

Week 9 Tutorial Solutions

Hoare Logic

The Warmup Exercises

a) $\{j = a\} j := j + 1 \{a = j + 1\}$

False. Consider the case of $a = j = 0$; the precondition is satisfied, but after the assignment $a = 0$ and $j = 1$ and the postcondition is false.

b) $\{i = j\} i := j + i \{i > j\}$

False. Consider the case of both i and j being negative. If $i = j = -1$ initially, then the postcondition will be false.

c) $\{j = a + b\} i := b; j := a \{j = 2 * a\}$

False. A counterexample is an initial state where $a = b = 1$ and $j = 2$. After the assignment, $a = j = 1$ and the postcondition does not hold.

d) $\{i > j\} j := i + 1; i := j + 1 \{i > j\}$ **True.** Regardless of the initial state, the computation ends by making $i = j + 1$ and so $i > j$ is satisfied.

e) $\{i \neq j\} i > j \text{ then } m := i - j \text{ else } m := j - i \{m > 0\}$

True. If $i > j$ then m will be assigned a positive quantity. The other possibility is that $i < j$, given that the precondition holds, and again m will be assigned a positive number.

f) $\{i = 3 * j\} i > j \text{ then } m := i - j \text{ else } m := j - i \{m - 2 * j = 0\}$

False. If $i = -3$ and $j = -1$ then m will end up as 2 and so the postcondition will not hold.

g) $\{x = b\} \text{ while } x > a \text{ do } x := x - 1 \{b = a\}$

False. If initially $x = b = 1$ and $a = 2$ then the loop terminates with no iterations, a and b will be unchanged and the postcondition will be false.

Single Assignment Statements

a) $\{i = 5\} \mathbf{a:=i+2} \{(a = 7) \wedge (i = 5)\}$

The Assignment Axiom gives $\{(i + 2 = 7) \wedge (i = 5)\} \mathbf{a:=i+2} \{(a = 7) \wedge (i = 5)\}$.
The precondition then simplifies to $\{i = 5\}$ as required.

b) $\{i = 5\} \mathbf{a:=i+2} \{a = 7\}$

This can be attacked in the same way as in part (a), or we can just use the result of part (a) with postcondition weakening.

c) $\{i = 5\} \mathbf{a:=i+2} \{(a = 7) \wedge (i > 0)\}$

Another example where we can either use assignment axiom followed by precondition strengthening or can use postcondition weakening on the result of part (a).

d) $\{(i = 5) \wedge (a = 3)\} \mathbf{a:=i+2} \{a = 7\}$

Part (b) followed from the Assignment Axiom; having got that far we just need to strengthen the precondition with the theorem (which we get by invoking “Standard Mathematics”): $((i = 5) \wedge (a = 3)) \rightarrow (i = 5)$.

e) $\{a = 7\} \mathbf{i:=i+2} \{a = 7\}$

This question is a one-liner that can be done in one’s head. The importance of the example is that you often want to know that some variables value is not affected by the execution of some given code.

f) $\{i = a - 1\} \mathbf{i:=i+2} \{i = a + 1\}$

Assignment axiom and simplifying the precondition, as above.

g) $\{i = 5\} \mathbf{i:=i+2} \{i > 0\}$

So, how about a properly set out proof?

- | | |
|--|----------------|
| 1. $\{i + 2 > 0\} \mathbf{i:=i+2} \{i > 0\}$ | Ass. Ax. |
| 2. $(i = 5) \rightarrow (i + 2 > 0)$ | Std. Math. |
| 3. $\{i = 5\} \mathbf{i:=i+2} \{i > 0\}$ | Prec. Str. 1,2 |

QED

h) $\{True\} \mathbf{a:=i+2} \{a = i + 2\}$

Another easy one. The only thing that is a little unusual is the constant *True* as the precondition, so discuss what this means (that if the code fragment terminates then it does so fulfilling the postcondition, whatever the opening state might have been). What might it mean if *False* was the precondition?

Dealing with Control Structures

a) $\{a > b\} \text{ m:=1; n:=a-b } \{m * n > 0\}$

The Assignment axiom gives $\{m * (a - b) > 0\} \text{ n:=a-b } \{m * n > 0\}$ and that defines the intermediate state for the sequence's execution. Applying the Assignment Axiom again gives $\{1 * (a - b) > 0\} \text{ m:=1 } \{m * (a - b) > 0\}$. After a little bit of simplification we apply the Sequence rule and get $\{a > b\} \text{ m:=1; n:=a-b } \{m * n > 0\}$.

b) $\{s = 2^i\} \text{ i:=i+1; s:=s*2 } \{s = 2^i\}$

Use Assignment Axiom on the 2nd assignment: $\{s * 2 = 2^i\} \text{ s:=s*2 } \{s = 2^i\}$, and then again on the 1st assignment: $\{s * 2 = 2^{i+1}\} \text{ i:=i+1 } \{s * 2 = 2^i\}$. Simplifying the precondition and using the Sequence Rule gives us the desired result. No problems here!

Point out that $(s = 2^i)$ is an *invariant* for this code.

c) $\{True\} \text{ if } i < j \text{ then } \text{min:=i} \text{ else } \text{min:=j} \{(min \leq i) \wedge (min \leq j)\}$

The conditional Rule gives us subgoals – one for each assignment. The proof follows:

1. $\{(i \leq i) \wedge (i \leq j)\} \text{ min:=i } \{(min \leq i) \wedge (min \leq j)\}$ Ass. Ax.
2. $(True \wedge (i < j)) \rightarrow ((i \leq i) \wedge (i \leq j))$ Std. Math.
3. $\{True \wedge (i < j)\} \text{ min:=i } \{(min \leq i) \wedge (min \leq j)\}$ Prec. Str. 1,2
4. $\{(j \leq i) \wedge (j \leq j)\} \text{ min:=j } \{(min \leq i) \wedge (min \leq j)\}$ Ass. Ax.
5. $\{True \wedge \neg(i < j)\} \text{ min:=j } \{(min \leq i) \wedge (min \leq j)\}$ Simplify Precond. 4
6. $\{True\} \text{ if } i < j \text{ then } \text{min:=i} \text{ else } \text{min:=j} \{(min \leq i) \wedge (min \leq j)\}$ Cond. Rule 3,5

QED

d) $\{i > 0 \wedge j > 0\} \text{ if } i < j \text{ then } \text{min:=i} \text{ else } \text{min:=j} \{min > 0\}$

This is just as easy as part (c).

e) $\{s = 2^i\} \text{ while } i < n \text{ do } \text{i:=i+1; s:=s*2 } \{s = 2^i\}$

We use the result of part (b) which shows that $(s = 2^i)$ is an invariant for the loop's body. We should expect it to be the right invariant because it is the invariant we would use from a programming point of view. That is, $(s = 2^i)$ is the condition that documents our understanding of the intent of the loop.

Note: What we would much rather prove about this loop is:

$$\{s = 2^i \wedge i \leq n\} \text{ while } i < n \text{ do } \text{i:=i+1; s:=s*2 } \{s = 2^n\}$$

in which case the invariant we would choose is $\{s = 2^i \wedge i \leq n\}$.

Exercise: Complete the proof of this statement.