Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Module: CH-232
Date: 2022-10-14
Due: 2022-10-21

ICS 2022 Problem Sheet #6

Problem 6.1: *brandy after dinner* (1+3+1+3 = 8 points)

Leo, Mark and Nick often have dinner together, but we do not know who likes to have a brandy after the dinner. However, we know the following:

1. If Leo drinks a brandy, then so does Mark.
2. Mark and Nick never drink brandy together.
3. Leo or Nick drink a brandy (alone or together).
4. If Nick drinks a brandy, then so does Leo.

We introduce three boolean variables: The variable L is true if Leo enjoys a brandy, the variable M is true if Mark enjoys a brandy, and the variable N is true if Nick enjoys a brandy.

a) Provide a boolean formula for the function $D(L, M, N)$ that captures the three rules.

b) Construct a truth table that shows all interpretations of $D(L, M, N)$. Break things into meaningful steps so that we can award partial points in case things go wrong somewhere.

c) Out of the truth table, derive a simpler boolean formula defining $D(L, M, N)$.

d) Take the boolean formula from a) and algebraically derive the simpler boolean formula from c). Annotate each step of your derivation with the equivalence law that you apply so that we can follow along.

Problem 6.2: *brandy after dinner* (1+1 = 2 points)

Leo, Mark and Nick often have dinner together, but we do not know who likes to have a brandy after the dinner. However, we know the following:

1. If Leo drinks a brandy, then so does Mark.
2. Mark and Nick never drink brandy together.
3. Leo or Nick drink a brandy (alone or together).
4. If Nick drinks a brandy, then so does Leo.

We introduce three boolean variables: The variable L is true if Leo enjoys a brandy, the variable M is true if Mark enjoys a brandy, and the variable N is true if Nick enjoys a brandy.

a) Define a function

```
brandy :: Bool -> Bool -> Bool -> Bool
```

that implements the rules directly following the description given above and a function

```
brandy' :: Bool -> Bool -> Bool -> Bool
```

that implements a simplified boolean formula.

b) Define a function

```
truthTable :: (Bool -> Bool -> Bool -> Bool) -> [(Bool, Bool, Bool, Bool)]
```

that takes a (brandy) function as an argument and returns a list where each element is a 4-tuple representing three input values passed to the (brandy) function followed by the function's result.

Submit your Haskell code as a plain text file.

Below is a unit test template that you can use to fill in your code.

```haskell
module Main (main) where

import Test.HUnit

-- The function brandy implements the rules directly.
brandy :: Bool -> Bool -> Bool -> Bool
brandy l m n = undefined

-- The function brandy' implements the simplified formula.
brandy' :: Bool -> Bool -> Bool -> Bool
brandy' l m n = undefined

-- The function truthTable takes a function as an argument and returns
-- a list where each element is a 4-tuple representing three input
-- values passed to the function followed by the function's result.
truthTable :: (Bool -> Bool -> Bool -> Bool) -> [(Bool, Bool, Bool, Bool)]
truthTable f = undefined

-- Test whether the two truth tables returned are the same (which is
-- not a very sharp test but I do not want to reveal too many details).
-- You may want to add your own test cases...
brandyTests = TestList [ truthTable brandy ~?=  truthTable brandy' ]

main = runTestTT brandyTests
```