Deductive Verification of Software
Examples discussed on 26 Nov 2013

(6.0 VU Formal Methods in Computer Science)

Gernot Salzer

WS 2013

Exercise 1

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$G[v/e]$ contains only those occurrences of v that are introduced by e; all other occurrences have been replaced by e. Therefore $G[v/e][v/v']$ is the same as $G[v/e[v/v']]$.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 2

Show that the following assertion is totally correct. Describe the function computed by the program.

$\{\, Pre\colon x \geq 1 \wedge y \geq 2 \,\}$
$u := y;$
$z := 0;$
while $u \leq x$ do
  $u := u * y;$
  $z := z + 1$
od
$\{\, Post\colon y^z \leq x < y^{z+1} \,\}$

Hint: Use the invariant $Inv\colon u = y^{z+1} \wedge y \leq u \leq xy \wedge y \geq 2$.

Solution

We start by adding further assertions to the program, according to the rules of the annotation calculus. The formulas are numbered in to the order in which they are added.

$\{\, Pre\colon x \geq 1 \wedge y \geq 2 \,\}$
$\{\, F_6\colon Inv[z/0][u/y] \,\}$
$u := y;$
$\{\, F_5\colon Inv[z/0] \,\}$
$z := 0;$
$\{\, F_1\colon Inv \,\}$
while $u \leq x$ do
  $\{\, F_2\colon Inv \wedge u \leq x \wedge t = t_0 \,\}$
  $\{\, F_8\colon (Inv \wedge (u \leq x \Rightarrow 0 \leq t < t_0))[z/z+1][u/uy] \,\}$
  $u := u * y;$
  $\{\, F_7\colon (Inv \wedge (u \leq x \Rightarrow 0 \leq t < t_0))[z/z+1] \,\}$
  $z := z + 1$
  $\{\, F_3\colon Inv \wedge (u \leq x \Rightarrow 0 \leq t < t_0) \,\}$
od
$\{\, F_4\colon Inv \wedge \neg(u \leq x) \,\}$
$\{\, Post\colon y^z \leq x < y^{z+1} \,\}$

It remains to show the validity of the three implications $Pre \Rightarrow F_6$, $F_4 \Rightarrow Post$, and $F_2 \Rightarrow F_8$.

***Pre* $\Rightarrow$ F_6:**

$$(x \geq 1 \wedge y \geq 2) \Rightarrow Inv[z/0][u/y]$$
$$(x \geq 1 \wedge y \geq 2) \Rightarrow (y = y^{0+1} \wedge y \leq y \leq xy \wedge y \geq 2)$$

We have to show that each conjunct in the conclusion is true (under the assumption that the premises are true).

conclusion	why it holds
$y = y^{0+1}$	Equivalent to the valid formula $y = y$
$y \leq y$	Valid (reflexivity of $\leq$)
$y \leq xy$	Since y is positive (premise $y \geq 2$), multiplying the premise $x \geq 1$ by y yields the conclusion $y \leq xy$.
$y \geq 2$	Is one of the premises.

$F_4 \Rightarrow Post$:

$$Inv \wedge \neg(u \leq x) \Rightarrow Post$$
$$(u = y^{z+1} \wedge y \leq u \leq xy \wedge y \geq 2 \wedge u > x) \Rightarrow y^z \leq x < y^{z+1}$$

conclusion	why it holds
$y^z \leq x$	From the premises $u = y^{z+1}$ and $u \leq xy$ we obtain $y^{z+1} \leq xy$. Since y is positive (premise $y \geq 2$), dividing the inequality by y yields the conclusion $y^z \leq x$.
$x < y^{z+1}$	Follows from the premises $u = y^{z+1}$ and $u > x$.

$F_2 \Rightarrow F_8$:

$$(Inv \wedge u \leq x \wedge t = t_0) \Rightarrow (Inv \wedge (u \leq x \Rightarrow 0 \leq t < t_0))[z/z+1][u/uy]$$
$$(Inv \wedge u \leq x \wedge t = t_0) \Rightarrow (Inv[z/z+1][u/uy] \wedge (uy \leq x \Rightarrow 0 \leq t[z/z+1][u/uy] < t_0))$$

Proving an implication $F \Rightarrow (G \wedge H)$ is equivalent to proving the implications $F \Rightarrow G$ and $F \Rightarrow H$ separately. Here the first implication corresponds to partial correctness and the second one to termination.

Partial correctness: We omit the premise $t = t_0$ as it cannot contribute anything to the proof; it just states that some term t equals a variable t_0 that doesn't occur anywhere else in the formula.

$$(Inv \wedge u \leq x) \Rightarrow Inv[z/z+1][u/uy]$$
$$(u = y^{z+1} \wedge y \leq u \leq xy \wedge y \geq 2 \wedge u \leq x) \Rightarrow (uy = y^{z+2} \wedge y \leq uy \leq xy \wedge y \geq 2)$$

conclusion	why it holds
$uy = y^{z+2}$	Obtained by multiplying the premise $u = y^{z+1}$ by y.
$y \leq uy$	Since y is positive (premise $y \geq 2$), its product with another positive number (the premises $y \leq u$ and $y \geq 2$ imply $u \geq 2$) is greater than or equal to y.
$uy \leq xy$	Since y is positive (premise $y \geq 2$), multiplying the premise $u \leq x$ by y yields the conclusion $uy \leq xy$.
$y \geq 2$	Is one of the premises.

Termination:

$$\begin{aligned}(\mathit{Inv} \wedge u \le x \wedge t = t_0) &\Rightarrow (uy \le x \Rightarrow 0 \le t[z/z+1][u/uy] < t_0)\\ (\mathit{Inv} \wedge u \le x) &\Rightarrow (uy \le x \Rightarrow 0 \le t[z/z+1][u/uy] < t)\\ (\mathit{Inv} \wedge u \le x \wedge uy \le x) &\Rightarrow (0 \le t[z/z+1][u/uy] < t)\end{aligned}$$

In the first step we use the equation $t = t_0$ to eliminate t_0 on the righthand side; after that we don't need the equation anymore and omit it. In the second step we use the fact that $F \Rightarrow (G \Rightarrow H)$ is equivalent to $(F \wedge G) \Rightarrow H$.

To guess a suitable variant t we rewrite the loop condition $u \le x$ to $x - u \ge 0$ and choose $t \stackrel{\text{def}}{=} x - u$. The loop condition ensures that $t \ge 0$ holds within the loop; moreover, since the loop increases u we may expect that t decreases. It remains to prove these two properties formally by showing the validity of the implication.

$$\begin{aligned}(\mathit{Inv} \wedge u \le x \wedge uy \le x) &\Rightarrow (0 \le t[z/z+1][u/uy] < t)\\ (u = y^{z+1} \wedge y \le u \le xy \wedge y \ge 2 \wedge u \le x \wedge uy \le x) &\Rightarrow (0 \le x - uy < x - u)\end{aligned}$$

conclusion	why it holds
$0 \le x - uy$	Equivalent to the premise $uy \le x$.
$x - uy < x - u$	This inequality is equivalent to $u < uy$. Since u is positive (the premises $y \le u$ and $y \ge 2$ imply $u \ge 2$) and y is greater than one (premise $y \ge 2$), the product uy is strictly greater than u.

Therefore the given assertion is totally correct.

Function computed by the program: To see what the program does it suffices to analyse the postcondition and to express z as a function of x and y.

$$\begin{aligned}&y^z \le x < y^{z+1}\\ &z \le \log_y(x) < z + 1 && \text{(Take the logarithm to base } y.)\\ &z = \lfloor \log_y(x) \rfloor && (a = \lfloor b \rfloor \text{ is equivalent to } a \le b < a + 1.)\end{aligned}$$

Thus the program computes the integer logarithm to base y of x.

Exercise 3

Let p be the program $\mathsf{while}\ i > j\ \mathsf{do}\ i := j/2;\ j := i - 2\ \mathsf{od}$. For each of the four correctness assertions $\{\,\phi\,\}\,p\,\{\,\mathrm{true}\,\}$, $\{\,\phi\,\}\,p\,\{\,\mathrm{false}\,\}$, $\{\,\mathrm{true}\,\}\,p\,\{\,\phi\,\}$, and $\{\,\mathrm{false}\,\}\,p\,\{\,\phi\,\}$ find formulas ϕ that are neither equivalent to true nor to false such that the assertion is partially correct, partially but not totally correct, totally correct, or totally but not partially correct. In total, these may be up to 16 formulas. Note that in some cases the required formula ϕ may not exist.

Solution

$p \equiv$ **while** $i > j$ **do** $i := j/2;\ j := i - 2$ **od** $\quad \phi \neq \text{true}, \text{false}$

	partially correct	partially but not totally cor.	totally correct	totally but not partially cor.
$\{\,\phi\,\}\,p\,\{\,\text{true}\,\}$	any formula except those equivalent to true or false	e.g. $i > j$ or $i \neq j$	e.g. $i \leq j$	does not exist
$\{\,\phi\,\}\,p\,\{\,\text{false}\,\}$	e.g. $i > j$	e.g. $i > j$	does not exist	does not exist
$\{\,\text{true}\,\}\,p\,\{\,\phi\,\}$	e.g. $i \leq j$	$i \leq j$	does not exist	does not exist
$\{\,\text{false}\,\}\,p\,\{\,\phi\,\}$	any formula except those equivalent to true or false	does not exist	any formula except those equivalent to true or false	does not exist

Exercise 4

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 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.

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

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

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, $z_1 = f_1(\ldots)$, to assertion A_2. By sequential composition we obtain a true assertion about a program computing both functions:

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