Formal Verification of Software – Exercises

Harald Glanzer Bernd-Peter Ivanschitz Lukas Petermann

May 2012

Exercise 1 (1 point) Show that the givenTPL program is syntactically correct:
$x := x + y; if\ x < 0\ then\ abort; else\ while\ x \neq y\ do\ x := x + 1; y := y + 2; od\ fi$

- $\mathcal{P}$
- $\Rightarrow \mathcal{V} := \mathcal{E}$
- $\Rightarrow x := (\mathcal{E}\mathcal{B}\mathcal{E})$
- $\Rightarrow x := (\mathcal{V} + \mathcal{V})$
- $\Rightarrow$ x:=x + y;
- $\Rightarrow$ x:=x + y; if $\mathcal{E}$ then $\mathcal{P}$ else $\mathcal{P}$ fi
- $\Rightarrow$ x:=x + y; if $(\mathcal{E}\mathcal{B}\mathcal{E})$ then $\mathcal{P}$ else $\mathcal{Q}$ if
- $\Rightarrow$ x:=x + y; if $\mathcal{V} < \mathcal{N}$ then $\mathcal{P}$ else $\mathcal{Q}$ if
- $\Rightarrow$ x:=x + y; if x $<$ 0 then $\mathcal{P}$ else $\mathcal{Q}$ if
- $\Rightarrow$ x:=x + y; if x $<$ 0 then abort; else $\mathcal{Q}$ if
- $\Rightarrow$ x:=x + y; if x $<$ 0 then abort; else while $\mathcal{E}$ do $\mathcal{P}$ od if
- $\Rightarrow$ x:=x + y; if x $<$ 0 then abort; else while $(\mathcal{E}\mathcal{B}\mathcal{E})$ do $\mathcal{P}$ od if
- $\Rightarrow$ x:=x + y; if x $<$ 0 then abort; else while $(\mathcal{E} \neq \mathcal{E})$ do $\mathcal{P}$ od if
- $\Rightarrow$ x:=x + y; if x $<$ 0 then abort; else while $(\mathcal{V} \neq \mathcal{E})$ do $\mathcal{P}$ od if
- $\Rightarrow$ x:=x + y; if x $<$ 0 then abort; else while $(\mathcal{V} \neq \mathcal{V})$ do $\mathcal{P}$ od if
- $\Rightarrow$ x:=x + y; if x $<$ 0 then abort; else while x$\neq$y do $\mathcal{P}; \mathcal{P}$ od if
- $\Rightarrow$ x:=x + y; if x $<$ 0 then abort; else while x$\neq$y do $\mathcal{E}; \mathcal{P}$ od if
- $\Rightarrow$ x:=x + y; if x $<$ 0 then abort; else while x$\neq$y do $(\mathcal{E}\mathcal{B}\mathcal{E}); \mathcal{P}$ od if

- $\Rightarrow$ x:=x + y; if x < 0 then abort; else while x≠y do $(\mathcal{V}+\mathcal{N});\mathcal{P}$ od if
- $\Rightarrow$ x:=x + y; if x < 0 then abort; else while x≠y do x + 1;$\mathcal{P}$; od if
- $\Rightarrow$ x:=x + y; if x < 0 then abort; else while x≠y do x + 1;$\mathcal{E}$; od if
- $\Rightarrow$ x:=x + y; if x < 0 then abort; else while x≠y do x + 1;$(\mathcal{EBE})$; od if
- $\Rightarrow$ x:=x + y; if x < 0 then abort; else while x≠y do x + 1;$(\mathcal{V}+\mathcal{N})$; od if
- $\Rightarrow$ x:=x + y; if x < 0 then abort; else while x≠y do x + 1; y + 2; od if

Here the idea is to construct the wanted(given) program by starting with with an 'empty' program and extending this program by substitution until we get the final program.

Exercise 2 (1 point) Let σ be a state satisfying $\sigma(x) = \sigma(y) = 1$, and let p be the program given in exercise 3. Compute $[p]\,\sigma$, using

(a) the structural operational semantics

FIXME!!!! TODO

(b) the natural semantics

- $p[\sigma] = [x := x + y; if\ x < 0\ then\ abort; else\ while\ x \neq y\ do\ x := x + 1; y := y + 2; od\ fi]\sigma$

  Regel: $[p;q]\sigma = [q][p]\sigma$

  $[x := x + y; if...]\sigma = [if...][x := x + y]\sigma$

  $\sigma : x \mapsto 1, y \mapsto 1$
- $= [if\ x < 0\ then\ abort; else\ while\ x \neq y\ do\ x := x + 1; y := y + 2; od\ fi]\sigma_1$

  Regel: $[if\ e\ then\ p\ else\ q\ fi]\sigma = \begin{cases} [p]\sigma, & if[e]\sigma \neq 0 \\ [q]\sigma, & if[e]\sigma = 0 \end{cases}$

  $\sigma_1 : x \mapsto 2, y \mapsto 1$

  $[x < 0]\sigma_1 = 0(false)$
- $= [while\ x \neq y\ do\ x := x + 1; y := y + 2; od]\sigma_2 =$

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

  $\sigma_2 : x \mapsto 2, y \mapsto 1$

  $[x \neq y] = 1(true)$

  $= [while\ x \neq y...][y := y + 2; x := x + 1]\sigma_2$

  $\sigma_3 : x \mapsto [x + 1]\sigma_2 = 3, y \mapsto 1$

- $= [while\ x \neq y...][y := y+2]\sigma_3$

  $\sigma_4 : x \mapsto 3, y \mapsto [y+2]\sigma_3 = 3$
- $= [while\ x \neq y\ do...od]\sigma_4$

  $\sigma_4 : x \mapsto 3, y \mapsto = 3$

  $[x \neq y] = 0(false)$
- $= \sigma_4$

of TPL.

Exercise 3 (1 point) 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
```

Show that $\{\, x = 2y \wedge y > 2 \,\}\, p \,\{\, x = y \,\}$ is totally correct by computing the weakest precondition of the program.
We search for the weakest precondition witch satisfies:
$Wp(p, S_{out})p(S_{out})$

- $wp(x := x+y; if\ x < 0\ then\ abort; else\ while\ x \neq y\ do\ x := x+1; y := y+2; od$ fi,x=y)
- $= wp(x := x+y; wp(if\ x < 0\ then...fi$,x=y))
- = wp($x := x+y$; $(x < 0 \wedge wp(abort)$,$x = y$)$\vee$ $(x \geq y \wedge wp(while...., x = y)))$
- = wp($x := x+y$; $(x < 0 \wedge FALSE$,$x = y$)$\vee$ $(x \geq y \wedge$ (*)$wp(while...., x = y)))$

- (*) = (while $x \neq y$ do $x := x+1; y := y+2;$ od , x=y)
- $\rightarrow F_1 = (x = y \wedge x = y)$
- $\rightarrow F_2 = (x \neq y \wedge wp(x := x+1; y := y+2, F_1)$=$(x \neq y \wedge wp(x := x+1$ $wp(y := y+2, F_1))$
  $= (x \neq y \wedge x = (y+2) - 1) = (x \neq y \wedge x = (y+1)$
  guess:

- $\rightarrow F_i = (x \neq y \wedge wp(x := x+1; y := y+2, F_{i-1})$ $=(x \neq y \wedge wp(x := x+1$ $wp(y := y+2, F_{i-1}))$
  $= (x \neq y \wedge x = (y+i))$
- $\rightarrow F_{i+1} = (x \neq y \wedge wp(x := x+1; y := y+2, F_i)$ $=(x \neq y \wedge wp(x := x+1$ $wp(y := y+2, F_i))$
  $= (x \neq y \wedge x+1 = (y+i+2)) = (x \neq y \wedge x = (y+i+1))$
  $\rightarrow wp(while...) = \exists i((i \geq 0) \wedge x = y+i) = ((i \geq 0) \wedge x - y = 1) = x - y \geq 0$

- = wp($x := x+y$; $(x < 0 \wedge FALSE$,$x = y$ $)\vee$ $(x \geq y \wedge (*)wp(while...., x = y)))$
- = wp($x := x+y$; wp(while...),$x = y$)
- $= (x := x+y \wedge (x+y) - y \geq 0)$
- $= (x := x+y \wedge x \geq 0)$ = Weakest Precondition

Exercise 4 (1 point) Let p be the program given in exercise 3. Use the Hoare calculus to show that

$$\{\, x = 2y \wedge y > 2 \,\}\, p \,\{\, x = y \,\}$$

is totally correct.

Exercise 5 (1 point) Extend our toy language by statements of the form "assert e". When the condition e evaluates to true, the program continues, otherwise the program aborts.

Specify the syntax and semantics of the extended language. Determine the weakest precondition, the weakest liberal precondition, the strongest postcondition, and Hoare rules (partial and total correctness) for assert-statements. Show that they are correct.

Treat the assert-statement as a first-class citizen, i.e., do not refer to other program statements in the final result. However, you may use other statements as intermediate steps when deriving the rules.

Exercise 6 (1 point) 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 ≠ y do
   x := x + 1;
   y := y + 1;
od
```

To prove total correctness we must show that the program is partitial correct and that it terminates.

After using some testvalues for x it is expected that the program terminates and seems to give the correct result only for inputvalues greater than 0, so the precondition is

$$x > 0$$

We want to prove that the program takes the input value of x and doubles this value, which is greater than zero(see precondition), so the postcondition is

$$x = 2 * x_0 \bigwedge x > 0 \bigwedge x > x_0$$

The invariant is assumed to be

$$INV = 2 * x_0 + x \bigwedge x_0 > 0$$

$$\{\, F \,\}\, y := 3 * x \mathsf{while}\ e\ do\ x := x + 1; y := y + 1\ od \,\{\, G \,\}$$

$$\dfrac{\{\, F \,\}\, y := 3 * x \,\{\, Inv \,\} \quad \{\, Inv \,\}\, \mathsf{while}\ e\ do ... od \,\{\, G \,\}}{\{\, F \,\}\, y := 3 * x\ \mathsf{while}\ e\ do\ x := x + 1; y := y + 1\ od \,\{\, G \,\}}\ \text{(wh)}$$

$$\dfrac{\overset{(1)}{F \Rightarrow Inv} \qquad \dfrac{\dfrac{\dfrac{\dfrac{Inv \wedge e \wedge e' \overset{(2)}{\Rightarrow} Inv[x/x+1] \quad \overset{\text{(sk)}}{\{\, Inv[x/x+1] \,\}\, \mathsf{skip} \,\{\, Inv[x/x+1] \,\}}}{\{\, Inv \wedge e \wedge e' \,\}\, \mathsf{skip} \,\{\, Inv[x/x+1] \,\}}\ \text{(lc)} \quad \overset{\text{(ab)}}{\{\, Inv \wedge e \wedge \neg e' \,\}\, \mathsf{abort} \,\{\, Inv[x/x+1] \,\}}}{\{\, Inv \wedge e \,\}\, \mathsf{if}\ e'\ \mathsf{then\ skip\ else\ abort\ fi} \,\{\, Inv[x/x+1] \,\}}\ \text{(if)} \quad \{\, I}{\{\, Inv \wedge e \,\}\, \mathsf{if}\ e'\ \mathsf{then\ skip\ else\ abort\ fi};\ x := x + 1 \,\{\, Inv}}{\{\, Inv \,\}\, \mathsf{while}\ e\ \mathsf{do\ if}\ e'\ \mathsf{then\ skip\ else\ abort\ fi};\ x := x + 1\ \mathsf{od} \,\{}}{\{\, F \,\}\, \mathsf{while}\ e\ \mathsf{do\ if}\ e'\ \mathsf{then\ skip\ else\ abort\ fi};\ x := x + 1\ \mathsf{od} \,\{\, G \,\}}$$

$$\dfrac{\overset{(1)}{F \Rightarrow Inv} \qquad \dfrac{\dfrac{\dfrac{\dfrac{Inv \wedge e \wedge e' \overset{(2)}{\Rightarrow} Inv[x/x+1] \quad \overset{\text{(sk)}}{\{\, Inv[x/x+1] \,\}\, \mathsf{skip} \,\{\, Inv[x/x+1] \,\}}}{\{\, Inv \wedge e \wedge e' \,\}\, \mathsf{skip} \,\{\, Inv[x/x+1] \,\}}\ \text{(lc)} \quad \overset{\text{(ab)}}{\{\, Inv \wedge e \wedge \neg e' \,\}\, \mathsf{abort} \,\{\, Inv[x/x+1] \,\}}}{\{\, Inv \wedge e \,\}\, \mathsf{if}\ e'\ \mathsf{then\ skip\ else\ abort\ fi} \,\{\, Inv[x/x+1] \,\}}\ \text{(if)} \quad \{\, I}{\{\, Inv \wedge e \,\}\, \mathsf{if}\ e'\ \mathsf{then\ skip\ else\ abort\ fi};\ x := x + 1 \,\{\, Inv}}{\{\, Inv \,\}\, \mathsf{while}\ e\ \mathsf{do\ if}\ e'\ \mathsf{then\ skip\ else\ abort\ fi};\ x := x + 1\ \mathsf{od} \,\{}}{\{\, F \,\}\, \mathsf{while}\ e\ \mathsf{do\ if}\ e'\ \mathsf{then\ skip\ else\ abort\ fi};\ x := x + 1\ \mathsf{od} \,\{\, G \,\}}$$

Exercise 7 (1 point) 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.

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

Exercise 8 (1 point) 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.

Exercise 9 (2 points) Determine the weakest liberal precondition of while-loops, i.e., find a formula equivalent to $\mathrm{wlp}(\mathsf{while}\ e\ \mathsf{do}\ p\ \mathsf{od}, G)$ 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.