Functional Programming: Assignment 1

Author: Brian Mc George (MCGBRI004)

Date: 17-04-2015

Important Notices

Variations of generateSuccessorStates

- 2 variations of generateSuccessorStates were created:
    - *generateSuccessorStates*
    - *generateOptimisedSuccessorStates*
- *generateSuccessorStates* produces the values expected by the auto-marker and the provided unit tests.
- *generateOptimisedSuccessorStates* implements the optimisation mentioned in Question 4 whereby rotations that undo the last move are ignored.

Variations of genStates

- 4 variations of *genStates* were created.
- The order in which the variations were initially written is as follows:
    1. *genStatesTailRecursive*
    2. *genStatesOptimisedTailRecursive*
    3. *genStates*
    4. *genStatesOptimised*
- *genStatesTailRecursive* was the initial implementation of genStates, it is a tail recursive implementation and produces the values expected by the auto-marker and the provided unit tests. It was very slow for n >= 7 and hence not used as the primary *genStates*.
- *genStatesOptimisedTailRecursive* is *genStatesTailRecursive* that uses *generateOptimisedSuccessorStates* instead of *generateSuccessorStates*.
- *genStates* is a **non** tail recursive implementation and produces the values expected by the auto-marker and the provided unit tests. It was written in a completely different way as I was not happy with the performance of *genStatesTailRecursive*. In order to obtain this improvement in completion time a non-tail recursive approach was taken. Some memory efficiency was sacrificed to get a major improvement in completion time (see theoretical questions pdf for more). In testing it was much faster than *genStatesTailRecursive* and had acceptable memory usage for n <= 7
- *genStatesOptimised* makes use of *generateOptimisedSuccessorStates* to further speed up completion time and reduce memory usage.

Variations of solveCube

- 3 variations of solveCube were created:
    - *solveCube*
    - *solveCubeSafe*
    - *solveCubeSlow*
- *solveCube* makes use of *genStatesOptimised* to provide the fastest completion time
- *solveCubeSafe* makes use of *genStatesOptimisedTailRecursive* to provide decent performance as well as a tail recursive approach
- *solveCubeSlow* makes use of *genStatesTailRecursive* and is very slow for n>=7

Trace of functions

All the main functions include a trace to show that the function is tail recursive. Please note that *genStates* and *genStatesOptimised* is **not** tail recursive by design. The trace output is located in the *trace_output* folder.

Theoretical Questions

The theoretical questions are answered in *Theortical_Questions.pdf*. It also includes some additional comparisons and analysis based on the questions provided.

Test System & Interpreter

The code was written and run on Windows 8.1 x64 using the Gambit interpreter (v 4.7.4). The code has been tested to run on the gambit interpreter on Ubuntu in the senior lab.

Files provided

- assignment3.scm
- Theoretical_Questions.pdf
- Readme.md
- Readme.pdf
- Theoretical_Questions.tex
- trace_output
    - rotate.out
    - generateSuccessor.out
    - generateOptimisedSuccessor.out
    - genStates.out
    - genStatesOptimised.out
    - genStatesTailRecursive.out
    - genStatesOptimisedTailRecursive.out
    - solveCube.out

    - solveCubeSafe.out
    - solveCubeSlow.out
- memory_usage.jpg
- mem_usage2.jpg
- mem_usage3.jpg
- theory_data.dat
- theory_data_optimised.dat
- theory_data.xlsx
- .gitignore