Workshop Testing and Formal Methods (Week 6), with Answers

Note: this is the theory part of the Exam Software Specification and Testing held on October 23, 2014

```
module Workshop6Answers

where
import Data.List
```

There are six questions.

Question 1 A property p on a domain D is a function $D \to \{\top, \bot\}$, where $\top$ and $\bot$ are the two truth values. A property p on D is stronger than a property q on D if it holds for all $x \in D$ that $p(x)$ implies $q(x)$. A property p on D is weaker than a property q on D if q is stronger than p. (You have seen these notions during the workshop sessions.)

Which of the following pairs of properties on the integers is stronger?

1. $\lambda x \mapsto x = 0$ and $\lambda x \mapsto x \leq 0$.
2. $\lambda x \mapsto x^2 \neq 4$ and $\lambda x \mapsto x \neq 2$.
3. $\lambda x \mapsto x^2 \geq 49$ and $\lambda x \mapsto x \geq 7 \vee x \leq -7$.
4. $\lambda x \mapsto x^2 \geq 0$ and $\lambda x \mapsto \bot$.
5. $\lambda x \mapsto x < 3$ and $\lambda x \mapsto x \neq 0$.

Answer Let's check:

```
infix 1 ==>

(==>) :: Bool -> Bool -> Bool
p ==> q = (not p) || q

forall = flip all

stronger :: [a] -> (a -> Bool) -> (a -> Bool) -> Bool
stronger xs p q = forall xs (\ x -> p x ==> q x)

compar :: [a] -> (a -> Bool) -> (a -> Bool) -> String
compar xs p q = let pq = stronger xs p q
                    qp = stronger xs q p
                in
                  if pq && qp then "equivalent"
                  else if pq  then "stronger"
                  else if qp  then "weaker"
                  else             "incomparable"
```

```
test1 = compar [-9..9] (==0) (<=0)
test2 = compar [-9..9] (\x -> x^2 /=4) (/=2)
test3 = compar [-9..9] (\x -> x^2>=49) (\x -> x>=7 || x<= -7)
test4 = compar [-9..9] (\x -> x^2>=0) (\_ -> False)
test5 = compar [-9..9] (\x -> x<3) (\x -> x/=0)
```

This gives:

```
PaperExamAnswers> [test1,test2,test3,test4,test5]
["stronger","stronger","equivalent","weaker","incomparable"]
```

So the answers are:

1. first property stronger,
2. first property stronger,
3. properties equally strong (equivalent),

4. first property weaker.
5. properties incomparable.

Question 2 Suppose function $f : a \to b$ has precondition $p : a \to \text{Bool}$ and postcondition $q : b \to \text{Bool}$. Suppose function $g : b \to c$ has precondition q and postcondition $r : c \to \text{Bool}$. Suppose the precondition of $g \cdot f$ is p. Derive from this a strongest postcondition for the function $g \cdot f$.

Answer If the precondition of $g \cdot f$ is p, this means that p can be assumed to hold for the input x. This means in turn, that q holds of fx, and next, that r holds of $g(fx)$. Thus, the strongest postcondition is r.

Question 3 Consider the following definition of expressions (you have seen this in the course):

```
data Expr = I Int | V String
          | Add Expr Expr
          | Subtr Expr Expr
          | Mult Expr Expr
          deriving (Eq,Show)
```

To evaluate expressions, one needs values for the variables that occur in them. Write a function that collects these variables:

```
vars :: Expr -> [String]
vars ...
```

Answer

```
vars :: Expr -> [String]
vars (I _) = []
vars (V s) = [s]
vars (Add e1 e2)  = nub (vars e1 ++ vars e2)
vars (Mult e1 e2) = nub (vars e1 ++ vars e2)
```

Question 4 Give all binary relations on the set $\{0,1\}$ that are *not* transitive. First express these relations as sets of pairs, next draw pictures of them.

Answer A relation is non-transitive if there is a triple a, b, c with (a, b) and (b, c) in the relation, but (a, c) not in the relation. Therefore any non-transitive relation on $\{0,1\}$ must have $(0,1)$ and $(1,0)$ in the relation. Thus, there are three non-transitive relations on $\{0,1\}$:

$$R_1 = \{(0,1),(1,0)\}, R_2 = \{(0,1),(1,0),(0,0)\}, R_3 = \{(0,1),(1,0),(1,1)\}.$$

Here are pictures:

$$\begin{tikzcd} 0 \arrow[r, bend left] & 1 \arrow[l, bend left] \end{tikzcd} \qquad \begin{tikzcd} 0 \arrow[loop left] \arrow[r, bend left] & 1 \arrow[l, bend left] \end{tikzcd} \qquad \begin{tikzcd} 0 \arrow[r, bend left] & 1 \arrow[l, bend left] \arrow[loop right] \end{tikzcd}$$

Question 5 Consider the following substitution procedure s on binary strings: replace each occurrence of 0 by 01 and each occurrence of 1 by 0. Start with 0, and repeat the procedure. This gives the following list of binary strings.

$$\begin{array}{lclcl}
s^0(0) & & & = & 0 \\
s^1(0) & = & s(0) & = & 01 \\
s^2(0) & = & ss(0) & = & 010 \\
s^3(0) & = & sss(0) & = & 01001 \\
s^4(0) & = & ssss(0) & = & 01001010 \\
s^5(0) & = & sssss(0) & = & 0100101001001 \\
 & \vdots & & &
\end{array}$$

The following implementation of the substitution operation is just for illustration:

```
s :: [Int] -> [Int]
s [] = []
s (0:xs) = 0:1:s xs
s (1:xs) = 0:  s xs

proc = iterate s [0]
```

This gives:

```
*Workshop6> take 7 proc
[[0],[0,1],[0,1,0],[0,1,0,0,1],[0,1,0,0,1,0,1,0],
[0,1,0,0,1,0,1,0,0,1,0,0,1],
[0,1,0,0,1,0,1,0,0,1,0,0,1,0,1,0,0,1,0,1,0]]
```

The Italian mathematician Leonardo Fibonacci, who lived around 1200, proposed this substitution scheme as a model of the breeding behaviour of rabbits. Let 0 represent a pair of mature rabbits and 1 a pair of young rabbits. Every month, each pair of mature rabbits produces a pair of young rabbits. This explains $0 \mapsto 01$. The young rabbits take one month to mature. This explains $1 \mapsto 0$.

Show by induction that the number of ones in $s^n(0)$ equals F_n and the number of zeros in $s^n(0)$ equals F_{n+1}. Here, F is the function for the Fibonacci numbers. Recall that the Fibonacci function F is defined by $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$.

Answer Basis. $s^0(0) = 0$. This string contains $0 = F_0$ ones and $1 = F_1$ zero. This is correct.

Induction step. Assume $s^n(0)$ contains F_n ones and F_{n+1} zeros. We have to show that $s^{n+1}(0)$ contains F_{n+1} ones and F_{n+2} zeros. Observe that each one in $s^{n+1}(0)$ is the result of applying the substitution s to a zero in $s^n(0)$. By the induction hypothesis, there are F_{n+1} zeros in $s^n(0)$, so there are F_{n+1} ones in $s^{n+1}(0)$. Each zero in $s^{n+1}(0)$ either results from a one in $s^n(0)$, or from a zero in $s^n(0)$. By the induction hypothesis, there are F_n ones and F_{n+1} zeros in $s^n(0)$. Thus, there are $F_n + F_{n+1} = F_{n+2}$ zeros in $s^{n+1}(0)$. Done.

Question 6 Consider the following definition of binary trees with information at the internal nodes.

```
data Btree a = Leaf | B a (Btree a) (Btree a) deriving (Eq,Show)
```

Consider the following insertion function.

```
insertT :: Ord a => a -> Btree a -> Btree a
insertT x Leaf = B x Leaf Leaf
insertT x (B y left right)
  | x < y     = B y (insertT x left) right
  | otherwise = B y left (insertT x right)
```

Call a binary tree *ordered* if either it is a single leaf, or its two subtrees are ordered, and moreover all items in its left subtree are less than or equal to the item at its root node, and the item at its root node is less than or equal to all items in its right subtree.

Show with induction on tree structure that if `tree` is ordered, then `insertT x tree` is also ordered.

Answer Basis. The result of inserting x in a leaf tree is the tree `B x Leaf Leaf`. This tree is ordered by definition, for the subtrees are trivially ordered, and it is also trivially true that all items in `Leaf` are less than or equal to x, and that x is less than or equal to all items in `Leaf`.

Induction step. Assume `B y left right` is ordered, and consider the result of inserting `x` in the tree. The induction hypothesis says that the result of inserting x in `left` gives an ordered tree, and so does the result of inserting x in `right`.

Case 1. $x < y$. In this case x gets inserted left. By induction hypothesis, the result of this is an ordered tree. Since x is before y, the tree `B y (insertT x left) right` is ordered.

Case 2. $x \geq y$. In this case x gets inserted right. By induction hypothesis, the result of this is an ordered tree. Since $x \geq y$, the tree `B y left (insertT x right)` is ordered.