm{wlp}(S_1,\ \mathrm{wlp}(S_2,\ Q))\ \}\ S_1\ \{\ \mathrm{wlp}(S_2,\ Q)\ \}$$

so that [$\mathrm{comp}_{\mathrm{p}}$] gives

$$\vdash_{\mathrm{p}} \{\ \mathrm{wlp}(S_1,\ \mathrm{wlp}(S_2,\ Q))\ \}\ S_1;S_2\ \{\ Q\ \}$$

We shall now prove that

$$\mathrm{wlp}(S_1;S_2,\ Q) \Rightarrow \mathrm{wlp}(S_1,\ \mathrm{wlp}(S_2,\ Q))$$

as then [$\mathrm{cons}_{\mathrm{p}}$] will give the required proof in the inference system. So assume that $\mathrm{wlp}(S_1;S_2,\ Q)\ s = \mathbf{tt}$ and we shall show that $\mathrm{wlp}(S_1,\ \mathrm{wlp}(S_2,\ Q))\ s = \mathbf{tt}$. This is obvious unless there is a state s' such that $\langle S_1,\ s\rangle \rightarrow s'$ and then we must prove that $\mathrm{wlp}(S_2,\ Q)\ s' = \mathbf{tt}$. However, this is obvious too unless there is a state s'' such that $\langle S_2,\ s'\rangle \rightarrow s''$ and then we must prove that $Q\ s'' = \mathbf{tt}$. But by [$\mathrm{comp}_{\mathrm{ns}}$] we have $\langle S_1;S_2,\ s\rangle \rightarrow s''$ so that $Q\ s'' = \mathbf{tt}$ follows from $\mathrm{wlp}(S_1;S_2,\ Q)\ s = \mathbf{tt}$.

The case `if` b `then` S_1 `else` S_2: The induction hypothesis applied to S_1 and S_2 gives

$$\vdash_{\mathrm{p}} \{\ \mathrm{wlp}(S_1,\ Q)\ \}\ S_1\ \{\ Q\ \} \text{ and } \vdash_{\mathrm{p}} \{\ \mathrm{wlp}(S_2,\ Q)\ \}\ S_2\ \{\ Q\ \}$$

Define the predicate P by

$$P = (\mathcal{B}[\![b]\!] \wedge \mathrm{wlp}(S_1,\ Q)) \vee (\neg\mathcal{B}[\![b]\!] \wedge \mathrm{wlp}(S_2,\ Q))$$

Then we have

$$(\mathcal{B}[\![b]\!] \wedge P) \Rightarrow \mathrm{wlp}(S_1,\ Q) \text{ and } (\neg\mathcal{B}[\![b]\!] \wedge P) \Rightarrow \mathrm{wlp}(S_2,\ Q)$$

so $[\mathrm{cons}_{\mathrm{p}}]$ can be applied twice and gives

$$\vdash_{\mathrm{p}} \{\ \mathcal{B}[\![b]\!] \wedge P\ \}\ S_1\ \{\ Q\ \} \text{ and } \vdash_{\mathrm{p}} \{\ \neg\mathcal{B}[\![b]\!] \wedge P\ \}\ S_2\ \{\ Q\ \}$$

Using $[\mathrm{if}_{\mathrm{p}}]$ we therefore get

$$\vdash_{\mathrm{p}} \{\ P\ \}\ \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \{\ Q\ \}$$

To see that this is the desired result it suffices to show that

$$\mathrm{wlp}(\texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ Q) \Rightarrow P$$

and this is straightforward by cases on the value of b.

The case `while` b `do` S: Define the predicate P by

$$P = \mathrm{wlp}(\texttt{while}\ b\ \texttt{do}\ S,\ Q)$$

We first show that

$$(\neg\mathcal{B}[\![b]\!] \wedge P) \Rightarrow Q \qquad (**)$$

$$(\mathcal{B}[\![b]\!] \wedge P) \Rightarrow \mathrm{wlp}(S,P) \qquad (***)$$

To verify (**) let s be such that $(\neg\mathcal{B}[\![b]\!] \wedge P)\ s = \mathbf{tt}$. Then it must be the case that $\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \rightarrow s$ so we have $Q\ s = \mathbf{tt}$. To verify (***) let s be such that $(\mathcal{B}[\![b]\!] \wedge P)\ s = \mathbf{tt}$ and we shall show that $\mathrm{wlp}(S,P)\ s = \mathbf{tt}$. This is obvious unless there is a state s' such that $\langle S,\ s\rangle \rightarrow s'$ in which case we shall prove that $P\ s' = \mathbf{tt}$. We have two cases. First we assume that $\langle \texttt{while}\ b\ \texttt{do}\ S,\ s'\rangle \rightarrow s''$ for some s''. Then $[\mathrm{while}^{\mathrm{tt}}_{\mathrm{ns}}]$ gives us that $\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \rightarrow s''$ and since $P\ s = \mathbf{tt}$ we get that $Q\ s'' = \mathbf{tt}$ using Fact 6.20. But this means that $P\ s' = \mathbf{tt}$ as was required. In the second case we assume that $\langle \texttt{while}\ b\ \texttt{do}\ S,\ s'\rangle \rightarrow s''$ does *not* hold for any state s''. But this means that $P\ s' = \mathbf{tt}$ holds vacuously and we have finished the proof of (***).

The induction hypothesis applied to the body S of the `while`-loop gives

$$\vdash_{\mathrm{p}} \{\ \mathrm{wlp}(S,P)\ \}\ S\ \{\ P\ \}$$

and using (***) together with $[\mathrm{cons}_{\mathrm{p}}]$ we get

$$\vdash_{\mathrm{p}} \{\ \mathcal{B}[\![b]\!] \wedge P\ \}\ S\ \{\ P\ \}$$

We can now apply the rule [while$_{\mathrm{p}}$] and get

$$\vdash_{\mathrm{p}} \{\ P\ \}\ \texttt{while}\ b\ \texttt{do}\ S\ \{\ \neg\mathcal{B}[\![b]\!] \wedge P\ \}$$

Finally, we use (**) together with [cons$_{\mathrm{p}}$] and get

$$\vdash_{\mathrm{p}} \{\ P\ \}\ \texttt{while}\ b\ \texttt{do}\ S\ \{\ Q\ \}$$

as required. □

Exercise 6.24 Prove that the inference system for the `while`-language extended with `repeat` S `until` b as in Exercise 6.11 is complete. (If not you should improve your rule for `repeat` S `until` b.) □

Exercise 6.25 * Prove the completeness of the inference system of Table 6.1 using the *strongest postconditions* of Exercise 6.22 rather than the weakest liberal preconditions as used in the proof of Lemma 6.23. □

Exercise 6.26 Define a notion of validity based on the denotational semantics of Chapter 4 and prove the soundness of the inference system of Table 6.1 using this definition, that is without using the equivalence between the denotational semantics and the operational semantics. □

Exercise 6.27 Use the definition of validity of Exercise 6.26 and prove the completeness of the inference system of Table 6.1. □

Expressiveness problems (in the intensional approach)

So far we have only considered the extensional approach where the preconditions and postconditions of the formulae are predicates. In the *intensional approach* they are formulae of some assertion language $\mathcal{L}$. The axioms and rules of the inference system will be as in Table 6.1, the only difference being that the preconditions and postconditions are formulae of $\mathcal{L}$ and that operations such as $P[x{\mapsto}\mathcal{A}[\![a]\!]]$, $P_1 \wedge P_2$ and $P_1 \Rightarrow P_2$ are operations on formulae of $\mathcal{L}$.

It will be natural to let $\mathcal{L}$ include the boolean expressions of **While**. The soundness proof of Lemma 6.17 then carries directly over to the intensional approach. Unfortunately, this is not the case for the completeness proof of Lemma 6.23. The reason is that the predicates wlp(S, Q) used as preconditions now have to be represented as formulae of $\mathcal{L}$ and that this may not be possible.

To illustrate the problems let S be a statement, for example a universal program in the sense of recursion theory, that has an undecidable Halting problem. Further, suppose that $\mathcal{L}$ only contains the boolean expressions of **While**. Finally, assume that there is a formula b_S of $\mathcal{L}$ such that for all states s

$\mathcal{B}[\![b_S]\!]\ s = \mathbf{tt}$ if and only if $\mathrm{wlp}(S, \mathtt{false})\ s = \mathbf{tt}$

Then also $\neg b_S$ is a formula of $\mathcal{L}$. We have

$\mathcal{B}[\![b_S]\!]\ s = \mathbf{tt}$ if and only if the computation of S on s loops

and hence

$\mathcal{B}[\![\neg b_S]\!]\ s = \mathbf{tt}$ if and only if the computation of S on s terminates

We now have a contradiction: the assumptions about S ensure that $\mathcal{B}[\![\neg b_S]\!]$ must be an undecidable function; on the other hand Table 1.2 suggests an obvious algorithm for evaluating $\mathcal{B}[\![\neg b_S]\!]$. Hence our assumption about the existence of b_S must be mistaken. Consequently we cannot mimic the proof of Lemma 6.23.

The obvious remedy is to extend $\mathcal{L}$ to be a much more powerful language that allows quantification as well. A central concept is that $\mathcal{L}$ must be *expressive* with respect to **While** and its semantics, and one then shows that Table 6.1 is *relatively complete* (in the sense of Cook). It is beyond the scope of this book to go deeper into these matters but we provide references in Chapter 7.

6.4 Extensions of the axiomatic system

In this section we shall consider two extensions of the inference system for partial correctness assertions. The first extension shows how the approach can be modified to prove *total correctness assertions* thereby allowing us to reason about termination properties. In the second extension we consider how to extend the inference systems to more language constructs, in particular recursive procedures.

Total correctness assertions

We shall now consider formulae of the form

$$\{\ P\ \}\ S\ \{\ \Downarrow Q\ \}$$

The idea is that

if the precondition P is fulfilled

then S is guaranteed to terminate (as recorded by the symbol $\Downarrow$)

and the final state will satisfy the postcondition Q.

This is formalized by defining validity of $\{\ P\ \}\ S\ \{\ \Downarrow Q\ \}$ by

$$\models_{\mathrm{t}} \{\ P\ \}\ S\ \{\ \Downarrow Q\ \}$$

$$[\text{ass}_t] \quad \{\ P[x \mapsto \mathcal{A}[\![a]\!]]\ \}\ x := a\ \{\ \Downarrow P\ \}$$

$$[\text{skip}_t] \quad \{\ P\ \}\ \texttt{skip}\ \{\ \Downarrow P\ \}$$

$$[\text{comp}_t] \quad \frac{\{\ P\ \}\ S_1\ \{\ \Downarrow Q\ \},\quad \{\ Q\ \}\ S_2\ \{\ \Downarrow R\ \}}{\{\ P\ \}\ S_1; S_2\ \{\ \Downarrow R\ \}}$$

$$[\text{if}_t] \quad \frac{\{\ \mathcal{B}[\![b]\!] \wedge P\ \}\ S_1\ \{\ \Downarrow Q\ \},\quad \{\ \neg\mathcal{B}[\![b]\!] \wedge P\ \}\ S_2\ \{\ \Downarrow Q\ \}}{\{\ P\ \}\ \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ \{\ \Downarrow Q\ \}}$$

$$[\text{while}_t] \quad \frac{\{\ P(\mathbf{z}{+}\mathbf{1})\ \}\ S\ \{\ \Downarrow P(\mathbf{z})\ \}}{\{\ \exists \mathbf{z}.P(\mathbf{z})\ \}\ \texttt{while}\ b\ \texttt{do}\ S\ \{\ \Downarrow P(\mathbf{0})\ \}}$$

where $P(\mathbf{z}{+}\mathbf{1}) \Rightarrow \mathcal{B}[\![b]\!]$, $P(\mathbf{0}) \Rightarrow \neg\mathcal{B}[\![b]\!]$

and $\mathbf{z}$ ranges over natural numbers (that is $\mathbf{z}{\geq}\mathbf{0}$)

$$[\text{cons}_t] \quad \frac{\{\ P'\ \}\ S\ \{\ \Downarrow Q'\ \}}{\{\ P\ \}\ S\ \{\ \Downarrow Q\ \}} \quad \text{where } P \Rightarrow P' \text{ and } Q' \Rightarrow Q$$

Table 6.2: Axiomatic system for total correctness

if and only if

for all states s, if $P\ s = \mathbf{tt}$ then there exists s' such that

$Q\ s' = \mathbf{tt}$ and $\langle S, s\rangle \to s'$

The inference system for total correctness assertions is very similar to that for partial correctness assertions, the only difference being that the rule for the `while`-construct has changed. The complete set of axioms and rules is given in Table 6.2. We shall write

$$\vdash_t \{\ P\ \}\ S\ \{\ \Downarrow Q\ \}$$

if there exists an inference tree with the formula $\{\ P\ \}\ S\ \{\ \Downarrow Q\ \}$ as root, that is if the formula is provably in the inference system.

In the rule [while$_t$] we use a parameterized family $P(\mathbf{z})$ of predicates for the invariant. The idea is that $\mathbf{z}$ is the number of unfoldings of the `while`-loop that will be necessary. So if the `while`-loop does not have to be unfolded at all then $P(\mathbf{0})$ holds and it must imply that b is false. If the `while`-loop has to be unfolded $\mathbf{z}{+}\mathbf{1}$ times then $P(\mathbf{z}{+}\mathbf{1})$ holds and b must hold *before* the body of the loop is executed; then $P(\mathbf{z})$ will hold *afterwards* so that we have decreased the total number of times the loop remains to be unfolded. The precondition of the conclusion of the rule expresses that there exists a bound on the number of times the loop has to be unfolded and the postcondition expresses that when the `while`-loop has terminated then no more unfoldings are necessary.

Example 6.28 The total correctness of the factorial statement can be expressed by the following assertion:

$$\{\ \mathtt{x} > 0 \wedge \mathtt{x} = \mathtt{n}\ \}$$
$$\mathtt{y} := 1;\ \mathtt{while}\ \neg(\mathtt{x}{=}1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1)$$
$$\{\ \Downarrow \mathtt{y} = \mathtt{n}!\ \}$$

where $\mathtt{y} = \mathtt{n}!$ is an abbreviation for the predicate

$$P \text{ where } P\ s = (s\ \mathtt{y} = (s\ \mathtt{n})!)$$

In addition to expressing that the final value of y is the factorial of the initial value of x the assertion also expresses that the program does indeed terminate on all states satisfying the precondition. The inference of this assertion proceeds in a number of stages. First we define the predicate $INV(\mathbf{z})$ that is going to be the invariant of the while-loop

$$INV(\mathbf{z})\ s = (s\ \mathtt{x} > \mathbf{0} \text{ and } (s\ \mathtt{y}) \star (s\ \mathtt{x})! = (s\ \mathtt{n})! \text{ and } s\ \mathtt{x} = \mathbf{z} + \mathbf{1})$$

We shall first consider the body of the loop. Using $[\mathrm{ass}_t]$ we get

$$\vdash_t \{\ INV(\mathbf{z})[\mathtt{x}{\mapsto}\mathtt{x}{-}1]\ \}\ \mathtt{x} := \mathtt{x}{-}1\ \{\ \Downarrow INV(\mathbf{z})\ \}$$

Similarly, we get

$$\vdash_t \{\ (INV(\mathbf{z})[\mathtt{x}{\mapsto}\mathtt{x}{-}1])[\mathtt{y}{\mapsto}\mathtt{y}{\star}\mathtt{x}]\ \}\ \mathtt{y} := \mathtt{y} \star \mathtt{x}\ \{\ \Downarrow INV(\mathbf{z})[\mathtt{x}{\mapsto}\mathtt{x}{-}1]\ \}$$

We can now apply the rule $[\mathrm{comp}_t]$ to the two assertions above and get

$$\vdash_t \{\ (INV(\mathbf{z})[\mathtt{x}{\mapsto}\mathtt{x}{-}1])[\mathtt{y}{\mapsto}\mathtt{y}{\star}\mathtt{x}]\ \}\ \mathtt{y} := \mathtt{y} \star \mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1\ \{\ \Downarrow INV(\mathbf{z})\ \}$$

It is easy to verify that

$$INV(\mathbf{z}{+}\mathbf{1}) \Rightarrow (INV(\mathbf{z})[\mathtt{x}{\mapsto}\mathtt{x}{-}1])[\mathtt{y}{\mapsto}\mathtt{y}{\star}\mathtt{x}]$$

so using the rule $[\mathrm{cons}_t]$ we get

$$\vdash_t \{\ INV(\mathbf{z}{+}\mathbf{1})\ \}\ \mathtt{y} := \mathtt{y} \star \mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1\ \{\ \Downarrow INV(\mathbf{z})\ \}$$

It is straightforward to verify that

$$INV(\mathbf{0}) \Rightarrow \neg(\neg(\mathtt{x}{=}1)), \text{ and}$$
$$INV(\mathbf{z}{+}\mathbf{1}) \Rightarrow \neg(\mathtt{x}{=}1)$$

Therefore we can use the rule $[\mathrm{while}_t]$ and get

$$\vdash_t \{\ \exists \mathbf{z}.INV(\mathbf{z})\ \}\ \mathtt{while}\ \neg(\mathtt{x}{=}1)\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}1)\ \{\ \Downarrow INV(\mathbf{0})\ \}$$

We shall now apply the axiom $[\mathrm{ass}_t]$ to the statement $\mathtt{y} := 1$ and get

$$\vdash_{\mathrm{t}} \{\ (\exists \mathbf{z}.INV(\mathbf{z}))[\mathtt{y}\mapsto\mathtt{1}]\ \}\ \mathtt{y} := \mathtt{1}\ \{\ \Downarrow \exists \mathbf{z}.INV(\mathbf{z})\ \}$$

so using [comp$_{\mathrm{t}}$] we get

$$\begin{array}{l}\vdash_{\mathrm{t}} \{\ (\exists \mathbf{z}.INV(\mathbf{z}))[\mathtt{y}\mapsto\mathtt{1}]\ \}\\ \quad \mathtt{y} := \mathtt{1};\ \mathtt{while}\ \neg(\mathtt{x}{=}\mathtt{1})\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}\mathtt{1})\\ \quad \{\ \Downarrow INV(\mathbf{0})\ \}\end{array}$$

Clearly we have

$$\mathtt{x} > \mathtt{0} \wedge \mathtt{x} = \mathtt{n} \Rightarrow (\exists \mathbf{z}.INV(\mathbf{z}))[\mathtt{y}\mapsto\mathbf{1}], \text{ and}$$

$$INV(\mathbf{0}) \Rightarrow \mathtt{y} = \mathtt{n}!$$

so applying rule [cons$_{\mathrm{t}}$] we get

$$\begin{array}{l}\vdash_{\mathrm{t}} \{\ \mathtt{x} > \mathtt{0} \wedge \mathtt{x} = \mathtt{n}\ \}\\ \quad \mathtt{y} := 1;\ \mathtt{while}\ \neg(\mathtt{x}{=}\mathtt{1})\ \mathtt{do}\ (\mathtt{y} := \mathtt{y}{\star}\mathtt{x};\ \mathtt{x} := \mathtt{x}{-}\mathtt{1})\\ \quad \{\ \Downarrow \mathtt{y} = \mathtt{n}!\ \}\end{array}$$

as required. □

Exercise 6.29 Suggest a total correctness inference rule for `repeat` S `until` b. You are not allowed to rely on the existence of a `while`-construct in the programming language. □

Lemma 6.30 The total correctness system of Table 6.2 is sound, that is for every total correctness formula $\{\ P\ \}\ S\ \{\ \Downarrow Q\ \}$ we have

$$\vdash_{\mathrm{t}} \{\ P\ \}\ S\ \{\ \Downarrow Q\ \} \text{ implies } \models_{\mathrm{t}} \{\ P\ \}\ S\ \{\ \Downarrow Q\ \}$$

Proof: The proof proceeds by induction on the shape of the inference tree just as in the proof of Lemma 6.17.

The case [ass$_{\mathrm{t}}$]: We shall prove that the axiom is valid, so assume that s is such that $(P[x\mapsto\mathcal{A}[\![a]\!]])\ s = \mathbf{tt}$ and let $s' = s[x\mapsto\mathcal{A}[\![a]\!]s]$. Then [ass$_{\mathrm{ns}}$] gives

$$\langle x := a,\ s\rangle \to s'$$

and from $(P[x\mapsto\mathcal{A}[\![a]\!]])\ s = \mathbf{tt}$ we get $P\ s' = \mathbf{tt}$ as was to be shown.

The case [skip$_{\mathrm{t}}$]: This case is immediate.

The case [comp$_{\mathrm{t}}$]: We assume that

$$\models_{\mathrm{t}} \{\ P\ \}\ S_1\ \{\ \Downarrow Q\ \}, \text{ and} \qquad (*)$$

$$\models_{\mathrm{t}} \{\ Q\ \}\ S_2\ \{\ \Downarrow R\ \} \qquad (**)$$

and we have to prove that $\models_{\mathrm{t}} \{\ P\ \}\ S_1;\ S_2\ \{\ \Downarrow R\ \}$. So let s be such that $P\ s = \mathbf{tt}$. From (*) we get that there exists a state s' such that $Q\ s' = \mathbf{tt}$ and

$$\langle S_1,\ s\rangle \to s'$$

Since $Q\ s' = \mathbf{tt}$ we get from (**) that there exists a state s'' such that $R\ s'' = \mathbf{tt}$ and

$$\langle S_2,\ s'\rangle \to s''$$

Using [comp_{ns}] we therefore get

$$\langle S_1;\ S_2,\ s\rangle \to s''$$

and since $R\ s'' = \mathbf{tt}$ we have finished this case.

The case [if_{t}]: Assume that

$$\models_{\text{t}} \{\ \mathcal{B}[\![b]\!] \wedge P\ \}\ S_1\ \{\ \Downarrow Q\ \}, \text{ and} \qquad (*)$$
$$\models_{\text{t}} \{\ \neg\mathcal{B}[\![b]\!] \wedge P\ \}\ S_2\ \{\ \Downarrow Q\ \}$$

To prove $\models_{\text{t}} \{\ P\ \}$ `if` b `then` S_1 `else` $S_2\ \{\ \Downarrow Q\ \}$ consider a state s such that $P\ s = \mathbf{tt}$. We have two cases. If $\mathcal{B}[\![b]\!]s = \mathbf{tt}$ then $(\mathcal{B}[\![b]\!] \wedge P)\ s = \mathbf{tt}$ and from (*) we get that there is a state s' such that $Q\ s' = \mathbf{tt}$ and

$$\langle S_1,\ s\rangle \to s'$$

From [if_{ns}] we then get

$$\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \to s'$$

as was to be proved. If $\mathcal{B}[\![b]\!]s = \mathbf{ff}$ the result follows in a similar way from the second assumption.

The case [while_{t}]: Assume that

$$\models_{\text{t}} \{\ P(\mathbf{z}{+}\mathbf{1})\ \}\ S\ \{\ \Downarrow P(\mathbf{z})\ \}, \qquad (*)$$
$$P(\mathbf{z}{+}\mathbf{1}) \Rightarrow \mathcal{B}[\![b]\!], \text{ and}$$
$$P(\mathbf{0}) \Rightarrow \neg\mathcal{B}[\![b]\!]$$

To prove $\models_{\text{t}} \{\ \exists\mathbf{z}.P(\mathbf{z})\ \}$ `while` b `do` $S\ \{\ \Downarrow P(\mathbf{0})\ \}$ it is sufficient to prove that for all natural numbers $\mathbf{z}$

$$\begin{array}{l}\text{if } P(\mathbf{z})\ s = \mathbf{tt} \text{ then there exists a state } s' \text{ such that}\\ P(\mathbf{0})\ s' = \mathbf{tt} \text{ and } \langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \to s'\end{array} \qquad (**)$$

So consider a state s such that $P(\mathbf{z})\ s = \mathbf{tt}$. The proof is now by numerical induction on $\mathbf{z}$.

First assume that $\mathbf{z} = \mathbf{0}$. The assumption $P(\mathbf{0}) \Rightarrow \neg\mathcal{B}[\![b]\!]$ gives that $\mathcal{B}[\![b]\!]s = \mathbf{ff}$ and from [$\text{while}^{\text{ff}}_{\text{ns}}$] we get

$$\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \to s$$

Since $P(\mathbf{0})\ s = \mathbf{tt}$ this proves the base case.

For the induction step assume that (**) holds for all states satisfying $P(\mathbf{z})$ and that $P(\mathbf{z{+}1})\ s = \mathbf{tt}$. From (*) we get that there is a state s' such that $P(\mathbf{z})\ s' = \mathbf{tt}$ and

$$\langle S,\ s\rangle \to s'$$

The numerical induction hypothesis applied to s' gives that there is some state s'' such that $P(\mathbf{0})\ s'' = \mathbf{tt}$ and

$$\langle \texttt{while}\ b\ \texttt{do}\ S,\ s'\rangle \to s''$$

Furthermore, the assumption $P(\mathbf{z{+}1}) \Rightarrow \mathcal{B}[\![b]\!]$ gives $\mathcal{B}[\![b]\!]s = \mathbf{tt}$. We can therefore apply $[\text{while}^{\text{tt}}_{\text{ns}}]$ and get that

$$\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \to s''$$

Since $P(\mathbf{0})\ s'' = \mathbf{tt}$ this completes the proof of (**).

The case $[\text{cons}_{\text{t}}]$: Suppose that

$$\models_{\text{t}} \{\ P'\ \}\ S\ \{\ \Downarrow Q'\ \},$$
$$P \Rightarrow P', \text{ and}$$
$$Q' \Rightarrow Q$$

To prove $\models_{\text{t}} \{\ P\ \}\ S\ \{\ \Downarrow Q\ \}$ consider a state s such that $P\ s = \mathbf{tt}$. Then $P'\ s = \mathbf{tt}$ and there is a state s' such that $Q'\ s' = \mathbf{tt}$ and

$$\langle S,\ s\rangle \to s'$$

However, we also have that $Q\ s' = \mathbf{tt}$ and this proves the result. □

Exercise 6.31 Show that the inference rule for `repeat` S `until` b suggested in Exercise 6.29 preserves validity. Argue that this means that the entire proof system consisting of the axioms and rules of Table 6.2 together with the rule of Exercise 6.29 is sound. □

Exercise 6.32 * Prove that the inference system of Table 6.2 is complete, that is

$$\models_{\text{t}} \{\ P\ \}\ S\ \{\ \Downarrow Q\ \} \text{ implies } \vdash_{\text{t}} \{\ P\ \}\ S\ \{\ \Downarrow Q\ \}$$ □

Exercise 6.33 * Prove that

$$\text{if } \vdash_{\text{t}} \{\ P\ \}\ S\ \{\ \Downarrow Q\ \} \text{ then } \vdash_{\text{p}} \{\ P\ \}\ S\ \{\ Q\ \}$$

Does the converse result hold? □