Paper Exam Software Specification and Testing, October 19 , 2015
With Answers

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. $\lambda x \mapsto x^2 \neq 9$ and $\lambda x \mapsto x \neq 3$.
3. $\lambda x \mapsto x^2 \geq 16$ and $\lambda x \mapsto x \geq 4 \vee x \leq -4$.
4. $\lambda x \mapsto x^2 \geq 0$ and $\lambda x \mapsto \top$.
5. $\lambda x \mapsto x < 3$ and $\lambda x \mapsto x \neq 0$.

Answer

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

Question 2 Give all equivalence relations on the set $\{0, 1, 2\}$. Express these relations as sets of pairs, and draw pictures of them.

Answer

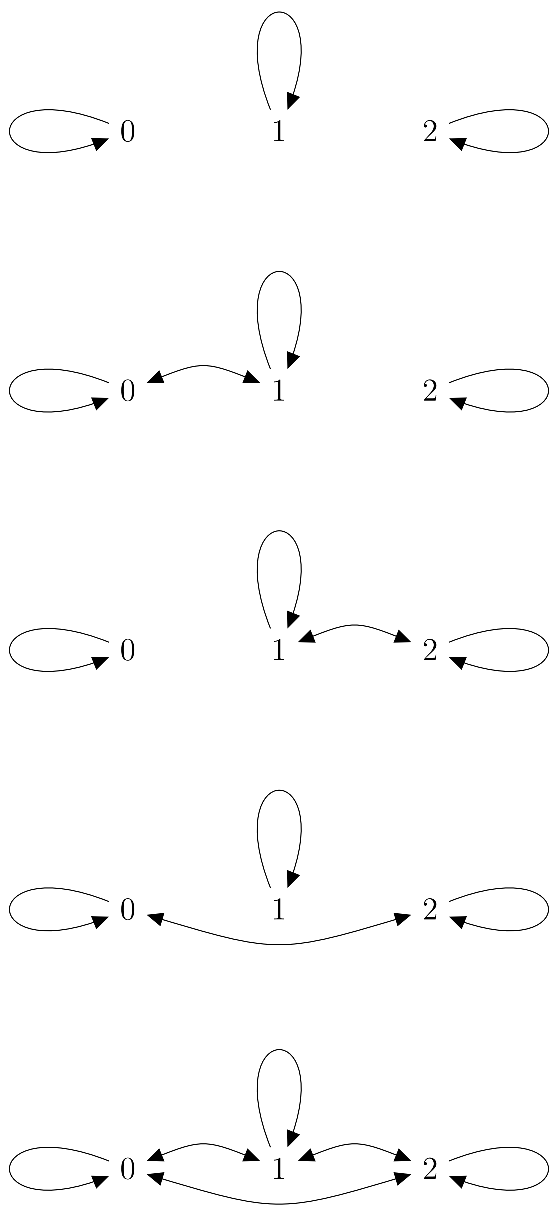
There are five, corresponding to the five partitions of $\{0, 1, 2\}$:

$$\{\{0\}, \{1\}, \{2\}\}, \{\{0, 1\}, \{2\}\}, \{\{0\}, \{1, 2\}\}, \{\{1\}, \{0, 2\}\}, \{\{0, 1, 2\}\}.$$

As sets of pairs:

$$
\begin{aligned}
&\{(0,0),(1,1),(2,2)\},\\
&\{(0,0),(0,1),(1,0),(1,1),(2,2)\},\\
&\{(0,0),(1,1),(1,2),(2,1),(2,2)\},\\
&\{(0,0),(0,2),(1,1),(2,0),(2,2)\},\\
&\{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\}.
\end{aligned}
$$

In a picture:



Question 3 Let the relation R on the set $A = \{0, 1, 2, 3, 4\}$ be given by

$$R = \{(0, 1), (1, 0), (2, 3), (3, 4), (4, 2)\}.$$

1. Determine $R \cup R^2$
2. Determine $R \cup R^2 \cup R^3$
3. Determine $R \cup R^2 \cup R^3 \cup R^4$
4. Which of these, if any, is the transitive closure of R?

Answer

1. $R \cup R^2 = \{(0, 1), (1, 0), (2, 3), (3, 4), (4, 2), (0, 0), (1, 1), (2, 4), (3, 2), (4, 3)\}$,
2. $R \cup R^2 \cup R^3 = \{(0, 1), (1, 0), (2, 3), (3, 4), (4, 2), (0, 0), (1, 1), (2, 4), (3, 2), (4, 3), (2, 2), (3, 3), (4, 4)\}$.
3. $R \cup R^2 \cup R^3 \cup R^4 = \{(0, 1), (1, 0), (2, 3), (3, 4), (4, 2), (0, 0), (1, 1), (2, 4), (3, 2), (4, 3), (2, 2), (3, 3), (4, 4)\}$.
4. In the step from $R \cup R^2 \cup R^3$ to $R \cup R^2 \cup R^3 \cup R^4$ no pairs get added. This indicates that both $R \cup R^2 \cup R^3$ and $R \cup R^2 \cup R^3 \cup R^4$ give the transitive closure of R.

Question 4 Let `factors :: Integer -> [Integer]` be a function that computes the list of (prime) factors of an integer. For example, the output for `factors 12` would be `[2,2,3]`.

Here is an assertion wrapper, as discussed in the course:

```
assert :: (a -> b -> Bool) -> (a -> b) -> a -> b
assert p f x = if p x (f x) then f x
                            else error "assert"
```

What would be a reasonable assertive version of the `factors` function?

```
factorsA :: Integer -> [Integer]
factorsA = assert ... factors
```

Fill in the dots, and explain.

Answer

Example implementation of `factors`, together with an assertive version:

```
factors :: Integer -> [Integer]
factors n = factors' n 2 where
  factors' 1 _ = []
  factors' n m
    | n `mod` m == 0 = m : factors' (n `div` m) m
    | otherwise      =     factors' n (m+1)

factorsA :: Integer -> [Integer]
factorsA = assert (\ x xs -> x == product xs) factors
```

Explanation: the product of all factors should give us the number back.

Question 5 Recall from the Lab Exam this morning:

A *frequency table* is a list of pairs `(char,int)`, where `int` specifies the number of occurrences of `char` in some string.

A *Huffman tree* is a binary tree with characters at the leaf nodes, and weight information (given by an integer) at every node.

```
data HTree  = Leaf Char Int
            | Fork HTree HTree Int
            deriving (Show)
```

The weight of a tree is given by:

```
weight :: HTree -> Int
weight (Leaf _ w)    = w
weight (Fork _ _ w)  = w
```

Call the following property the *Huffman property*:

```
prop_huffman :: HTree -> Bool
prop_huffman (Leaf _ _) = True
prop_huffman (Fork t1 t2 w) = prop_huffman t1 && prop_huffman t2
                     && weight t1 + weight t2 == w
```

Suppose we take care to build Huffman trees with the following tree `merge` function:

```
merge t1 t2 = Fork t1 t2 (weight t1 + weight t2)
```

Show by means of an induction proof that

```
createTree :: [(Char,Int)] -> HTree
createTree [(c,i)] = Leaf c i
createTree ((c,i):t) = merge (Leaf c i) (createTree t)
```

for any non-empty frequency table `t` creates a tree `createTree t` that satisfies the Huffman property.

Answer

Induction proof of the fact that for all frequency tables `t`, type, `createTree t` has the Huffman property.

Basis: the input is a unit list `[(c,i)]`. In this case `createTree t` equals `Leaf c i`, and this Huffman tree has the Huffman property by definition.

Induction step. Assume that the property holds for any frequency table `t` of length n. Now consider a table `(c,i):t` of length $n + 1$. We can assume, by the induction hypothesis, that `createTree t` has the Huffman property.

Now observe that `createTree (c,i):t` equals `merge (Leaf c i) (createTree t)`, by the second clause in the definition of `createTree`.

By the induction hypothesis, `createTree t` has the Huffman property. By the definition of `merge`, `merge  (Leaf c i) (createTree t)` creates a new tree with a weight that is the sum of i and the weight of `createTree t`. But this means that the result has the Huffman property. QED.