COMP2209 Programming III Coursework Report

Jack Chiplin (jc16g22

January 2024

1 Development and Testing

Each of the challenges in this coursework I completed by testing the code as I created the function. Once the function was complete, I created a testing file, Tests.hs, that imports the functions from Challenges and tests whether different cases output the correct answer. Within the test file, the correct answers were calculated by manually working through the problem. For each exercise, there is a function which can be run with no arguments that outputs a tuple containing Boolean values. Each value should evaluate to True if the functions are working correctly.

Challenge one was completed using two main functions, one of which checks whether all sources and sinks are linked and the other checks that every wire is connected. I tested many cases that are not fully complete but have certain aspects to them which are complete. I also tested several complete puzzles of different varieties to ensure they were counted as complete.

Challenge two was harder and I applied a brute-force method to complete it. I tested a couple of unsolvable puzzles, one of which could be come connected but had no sinks or sources and one of which had the opposite. I also tested many completable puzzles of different sizes containing different tiles, loops, several sinks and sources and more.

Challenge three was much easier and most of it could be completed within one function using pattern-matching. The testing for this challenge was also

easy as all I needed to do was test each pattern and make sure the output was correct. One important thing to test was nested abstractions as they format very differently as a string than as a LExpr.

Challenge four was completed using separate parsers for each type of expression. The testing process was very similar to the previous challenge; just test each different case separately and ensure that everything outputs correctly.

Challenge five was a similar pattern-matching exercise to challenge three. The variables evaluate differently to the coursework specification; anything (Var x) or (V x) maintain the same name through conversion and any newly-created variables are given a unique name as not to interfere with each other. My answers are alpha-equivalent to the coursework specification examples. The testing process was very similar to the previous two challenges as it is just testing different cases and ensuring that variables are given correct names.

2 Bibliography

The 'many1' and 'chainl1' functions were adapted from the following source: Hutton, G. & Meijer, E. (1998) - "Functional Pearls: Monadic Parsing in Haskell"