Riley Wood

EE26 Chang

4/2/15

Quine McCluskey method implemented in Python

Contained in this folder is a Python implementation of the Quine-McCluskey method for reducing single-output Boolean functions.

The code was written using the PyCharm IDE for Python. An object-oriented approach was taken in its design. Classes were created to encapsulate the functionality of NCubes, Minterms, Boolean functions, and the steps of the Quine McCluskey method itself. These pieces then fit together quite nicely.

FILES:

* main.py – The file to run at the command line which will accept input and then generate and print the solutions to the function it gets in.
* Function.py – Contains the Function class, which models all attributes of a Boolean algebra function
* inputParser.py – Receives input and parses them to Boolean functions from either the command line or a file
* Minterm.py – Contains the Minterm class, which models a Boolean Minterm
* NCube.py – Contains the NCube class, which models a Boolean NCube. Offers functionality to check if one NCube is adjacent to (and thus able to be combined with) another NCube, among other things
* QuineMcCluskey.py – Contains functions to perform each step of the QM method. Operates on Function, Minterm, and NCube objects.
* SolutionPrinter.py – Takes the output of the QuineMcCluskey solver and prints it to the console in the specified format.
* testinput.txt – A file containing Boolean functions to be passed into the program for demo purposes

HOW TO RUN:

This program accepts input in two ways:

From a file:

Run **python main.py FILEPATH/FILENAME.TXT**

Where FILENAME.TXT is a plaintext file containing one function per line in the format:

m(0,2,4,5, ...) + d(1,3,6, …)

Where the minterms appear in the parentheses prefixed with an “m” and the don’t care states appear in the parentheses prefixed with a “d”.

From the command line:

Run **python main.py**

The program will prompt the user for functions one at a time in the format:

m(0,2,4,5, ...) + d(1,3,6, …)

Where the minterms appear in the parentheses prefixed with an “m” and the don’t care states appear in the parentheses prefixed with a “d”.

The user must type “done” and press Enter once all functions have been entered for the solver to begin operation.

OUTPUT OF THE PROGRAM:

For each function, the program will print a representation of the function with minterms and don’t cares in ascending order. On the next line it will print the SOP minimization. On the line after that it will print the POS minimization. Finally it will print the time taken to compute both minimizations for that function. It will also print a string of dashes in order to separate function solutions visually.

TIME COMPLEXITY:

Because solving Boolean functions for a minimal implementation is an NP problem and the number of operations increases rapidly with the number of inputs, certain functions may take a long time to simplify. On a relatively powerful desktop computer, I was able to simplify the function

f = m(0,512)

and get both its SOP and POS representations in about 2.5 minutes. I expect on an average computer it would take longer. This is a function containing 10 inputs, which is all we are meant to be responsible for, but I expect other functions with 10 inputs may take longer. With this kind of problem, you come up against the limits of the computer’s processing power. I think my implementation is fairly efficient and even so, it still lags on larger inputs such as the one above. I’m hoping this will be acceptable given that the problem being solved by the computer is a very difficult one.

HANDLING MULTIPLE SOLUTIONS:

Sometimes there are multiple minimizations of the same cost. In these cases, the program will choose one of these lowest-cost solutions and report it as the solution to the function. It does not report all minimizations. This may lead to the situation where the solution for a particular function is listed elsewhere as one thing and this program will give another answer. Rather than assume the program’s answer is wrong, one should look into whether or not the program’s solution is any more costly that the printed solution while still covering all minterms. An example is this function from the project specification:

m(1,5,3)+d(2,4) is minimized to A'C + AB' by my program, however the project specification gives B’C+A’C as the minimization. In reality, both of these solutions are correct and both are of the same cost, so this program’s answer should be acceptable.