

Software Testing Assignment 2

Cindy Berghuizen, Omar Pakker , Chiel Peter, Maria Gouseti

10 September , 2013

Triangle Exercise

Time spent: 1 hour

```
triangle :: Integer -> Integer -> Integer -> Shape
triangle x y z
  | (a + b <= c) || (a < 1)      = NoTriangle
  | (a == b && b == c)           = Equilateral
  | ((a^2 + b^2) == c^2)        = Rectangular
  | (a == b) || (b == c)        = Isosceles
  | otherwise                   = Other
  where [a, b, c] = sort [x, y, z]

testTriangle :: Shape -> Integer -> [(Integer, Integer, Integer), Shape]
testTriangle s maxN = [(a, b, c), s] | a <- [1..maxN],
                                     b <- [a..maxN],
                                     c <- [b..maxN],
                                     triangle a b c == s]
```

The function testTriangle takes two input arguments. The first is the type of triangle and the second is the maximum side length. To avoid permutations (b,c) only range over there value of there predecessor and the maximum value as these permutations are mapped to the same values by the sort algortihm inside the triangle function. 10 was chosen as a maximum value to keep the results acceptable to manual inspection. The results are shown below and where verified to be correct.

```

Type :? for help
Lab2> testTriangle NoTriangle 10
[( (1,1,2),NoTriangle), ( (1,1,3),Isosceles), ( (1,1,4),NoTriangle), ( (1,1,5),NoTriangle), ( (1,1,6),NoTriangle), ( (1,1,7),NoTriangle), ( (1,1,8),NoTriangle),
( (1,1,9),NoTriangle), ( (1,1,10),NoTriangle), ( (1,2,3),NoTriangle), ( (1,2,4),NoTriangle), ( (1,2,5),NoTriangle), ( (1,2,6),NoTriangle), ( (1,2,7),NoTriangle),
( (1,2,8),NoTriangle), ( (1,2,9),NoTriangle), ( (1,2,10),NoTriangle), ( (1,3,4),NoTriangle), ( (1,3,5),NoTriangle), ( (1,3,6),NoTriangle), ( (1,3,7),NoTriangle),
( (1,3,8),NoTriangle), ( (1,3,9),NoTriangle), ( (1,3,10),NoTriangle), ( (1,4,5),NoTriangle), ( (1,4,6),NoTriangle), ( (1,4,7),NoTriangle), ( (1,4,8),NoTriangle),
( (1,4,9),NoTriangle), ( (1,4,10),NoTriangle), ( (1,5,6),NoTriangle), ( (1,5,7),NoTriangle), ( (1,5,8),NoTriangle), ( (1,5,9),NoTriangle), ( (1,5,10),NoTriangle),
( (1,6,7),NoTriangle), ( (1,6,8),NoTriangle), ( (1,6,9),NoTriangle), ( (1,6,10),NoTriangle), ( (1,7,8),NoTriangle), ( (1,7,9),NoTriangle), ( (1,7,10),NoTriangle),
( (1,8,9),NoTriangle), ( (1,8,10),NoTriangle), ( (1,9,10),NoTriangle), ( (2,2,4),NoTriangle), ( (2,2,5),NoTriangle), ( (2,2,6),NoTriangle), ( (2,2,7),NoTriangle),
( (2,2,8),NoTriangle), ( (2,2,9),NoTriangle), ( (2,2,10),NoTriangle), ( (2,3,5),NoTriangle), ( (2,3,6),NoTriangle), ( (2,3,7),NoTriangle), ( (2,3,8),NoTriangle),
( (2,3,9),NoTriangle), ( (2,3,10),NoTriangle), ( (2,4,6),NoTriangle), ( (2,4,7),NoTriangle), ( (2,4,8),NoTriangle), ( (2,4,9),NoTriangle), ( (2,4,10),NoTriangle),
( (2,5,7),NoTriangle), ( (2,5,8),NoTriangle), ( (2,5,9),NoTriangle), ( (2,5,10),NoTriangle), ( (2,6,8),NoTriangle), ( (2,6,9),NoTriangle), ( (2,6,10),NoTriangle),
( (2,7,9),NoTriangle), ( (2,7,10),NoTriangle), ( (2,8,10),NoTriangle), ( (3,3,6),NoTriangle), ( (3,3,7),NoTriangle), ( (3,3,8),NoTriangle), ( (3,3,9),NoTriangle),
( (3,3,10),NoTriangle), ( (3,4,7),NoTriangle), ( (3,4,8),NoTriangle), ( (3,4,9),NoTriangle), ( (3,4,10),NoTriangle), ( (3,5,8),NoTriangle), ( (3,5,9),NoTriangle),
( (3,5,10),NoTriangle), ( (3,6,9),NoTriangle), ( (3,6,10),NoTriangle), ( (3,7,10),NoTriangle), ( (4,4,8),NoTriangle), ( (4,4,9),NoTriangle), ( (4,4,10),NoTriangle),
( (4,5,9),NoTriangle), ( (4,5,10),NoTriangle), ( (4,6,10),NoTriangle), ( (5,5,10),NoTriangle) ]
Lab2> testTriangle Equilateral 10
[( (1,1,1),Equilateral), ( (2,2,2),Equilateral), ( (3,3,3),Equilateral), ( (4,4,4),Equilateral), ( (5,5,5),Equilateral), ( (6,6,6),Equilateral), ( (7,7,7),Equilateral),
( (8,8,8),Equilateral), ( (9,9,9),Equilateral), ( (10,10,10),Equilateral) ]
Lab2> testTriangle Rectangular 10
[( (3,4,5),Rectangular), ( (6,8,10),Rectangular) ]
Lab2> testTriangle Isosceles 10
[( (1,2,2),Isosceles), ( (1,3,3),Isosceles), ( (1,4,4),Isosceles), ( (1,5,5),Isosceles), ( (1,6,6),Isosceles), ( (1,7,7),Isosceles), ( (1,8,8),Isosceles),
( (1,9,9),Isosceles), ( (1,10,10),Isosceles), ( (2,2,3),Isosceles), ( (2,3,3),Isosceles), ( (2,4,4),Isosceles), ( (2,5,5),Isosceles), ( (2,6,6),Isosceles),
( (2,7,7),Isosceles), ( (2,8,8),Isosceles), ( (2,9,9),Isosceles), ( (2,10,10),Isosceles), ( (3,3,4),Isosceles), ( (3,3,5),Isosceles), ( (3,4,4),Isosceles),
( (3,5,5),Isosceles), ( (3,6,6),Isosceles), ( (3,7,7),Isosceles), ( (3,8,8),Isosceles), ( (3,9,9),Isosceles), ( (3,10,10),Isosceles), ( (4,4,5),Isosceles),
( (4,4,6),Isosceles), ( (4,4,7),Isosceles), ( (4,5,5),Isosceles), ( (4,6,6),Isosceles), ( (4,7,7),Isosceles), ( (4,8,8),Isosceles), ( (4,9,9),Isosceles),
( (4,10,10),Isosceles), ( (5,5,6),Isosceles), ( (5,5,7),Isosceles), ( (5,5,8),Isosceles), ( (5,5,9),Isosceles), ( (5,6,6),Isosceles), ( (5,7,7),Isosceles),
( (5,8,8),Isosceles), ( (5,9,9),Isosceles), ( (5,10,10),Isosceles), ( (6,6,7),Isosceles), ( (6,6,8),Isosceles), ( (6,6,9),Isosceles), ( (6,6,10),Isosceles),
( (6,7,7),Isosceles), ( (6,8,8),Isosceles), ( (6,9,9),Isosceles), ( (6,10,10),Isosceles), ( (7,7,8),Isosceles), ( (7,7,9),Isosceles), ( (7,7,10),Isosceles),
( (7,8,8),Isosceles), ( (7,9,9),Isosceles), ( (7,10,10),Isosceles), ( (8,8,9),Isosceles), ( (8,8,10),Isosceles), ( (8,9,9),Isosceles), ( (8,10,10),Isosceles),
( (9,9,10),Isosceles), ( (9,10,10),Isosceles) ]
Lab2> testTriangle Other 10
[( (2,3,4),Other), ( (2,4,5),Other), ( (2,5,6),Other), ( (2,6,7),Other), ( (2,7,8),Other), ( (2,8,9),Other), ( (2,9,10),Other), ( (3,4,6),Other), ( (3,5,6),Other),
( (3,5,7),Other), ( (3,6,7),Other), ( (3,6,8),Other), ( (3,7,8),Other), ( (3,7,9),Other), ( (3,8,9),Other), ( (3,8,10),Other), ( (3,9,10),Other), ( (4,5,6),Other),
( (4,5,7),Other), ( (4,5,8),Other), ( (4,6,7),Other), ( (4,6,8),Other), ( (4,6,9),Other), ( (4,7,8),Other), ( (4,7,9),Other), ( (4,7,10),Other), ( (4,8,9),Other),
( (4,8,10),Other), ( (4,9,10),Other), ( (5,6,7),Other), ( (5,6,8),Other), ( (5,6,9),Other), ( (5,6,10),Other), ( (5,7,8),Other), ( (5,7,9),Other), ( (5,7,10),Other),
( (5,8,9),Other), ( (5,8,10),Other), ( (5,9,10),Other), ( (6,7,8),Other), ( (6,7,9),Other), ( (6,7,10),Other), ( (6,8,9),Other), ( (6,9,10),Other), ( (7,8,9),Other),
( (7,8,10),Other), ( (7,9,10),Other), ( (8,9,10),Other) ]
Lab2>

```

Logic

Time spent: 1 hour

```

contradiction :: Form -> Bool
contradiction f = not (satisfiable f)

tautology :: Form -> Bool
tautology f = all (\ v -> eval v f) (allVals f)

entails :: Form -> Form -> Bool
entails a b = all (\ v -> eval v f) (allVals f) where f = (Impl a b)

equiv :: Form -> Form -> Bool
equiv a b = all (\ v -> eval v f) (allVals f) where f = (Equiv a b)

```

Testing can be done by inserting propositions which we have already validated by hand. formC is a contradiction, formT is a tautology and formS is a proposition which is satisfiable. For equiv we can check DeMorgan Law's because we already know they are equivalent. DeMorgan can also be used for the entails function. Because if \leftrightarrow holds than \rightarrow will also hold.

```

formC = Cnj[p, (Neg p)]
formT = Equiv (Impl p q)(Impl (Neg q)(Neg p))
formS = Dsj[p, (Neg q)]
deMorgan1 = Neg (Cnj[p,q])
deMorgan2 = Dsj[Neg p, Neg q]
deMorgan3 = Neg (Dsj[p,q])
deMorgan4 = Cnj[Neg p, Neg q]

```

```

*Lab2> tautology formS
False
*Lab2> tautology formC
False
*Lab2> tautology formI
True
*Lab2> contradiction formS
False
*Lab2> contradiction formC
True
*Lab2> contradiction formI
False
*Lab2> equiv deMorgan1 deMorgan2
True
*Lab2> equiv deMorgan2 deMorgan3
False
*Lab2> entails deMorgan3 deMorgan4
True
*Lab2> entails deMorgan1 deMorgan4
False
*Lab2>

```

Conjunctive Normal Form (CNF)

Time spent: 3 hours

```

-- Precondition: Form is arrowfree and in negative normal form
-- Postcondition: Form is in conjunctive normal form
cnf :: Form -> Form
cnf (Prop x)           = Prop x
cnf (Neg (Prop x))     = Neg (Prop x)
cnf (Cnj f)            = Cnj (map cnf f)
cnf (Dsj [f, g])       = dist (cnf f) (cnf g)
cnf (Dsj (f:fs))       = dist (cnf f) (cnf (Dsj fs))

-- Precondition: Forms are in conjunctive normal form
-- Postcondition: Form is the the conjunctive normal form of (form1 v form2)
dist :: Form -> Form -> Form
dist (Cnj fs) g        = Cnj (map (dist g) fs)
dist f (Cnj gs)        = Cnj (map (dist f) gs)
dist f g               = Dsj [f,g]

```

In order to define test cases arbitrary formulas need to be defined first and converted to conjunctive normal form manually. In the workshop four functions were defined and converted to CNF therefore these four functions will be used to test our CNF. The four test cases are:

```

test1 = Neg (Neg (Neg p))
test2 = Neg (Dsj [p, Neg q])
test3 = Neg (Cnj [Neg p, Neg q])
test4 = Equiv (Impl p q) (Impl (Neg q) (Neg p))

```

The results are stated in the picture below and are verified by hand.

```

Lab2> cnf (nnf (arrowfree test1))
-1
Lab2> cnf (nnf (arrowfree test2))
*(-1 2)
Lab2> cnf (nnf (arrowfree test3))
+(1 2)
Lab2> cnf (nnf (arrowfree test4))
*(*(*(+((-1 2) 1) +((-1 2) -2)) *((+((-1 2) -2) +((-1 2) 1)))
*(+((2 -1) 1) +((2 -1) -2)) *((+((2 -1) -2) +((2 -1) 1))))

```

To get the supplied formula to match with the pre-condition for 'cnf', we chain the 'arrowfree' and 'nnf' functions from Week2.hs:

arrowfree

```
-- Precondition: Any formula
-- Postcondition: A formula with the arrows (=> and <==>) replaced with disjunctions.
```

nnf

```
-- Precondition: Arrow free formula
-- Postcondition: Negative normal form of the arrow free formula
```

The combination of 'arrowfree' and 'nnf' gives us a pre-condition where any formula can be used and a post-condition of an arrow free, negative normal formula which is what we need in our 'cnf' function.

Time spent making document and finishing it: 2 hours

Total time spent: 7 hours