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:

```
1 $x := x + y$;
2 if $x < 0$ then
3  abort
4 else
5  while $x \neq y$ do
6     $x := x + 1$;
7     $y := y + 2$
8  od
9 fi
```

To prove the progam for correct syntax, every line of the given code must be examined to show that every statement is part of our programming language. The TPL syntax was introduced in the first lecture, see the corresping pdf, pages 12 and 13.

- Line 1: $x := x + y$;
  Assignment: $\nu ::= \epsilon, \nu ::= variable|integer|binary\ expression|unary\ expression$

  Left side of an assignment must be a variable, which must not be a keyword, which is not the case, x is not reserved, as well as y. Right side must be an expression ϵ, which can itself be another variable, an integer or one or two expressions, connected with an uniary or binary operator.

- Line 2, 4, 9: if $x < 0$ then ... else ... fi
  If-Then-Else - Statement: must start with the keyword IF, followed by an expression ϵ. In this case, the expression consists of 2 variables with a binary operator, as defined recursevly in the lecture notes. After this expression another keyword, $THEN$, must occur, which is the case. A programstatement must follow afterwards(will be examined in the next steps), until another keyword, $ELSE$, occurs. Afterwards, another programstatement follows, until the construct is closed by the final keyword FI.

- Line 3: abort
  valid keyword, which terminates the actual program and signals an error condition

- Line 5,8: while $x \neq y$ do ... od
  keyword $WHILE$, followed by a valid binary expression, followd by keyword DO, followed by a program statement, terminated by keyword OD

- Line 6: $x := x + 1;$
  valid assignment, left side is a variable(no keyword allowed), right side consists of a binary expression, in this case a variable, an integer and the defined operator $+$, which is a valid expression as defined in a recursive manner on page 12 in the lecture notes

- Line 7: $y := y + 2;$
  same as Line 6

Because the program only contains valid expressions that are all well-defined, it belongs to the family of TPL - programs.

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

(b) the natural semantics

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

  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: $[if\ e\ then\ p\ else\ q\ fi]\sigma$

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

- $= [while\ x \neq y\ do\ x := x+1; y := y+2; od]\sigma_2$

  Regel: $[while\ e\ do\ p\ od]\sigma =$

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

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

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

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

  $\sigma_4 : x \mapsto 3, y \mapsto [y+2]\sigma_3 = 3$

- $= [while\ x \neq y\ do...od]\sigma_4$

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

- $= \sigma_4$

of Tpl.

Exercise 3 (1 point) Let p be the following program:

```
$x := x + y$;
if $x < 0$ then
   abort
else
   while $x \neq y$ 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.

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 $2x \neq y$ do
   $x := x + 1$;
   $y := y + 1$;
od
```

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

$$\frac{\{\, Inv \wedge e \,\}\, p \,\{\, Inv \,\}}{\{\, Inv \,\}\, \textsf{while}\ e\ \textsf{do}\ p\ \textsf{od}\, \{\, Inv \wedge \neg e \,\}}\ {\scriptstyle (\mathrm{wh})}$$

is correct regarding partial correctness, i.e., show that $\{\, Inv \,\}$ while e do p 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 wlp(while e do p od$, G)$ similar to the weakest precondition in the course.

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

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

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