T-79.3001 Logic in computer science: foundations **Spring 2007**
Exercise 12 ([Huth & Ryan, 2000], Chapter 4)
May 2–3, 2007

Solutions to demonstration problems

4. Use propositional logic to prove the equivalence of the following statements.

(a) `!(a==b||a<b)`

(b) `a!=b&&!(b>a)`

Solution. Boolean statements can be represented using basic cases, thus

$$\texttt{a==b} \equiv_{def} \texttt{!(a>b)\&\&!(b>a)}$$

and

$$\begin{aligned} \texttt{a<b} &\equiv_{def} \texttt{b>a} \\ \texttt{a!=b} &\equiv_{def} \texttt{!(a==b)} \end{aligned}$$

We choose $A =$“`a>b`” and $B =$“`b>a`” as atomic propositions. This way the statement in item (a) is

$$\neg((\neg A \wedge \neg B) \vee B)$$

and respectivelly, in item (b):

$$\neg(\neg A \wedge \neg B) \wedge \neg B$$

Notice that the sesond proposition is obtained from the first by applying de Morgan’s rule and thus the statements are logically equivalent.

5. Prove the partial correctness in the following cases.

(a) $\models_p$ [`x>0`] `y=x+1` [`y>1`]

Solution. Starting from the postcondition and applying the rule for assignment backwards, we obtain [`x+1>1`] `y=x+1` [`y>1`]

`x>0` is equivalent to `x+1>1`, and the claim holds.

(b) $\models_p$ [`true`] `y=x;y=x+x+y` [`y==3*x`]

Solution. Applying twise the assignment rule, we obtain:

[`x+x+y==3*x`] `y=x+x+y` [`y==3*x`]
[`x+x+x==3*x`] `y=x` [`x+x+y==3*x`]

and furthermore using the rule for composition:

[`x+x+x==3*x`] `y=x;y=x+x+y` [`y==3*x`].

Statement `x+x+x==3*x` evaluates to true for all integers, and thus the claim holds.

(c) $\models_p$ [`x>1`] `a=1;y=x;y=y-a` [`y>0&&x>y`]

Solution.

[`y-a>0&&x>y-a`] `y=y-a` [`y>0&&x>y`]
[`x-a>0&&x>x-a`] `y=x` [`y-a>0&&x>y-a`]
[`x-1>0&&x>x-1`] `a=1` [`x-a>0&&x>x-a`].

Now, the latter part of `x-1>0&&x>x-1` evaluates to true for all integers and `x-1>0` is equivalent to `x>1`. Thus the claim holds.

6. Show that $\models_p$ [`true`] `P` [`z==` min`(x,y)`], where `P` is the following program:

```
if(x>y) then {
    z=y
} else {
    z=x
}
```

Solution.

[`true&&!(x>y)`][`!(x>y)`][`x<=y`][`x ==` min`(x,y)`] `z=x` [`z==` min`(x,y)`] and
[`true&&(x>y)`][`(x>y)`][`y ==` min`(x,y)`] `z=y` [`z==` min`(x,y)`].

Thus,

```
[true]
if(x>y) then {
    z=y
} else {
    z=x
} [z == min(x, y)]
```

7. Show that

(a) $\models_p$ [true] Sum [z==x+y]

(b) $\models_t$ [0<=y] Sum [z==x+y]

where Sum is the following program:

```
z=x;
v=y;
while(!(v==0)){
    z=z+1;
    v=v-1
}
```

Solution. First, we need and invariant for the loop. Inspecting the code, we note that the value of variable *z* increases while the value for variable *v* decreases. Moreover, the sum of *z* and *v* stays constant. This constant is obtained for the initila values of *z* and *v*, as thus we have invariant *I*: z+v==x+y.

We check that *I* really is an invariant:

```
[z+v-1==x+y] v=v-1 [z+v==x+y]
[z+v==x+y][z+1+v-1==x+y] z=z+1 [z+v-1==x+y]
```

Thus:

```
[z +v ==x +y]
while(!(v ==0)){
    z=z+1;
    v=v-1
}[z +v ==x +y && v ==0]
```

Finally, we need to find the preconditions for the assignments before the loop:

```
[z+y==x+y] v=y [z+v==x+y]
[x+y==x+y] z=x [z+y==x+y]
```

Now x+y==x+y evaluates to true for all integers.

(b) $\models_t$ [0<=y] Sum [z==x+y]

To prove total correctness, we need to make sure the program terminates.

```
[0 <=y][x +y ==x +y && 0 <=y] z=x [z +y ==x +y && 0 <=y]
[z +y ==x +y && 0 <=y] v=y [z +v ==x +y && 0 <=v]
while(!(v ==0)){
    [z +v ==x +y && 0 <=v -1 && v -1 <n] z=z +1;
                         [z +v -1 ==x +y && 0 <=v -1 && v -1 <n]
    [z +v -1 ==x +y && 0 <=v -1 && v -1 <n] v=v -1
                         [z +v ==x +y && 0 <=v && v <n]
}[z +v ==x +y && 0 <=v && v ==0][z ==x+y]
```