Solutions to selected exercises from Homework 4

Comments on homework 4

Magic (also known as pulling rabbits out of hats): in 2.30 on what evidence do you assume that you can limit looking at f(), g(), and h() to parameter values less than some arbitrary n or even separate limits m, n, p for the different functions? {In my solution below I seemingly pull min out of a hat – but I justify its existence from the givens of the problems and I use it only in the proof and not as a program variable.}

Changing the rules: example: if the question is about the at-most-once property it means "according to the definition", not "according to some changed version of the definition that you just made up". If you don't know how to solve a problem as given, it *may* be permissible to alter the problem definition to one you can solve (researchers do this all the time, of course, in order to find tractable subproblems of the problems they would really like to solve.) However, if you change the problem definition you are obligated to state the original problem definition, state why you believe it is intractable as given, explain how you modified it and explain why you made the particular modifications that you did.

Suggestion:make it a habit on homeworks, exams, projects in this class and others to check your solutions for type-checking failures, use of magic, and places where you've changed the rules. If any of these occur, your solution is not a solution and needs to be changed. I find that type-checking is often a good algorithm to run mentally while listening to technical talks. By tracking "what kind of thing" each symbol refers to I find that I understand better what is being said and sometimes gain insights that even the presenter has missed.

2.12

In part a, since the two statements are atomic, the only possible executions are one first and then the other. So the results are either (x=5,y=15) or (x=8,y=6). If the statements are not atomic, both expressions can be evaluated before either variable is assigned so a third possibility arises (x=5,y=6).

2.22

The for loop written as an annotated while loop looks like:

```
i=1
while (i≤n)
   S;
   i = i+1;
```

Our goal is to find a statement of the form if xxx and yyy and ... are theorems of programming logic then {*something*} for i=1 to n {*something else*} is a theorem of programming logic.

Begin by observing that we would like at the end of the for loop to conclude, at least, that i==n+1. Thus {*something else*} should be of the form {*something else'* $\wedge$ i==n+1}, henceforth, {P $\wedge$ i==n+1}. Now observe that the WHILE rule tells us that after the loop we have {i$\geq$n+1}, which suggests that the loop invariant should be {P$\wedge$i$\leq$n+1}. So now our proof outline looks like:

```
i=1
{P∧i≤n+1}
while (i≤n)
   {P∧i≤n}
   S;
   i = i+1
   {P∧i≤n+1}
{P∧i==n+1}
```

Let's push the invariants through the assignment statements:

```
{P∧1≤n+1}
{P∧0≤n}
i=1
{P∧i≤n+1}
while (i≤n)
   {P∧i≤n}
   S;
   {P∧i≤n}
   i = i+1
   {P∧i≤n+1}
{P∧i==n+1}
```

The only programming logic triple in this outline that is not a theorem of programming logic is the one containing S, the body of the loop. This triple becomes the hypothesis of our proof rule for the FOR statement. So, we have if {P$\wedge$i$\leq$n} S {P$\wedge$i$\leq$n} is a theorem of programming logic, then so is {P$\wedge$0$\leq$n} FOR i =1 to n {P$\wedge$i==n+1}.

2.30

In this problem we're asked to find the least values of i,j,k such that f(i)==g(j)==h(k) given that each function is a strictly increasing function of non-negative integers, and that there exists such values. The problem is to show that we have computed the smallest such i,j,k. In order to do this we'll define a *logical variable* min to be the required

value of f (and g and h). We're told it exists as part of the problem; we don't know what its value is but we know enough of its properties to make use of it in the proof.

The problem stated in the book requires that we perform the tests concurrently.

```
(2)  i=0; j=0; k=0;
(3)  {I: f(i)≤min ∧ g(j)≤min ∧ h(k)≤min}
(1)  while f(i)≠g(j)∨g(j)≠h(k)
        co
(6)        {I: f(i)≤min ∧ g(j)≤min ∧ h(k)≤min}
(5)        if f(i)<g(j) ∨ f(i)<h(k) then
(7)           {f(i)<min ∧ g(j)≤min ∧ h(k)≤min}
(5)           i=i+1
           {I: f(i)≤min ∧ g(j)≤min ∧ h(k)≤min}
        //
(6)        {I: f(i)≤min ∧ g(j)≤min ∧ h(k)≤min}
(5)        if g(j)<f(i) ∨ g(j)<h(k) then
(7)           {f(i)≤min ∧ g(j)<min ∧ h(k)≤min}
(5)           j=j+1
           {I: f(i)≤min ∧ g(j)≤min ∧ h(k)≤min}
        //
(6)        {I: f(i)≤min ∧ g(j)≤min ∧ h(k)≤min}
(5)        if h(k)<f(i) ∨ h(k)<g(j) then
(7)           {f(i)≤min ∧ g(j)≤min ∧ h(k)<min}
(5)           k=k+1
           {I: f(i)≤min ∧ g(j)≤min ∧ h(k)≤min}
        oc
(3)     {f(i)≤min ∧ g(j)≤min ∧ h(k)≤min}
(4)  f(i)==g(j) ∧ g(j)==h(k) ∧ I
```

You should check that each of the required sequential proofs is present in this proof outline.

Non-interference:

The only assignment actions are in the lines labelled (5). The invariant is a critical assertion; let's examine the case for non-interference of I by the assignment i=i+1 in the first branch. The theorem we need is

{f(i)≤min ∧ g(j)≤min ∧ h(k)≤min ∧ f(i)<min ∧ g(j)≤min ∧ h(k)≤min}i=i+1 {I}

But this is just the same theorem as already surrounds i=i+1 which we already established. The proofs of non-interference with the critical assertions on lines labelled (7) are similar.

Instead of a co for each interation of a loop, we could write a co containing 3 loops:

```
i=0; j=0; k=0;
```

```
co
   while f(i)≠g(j)∨g(j)≠h(k)
      if f(i)<g(j) ∨ f(i)<h(k) then
         i=i+1
//
   while f(i)≠g(j)∨g(j)≠h(k)
      ...
//
   while f(i)≠g(j)∨g(j)≠h(k)
      ...
oc
```

The proof is almost the same, but the loops do not necessarily make progress toward a solution on each iteration. This program can waste a lot of computer cycles.

Some submitted solutions used a loop containing co of two other loops. That's ok though maybe the proofs become a little harder.