

Homework 2 - Axiomatic semantics and invariants

Here is some practice with the techniques of axiomatic semantics and invariants.

1. Determine the weakest precondition for the given statements and postconditions (each line is a separate problem). Show the intermediate assertions for the compound statements.

① $x = x+1 \{0 \leq x+y < z\}$
 $\{0 \leq (x+1) + y < z\}$
(substitute $(x+1)$ for x in the post-condition)

② $x = x-1; y = y+x \{x \geq 0 \ \&\& \ y \leq z\}$
 $\text{wp}(x = x-1, \text{wp}(y = y+x, \{x \geq 0 \ \&\& \ y \leq z\})) =$
 $\text{wp}(x = x-1, \{x \geq 0 \ \&\& \ (y+x) \leq z\}) =$
 $\{x-1 \geq 0 \ \&\& \ y+(x-1) \leq z\}$
you might simplify this to
 $\{x \geq 1 \ \&\& \ y+x \leq z+1\}$

③ $y=x; x = 3(x+1) \{0 < y < x\}$
 $\text{wp}(y = x, \text{wp}(x = 3(x+1), \{0 < y < x\})) =$
 $\text{wp}(y = x, \{0 < y < 3x + 3\}) =$
 $\{0 < x < 3x + 3\}$
simplifies to
 $\{0 < x\}$

④ $y=x+1; x = 3(y+1) \{0 < x < y\}$
 $\text{wp}(y = x+1, \text{wp}(x = 3(y+1), \{0 < x < y\})) =$
 $\text{wp}(y = x+1, \{0 < 3y+3 < y\})$
 $\{0 < 3(x+1) + 3 < x + 1\} = \{x < -5/2 \ \&\& \ x > -2\} = \text{False}.$

In no state can you execute this sequence of statements and end up in a state satisfying the postcondition.

2. Some problems involving repeated actions are naturally solved using invariants. Here is a classic: suppose you remove two diagonally opposite corners of a chessboard. Note that those two squares are the same color. Now place rectangles on the board with each rectangle covering exactly two horizontally or vertically adjacent squares. Can you completely cover the mutilated chessboard using these rectangles? To answer the question find and state an invariant property of the board as it is gradually covered with rectangles. (Just state the invariant as an English sentence -- you don't have to formalize it in mathematics.) When no more rectangles can be placed (i.e. the loop exits) what can you conclude about the coverage of the board.

Because two adjacent squares are not same color, when you place a rectangle on the board you reduce the number of exposed square of each color by 1.

Removing 2 opposite-corner squares removes two squares of the same color (say black), so the invariant is number-of-exposed-black+2 = number-of-exposed-red. When no more moves are possible the invariant still holds, so there are at least 2 red squares not covered.

3. In this program

```
s = 0;
n = 1;
while x > 0 do
  s = s+n;
  n = n + 2;
  x = x - 1;
end
```

show that $\{s = ((n-1)/2)^2\}$ is an invariant of the loop. Be sure to show that the invariant holds the first time the loop test is reached as well as each subsequent time. For this problem write out the necessary assertions using formal, mathematical notation. From the invariant and the loop test what can you conclude about the state at the completion of the loop (after the last iteration)?

```
{True}
{0=0}
s = 0;
{s=((1-1)/2)**2}=0}
n = 1;
{s = ((n-1)/2)**2}
while x > 0 do
  wp{s=s+n, {s=((n+1)/2)**2}) == { s+n = ((n+1)/2)**2 } ==
  {s+n = (n**2)/4 + n/2 + 1/4} == {s=(n**2)/4 - n/2 + 1/4} ==
  {s=((n-1)/2)**2}
  s = s+n;
  wp(n = n + 2, {s = ((n-1)/2)**2}) == { s=((n+1)/2)**2}
  n = n + 2;
  {s = ((n-1)/2)**2}
  x = x - 1;
  {s = ((n-1)/2)**2}
end
{s = ((n-1)/2)**2 && x<=0}
```

So, even though adding up the first x odd numbers produces x^2 you can't conclude that because this invariant is too weak – it doesn't say enough about the initial value of x .

4. Write a code to calculate the integer quotient of two positive integers n , the dividend, and k , the divisor. Use repeated subtraction; do NOT use the $/$ operator. Assume n and k are both > 0 . The answer is to be in variable q

One key to this kind of problem is determining how to express the post-condition that is to be achieved. We will go over this in class. Once you know the postcondition think about the loop test and the invariant and how you will write the code. What is the postcondition? What will you try to use as the loop invariant (I)? What is the boolean test (B)? You must show, using the statement sequence rule that your claimed invariant actually is an invariant for your loop.

```

q = 0
div = n
while ( k<= div) begin
    { n = div + q * k && div > 0 && k<=div } =>      --invariant && looptest
    { n = div-k+q*k+k && div-k>=0 }      --wp(loop body, {invariant})
    {n=div-k+(q+1)*k && div-k >= 0}
    div = div-k
    { n=div + (q+1)*k && div >= 0
    q = q +1
    { n = div + q * k && div >= 0}
end

```

Invariant of loop:

```
{ n = div + q * k ^ div >= 0}
```