Varying the Synchronicity of Conway's Game of Life Using Forth

Isabel Jones

***Abstract*—This investigation looks at the effects of varying the synchronicity of the Conway's Game of Life, by changing the fraction of cells, *s*, that the rules are applied to. This was done by learning how to use Forth, a programming language that is efficient and thus useful when using large data sets, and then using it to build the Game of Life. We found the steady-state density of live cells for each value of s and plotted them on a graph. We used a nonlinear Levenberg-Marquardt fit to find the critical point and repeated this for each system size. We found that the critical points converge for large grid sizes to a value of $0.972 \,^{+}/\!- \,0.004$ and compared this value to Blok *et al.* [2] who found a value of $S_c^{(\infty)} = 0.906 \,^{+}/\!- \,0.004$. We discuss the limitations of our model and why these results may differ. Also, how the results imply that the Game of Life exhibits self-organizes criticality.**

I. Introduction

Conway's Game of Life was devised in the 1970's by John Horton Conway and was one of the first of many 'simulation games' [1]. The game was named after life due to its ability to display how simple systems could transform into complex and rich ones, much like the theory of evolution. Conway's Life was also the catalyst for the popularity of cellular automation which can now be used for simulating a variety of real-world systems from sociological to biological.

The process of building Conway's game requires efficient computation to produce large amounts of data. Therefore, this experiment was carried out using Forth. Forth is a programing language that was developed in 1970. It is suited for use in numerically intensive problems because of the way it works closely to the hardware of computers. Due to its efficiency, Forth is often used in astronomical applications and has also been applied to embedded systems.

In this investigation, we aimed to learn Forth, a previously unseen computer language, and apply it to Conway's Life. Then, using the vast amount of data generated, we varied the synchronicity. This was done by changing the probability of the rules being applied to a fraction of the number of cells in the game. The aim was to analyse the phase transition exhibited by Conway's Life and show that the model has characteristics of self-organized criticality.

II. Theory

A. Game of Life

Conway's Life is a 2D square grid made up of cells that are either 'alive' or 'dead'. The rules obeyed by the cells in Conway's Life can be defined as follows:

- If a cell is dead and has three neighbors, it becomes alive.
- If a cell is alive and has two or three neighbors, it stays alive.
- If a cell is alive and has either 0,1,4,5,6,7, or 8 neighbors, it dies.

These rules can be considered as representative of a species population whereby a cell or 'organism' dies if there is either overpopulation or underpopulation and reproduction is only possible with a specific population size (three neighbors in the games case).

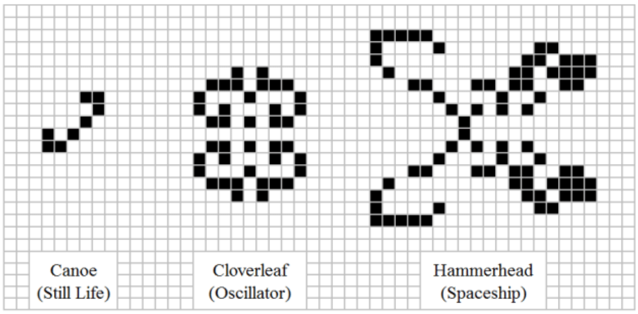

Figure 1: Three examples of different types of patterns that occur in the steady state of Conway's Life. Patterns can be split into three categories, still life, oscillator, and spaceship [2].

There are a few patterns that occur repeatedly from different starting points. Methuselahs patterns are patterns which take a long time to stabilize. Once the game has stabilized into a still life, oscillating or spaceship pattern, it is considered to be in a steady state. Examples of these different types of patterns can be seen in Figure 1.

B. Synchronicity

In computer simulations time is usually represented as discrete. In this experiment we represent time as being the number of cycles that have passed. This implies all cells are updated at the same time which is known as *synchronous* updating [3]. In nature, synchronous updating is impossible because time is continuous rather than discrete. Therefore, methods of asynchronous updating have been developed. This experiment investigates the effects of changing the synchronicity by using a probability *s* that a fraction of the cells will update in each cycle.

C. Phase Transitions and Self-organized Criticality

At a critical point, complex systems can undergo a phase transition where the rules on the microscale lead to macroscopic properties [4]. After the phase transition, the system is in the steady-state, and before the transition it is in a transient state. This scale-invariant behavior can be described by a power-law and for the system to reach the critical point, tuning is usually required. For example, in the Ising model there is an external field which influences the way an individual node is orientated. Thus, it influences when there will be an overall alignment in the system i.e. a phase transition [5].

However, self-organized criticality describes systems that evolve naturally towards the critical point without tuning. This result was first shown with the sand pile model in 1987 [6] and demonstrates that macroscopic properties can occur spontaneously. Changing the parameters will, for the most part, not stop these systems reaching the phase transition. This result has important implications ranging from biology and evolution, to the stock-markets and social science.

In Conway's Life, when the system had reached the steady-state we estimated the critical point S_c with a nonlinear Levenberg-Marquardt fit as is done in [2].

$$\rho - \rho_0 \propto (S_c - S)^{\beta} \tag{1}$$

Where β is the power-law critical exponent, ρ is the density of live states and $\rho_0 = 0.026$ for large S.

The activity in Conway's Life is the fraction of cells that flip states when updated. It can be described as,

$$\alpha \propto (S_c - S)^{\beta'} \tag{2}$$

Where α is the activity and β' is the critical exponent.

III. Programming in Forth

The first two weeks of this experiment were spent gathering tools to learn and apply Forth. Forth is a language that works closely to the hardware of computers in order to be more efficient. It works by storing numbers on a 'stack' which can be manipulated with mathematical operations. Forth also allows you to store data in variables but will only retrieve the memory location of the data unless it is specified that you want the value. Functions in Forth are called 'words' and they can also be used to build loops.

To build a 2D square array in Forth, we built a 1D array with a length the same size as the total number of cells that would be in our desired 2D array. This 1D array could then be treated as a 2D array by multiplying the y-axis value by the grid dimension to locate the coordinate position in the 1D array. We designed functions that could change and retrieve the values in the grid at the desired coordinates. This, and other functions, would later be applied to the game.

To build Conway's Game of Life, we first created a grid made up of random 1's and 0's. This was made as a 1D grid, later to be manipulated as 2D. We used the RND word provided by the department to randomly generate the numbers to put into the grid. We then created a word that would display this array so that it appeared as a 2D grid on the console.

Next, we created a variable that would store the number of live neighbors each cell had. We also made a word to initialize a grid the same size as our initial grid that would contain the cells after the rules were applied. A word was made to display this on the console as before.

Since the grid is finite, we needed to consider the how to treat the cells on the borders. If this system was found, for example in evolution, the grid would not be finite so would not have these boundary effects. To handle them, we decided to kill all the cells at the border i.e. set them to zero. This was done by looping over each of the edges and this process had to be completed before we started summing the live neighbors of each cell. To create the word for putting the live neighbors sum into the variable we created earlier, we needed another loop and if statements to find the number of alive cells in the eight surrounding cells for each cell in the initial grid.

This word called 'sum_cells' was then used to apply the rules. We used the 'case' function in Forth within a loop over the initial grid to assign values of the new cells to the new grid we made to store them. The values assigned to the new cells depended on the number of neighbors they had via the Life rules.

To test whether these rules worked before exporting the data, we could display the arrays on the console and manually check that the rules had been applied correctly as is shown in Figure 2.

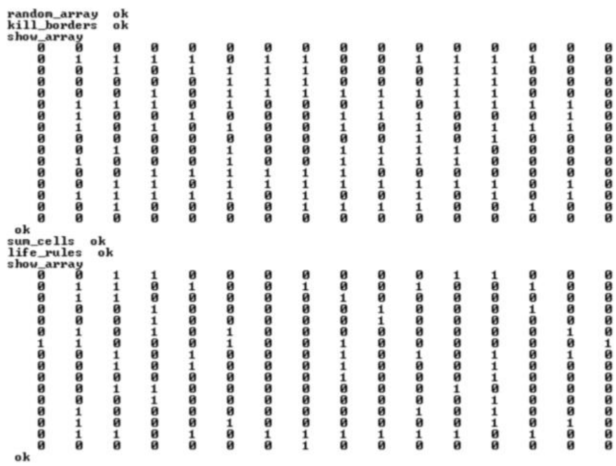

Figure 2: A display of the Forth console. The top array is randomly generated with 1's and 0's, and the borders have been set to zero. The bottom array is after the life rules have been applied to show that the algorithm has been effective.

IV. Results and Discussion

To produce the results for varying synchronicity, we updated the method such that only a fraction, *s*, of the cells would have rules applied to them. We produced graphics as shown in Figure 3 of the Game of Life at the end of 500 cycles for s=0.1 and s=0.89.

When s=0.89, the system formed a still-life pattern once it was in the steady state and had a low-density of live cells. For s=0.1, the grid reached a randomly fluctuating pattern after the transient state.

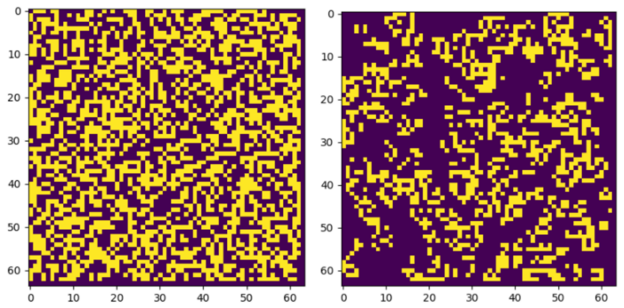

Figure 3: Graphical displays of the Game of Life grid with dimensions 64x64. The figures were taken after the system reached the steady state. On the left s=0.89, and on the right s=0.1.

We plotted a graph of the steady-state live cells density against varying value of s, as can be seen in Figure 4. This was done for a grid-size of 32x32 and we fitted a curve fit using Equation (1). This yielded a critical point, S_c of 0.964 with an error of 0.009 and a critical exponent, β value of 0.190 with error 0.018.

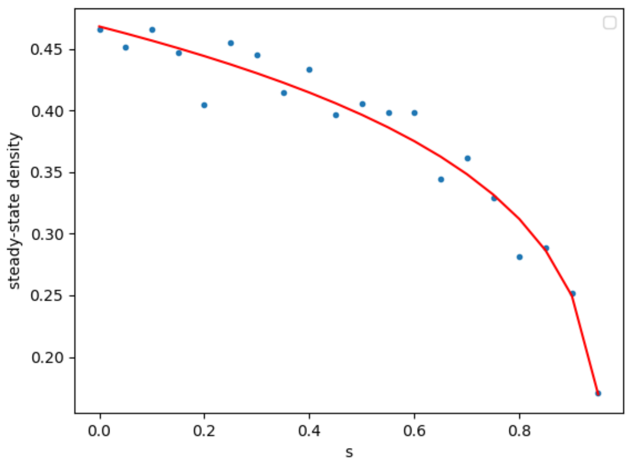

Figure 4 shows a plot of the density of live cells in the steady state plotted against the probability s. The red line is the curve-fit from Equation (1). This yields a critical point of 0.964 for a grid of 32x32.

This process was then repeated for each system size and the value of the critical point, found by plotting the curve fit, was plotted against the system size in Figure 5. This Figure shows that as the system size increased the critical point converged to a dotted line at $S_c^{(\infty)} = 0.972 \ {}^{+}/_{-}\ 0.004$, implying the system has scale-free behavior. This line was calculated by only using the last three points as the error margin was smaller for these points.

The value that the critical point converged to for larger system sizes in paper [2] was $S_c^{(\infty)} = 0.906 \ {}^{+}/_{-}\ 0.004$. Our value does not fall within the error margins of this number but is reasonably close. One reason for the difference could be because in the paper the line of best fit as seen in Figure 4 appears to only have been applied to values of s above 0.5, whereas ours was applied to all values of s. This may have had an important effect because as mentioned above, lower values of *s* tend to produce a randomly fluctuating pattern in the steady state, so perhaps the live density varies too widely for smaller values of *s* to be reliable.

Moreover, the paper mentions that "to keep the average rate of events uniform time is rescaled by s". We were unaware quite how they did this, so this could've been something that was neglected in our investigation that impacted the results. It could also be possible that dealing with the boundaries of Life by setting them to zero is less effective than using wrapped edges. This may have contributed to our difference in results.

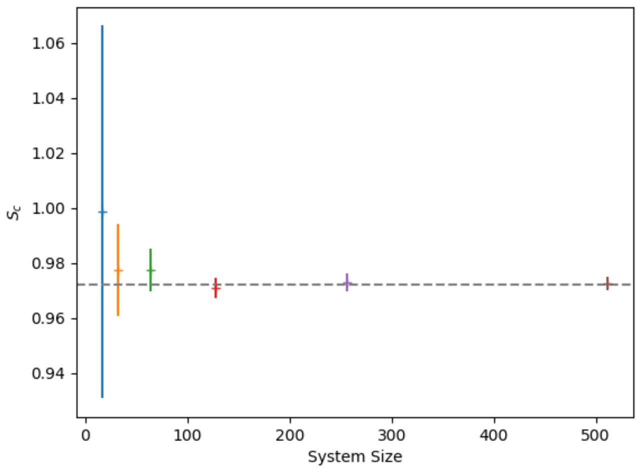

Figure 5 shows the critical point plotted against 6 different system sizes. The line of best fit is a horizontal line through 0.972.

Overall, our results show that as *s* is decreased from 1, there is a phase transition in the Game of Life, independent of the size of the system, at a critical point. This implies that Conway's Game of Life obeys self-organized criticality. Moreover, the vast difference of outcomes for varying values of synchronicity shows the importance and need for the development of asynchronous updating methods.

V. Conclusion

Conway's Game of Life is a complex system often hailed as a way to understand evolution. In this experiment, we aimed to analyse some of the games' properties, namely when the synchronicity of the system is varied. To do this we needed to utilize a new programming language, Forth.

The first portion of the experiment was spent building tools in forth that would allow us to build the final Life model. The model was built and tested and then could be analysed. We applied the rules to only a fraction of the cells, *s*. The steady-state density of the live cells was found for each value of *s* and a critical point was calculated for each grid size using a nonlinear Levenberg-Marquardt fit. This demonstrated that the critical point converged for larger system sizes implying that the game of life exhibits self-organized criticality i.e. will reach a phase transition and macroscopic results independent of the parameters.

Overall, the experiment was successful because we demonstrated the properties of the Game of Life and the importance that synchronicity plays in systems. However, our final result for the critical point was not quite close enough to what was found in paper [2], therefore, we discussed the ways in which our experiment could be improved.

References

[1] M. Gardner, “MATHEMATICAL GAMES The fantastic combinations of John Conway’s new solitaire game ‘life,’” 1970. Accessed: Apr. 13, 2021. [Online]. Available: http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/lis_projekt/proj_gamelife/ConwayScientificAmerican.htm.

[2] D. Marolt, J. Scheible, G. Jerke, and V. Marolt, “SWARM: A Self-Organization Approach for Layout Automation in Analog IC Design,” *Int. J. Electron. Electr. Eng.*, pp. 374–385, 2016, doi: 10.18178/ijeee.4.5.374-385.

[3] H. J. Blok and B. Bergersen, “Synchronous versus asynchronous updating in the ‘‘game of Life.’’”

[4] R. V. Solé, S. C. Manrubia, B. Luque, J. Delgado, and J. Bascompte, “Phase transitions and complex systems: *Simple, nonlinear models capture complex systems at the edge of chaos*,” *Complexity*, vol. 1, no. 4, pp. 13–26, Mar. 1996, doi: 10.1002/cplx.6130010405.

[5] L. Li, Y. Fan, A. Zeng, and Z. Di, “Binary opinion dynamics on signed networks based on Ising model,” *Phys. A Stat. Mech. its Appl.*, vol. 525, pp. 433–442, Jul. 2019, doi: 10.1016/j.physa.2019.03.011.

[6] P. Bak, C. Tang, and K. Wiesenfeld, “Self-organized criticality: An explanation of the 1/f noise,” *Phys. Rev. Lett.*, vol. 59, no. 4, pp. 381–384, Jul. 1987, doi: 10.1103/PhysRevLett.59.381.