COMP2K

Game of Life

Laboratory

Instructions

Game of Life (GoL)
and Turing Completeness

Recommended you complete this part by the end of Week 7. You should demo this lab in your Week 8 or 9 practical sessions.

Version 1.1

Game of Life and Turing Completeness

[This is the lab sheet for the Game of Life Demonstration. You must demonstrate it to the instructor in one of your practical sessions BEFORE the due date in order to be awarded marks. Please check the ECP for the correct due date. Note that sections are 'complete' and marks are awarded by attempting each task AND correctly answering related questions to the satisfaction of the instructor.]

After John von Neumann designed the computer, he envisioned the next evolution of Turing machines given Alonzo Church's and Alan Turing's result on the Decision Problem and Universal Computation. In his attempt to create a simplified simulation of the human brain, he proposed the use of a 'universe' of cells (i.e. an 2D array of values) that communicate with themselves through their neighbours with a set of rules. We used the idea of the Universal Turing Machine to show that a machine could exist within this universe that could self-replicate by maintaining and executing a set of instructions on a tape that provide the necessary rules to allow it to do so. He would create a theory of self-replicating machines that uses a complex set of 29 rules that achieved this and created the area we know as cellular automation in the process (Von Neumann, 1966).

Over the latter half of the 20th century, many researchers have sought to simplify these rules, since von Neumann rules were so computationally complex that it was not realized until the 1990s. The Game of Life (GoL) originally developed by John Conway is the culmination of that research where the rules have been reduced to simply a few rules. The game involves a set of cells within an $N \times N$ grid whose state is either alive or dead (i.e. 1 or 0 respectively). The grid effectively represents the 'universe' that will be simulated and the alive cells the life within it. The game is governed by a set of simple rules that dictate the next state of each cell in this universe depending on its current state and the states of its neighbours. The rules of GoL depend on the 8-connected neighbours of a cell as follows:

1. **Underpopulation:** A live cell that has < 2 live neighbouring cells will die
2. **Survival:** A live cell that has 2-3 live neighbouring cells will remain alive
3. **Overpopulation:** A live cell with more than 3 live neighbours will die
4. **Reproduction:** A dead cell with exactly 3 live neighbours will become alive

The game begins with an initial state of the universe with a pattern of live cells. The universe is evolved by applying the above rules to each cell of the universe to determine the next iteration of the simulation. The evolution of the universe is observed by continual computing the next iteration of the universe. See chapter 7, section 7.6.4 of (Moore and Mertens, 2011) for more theoretical details.

In this laboratory, you will create a simulation of the GoL using Python based on an initial class provided. In the following parts of the lab, you will be required to code up the algorithms related to the computation, importation and evaluation of the GoL.

Important Notes

For this practical you will need to install/already have numpy, scipy and matplotlib on your machine. Use either Anaconda Python or WinPython to have these setup for you quickly and hassle free. See my video series on setting up Python environments on Windows for help.

For this practical, we will only accept solutions in pure Python and all animations must be in matplotlib (not tkinter or turtle, etc).

Use of Generative AI

This task has been designed to be challenging, authentic and complex. Whilst students may use AI technologies, successful completion of assessment in this course will require students to demonstrate and engage in specific contexts and questions during their demonstration where the use of generative Artificial Intelligence (AI) tools will not be permitted during the demonstration. A failure to reference generative AI along with their code submission use may constitute student misconduct under the Student Code of Conduct. Any attempted use of Generative AI during the in-person demonstration may constitute student misconduct under the Student Code of Conduct. To pass this assessment, students will be required to demonstrate their comprehension and coded solutions independent of AI tools.

Section I – Game of Life Simulation (4 Marks)

An initial class called "conway.py" is provided with the necessary hooks required for this part of the lab. An example test script is provided that enables the animation of the simulation. Another script is also provided without animation for debugging purposes, especially for implementing the GoL rules.

[See scripts `conway.py`, `test_gameoflife_glider_simple.py` and `test_gameoflife_glider.py`]

a) Implement the four GoL rules as mentioned above in the relevant parts of the `conway.py` module and test your simulation on the 'blinker' initial pattern. You may use the 'simple' script first to ensure your algorithm is working correctly.

b) Change the initial pattern to the glider (already implemented in `conway.py`) and run the animation to verify that the rules are working correctly. How can you tell your code is working correctly?

c) Change the initial pattern to the glider gun (already implemented in `conway.py`) and run the animation. This pattern should produce gliders at a steady rate, but pattern must be exact, otherwise it will not run as expected. Fix the glider gun pattern so that its runs correctly. Hint: One of the lines for the glider gun member is incorrectly alive, e.g. see the sixth set of alive values.

d) Construct at least three different patterns from the LifeWiki for the `conway.py` module by implementing a plaintext reader for the module as a `insertFromPlainText()` member (see stub provided). This member should accept a string of the pattern in human readable form as defined by the format as a single string (as provided by the standard Python file reader after suitably handling comments etc.). Demonstrate multiple initial patterns that are greater than 20x20 in size.

Section II – Turing Completeness of the Game of Life Simulation (6 Marks)

An initial class called "rle.py" is provided with the necessary hooks required for this part of the lab. An example test script is provided that enables the running of the relevant patterns.

[See scripts `conway.py`, `rle.py` and `test_gameoflife_turing.py`]

e) Implement a fast method for computing the weights for the rules based on convolution and run a large simulation ($N > 1024$) with an appropriately large pattern (at least of the order of $N/4$ or one that is acceptable to your demonstrator).

f) Construct at least three different patterns from the LifeWiki for the `conway.py` module by implementing run length encoded (RLE) reader for the module as a `insertFromRLE()` member (see stub provided) using the `rle.py` module provided. This member should accept a string of the pattern in run length encoded form as defined by the format as a single string (as provided by the standard Python file reader). Demonstrate multiple initial patterns that are greater than 20x20 in size.

g) Demonstrate a running GoL Turing Machine pattern by using your RLE reader from the previous section to load and run the pattern.

h) Given the Turing machine pattern runs within GoL, comment on whether GoL is Turing complete. Justify your answer by referencing the theory of Turing machines and the different components of the Turing machine pattern provided using this link.

Interesting Links

Shakes' Windows Deep Learning Python Setup Series
https://www.youtube.com/playlist?list=PLC0kkV5axv-X4OpBHlIPlIz15XNNGd3OE

Video of an 8-bit Programmable computer in GoL!
https://www.youtube.com/watch?v=8unMqSp0bFY

Various GoL and Langton's Ant videos on the course Computation YouTube Playlist
https://www.youtube.com/playlist?list=PLC0kkV5axv-X3JOXeHGoedTMCXGakoYmt

References

Moore, C., Mertens, S., 2011. The Nature Of Computation. Oxford University Press.

Von Neumann, J., 1966. Theory of self-reproducing automata / Edited and completed by Arthur W. Burks. University of Illinois Press, Urbana Ill.}.

End of Laboratory