Hoare Logic Recap

Software Verification 2010

13 October 2010

1 Factorial

- Write a routine that computes the factorial of its input argument n.
- Annotate the routine with pre and postcondition.
- Prove that your implementation is correct.

```
fact (n: INTEGER): INTEGER
  require n ≥ 0
  local i: INTEGER
  do
    from
      i := 0
      Result := 1
    until i = n
    loop
      i := i + 1
      Result := Result * i
    end
  ensure Result = n! end
```

With standard notation, our goal is to prove that the following Hoare triple is valid.

```
{ n ≥ 0 }
from
  i := 0
  Result := 1
until i = n
loop
  i := i + 1
  Result := Result * i
end
{ Result = n! }
```

Let *Inv* denote the loop invariant. The following is a proof outline of a partial correctness proof, based on the inference rule for loops.

```
{ n ≥ 0 }
from
```

$i := 0$
Result $:= 1$
$\{$ *Inv* $\}$
until $i = n$
loop
$\{$ *Inv* $\wedge i \neq n$ $\}$
$i := i + 1$
Result $:=$ **Result** $* \; i$
$\{$ *Inv* $\}$
end
$\{$ *Inv* $\wedge i = n$ $\}$
$\{$ **Result** $= n!$ $\}$

Once we find a suitable invariant, we can verify each block separately, thanks to the composition and the loop inference rules.

To determine the invariant, consider the values of i and **Result** over a few iterations:

i	**Result**
0	1
1	1
2	2
3	6
4	24

It should be clear that **Result** $= i\,!$ is an invariant characterizing the loop.

Finally, prove each block correct with backward substitution (the assignment rule). The first block:

$\{$ $n \geq 0$ $\}$
$\{$ $1 = 0\,!$ $\}$
$i := 0$
$\{$ $1 = i\,!$ $\}$
Result $:= 1$
$\{$ **Result** $= i\,!$ $\}$

is correct because indeed $1 = 0!$.

The second block:

$\{$ **Result** $= i\,! \wedge i \neq n$ $\}$
$\{$ **Result** $* \; (i + 1) = (i + 1)!$ $\}$
$i := i + 1$
$\{$ **Result** $* \; i = i\,!$ $\}$
Result $:=$ **Result** $* \; i$
$\{$ **Result** $= i\,!$ $\}$

is correct because **Result** $= i\,!$ implies **Result** $* \; (i + 1) = (i\,!) * (i+1) = (i+1)!$ by elementary arithmetic.

The third block is also correct, because **Result** $= i\,! \wedge i = n$ implies **Result** $= n!$ by elementary arithmetic.

To prove termination, consider the variant $n - i$. It decreases at every iteration because i increases but n does not change:

$\{n - i = x\}$ $i := i + 1$; **Result** $:=$ **Result** $* \; i$ $\{n - i < x\}$

Also, $i \leq n$ is a loop invariant, which implies that $n - i \geq 0$, hence the variant has a lower bound. This concludes the termination proof.

2 Primality testing

The following piece of code sets *pr* to **True** iff x — assumed to be greater than one — is a prime number. Prove correctness.

```
{ x > 1 }
from i := 2 ; pr := True
until i ≥ x
loop
   if x mod i = 0 then
      pr := False
   end
   i := i + 1
end
{ (¬ pr ⇒ ∃ y (1 < y < x ∧ x mod y = 0))
  ∧ (pr ⇒ ∀ y (1 < y < x ⇒ x mod y ≠ 0)) }
```

The proof follows the usual proof outline, based on the inference rule for loops, with *Inv* denoting the loop invariant.

```
{ x > 1 }
from i := 2 ; pr := True
{ Inv }
until i ≥ x
loop
   { Inv ∧ i < x }
   if x mod i = 0 then
      pr := False
   end
   i := i + 1
   { Inv }
end
{ Inv ∧ i ≥ x }
{ (¬ pr ⇒ ∃ y (1 < y < x ∧ x mod y = 0))
  ∧ (pr ⇒ ∀ y (1 < y < x ⇒ x mod y ≠ 0)) }
```

The invariant must imply, together with $i \geq x$, the postcondition, hence it is probably very close to it syntactically. Indeed, since the loop proceeds by increasing i from 2 up until x, a loop invariant is obtained by replacing x with i in the postcondition. Another clause in the loop invariant specifies the obvious bounds for i: $1 < i \leq x$.

$$
\begin{aligned}
\mathit{Inv} \triangleq 1 < i \leq x \wedge & (\neg\, pr \Rightarrow \exists\, y\ (1 < y < i \wedge x \ \mathbf{mod}\ y = 0)) \\
& \wedge (pr \Rightarrow \forall\, y\ (1 < y < i \Rightarrow x \ \mathbf{mod}\ y \neq 0))
\end{aligned}
$$

2.1 Initialization

The first block (initialization) corresponds to the triple:

$\{\ x > 1\ \}$
from $i := 2$; $pr :=$ **True**
$\{\ Inv\ \}$

The backward substitution of Inv yields:

$$1 < 2 \leq x \wedge (\neg\ \mathbf{True} \Rightarrow \exists\ y\ (1 < y < 2 \wedge x\ \mathbf{mod}\ y = 0))$$
$$\wedge (\mathbf{True} \Rightarrow \forall\ y\ (1 < y < 2 \Rightarrow x\ \mathbf{mod}\ y \neq 0))$$

Then:

- $2 \leq x$ is equivalent to the precondition $x > 1$.
- The first implication holds trivially because its antecedent if **False**.
- The second implication holds trivially because the interval $1 < y < 2$ is empty for all integer values of y.

2.2 Loop iteration

The second block requires to prove:

$\{\ Inv \wedge i < x\ \}$
if x **mod** $i = 0$ **then** $pr :=$ **False** **end**
$i := i + 1$
$\{\ Inv\ \}$

Using the inference rule for **if**, split the proof into two branches.

2.2.1 Then branch

$\{\ Inv \wedge i < x \wedge x\ \mathbf{mod}\ i = 0\ \}$
$pr :=$ **False**
$i := i + 1$
$\{\ 1 < i \leq x \wedge (\neg\ pr \Rightarrow \exists\ y\ (1 < y < i \wedge x\ \mathbf{mod}\ y = 0))$
$\wedge (pr \Rightarrow \forall\ y\ (1 < y < i \Rightarrow x\ \mathbf{mod}\ y \neq 0))\ \}$

Backward substitution yields:

$\{\ 1 < i{+}1 \leq x \wedge (\neg\ \mathbf{False} \Rightarrow \exists\ y\ (1 < y < i{+}1 \wedge x\ \mathbf{mod}\ y = 0))$
$\wedge (\mathbf{False} \Rightarrow \forall\ y\ (1 < y < i{+}1 \Rightarrow x\ \mathbf{mod}\ y \neq 0))\ \}$

- The clauses $1 < i < x$ imply the clause $1 < i{+}1 \leq x$, as we are dealing with integer variables.
- The first implication requires to establish $\exists\ y\ (1 < y < i{+}1 \wedge x\ \mathbf{mod}\ y = 0)$, which is implied by $x\ \mathbf{mod}\ i = 0$ in the precondition for $\overline{y} = i < i + 1$.
- The second implication is trivial as its antecedent is false.

2.2.2 Else branch

$\{\ Inv \wedge i < x \wedge x\ \mathbf{mod}\ i \neq 0\ \}$
$i := i + 1$
$\{\ 1 < i \leq x \wedge (\neg\ pr \Rightarrow \exists\ y\ (1 < y < i \wedge x\ \mathbf{mod}\ y = 0))$
$\wedge (pr \Rightarrow \forall\ y\ (1 < y < i \Rightarrow x\ \mathbf{mod}\ y \neq 0))\ \}$

Backward substitution yields:

```
{ 1 <i+1 ≤x ∧(¬ pr ⇒∃ y (1 <y <i+1 ∧x mod y = 0))
                    ∧(pr ⇒∀ y (1 <y <i+1 ⇒x mod y ≠0)) }
```

First notice that The clauses $1 < i < x$ imply the clause $1 < i{+}1 \leq x$, as we are dealing with integer variables. Then, the proof follows a case discussion:

1. CASE $pr = \mathbf{False}$.
   We have to establish only the first implication, as the second has false antecedent. The precondition, for $pr = \mathbf{False}$, says in particular that $\exists\ y\ (1 < y < i \wedge x \ \mathbf{mod}\ y = 0)$. The value $\overline{y}$ that satisfies the existential quantification also satisfies the weaker quantification $\exists\ y\ (1 < y < i{+}1 \wedge x \ \mathbf{mod}\ y = 0)$ over the larger interval $(1, i + 1)$.

2. CASE $pr = \mathbf{True}$.
   We have to establish only the second implication, as the first has false antecedent. In the precondition with $pr = \mathbf{True}$, we combine the facts $\forall\ y\ (1 < y < i \Rightarrow x \ \mathbf{mod}\ y \neq 0)$ and $x \ \mathbf{mod}\ i \neq 0$ to get $\forall\ y\ (1 < y < i{+}1 \Rightarrow x \ \mathbf{mod}\ y \neq 0)$, the stronger quantification over the larger interval $(1, i + 1)$.

2.3 Conclusion

The loop invariant clause $i \leq x$ and $i \geq x$ imply $i = x$. Substituting x for i in the other loop invariant clauses yields the postcondition of the program.

2.4 Termination

The variant $x - i$ and the invariant clause $1 < i \leq x$ can be combined to prove termination.

3 Least common multiple

Consider a simple program computing the least common multiple (LCM) of two integers x, y, with the following specification.

```
{ x ≥ 1 ∧y ≥ 1 }
  from z := 1
  until z mod x = 0 ∧z mod y = 0
  loop z := z + 1
  end
{ z mod x = 0 ∧z mod y = 0 ∧
  ∀ w (1 ≤w <z ⇒(w mod x ≠0 ∨w mod y ≠0)) }
```

Prove its correctness.

The partial correctness proof follows the usual outline, for a suitable loop invariant *Inv*.

```
{ x ≥ 1 ∧y ≥ 1 }
  from z := 1
{ Inv }
  until z mod x = 0 ∧z mod y = 0
  loop
```

$\{$ *Inv* $\wedge (z$ **mod** $x \neq 0 \vee z$ **mod** $y \neq 0)$ $\}$
  $z := z + 1$
$\{$ *Inv* $\}$
  end
$\{$ *Inv* $\wedge z$ **mod** $x = 0 \wedge z$ **mod** $y = 0$ $\}$
$\{$ z **mod** $x = 0 \wedge z$ **mod** $y = 0 \wedge$
  $\forall\ w\ (1 \leq w < z \Rightarrow (w$ **mod** $x \neq 0 \vee w$ **mod** $y \neq 0))$ $\}$

The loop invariant should mirror the last conjunct of the postcondition, hence:

Inv $\triangleq \forall\ w\ (1 \leq w < z \Rightarrow (w$ **mod** $x \neq 0 \vee w$ **mod** $y \neq 0))$

3.1 Initialization

Backward substitution of Inv through the **from** block yields:

$\forall\ w\ (1 \leq w < 1 \Rightarrow (w$ **mod** $x \neq 0 \vee w$ **mod** $y \neq 0))$

which holds trivially because the interval $[1, 1)$ is empty.

3.2 Loop iteration

The loop body is very simple, hence just apply backward substitution of Inv through $z := z + 1$ to get:

$I' \triangleq \forall\ w\ (1 \leq w < z{+}1 \Rightarrow (w$ **mod** $x \neq 0 \vee w$ **mod** $y \neq 0))$

Inv implies I' for values of w less than z; combined with the other conjunct $(z$ **mod** $x \neq 0 \vee z$ **mod** $y \neq 0)$, it is equivalent to I'.

3.3 Conclusion

Inv and the exit condition z **mod** $x = 0 \wedge z$ **mod** $y = 0$ is exactly the postcondition.

3.4 Termination

Use the variant $x{*}y - z$ and the invariant $x{*}y - z \geq 0$ to prove termination. (Recall that $x{*}y$ **mod** $x = x{*}y$ **mod** $y = 0$).