Week 2 test report

by Tuba Kaya Chomette, Sander Leer, Martijn Stegeman

Question 1 - Triangles

Triangles are represented using a triple of integer values and defined to be in one of five categories: *not a triangle*, *equilateral*, *rectangular*, *isosceles* and *other*. Because some triangles can by mathematical definition be in more than category, they are assigned such a category in the stated order.

Checking the program that assigns the correct category to a triangle based upon the lengths of its sides is done as follows:

`NoTriangle`

We systematically check all combinations of triangles with lengths 1 where one or more of the lengths are replaced by a zero.

```
1 1 0
1 0 1
0 1 1
1 0 0
0 1 0
0 0 1
0 0 0
```

Also, lengths cannot be negative by definition, so there are three cases to verify that negative lengths yield `NoTriangle`[^1].

[^1]: This is not in the actual implementation of `triangle`, so the check fails.

```
 1  1 -1
 1 -1  1
-1  1  1
```

We add one case to check that a triangle with all non-zero lengths is *not* awarded the `NoTriangle` predicate.

```
1 1 1
```

`Equilateral`

First, we explicitly check that a proposed triangle with one or three zero lengths are *not* awarded the `Equilateral` predicate.

This is useful for cases where a definition of `NoTriangle` is removed. In that case, the definition of `Equilateral` should stand. This check can be done more systematically by including all cases that check the definition of `NoTriangle`.

After those, we check cases where the lengths are not equal (all three different, or just one), which are by definition not `Equilateral`.

Finally, we check three cases where the lengths are actually equal.

`Rectangular`

First, we check again for cases where there is a zero side, which should yield `NoTriangle`, but more

importantly here, should *not* yield `Rectangular`.

Then we check two known rectangular triangles. The first one, $$(3, 4, 5)$$, is permuted in order to check that order is not relevant for the designation of `Rectangular`.

`Isosceles`

Here, we included cases of `Equilateral` triangles as hinted upon in the assignment, that should not yield `Isosceles`.

Then there are three cases that check if the order is disregarded in yielding `Isosceles`.

`Other`

`Other` only contains one case for every other category in order to check that it does not yield `Other`.

Analysis

Because the categories are awarded in a certain order, all combinations of three lengths can be divided into five non-overlapping classes of triangles. In order to verify correct non-overlapping assignment of each of the classes, the checks for each class should contain positive as well as negative cases.

For the `NoTriangle` case, systematically checking all variations is relatively easy, because generating all cases with one or more zeroes is trivial. However, we do not check any cases where lengths other than 0 or 1 are involved.

Lengths are not limited (they are positive integers), so the checks are bound by the fact that we can only generate a limited amount of checks by hand. Even when computationally generating the checks cases according to a pattern, we can only run a finite amount of them if we want to be able to actually draw a conclusion.

In all of the categories, order of lengths is unimportant (this is a property of the triangle). This can be checked by generating all permutations of relevant check cases.

Question 2 - Formula properties

Using the sample formulae `form1`, `form2` and `form3`, we can only check the correctness of our programs for an arbitrary amount (and kind) of cases.

To check more rigorously, we use formulae of which we hand-checked the properties using truth tables during the workshop session.

For the logical equivalence we also add cases for all theorems which are defined for equivalence in propositional logic (see Theorem 2.10, Haskell Road to Logic, p.46).

Then we send these formulas as parameters to the definitions and checked whether the results are in line with our expectations.

Question 3 - CNF conversion

To verify the correct output of the CNF converter, we simply check equivalence of the input formula with the converted formula, for the three formulae that came with `Week2`, as well as the trivial formulae containing only one connective.