

# Homework 2

CSC 7101, Spring 2009

Due: 2 March 2009

1. (5 pts)

What set of states does the formula ‘ $\forall x.x = 3$ ’ characterize? What set of states does the formula ‘ $\forall x.x = x$ ’ characterize? Explain your answers.

2. (10 pts)

Which of the following Hoare triples are valid:

(a)  $\{ \text{true} \} x := 2 \{ \text{true} \}$

(b)  $\{ \text{true} \} x := x \{ \text{false} \}$

(c)  $\{ \text{false} \} x := 2 \{ \text{true} \}$

(d)  $\{ \text{false} \} x := 2 \{ \text{false} \}$

(e)  $\{ \text{true} \} \text{while true do } x := 2 \text{ end } \{ \text{false} \}$

(f)  $\{ \text{true} \} x := x + 1 \{ x = x + 1 \}$

(g)  $\{ x = y \} t := x; x := y; y := t \{ x = y \}$

(h)  $\{ x \geq 0 \} x := y \{ y \geq 0 \}$

3. (5 pts)

The following axiom for the assignment isn’t sound:

$$\frac{}{\{ \text{true} \} x := e \{ x = e \}}$$

Find an example that demonstrates the unsoundness of the rule, i.e., a Hoare triple that can be proved with this rule but that isn’t operationally valid.

4. (10 pts)

Give a good (i.e., sound and complete) rule for the assignment statement where the postcondition is determined by the precondition. In other words, fill in the question mark in the following proof rule. Explain why your rule is good.

$$\frac{}{\{ P \} x := e \{ ? \}}$$

Hint: make sure you don’t lose the information contained in P.

5. (10 pts)

Find a counterexample to the following proposed proof rule for the while loop:

$$\frac{\{ P \wedge B \} S \{ Q \}}{\{ P \} \text{ while } B \text{ do } S \text{ end } \{ \neg B \wedge Q \}}$$

Hint: since this rule does not require the loop invariant to be valid after an execution of the loop body, this suggests we should look for an example that works for a single execution of  $S$ , but not for repeated execution of  $S$ .

6. (10 pts)

The following Egyptian Multiplication algorithm takes two integers  $x$  and  $y$  as input and computes the product  $x * y$  as the value of  $s$ .

Provide a sketch of the **partial** correctness proof of the Egyptian Multiplication algorithm. The loop invariant has been provided.

```
{ true }
a := x;
b := y;
s := 0;
{ x * y = a * b + s }
while b != 0 do
  if odd(b) then
    s := s + a;
  a := 2 * a;
  b := b div 2;
end
{ s = x * y }
```

where `div` is regular integer division, and `odd()` returns true if its argument is an odd integer and false otherwise.

Clearly separate the proof into the three parts before, in, and after the while loop, and indicate the implications resulting from the rule of consequence.