Deductive Verification of Software – Exercises and Solutions

(6.0 VU Formal Methods in Computer Science)

WS 2012/2013

Exercise 1

Let p be the following program:

```
x := x + y;
if x < 0 then
   abort
else
   while x ≠ y do
      x := x + 1;
      y := y + 2
   od
fi
```

(a) Show that p is syntactically correct with respect to the definition of TPL.

(b) Let σ be a state such that $\sigma(x) = 1$ and $\sigma(y) = 5$. Compute $[p]\,\sigma$, using

- the structural operational semantics
- the natural semantics

of TPL.

(c) Show that $\{\, x = 2y \wedge y > 2 \,\}\, p \,\{\, x = y \,\}$ is totally correct

- by computing the weakest precondition of the program;
- using the Hoare calculus;
- using annotation rules.

Solution

(a) ***Syntax check:*** We give a parallel derivation[1] for the program, using the productions of the context-free grammar for TPL (see slides).

$$
\begin{aligned}
&\mathcal{P} \Rightarrow_p \mathcal{P};\mathcal{P}\\
&\quad\Rightarrow_p \mathcal{V} := \mathcal{E};\ \mathsf{if}\ \mathcal{E}\ \mathsf{then}\ \mathcal{P}\ \mathsf{else}\ \mathcal{P}\ \mathsf{fi}\\
&\quad\Rightarrow_p x := (\mathcal{E}\,\mathcal{B}\,\mathcal{E});\ \mathsf{if}\ (\mathcal{E}\,\mathcal{B}\,\mathcal{E})\ \mathsf{then\ abort\ else\ while}\ \mathcal{E}\ \mathsf{do}\ \mathcal{P}\ \mathsf{od\ fi}\\
&\quad\Rightarrow_p x := (\mathcal{V}+\mathcal{V});\ \mathsf{if}\ (\mathcal{V}<\mathcal{N})\ \mathsf{then\ abort\ else\ while}\ (\mathcal{E}\,\mathcal{B}\,\mathcal{E})\ \mathsf{do}\ \mathcal{P};\mathcal{P}\ \mathsf{od\ fi}\\
&\quad\Rightarrow_p x := (x+y);\ \mathsf{if}\ (x<0)\ \mathsf{then\ abort\ else\ while}\ (\mathcal{V}\neq\mathcal{V})\ \mathsf{do}\ \mathcal{V}:=\mathcal{E};\ \mathcal{V}:=\mathcal{E}\ \mathsf{od\ fi}\\
&\quad\Rightarrow_p x := (x+y);\ \mathsf{if}\ (x<0)\ \mathsf{then\ abort\ else\ while}\ (x\neq y)\ \mathsf{do}\ x:=(\mathcal{E}\,\mathcal{B}\,\mathcal{E});\ y:=(\mathcal{E}\,\mathcal{B}\,\mathcal{E})\ \mathsf{od\ fi}\\
&\quad\Rightarrow_p x := (x+y);\ \mathsf{if}\ (x<0)\ \mathsf{then\ abort\ else\ while}\ (x\neq y)\ \mathsf{do}\ x:=(\mathcal{V}+\mathcal{N});\ y:=(\mathcal{V}+\mathcal{N})\ \mathsf{od\ fi}\\
&\quad\Rightarrow_p x := (x+y);\ \mathsf{if}\ (x<0)\ \mathsf{then\ abort\ else\ while}\ (x\neq y)\ \mathsf{do}\ x:=(x+1);\ y:=(y+2)\ \mathsf{od\ fi}
\end{aligned}
$$

(b) ***Structural operational semantics:*** We compute a complete program run. The final state is, by the definition of the semantics, the result of $[p]\,\sigma$.

$$
\begin{aligned}
(p,\sigma) &= (x := x+y; \mathsf{if}\ \cdots\ \mathsf{fi},\ \sigma)\\
&\quad\left[\begin{array}{l}(x := x+y,\ \sigma)\\ \Rightarrow \sigma_1 \qquad \text{where } \sigma_1(x) = [x+y]\,\sigma = 6 \text{ and } \sigma_1(y) = \sigma(y) = 5\end{array}\right.\\
&\Rightarrow (\mathsf{if}\ x<0\ \mathsf{then\ abort\ else\ while}\ x\neq y\ \mathsf{do}\ \cdots\ \mathsf{od\ fi},\ \sigma_1)\\
&\Rightarrow (\mathsf{while}\ x\neq y\ \mathsf{do}\ x:=x+1;\ y:=y+2\ \mathsf{od},\ \sigma_1)\\
&\quad \text{since } [x<0]\,\sigma_1 = (\sigma_1(x)<0) = (6<0) = 0\\
&\Rightarrow (x := x+1; y := y+2; \mathsf{while}\ x\neq y\ \mathsf{do}\ x:=x+1;\ y:=y+2\ \mathsf{od},\ \sigma_1)\\
&\quad \text{since } [x\neq y]\,\sigma_1 = (\sigma_1(x)\neq\sigma_1(y)) = (6\neq 5) = 1\\
&\quad\left[\begin{array}{l}(x := x+1; y := y+2,\ \sigma_1)\\ \quad\left[\begin{array}{l}(x := x+1,\ \sigma_1)\\ \Rightarrow \sigma_2 \qquad \text{where } \sigma_2(x) = [x+1]\,\sigma_1 = 7 \text{ and } \sigma_2(y) = \sigma_1(y) = 5\end{array}\right.\\ \Rightarrow (y := y+2,\ \sigma_2)\end{array}\right.\\
&\Rightarrow (y := y+2; \mathsf{while}\ x\neq y\ \mathsf{do}\ x:=x+1;\ y:=y+2\ \mathsf{od},\ \sigma_2)\\
&\quad\left[\begin{array}{l}(y := y+2,\ \sigma_2)\\ \Rightarrow \sigma_3 \qquad \text{where } \sigma_3(y) = [y+2]\,\sigma_2 = 7 \text{ and } \sigma_3(x) = \sigma_2(x) = 7\end{array}\right.\\
&\Rightarrow (\mathsf{while}\ x\neq y\ \mathsf{do}\ x:=x+1;\ y:=y+2\ \mathsf{od},\ \sigma_3)\\
&\Rightarrow \sigma_3\\
&\quad \text{since } [x\neq y]\,\sigma_3 = (\sigma_3(x)\neq\sigma_3(y)) = (7\neq 7) = 0
\end{aligned}
$$

Therefore we have $[p]\,\sigma = \sigma_3$.

[1] In a *parallel derivation* a derivation step consists in replacing all variables in the expression simultaneously (in parallel) by the right-hand side of some production.

Natural semantics:

$$
\begin{aligned}
[p]\,\sigma &= [x := x + y; \mathsf{if}\ \cdots\ \mathsf{fi}]\,\sigma \\
&= [\mathsf{if}\ \cdots\ \mathsf{fi}]\,[x := x + y]\,\sigma \\
&= [\mathsf{if}\ x < 0\ \mathsf{then\ abort\ else\ while}\ x \neq y\ \mathsf{do}\ \cdots\ \mathsf{od\ fi}]\,\sigma_1 \\
&\qquad \text{where } \sigma_1(x) = [x + y]\,\sigma = 6 \text{ and } \sigma_1(y) = \sigma(y) = 5 \\
&= [\mathsf{while}\ x \neq y\ \mathsf{do}\ x := x + 1;\ y := y + 2\ \mathsf{od}]\,\sigma_1 \\
&\qquad \text{since } [x < 0]\,\sigma_1 = (\sigma_1(x) < 0) = (6 < 0) = 0 \\
&= [x := x + 1; y := y + 2; \mathsf{while}\ x \neq y\ \mathsf{do}\ x := x + 1;\ y := y + 2\ \mathsf{od}]\,\sigma_1 \\
&\qquad \text{since } [x \neq y]\,\sigma_1 = (\sigma_1(x) \neq \sigma_1(y)) = (6 \neq 5) = 1 \\
&= [\mathsf{while}\ x \neq y\ \mathsf{do}\ x := x + 1;\ y := y + 2\ \mathsf{od}]\,[x := x + 1; y := y + 2]\,\sigma_1 \\
&= [\mathsf{while}\ x \neq y\ \mathsf{do}\ x := x + 1;\ y := y + 2\ \mathsf{od}]\,[y := y + 2]\,[x := x + 1]\,\sigma_1 \\
&= [\mathsf{while}\ x \neq y\ \mathsf{do}\ x := x + 1;\ y := y + 2\ \mathsf{od}]\,[y := y + 2]\,\sigma_2 \\
&\qquad \text{where } \sigma_2(x) = [x + 1]\,\sigma_1 = 7 \text{ and } \sigma_2(y) = \sigma_1(y) = 5 \\
&= [\mathsf{while}\ x \neq y\ \mathsf{do}\ x := x + 1;\ y := y + 2\ \mathsf{od}]\,\sigma_3 \\
&\qquad \text{where } \sigma_3(y) = [y + 2]\,\sigma_2 = 7 \text{ and } \sigma_3(x) = \sigma_2(x) = 7 \\
&= \sigma_3 \\
&\qquad \text{since } [x \neq y]\,\sigma_3 = (\sigma_3(x) \neq \sigma_3(y)) = (7 \neq 7) = 0
\end{aligned}
$$

(c) ***Correctness proof via weakest preconditions:*** We show that the precondition implies the weakest precondition of the program with respect to the postcondition.

$\mathrm{wp}(x := x + y; \mathsf{if}\ \cdots\ , x = y) = \mathrm{wp}(x := x + y,\ \mathrm{wp}(\mathsf{if}\ \cdots\ , x = y))$

$$
\begin{aligned}
&\mathrm{wp}(\mathsf{if}\ \cdots\ , x = y) \\
&= ((x < 0 \wedge \mathrm{wp}(\mathsf{abort}, x = y)) \vee (x \geq 0 \wedge \mathrm{wp}(\mathsf{while}\ \cdots\ , x = y))) \\
&= ((x < 0 \wedge \mathrm{false}) \vee (x \geq 0 \wedge \mathrm{wp}(\mathsf{while}\ \cdots\ , x = y))) \\
&= (\mathrm{false} \vee (x \geq 0 \wedge \mathrm{wp}(\mathsf{while}\ \cdots\ , x = y))) \\
&= (x \geq 0 \wedge \mathrm{wp}(\mathsf{while}\ \cdots\ , x = y))
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{wp}(\mathsf{while}\ \cdots\ , x = y) = \exists i \geq 0\, F_i \\
&\quad F_0 : \neg(x \neq y) \wedge x = y \\
&\quad F_1 : x \neq y \wedge \mathrm{wp}(x := x + 1; y := y + 2,\ x = y) \\
&\qquad = (x \neq y \wedge x + 1 = y + 2) \\
&\qquad = (x = y + 1) \\
&\quad F_i : x = y + i \quad \text{(guess)} \\
&\quad F_{i+1} : x \neq y \wedge \mathrm{wp}(x := x + 1; y := y + 2,\ F_i) \\
&\qquad = x \neq y \wedge (x + 1) = (y + 2) + i \\
&\qquad = x \neq y \wedge x = y + i + 1 \\
&\qquad = (x = y + i + 1) \quad \text{(proof)} \\
&= \exists i \geq 0\, (x = y + i) \\
&= x \geq y
\end{aligned}
$$

$= (x \geq 0 \wedge x \geq y)$

$= \text{wp}(x := x + y,\ x \geq 0 \wedge x \geq y)$
$= (x + y) \geq 0 \wedge (x + y) \geq y$
$= (x + y) \geq 0 \wedge x \geq 0$

It remains to show that the precondition implies the weakest precondition.

$$\begin{aligned} x = 2y \wedge y > 2 &\Rightarrow \text{wp}(p, x = y) \\ &\Rightarrow (x + y) \geq 0 \wedge x \geq 0 \end{aligned}$$

The two conjuncts of the conclusion can be proven separately:

$$\begin{aligned} x = 2y \wedge y > 2 &\Rightarrow (x + y) \geq 0 \\ x = 2y \wedge y > 2 &\Rightarrow x \geq 0 \end{aligned}$$

The first implication is valid, since $x = 2y$ and $y > 2$ imply $(x + y) = 3y > 6 \geq 0$. The second implication holds since $x = 2y$ and $y > 2$ imply $x > 4$, which implies the conclusion $x \geq 0$.

Correctness proof via the Hoare calculus: The Hoare derivation is shown in figure 1. It remains to find formulas F and G and an expression e such that the implications 1, 2, 4, 6 and 7 are valid.

We guess a suitable formula F by forward reasoning. The new value of x is the old one plus the value of y, hence the old value can be obtained after the assignment by evaluating $x - y$. We obtain $x - y = 2y \wedge y > 2$ as description of the states after the assignment, therefore we choose $F = (x = 3y \wedge y > 2)$. Now we are able to prove implications 1 and 2.

Validity of implication 1:

$$\begin{aligned} (x = 2y \wedge y > 2) &\Rightarrow F[x/x + y] \\ (x = 2y \wedge y > 2) &\Rightarrow (x + y = 3y \wedge y > 2) \end{aligned}$$

The first conjunct of the conclusion, $x + y = 3y$, is implied by the first conjunct of the premise, and the second conjunct of the conclusion, $y > 2$, is part of the premise.

Validity of implication 2:

$$\begin{aligned} (F \wedge x < 0) &\Rightarrow \text{false} \\ (x = 3y \wedge y > 2 \wedge x < 0) &\Rightarrow \text{false} \end{aligned}$$

The premise is contradictory: $y > 2$ implies $3y > 6$, while $x = 3y$ and $x < 0$ imply $3y < 0$. Therefore the implication holds in every state.

$$\dfrac{\dfrac{(G \wedge x \neq y \wedge e = z) \Rightarrow (G \wedge 0 \leq e < z)[y/y+2][x/x+1]^{(7)} \quad \{\,(G \wedge 0 \leq e < z)[y/y+2][x/x+1]\,\}\, x := x+1 \,\{\,(G \wedge 0 \leq e < z)[y/y+2]\,\}^{(\text{as})}}{\{\,G \wedge x \neq y \wedge e = z\,\}\, x := x+1 \,\{\,(G \wedge 0 \leq e < z)[y/y+2]\,\}}\,(\text{lc})}{\text{(5, from below)}}$$

$$\dfrac{\dfrac{\{\,G \wedge x \neq y \wedge e = z\,\}\, x := x+1 \,\{\,(G \wedge 0 \leq e < z)[y/y+2]\,\}^{(5,\text{ see above})} \quad \{\,(G \wedge 0 \leq e < z)[y/y+2]\,\}\, y := y+2 \,\{\,G \wedge 0 \leq e < z\,\}^{(\text{as})}}{\dfrac{\{\,G \wedge x \neq y \wedge e = z\,\}\, x := x+1; y := y+2 \,\{\,G \wedge 0 \leq e < z\,\}}{\{\,G\,\}\,\mathsf{while}\ x \neq y\ \mathsf{do}\ x := x+1; y := y+2\ \mathsf{od}\,\{\,G \wedge \neg x \neq y\,\}}\,(\text{wht}'')}\,(\text{sc})}{\dfrac{(F \wedge x \not< 0) \Rightarrow G^{(4)} \qquad (G \wedge \neg x \neq y) \Rightarrow x = y^{(6)}}{\{\,F \wedge x \not< 0\,\}\,\mathsf{while}\ x \neq y\ \mathsf{do}\ x := x+1; y := y+2\ \mathsf{od}\,\{\,x = y\,\}}\,(\text{lc})}}{\text{(3, from below)}}$$

$$\dfrac{\dfrac{(x = 2y \wedge y > 2) \Rightarrow F[x/x+y]^{(1)} \quad \{\,F[x/x+y]\,\}\, x := x+y \,\{\,F\,\}^{(\text{as})}}{\{\,x = 2y \wedge y > 2\,\}\, x := x+y \,\{\,F\,\}}\,(\text{lc}) \qquad \dfrac{(F \wedge x < 0) \Rightarrow \text{false}^{(2)} \quad \{\,\text{false}\,\}\,\mathsf{abort}\,\{\,x = y\,\}^{(\text{abt})}}{\{\,F \wedge x < 0\,\}\,\mathsf{abort}\,\{\,x = y\,\}}\,(\text{lc})}{\dfrac{\{\,F\,\}\,\mathsf{if}\ x < 0\ \mathsf{then\ abort\ else\ while}\ \cdots\ \mathsf{od\ fi}\,\{\,x = y\,\}}{\{\,x = 2y \wedge y > 2\,\}\, x := x+y; \mathsf{if}\ x < 0\ \mathsf{then\ abort\ else\ while}\ x \neq y\ \mathsf{do}\ x := x+1; y := y+2\ \mathsf{od\ fi}\,\{\,x = y\,\}}\,(\text{sc})}\,(\text{if})$$

$$\dfrac{\{\,F \wedge x \not< 0\,\}\,\mathsf{while}\ \cdots\ \mathsf{od}\,\{\,x = y\,\}^{(3,\text{ see above})}}{\{\,F\,\}\,\mathsf{if}\ x < 0\ \mathsf{then\ abort\ else\ while}\ \cdots\ \mathsf{od\ fi}\,\{\,x = y\,\}}\,(\text{sc})$$

It remains to find formulas F and G and an expression e such that the following implications become valid:

(1) $(x = 2y \wedge y > 2) \Rightarrow F[x/x+y]$

(2) $(F \wedge x < 0) \Rightarrow \text{false}$

(4) $(F \wedge x \not< 0) \Rightarrow G$

(6) $(G \wedge \neg x \neq y) \Rightarrow x = y$

(7) $(G \wedge x \neq y \wedge e = z) \Rightarrow (G \wedge 0 \leq e < z)[y/y+2][x/x+1]$

We deliberately choose the names G, e and z instead of Inv, t and t_0 to emphasise the fact that the latter are just placeholders for a formula, an expression and a variable characteristic of a particular while-loop. In general, different loops require different invariants and variants. Moreover, t_0 is not an initial instance of the variant t, but an auxiliary variable occurring neither in the loop nor in the invariant nor in the variant under consideration, which represents the value of the variant before the loop.

Figure 1: Derivation of $\{\,x = 2y \wedge y > 2\,\}\, p \,\{\,x = y\,\}$ in the Hoare calculus, where p is the program of exercise 1.

To obtain a suitable variant e we rewrite the loop condition $x \neq y$ as $x - y \neq 0$. We observe that the expression $x - y$ decreases in each iteration and equals zero when the loop terminates. Therefore we guess $e = x - y$.

The main purpose of invariant G is to guarantee the partial correctness of the loop: It has to be strong enough to imply the postcondition (implication 6), while being at the same time weak enough to be implied by the precondition of the loop (implication 4) and being maintained throughout the iterations (first conclusion in implication 7). In this example implication 6 is valid for any choice of G, since the negated loop condition already implies the postcondition. Hence we choose the weakest possible invariant, $G = \text{true}$, which obviously also satisfies the other implications.

The second purpose of the invariant is to ensure properties of the variables needed for showing that e is a variant. To prove $e \geq 0$ we need to assume $x \geq y$; obviously the loop only terminates for such states. This property has to be guaranteed at the start of the loop and after each iteration. Therefore we add it to the invariant and obtain $G = (\text{true} \wedge x \geq y) = x \geq y$.

Validity of implication 4:

$$\begin{aligned} (F \wedge x \not< 0) &\Rightarrow G \\ (x = 3y \wedge y > 2 \wedge x \geq 0) &\Rightarrow x \geq y \end{aligned}$$

$x \geq y$ follows from $x = 3y$ and the fact that y is non-negative (because of $y > 2$).

Validity of implication 6:

$$\begin{aligned} (G \wedge \neg x \neq y) &\Rightarrow x = y \\ (G \wedge x = y) &\Rightarrow x = y \end{aligned}$$

The conclusion is part of the premise.

Validity of implication 7:

$$\begin{aligned} (G \wedge x \neq y \wedge e = z) &\Rightarrow (G \wedge 0 \leq e < z)[y/y+2][x/x+1] \\ (x \geq y \wedge x \neq y \wedge x - y = z) &\Rightarrow (x \geq y \wedge 0 \leq x - y < z)[y/y+2][x/x+1] \\ (x \geq y \wedge x \neq y \wedge x - y = z) &\Rightarrow (x \geq (y{+}2) \wedge 0 \leq x - (y{+}2) < z)[x/x+1] \\ (x \geq y \wedge x \neq y \wedge x - y = z) &\Rightarrow (x{+}1 \geq y{+}2 \wedge 0 \leq (x{+}1) - (y{+}2) < z) \\ (x \geq y \wedge x \neq y \wedge x - y = z) &\Rightarrow (x \geq y + 1 \wedge 0 \leq x - y - 1 < z) \\ (x > y \wedge x - y = z) &\Rightarrow 0 \leq x - y - 1 < z \end{aligned}$$

(Note that $x \geq y{+}1$ is the same as $0 \leq x{-}y{-}1$.) The condition $x > y$ in the premise is equivalent to $0 \leq x - y - 1$, and the condition $x - y = z$ implies $x - y - 1 < z$.

Since all implications obtained by the Hoare calculus are valid, this initial correctness assertion is totally correct.

Correctness proof using annotation rules: We annotate the program with additional assertions. The order of rule applications is indicated by the numbering of formulas. The order as well as the kind of rules is not uniquely determined; other annotations are possible.

$\{\, 1\colon x = 2y \wedge y > 2 \,\}$
$x := x + y;$
$\{\, 3\colon \exists x'(x' = 2y \wedge y > 2 \wedge x = x' + y) \,\}$ as↓
if $x < 0$ then
  $\{\, 4\colon (\text{Formula } 4) \wedge x < 0 \,\}$ if↓
  $\{\, 6\colon \text{false} \,\}$ abt
  abort
  $\{\, 7\colon \text{false} \,\}$ abt
  $\{\, 8\colon x = y \,\}$ fi↑
else
  $\{\, 5\colon (\text{Formula } 4) \wedge x \not< 0 \,\}$ if↓
  $\{\, 10\colon G \,\}$ wht″
  while $x \neq y$ do
    $\{\, 11\colon G \wedge x \neq y \wedge e = z \,\}$ wht″
    $\{\, 15\colon (G \wedge 0 \leq e < z)[y/y+2][x/x+1] \,\}$ as↑
    $x := x + 1;$
    $\{\, 14\colon (G \wedge 0 \leq e < z)[y/y+2] \,\}$ as↑
    $y := y + 2$
    $\{\, 12\colon G \wedge 0 \leq e < z \,\}$ wht″
  od
  $\{\, 13\colon G \wedge \neg x \neq y \,\}$ wht″
  $\{\, 9\colon x = y \,\}$ fi↑
fi
$\{\, 2\colon x = y \,\}$

We start by simplifying formula 3:

$$\begin{aligned}\exists x'(x' = 2y \wedge y > 2 \wedge x = x' + y) &= \exists x'(x' = 2y \wedge y > 2 \wedge x = 2y + y)\\ &= y > 2 \wedge x = 3y \wedge \exists x'(x' = 2y)\\ &= y > 2 \wedge x = 3y\end{aligned}$$

For G and e we choose the same invariant and variant as above. It remains to prove the validity of five implications: $4 \Rightarrow 6$, $7 \Rightarrow 8$, $5 \Rightarrow 10$, $11 \Rightarrow 15$, and $13 \Rightarrow 9$. The formula $7 \Rightarrow 8$ is trivially true (false implies everything). The other four implications are the same as derived by the Hoare calculus above.

Exercise 2

Consider the following modified if-rule:

$$\frac{\{\,F\,\}\,p\,\{\,G\,\} \quad \{\,F\,\}\,q\,\{\,G\,\}}{\{\,F\,\}\ \mathsf{if}\ e\ \mathsf{then}\ p\ \mathsf{else}\ q\ \mathsf{fi}\ \{\,G\,\}}\ (\mathrm{if}'')$$

(a) Show that the rule is admissible (for partial and total correctness).

Hint: Derive the new rule using rules that we alreay know to be admissible.

(b) Show that the Hoare calculus is no longer complete, if the regular if-rules (if) and (if′) are replaced by the rule (if″).

Hint: Find a correctness assertion that is correct (argue why it is!) but that cannot be derived in the modified calculus (explain why it can't!).

Solution

Admissibility: We show that the conclusion of the rule (if″) can be derived from its premises using the rules (if) and (lc). Since the latter rules are correct for proving total correctness, the former is also correct.

$$\dfrac{\dfrac{F \wedge e \Rightarrow F \quad \{\,F\,\}\,p\,\{\,G\,\}}{\{\,F \wedge e\,\}\,p\,\{\,G\,\}}\ (\mathrm{lc}) \qquad \dfrac{F \wedge \neg e \Rightarrow F \quad \{\,F\,\}\,q\,\{\,G\,\}}{\{\,F \wedge \neg e\,\}\,p\,\{\,G\,\}}\ (\mathrm{lc})}{\{\,F\,\}\ \mathsf{if}\ e\ \mathsf{then}\ p\ \mathsf{else}\ q\ \mathsf{fi}\ \{\,G\,\}}\ (\mathrm{if})$$

The premises $F \wedge e \Rightarrow F$ and $F \wedge \neg e \Rightarrow F$ are tautologies, hence the total correctness of $\{\,F\,\}\,p\,\{\,G\,\}$ and $\{\,F\,\}\,q\,\{\,G\,\}$ implies the total correctness of $\{\,F\,\}\ \mathsf{if}\ e\ \mathsf{then}\ p\ \mathsf{else}\ q\ \mathsf{fi}\ \{\,G\,\}$.

Incompleteness: Consider the correctness assertion

$$\{\,\mathrm{true}\,\}\ \mathsf{if}\ x = y\ \mathsf{then}\ x := x + 1\ \mathsf{else\ skip\ fi}\ \{\,x \neq y\,\}\ .$$

The assertion is partially and totally correct, since we have:

$$\begin{array}{l}
\{\,1\colon \mathrm{true}\,\}\\
\mathsf{if}\ x = y\ \mathsf{then}\\
\quad \{\,7\colon \mathrm{true} \wedge x = y\,\} \quad \mathrm{if}{\downarrow}\\
\quad \{\,6\colon x + 1 \neq y\,\} \quad \mathrm{as}{\uparrow}\\
\quad x := x + 1\\
\quad \{\,4\colon x \neq y\,\} \quad \mathrm{fi}{\uparrow}\\
\mathsf{else}\\
\quad \{\,8\colon \mathrm{true} \wedge x \neq y\,\} \quad \mathrm{if}{\downarrow}\\
\quad \{\,5\colon x \neq y\,\} \quad \mathrm{sk}{\uparrow}\\
\quad \mathsf{skip}\\
\quad \{\,3\colon x \neq y\,\} \quad \mathrm{fi}{\uparrow}\\
\mathsf{fi}\\
\{\,2\colon x \neq y\,\}
\end{array}$$

and the two implications $\text{true} \wedge x = y \Rightarrow x + 1 \neq y$ $(7 \Rightarrow 6)$ and $\text{true} \wedge x \neq y \Rightarrow x \neq y$ $(8 \Rightarrow 5)$ are valid.

In the modified calculus only two rules are applicable to the assertion above: if$''$ and lc. Rule lc weakens the pre- and strengthens the postcondition. The precondition true cannot be weakened any further; the postcondition can be strengthened to a formula G such that $G \Rightarrow x \neq y$. Then if$''$ has to be applied, resulting in the premises $\{\,\text{true}\,\}\ x := x + 1\,\{\,G\,\}$ and $\{\,\text{true}\,\}\ \mathsf{skip}\,\{\,G\,\}$, both of which are wrong and therefore cannot be derived by a correct calculus.

Summarising, the assertion above is correct, but cannot be derived in the modified Hoare calculus. Therefore the calculus is not complete.

Exercise 3

Consider the rule

$$\frac{\{\,F\,\}\,p\,\{\,G\,\}}{\{\,H \wedge F\,\}\,p\,\{\,G \wedge H\,\}}$$

where F, G, H are formulas and p is a program.

(a) Show that the rule is not admissible in general.

(b) Show that the rule is admissible, if H does not contain any variable that occurs on the lefthand side of an assignment in p. Can you think of situations where this restricted rule might be useful?

Solution

(a) To show that the rule is not admissible it suffices to find a true correctness assertion, where application of the rule leads to a false assertion.

Consider the assertion $\{\,\text{true}\,\}\ x := 0\,\{\,x = 0\,\}$, which is obviously true. If we apply the rule for $H = (x = 1)$ we obtain the assertion $\{\,x = 1\,\}\ x := 0\,\{\,x = 0 \wedge x = 1\,\}$, or equivalently $\{\,x = 1\,\}\ x := 0\,\{\,\text{false}\,\}$. This assertion is false: For each state σ such that $\sigma(x) = 1$ the program terminates in a state that clearly does not satisfy false.

(b) We first show that $\{\,H\,\}\,p\,\{\,H\,\}$ is partially correct, if the formula H does not contain any variable that occurs as lefthand side of an assignment in program p. We perform an induction on the structure of p; as induction hypothesis we assume that $\{\,H\,\}\,p'\,\{\,H\,\}$ is partially correct for all proper sub-programs p' of p.

$p = \mathsf{skip}$: $\{\,H\,\}\,\mathsf{skip}\,\{\,H\,\}$ is an instance of the skip-axiom.

$p = \mathsf{abort}$: $\{\,H\,\}\,\mathsf{abort}\,\{\,H\,\}$ is an instance of the abort-axiom.

$p = v := e$: $\{\,H\,\}\,v := e\,\{\,H\,\}$ is an instance of the axiom $\{\,F[v/e]\,\}\,p\,\{\,F\,\}$, since $H[v/e]$ simplifies to H because H does not contain v. (Note that we need here (and only here) the restriction regarding H and assignments.)

$p = p_1; p_2$: By induction hypothesis the assertions $\{ H \}\, p_1 \,\{ H \}$ and $\{ H \}\, p_2 \,\{ H \}$ are partially correct. Applying the sequential composition rule (sc) we see that $\{ H \}\, p \,\{ H \}$ is also partially correct.

$$\frac{\{ H \}\, p_1 \,\{ H \} \quad \{ H \}\, p_2 \,\{ H \}}{\{ H \}\, p_1; p_2 \,\{ H \}}\ (\text{sc})$$

$p = \mathsf{if}\ e\ \mathsf{then}\ p_1\ \mathsf{else}\ p_2\ \mathsf{fi}$: By induction hypothesis the assertions $\{ H \}\, p_1 \,\{ H \}$ and $\{ H \}\, p_2 \,\{ H \}$ are partially correct. Applying the logical consequence rule and the if-rule we obtain:

$$\frac{\dfrac{H \wedge e \Rightarrow H \quad \{ H \}\, p_1 \,\{ H \}}{\{ H \wedge e \}\, p_1 \,\{ H \}}\ (\text{lc}) \qquad \dfrac{H \wedge \neg e \Rightarrow H \quad \{ H \}\, p_2 \,\{ H \}}{\{ H \wedge \neg e \}\, p_2 \,\{ H \}}\ (\text{lc})}{\{ H \}\, \mathsf{if}\ e\ \mathsf{then}\ p_1\ \mathsf{else}\ p_2\ \mathsf{fi} \,\{ H \}}\ (\text{if})$$

The implications $(H \wedge e) \Rightarrow H$ and $(H \wedge \neg e) \Rightarrow H$ are valid, therefore $\{ H \}\, p \,\{ H \}$ is partially correct. (See also the discussion of the modified if-rule in exercise 2.)

$p = \mathsf{while}\ e\ \mathsf{do}\ p_1\ \mathsf{od}$: By induction hypothesis the assertion $\{ H \}\, p_1 \,\{ H \}$ is partially correct. Applying the logical consequence rule and the while-rule we obtain:

$$\frac{\dfrac{\dfrac{(H \wedge e) \Rightarrow H \quad \{ H \}\, p_1 \,\{ H \}}{\{ H \wedge e \}\, p_1 \,\{ H \}}\ (\text{lc})}{\{ H \}\, \mathsf{while}\ e\ \mathsf{do}\ p_1\ \mathsf{od} \,\{ H \wedge \neg e \}}\ (\text{wh}) \qquad (H \wedge \neg e) \Rightarrow H}{\{ H \}\, \mathsf{while}\ e\ \mathsf{do}\ p_1\ \mathsf{od} \,\{ H \}}\ (\text{lc})$$

The implications $(H \wedge e) \Rightarrow H$ and $(H \wedge \neg e) \Rightarrow H$ are valid, therefore $\{ H \}\, p \,\{ H \}$ is partially correct.

We now return to the original problem.

If $\{ F \}\, p \,\{ G \}$ *is partially correct, then* $\{ H \wedge F \}\, p \,\{ G \wedge H \}$ *is partially correct:*
Let σ be an $(H \wedge F)$-state, i.e., σ is an H-state as well as an F-state. Suppose $\sigma' = [p]\,\sigma$ is defined. Since $\{ F \}\, p \,\{ G \}$ is true, we have that σ' is a G-state. From above we know that $\{ H \}\, p \,\{ H \}$ is also true; hence σ' is also an H-state. Therefore σ' is a $(G \wedge H)$-state. We conclude that $\{ H \wedge F \}\, p \,\{ G \wedge H \}$ is partially correct.

If $\{ F \}\, p \,\{ G \}$ *is totally correct, then* $\{ H \wedge F \}\, p \,\{ G \wedge H \}$ *is totally correct:*
Let σ be an $(H \wedge F)$-state, i.e., σ is an H-state as well as an F-state. Since $\{ F \}\, p \,\{ G \}$ is true, we know that $\sigma' = [p]\,\sigma$ is defined and that σ' is a G-state. From above we know that $\{ H \}\, p \,\{ H \}$ is also true; hence σ' is also an H-state. Therefore σ' is a $(G \wedge H)$-state. We conclude that $\{ H \wedge F \}\, p \,\{ G \wedge H \}$ is totally correct.

What is the rule good for? This rule is useful for proving properties of program parts locally and adding the information about the global context (i.e., about the other variables not relevant for the program part under consideration) afterwards.

As an example, suppose that under the precondition F_1 the program p_1 computes some function f_1, i.e., the assertion $A_1 = \{\, F_1 \,\}\, p_1 \,\{\, z_1 = f_1(\cdots) \,\}$ is true. Another program, p_2, computes function f_2, i.e., the assertion $A_2 = \{\, F_2 \,\}\, p_2 \,\{\, z_2 = f_2(\cdots) \,\}$ is true. Due to our new rule we may prove the correctness of A_1 and A_2 separately. Afterwards we add the precondition of the second program, F_2, to assertion A_1, and the postcondition of the first program, $z_1 = f_1(\dots)$, to assertion A_2. By sequential composition we obtain a true assertion about a program computing both functions:

$$\dfrac{\dfrac{\{\, F_1 \,\}\, p_1 \,\{\, z_1 = f_1(\cdots) \,\}}{\{\, F_1 \wedge F_2 \,\}\, p_1 \,\{\, z_1 = f_1(\cdots) \wedge F_2 \,\}} \quad \dfrac{\{\, F_2 \,\}\, p_2 \,\{\, z_2 = f_2(\cdots) \,\}}{\{\, z_1 = f_1(\cdots) \wedge F_2 \,\}\, p_2 \,\{\, z_1 = f_1(\cdots) \wedge z_2 = f_2(\cdots) \,\}}}{\{\, F_1 \wedge F_2 \,\}\, p_1; p_2 \,\{\, z_1 = f_1(\cdots) \wedge z_2 = f_2(\cdots) \,\}}\ (\text{sc})$$

Exercise 4

Determine the strongest postcondition of the weakest precondition of an assignment statement, i.e., compute $\mathrm{sp}(\mathrm{wlp}(v := e, G), v := e)$. Why is it different from G?

Solution

$$\begin{aligned}\mathrm{sp}(\mathrm{wlp}(v := e, G), v := e) &= \mathrm{sp}(G[v/e], v := e) \\ &= \exists v' \,(G[v/e][v/v'] \wedge v = e[v/v'])\end{aligned}$$

$G[v/e]$ contains only those occurrences of v that are introduced by e; all other occurrences have been replaced by e. Therefore $G[v/e][v/v']$ is the same as $G[v/e[v/v']]$.

$$= \exists v' \,(G[v/e[v/v']] \wedge v = e[v/v'])$$

By the second conjunct we know that $e[v/v']$ equals v.

$$\begin{aligned}&= \exists v' \,(G[v/v] \wedge v = e[v/v']) \\ &= \exists v' \,(G \wedge v = e[v/v']) \\ &= G \wedge \exists v' \,(v = e[v/v'])\end{aligned}$$

Why is the result different from G? G may admit states that cannot be obtained by executing the assignment and therefore do not appear in the strongest postcondition. The additional existential formula ensures that the value of v is the result of evaluating the expression e for some input state. For example, the postcondition $G = \text{true}$ after the assignment $x := 2x$ is also satisfied by states assigning an odd number to x; the additional formula $\exists x'(x = 2x')$ ('x is even') excludes such states.

Exercise 5

Show that wp and wlp are dual to each other, i.e., show that $\mathrm{wlp}(p, G) = \neg\mathrm{wp}(p, \neg G)$ holds. Use this relationship to find a formula for $\mathrm{wlp}(\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}, G)$ similar to the weakest precondition in the course.

Use your formula to compute the weakest liberal precondition of the program

$$z := 0;\ \mathsf{while}\ y \neq 0\ \mathsf{do} z := z + x;\ y := y - 1\ \mathsf{od}$$

with respect to the postcondition $z = x * y_0$. Compare the result to the weakest precondition computed in the course and explain the differences.

Solution

Given a program p and a postcondition G, there are three disjoint types of states: those states, for which p does not terminate; those states, for which p terminates in a G-state; and those states, for which p terminates in a $\neg G$-state.

$$\begin{aligned}\mathcal{S} = \quad & \{\, \sigma \in \mathcal{S} \mid [p]\,\sigma \text{ undefined} \,\} \\ \dot\cup\, & \{\, \sigma \in \mathcal{S} \mid [p]\,\sigma \text{ defined and } [G]\,[p]\,\sigma = \text{true} \,\} \\ \dot\cup\, & \{\, \sigma \in \mathcal{S} \mid [p]\,\sigma \text{ defined and } [\neg G]\,[p]\,\sigma = \text{true} \,\}\end{aligned}$$

where $\dot\cup$ denotes disjoint union. The first two sets can be interpreted as the weakest liberal precondition of p with respect to G, whereas the third one is the weakest preconditon of p with respect to $\neg G$.

$$\mathcal{S} = \mathrm{wlp}(p, \{\, G \,\}) \,\dot\cup\, \mathrm{wp}(p, \{\, \neg G \,\})$$

Subtracting the set $\mathrm{wp}(p, \{\, \neg G \,\})$ on both sides results in:

$$\mathcal{S} - \mathrm{wp}(p, \{\, \neg G \,\}) = \mathrm{wlp}(p, \{\, G \,\})$$

If we represent these state sets as formulas, the complement of a set with respect to the set of all states corresponds to negation. Hence we obtain:

$$\neg\mathrm{wp}(p, \neg G) = \mathrm{wlp}(p, G)$$

i.e., wp and wlp are indeed dual operators.

Applying this relationship to while-loops gives us a formula for computing wlp for while-loops.

$$\begin{aligned}\mathrm{wlp}(\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}, G) &= \neg\mathrm{wp}(\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}, \neg G) \\ &= \neg\exists i\,(i \geq 0 \wedge F_i) \\ &= \forall i\,(i \geq 0 \Rightarrow \neg F_i) \\ \text{where} \quad F_0 &= \neg e \wedge \neg G \\ F_{i+1} &= e \wedge \mathrm{wp}(p, F_i) = e \wedge \neg\mathrm{wlp}(p, \neg F_i)\end{aligned}$$

Since F_i occurs in negated form only, we rewrite the recursion such that it defines $\neg F_i$.

$$\begin{aligned}\neg F_0 &= e \vee G\\ \neg F_{i+1} &= \neg e \vee \mathrm{wlp}(p, \neg F_i)\end{aligned}$$

After renaming $\neg F_i$ to E_i we obtain the following compact definition of the weakest liberal precondition:

$$\begin{aligned}\mathrm{wlp}(\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}, G) = \forall i \geq 0\, E_i \ &= \forall i\, (i \geq 0 \Rightarrow E_i) = \forall i\, (i < 0 \vee E_i)\\ \text{where}\quad E_0 &= e \vee G\\ E_{i+1} &= \neg e \vee \mathrm{wlp}(p, E_i)\end{aligned}$$

As an example, the weakest liberal precondition of the multiplication program can be computed as follows.

$\mathrm{wlp}(z := 0; \mathsf{while}\ y \neq 0\ \mathsf{do}\ z := z + x; y := y - 1\ \mathsf{od},\ z = xy_0)$
$= \mathrm{wlp}(z := 0,\ \mathrm{wlp}(\mathsf{while}\ y \neq 0\ \mathsf{do}\ z := z + x; y := y - 1\ \mathsf{od},\ z = xy_0))$
$= \mathrm{wlp}(z := 0,\ \forall i\, (i < 0 \vee E_i))$

$\forall i\, (i < 0 \vee E_i)$: We compute E_i for some values of i, guess $E_i = (y \neq i \vee z = x(y_0 - i))$ and prove the guess by induction.

Base case: $E_0 = (y \neq 0 \vee z = xy_0) = e \vee G$ ✓

Induction hypothesis: $E_i = (y \neq i \vee z = x(y_0 - i))$ holds.

Induction step ($i \geq 0$):

$$\begin{aligned}E_{i+1} &= \neg e \vee \mathrm{wlp}(p, E_i)\\ &= (y = 0 \vee \mathrm{wlp}(z := z + x; y := y - 1,\ (y \neq i \vee z = x(y_0 - i))))\\ &= (y = 0 \vee (y - 1) \neq i \vee (z + x) = x(y_0 - i))\\ &= (y = 0 \vee y \neq (i + 1) \vee z = x(y_0 - (i + 1)))\\ &= (y \neq (i + 1) \vee z = x(y_0 - (i + 1)))\quad \checkmark\end{aligned}$$

$$\begin{aligned}\forall i \geq 0\, (i < 0 \vee E_i) &= \forall i\, (i < 0 \vee y \neq i \vee z = x(y_0 - i))\\ &= \forall i\, (y < 0 \vee y \neq i \vee z = x(y_0 - y))\\ &= y < 0 \vee \forall i\, (y \neq i) \vee z = x(y_0 - y)\\ &= y < 0 \vee \mathrm{false} \vee z = x(y_0 - y)\\ &= y < 0 \vee z = x(y_0 - y)\end{aligned}$$

$\mathrm{wlp}(z := 0,\ \forall i\, (i < 0 \vee E_i))$
$= \mathrm{wlp}(z := 0,\ (y < 0 \vee z = x(y_0 - y)))$
$= (y < 0 \vee 0 = x(y_0 - y))$
$= (y < 0 \vee x = 0 \vee y = y_0)$

Comparison of weakest and weakest liberal precondition: Above and in the course we have computed the following wp and wlp of the multiplication program q:

$$\begin{aligned}\mathrm{wp}(q,\, z = xy_0) &= y \geq 0 \wedge (x = 0 \vee y = y_0)\\ \mathrm{wlp}(q,\, z = xy_0) &= y < 0 \vee x = 0 \vee y = y_0\\ &= y < 0 \oplus (y \geq 0 \wedge (x = 0 \vee y = y_0))\\ &= y < 0 \oplus \mathrm{wp}(q,\, z = xy_0)\end{aligned}$$

($\oplus$ denotes exclusive disjunction.) The two preconditions differ in the condition $y < 0$, which characterises exactly the states for which the result of q is undefined.

Exercise 6

Verify that the following program doubles the value of x. For which inputs does it terminate? Choose appropriate pre- and postconditions and show that the assertion is totally correct. Use $y = 2x_0 + x$ as a starting point for the invariant, where x_0 denotes the initial value of x.

$y := 3x;$
while $2x \neq y$ **do**
  $x := x + 1;$
  $y := y + 1;$
od

Solution

$\{\, 1\colon x = x_0 \wedge x \geq 0 \,\}$
$\{\, 7\colon \mathit{Inv}[y/3x] \,\}$ as↑
$y := 3x;$
$\{\, \mathit{Inv}\colon y = 2x_0 + x \wedge y \geq 2x \,\}$ wht″
while $2x \neq y$ **do**
  $\{\, 4\colon \mathit{Inv} \wedge 2x \neq y \wedge t = t_0 \,\}$ wht″
  $\{\, 9\colon (\mathit{Inv} \wedge 0 \leq t < t_0)[y/y+1][x/x+1] \,\}$ as↑
  $x := x + 1;$
  $\{\, 8\colon (\mathit{Inv} \wedge 0 \leq t < t_0)[y/y+1] \,\}$ as↑
  $y := y + 1;$
  $\{\, 5\colon \mathit{Inv} \wedge 0 \leq t < t_0 \,\}$ wht″
od
$\{\, 6\colon \mathit{Inv} \wedge 2x = y \,\}$ wht″
$\{\, 2\colon x = 2x_0 \,\}$

We choose $y = 2x_0 + x \wedge 2x \leq y$ as invariant Inv and $y - 2x$ as variant t. It remains to

show that the three implications $1 \Rightarrow 7$, $4 \Rightarrow 9$, and $6 \Rightarrow 2$ are valid.

$$
\begin{aligned}
& 1 \Rightarrow 7 \\
& x = x_0 \wedge x \geq 0 \Rightarrow \mathit{Inv}[y/3x] \\
& x = x_0 \wedge x \geq 0 \Rightarrow 3x = 2x_0 + x \wedge 2x \leq 3x
\end{aligned}
$$

$3x = 2x_0 + x$ holds because of the first premise and $2x \leq 3x$ because of the second one.

$$
\begin{aligned}
& 4 \Rightarrow 9 \\
& \mathit{Inv} \wedge 2x \neq y \wedge t = t_0 \Rightarrow (\mathit{Inv} \wedge 0 \leq t < t_0)[y/y+1][x/x+1] \\
& \mathit{Inv} \wedge 2x \neq y \wedge t = t_0 \Rightarrow (y = 2x_0 + x \wedge 2x \leq y \wedge 0 \leq y - 2x < t_0)[y/y+1][x/x+1] \\
& \mathit{Inv} \wedge 2x \neq y \wedge t = t_0 \Rightarrow y + 1 = 2x_0 + x + 1 \wedge 2(x+1) \leq y + 1 \wedge 0 \leq y + 1 - 2(x+1) < t_0 \\
& \mathit{Inv} \wedge 2x \neq y \wedge t = t_0 \Rightarrow y = 2x_0 + x \wedge 2x + 1 \leq y \wedge 0 \leq y - 2x - 1 < t_0
\end{aligned}
$$

$y = 2x_0 + x$ is part of the invariant (first premise). $2x + 1 \leq y$ holds since $2x \leq y$ is part of the invariant (first premise) and $2x \neq y$ holds because of the second premise. $0 \leq y - 2x - 1$ is the same as $2x + 1 \leq y$, which we just showed to be true. $y - 2x - 1 < t_0$ is true since t_0 is the same as $y - 2x$ (third premise) and $t_0 - 1$ is obviously smaller than t_0.

$$
\begin{aligned}
& 6 \Rightarrow 2 \\
& \mathit{Inv} \wedge 2x = y \Rightarrow x = 2x_0
\end{aligned}
$$

Solving the two equalities $y = 2x_0 + x$ (from Inv) and $2x = y$ for x we obtain the conclusion $x = 2x_0$.

Exercise 7

Show that the following correctness assertion is totally correct. Describe the function computed by the program if we consider a as its input and c as its output.

$\{\, 1 \colon a \geq 0 \,\}$
$b := 1;$
$c := 0;$
$\{\, \mathit{Inv} \colon b = (c+1)^3 \wedge 0 \leq c^3 \leq a \,\}$
while $b \leq a$ **do**
  $d := 3 * c + 6;$
  $c := c + 1;$
  $b := b + c * d + 1$
od
$\{\, 2 \colon c^3 \leq a < (c+1)^3 \,\}$

Solution

$\{\, 1\colon a \geq 0 \,\}$
$\{\, 7\colon \mathit{Inv}[c/0][b/1] \,\}$ as↑
$b := 1;$
$\{\, 6\colon \mathit{Inv}[c/0] \,\}$ as↑
$c := 0;$
$\{\, \mathit{Inv}\colon b = (c+1)^3 \wedge 0 \leq c^3 \leq a \,\}$ wht‴
while $b \leq a$ **do**
  $\{\, 3\colon \mathit{Inv} \wedge b \leq a \wedge 0 \leq t = t_0 \,\}$ wht‴
  $\{\, 10\colon \mathit{Inv} \wedge (b \leq a \Rightarrow 0 \leq t < t_0)[b/b+cd+1][c/c+1][d/3c+6] \,\}$ as↑
  $d := 3 * c + 6;$
  $\{\, 9\colon \mathit{Inv} \wedge (b \leq a \Rightarrow 0 \leq t < t_0)[b/b+cd+1][c/c+1] \,\}$ as↑
  $c := c + 1;$
  $\{\, 8\colon \mathit{Inv} \wedge (b \leq a \Rightarrow 0 \leq t < t_0)[b/b+cd+1] \,\}$ as↑
  $b := b + c * d + 1$
  $\{\, 4\colon \mathit{Inv} \wedge (b \leq a \Rightarrow 0 \leq t < t_0) \,\}$ wht‴
od
$\{\, 5\colon \mathit{Inv} \wedge a < b \,\}$ wht‴
$\{\, 2\colon c^3 \leq a < (c+1)^3 \,\}$

Proof of the implications:

$1 \Rightarrow 7$**:** $a \geq 0$ obviously implies $\mathit{Inv}[c/0][b/1] = (1 = (0+1)^3 \wedge 0 \leq 0^3 \leq a)$

$5 \Rightarrow 2$**:** The third conjunct of the invariant, $c^3 \leq a$, and the negated loop condition together yield $c^3 \leq a < b$. Using $b = (c+1)^3$ we obtain formula 2.

We split implication $3 \Rightarrow 10$ into two parts, one for partial correctness and one for termination:

$$\mathit{Inv} \wedge b \leq a \Rightarrow \mathit{Inv}[b/b+cd+1][c/c+1][d/3c+6]$$
$$\mathit{Inv} \wedge b \leq a \wedge 0 \leq t = t_0 \Rightarrow (b \leq a \Rightarrow 0 \leq t < t_0)[b/b+cd+1][c/c+1][d/3c+6]$$

Partial correctness: We start by simplifying the right-hand side.

$$\begin{aligned}
&\mathit{Inv}[b/b+cd+1][c/c+1][d/3c+6] \\
&= (b = (c+1)^3 \wedge 0 \leq c^3 \leq a)[b/b+cd+1][c/c+1][d/3c+6] \\
&= (b + (c+1)(3c+6) + 1 = (c+2)^3 \wedge 0 \leq (c+1)^3 \leq a) \\
&= (b + 3c^2 + 9c + 6 + 1 = ((c+1)+1)^3 \wedge 0 \leq (c+1)^3 \leq a) \\
&= (b + 3c^2 + 9c + 7 = (c+1)^3 + 3(c+1)^2 + 3(c+1) + 1 \wedge 0 \leq (c+1)^3 \leq a) \\
&= (b + 3c^2 + 9c + 7 = (c+1)^3 + 3c^2 + 6c + 3 + 3c + 3 + 1 \wedge 0 \leq (c+1)^3 \leq a) \\
&= (b = (c+1)^3 \wedge 0 \leq (c+1)^3 \leq a)
\end{aligned}$$

So we have to prove the implication

$$b = (c+1)^3 \wedge 0 \leq c^3 \leq a \wedge b \leq a \Rightarrow b = (c+1)^3 \wedge 0 \leq (c+1)^3 \leq a$$

We consider the conjuncts on the right-hand side in turn:

- $b = (c+1)^3$: occurs also on the left-hand side.
- $0 \leq (c+1)^3$: follows from $0 \leq c^3$.
- $(c+1)^3 \leq a$: follows from $b \leq a$ and $b = (c+1)^3$.

Termination: As variant t we choose $a-b$ (alternatively: $a-c^3$). We start by simplifying the right-hand side.

$$\begin{aligned}&(b \leq a \Rightarrow 0 \leq a-b < t_0)[b/b+cd+1][c/c+1][d/3c+6]\\ &= ((b \leq a \Rightarrow 0 \leq a-b) \wedge (b \leq a \Rightarrow a-b < t_0))[b/b+cd+1][c/c+1][d/3c+6]\end{aligned}$$

The first implication is obviously true, it remains to prove the second one. In fact, we prove a stronger conclusion (without the assumption $(b \leq a)[b/\cdots][c/\cdots][d/\cdots]$):

$$\begin{aligned}&a-b < t_0[b/b+cd+1][c/c+1][d/3c+6]\\ &= a-(b+(c+1))(3c+6)+1) < t_0\end{aligned}$$

So we have to prove the implication

$$Inv \wedge b \leq a \wedge 0 \leq a-b = t_0 \Rightarrow a-(b+(c+1)(3c+6)+1) < t_0$$

Using $a-b = t_0$ the right-hand side becomes

$$a-(b+(c+1))(3c+6)+1) < a-b$$

and further

$$(b+(c+1))(3c+6)+1 > b$$

Because of *Inv* both, b and c, are non-negative, hence the inequality holds.

Function computed by the program: Integer cubic root, $i = \lfloor \sqrt[3]{a} \rfloor$.

Exercise 8

Prove that the rule

$$\frac{\{\, Inv \wedge e \,\}\, p \,\{\, Inv \,\}}{\{\, Inv \,\}\ \mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}\ \{\, Inv \wedge \neg e \,\}}\ \text{(wh)}$$

is correct regarding partial correctness, i.e., show that $\{\, Inv \,\}\ \mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}\ \{\, Inv \wedge \neg e \,\}$ is partially correct whenever $\{\, Inv \wedge e \,\}\, p \,\{\, Inv \,\}$ is partially correct.

Solution

The assertion $\{\, Inv \,\}\, \mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}\, \{\, Inv \wedge \neg e \,\}$ is partially correct, if we can show that

> For all states σ:
> If $[Inv]\,\sigma = \text{true}$ and $\tau = [\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}]\,\sigma$ is defined
> then $[Inv \wedge \neg e]\,\tau = \text{true}$.

The natural semantics of while is specified recursively as

$$[\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}]\,\sigma = \begin{cases} [\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}]\,[p]\,\sigma & \text{if } [e]\,\sigma \neq 0 \\ \sigma & \text{if } [e]\,\sigma = 0 \end{cases}$$

Therefore τ being defined means that there is a number n (the number of loop iterations) such that $\tau = [\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}]\,\sigma = [p]^{\,n}\sigma$. The statement thus can be written as

> For all states σ:
> If $[Inv]\,\sigma = \text{true}$ and if $\tau = [\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}]\,\sigma = [p]^{\,n}\sigma$ holds for some $n \geq 0$
> then $[Inv \wedge \neg e]\,\tau = \text{true}$.

which is the same as

> For all numbers $n \geq 0$:
> $A(n) =$ For all states σ:
> If $[Inv]\,\sigma = \text{true}$ and $\tau = [\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}]\,\sigma = [p]^{\,n}\sigma$
> then $[Inv \wedge \neg e]\,\tau = \text{true}$.

We prove the universal statement "$A(n)$ for all n" by induction on n.

Induction base: We prove $A(0)$. Let σ be a state such that $[Inv]\,\sigma = \text{true}$ and $\tau = [\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}]\,\sigma = [p]^{\,0}\sigma = \sigma$. This is the case if $[e]\,\tau = \text{false}$. Together with $[Inv]\,\sigma = [Inv]\,\tau$ we obtain $[Inv \wedge \neg e]\,\tau = \text{true}$.

Induction step: We show $A(n+1)$ under the assumption that $A(n)$ holds.

Let σ be a state such that $[Inv]\,\sigma = \text{true}$ and let $\tau = [\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}]\,\sigma = [p]^{\,n+1}\sigma$ be the state after $n+1$ iterations. Since there is at least one iteration, we have $[e]\,\sigma = \text{true}$, i.e., $[Inv \wedge e]\,\sigma = \text{true}$. Moreover the state $\sigma' = [p]\,\sigma$ after the first iteration is defined.

By assumption, the premise $\{\, Inv \wedge e \,\}\, p\, \{\, Inv \,\}$ is partially correct, therefore $[Inv \wedge e]\,\sigma = \text{true}$ and $\sigma' = [p]\,\sigma$ being defined implies $[Inv]\,\sigma' = \text{true}$.

Now we can apply the induction hypothesis. σ' is a state such that $[Inv]\,\sigma' = \text{true}$ and $\tau = [\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}]\,\sigma' = [p]^{\,n}\sigma'$ is the state after n further iterations. Therefore, by $A(n)$, we obtain $[Inv \wedge \neg e]\,\tau = \text{true}$.