Automating Large-Scale Detection and Classification of Larger than Life Cellular Automata Patterns

Authors:
Brandon Ismalej[1]
Dr. Kellie Evans[2]

Department of Computer Science[1]
Department of Mathematics[2]
California State University, Northridge

Larger than Life Cellular Automata

- **Cellular automata (CA):**
  - A class of discrete, grid-based computational models - based on simple rules and algorithms
  - Cell states: typically “live” or “dead”; 0 or 1

- **Conway’s Game of Life (*Life*):**
  - A CA with simples rule for cell birth, survival, death [1]
  - Can lead to dynamic patterns: spaceships

- **Larger than Life (LtL):**
  - Generalization of *Life,* extended to larger neighborhoods [2]
  - Uses intervals for birth and survival thresholds
  - Complexity - Exploration of more intricate patterns: bugs

How to explore these CA?

Golly

- Open-source software for exploration and simulation of CA, including LtL [3]
- Written in C++
- Supports scripting in Python and Lua
- Supports bounded and unbounded universes

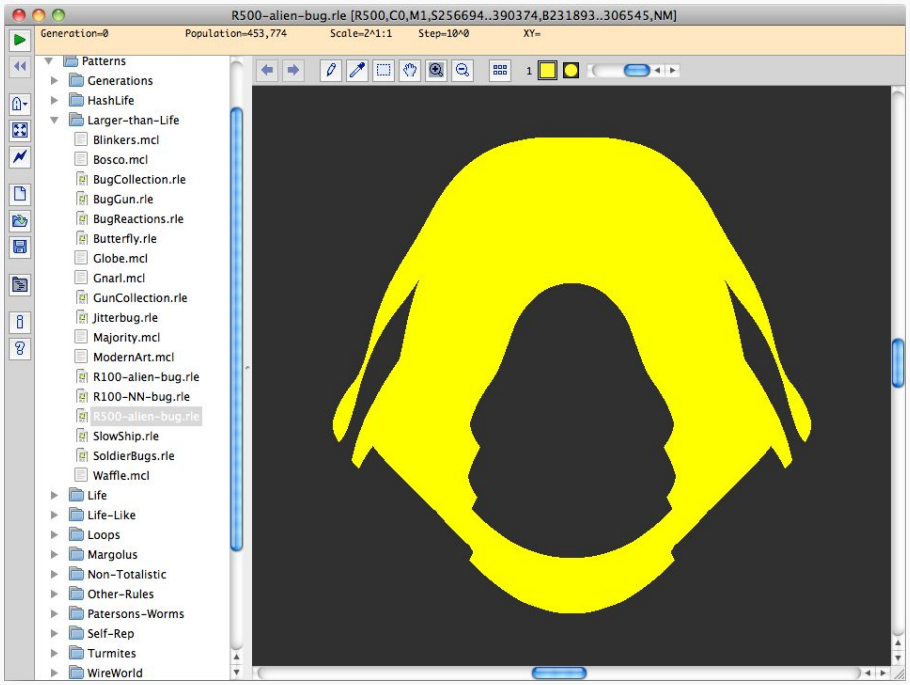

Figure 1: Snapshot of Golly GUI [3]

Motivation

- In Life, the most intriguing patterns are known as “spaceships”
  - Can carry information across time as space updates
  - The most famous spaceship is known as a glider

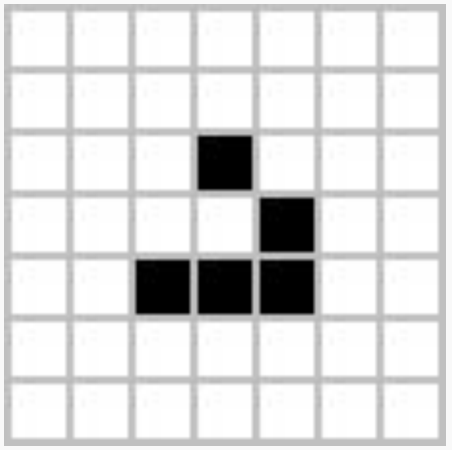

Figure 2: Life’s glider [4]

- In LtL, generalizations of *Life’s* spaceships are known as “bugs”
  - Exhibit complex dynamics and behaviors

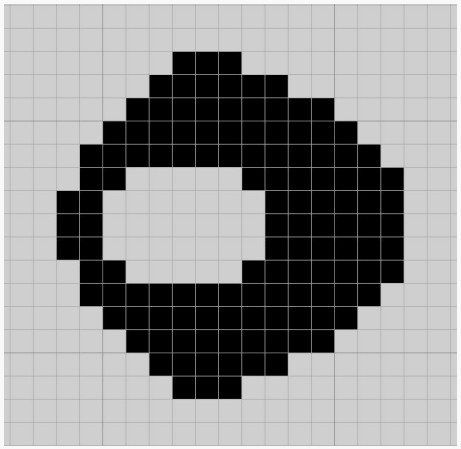

Figure 3: LtL Range 25 Bug

Big Question

If cellular automata follow specific rules and algorithms, how can we systematically discover and identify complex patterns like “bugs” within these systems?

Current Methodology

- **Random Soup Search**
  - “apgsearch” - automated search program written by Adam P. Goucher [5]
  - Generates large amounts of random asymmetrical soups

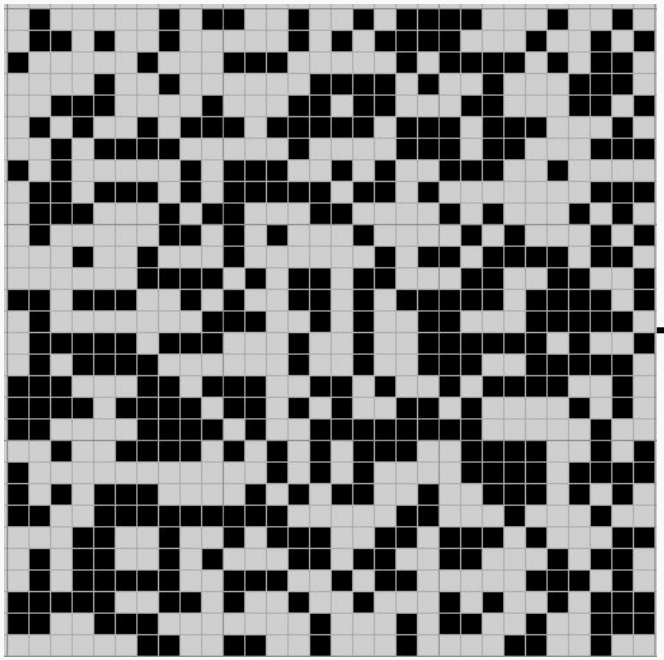

Figure 4: 30 by 30 random soup, density=0.5

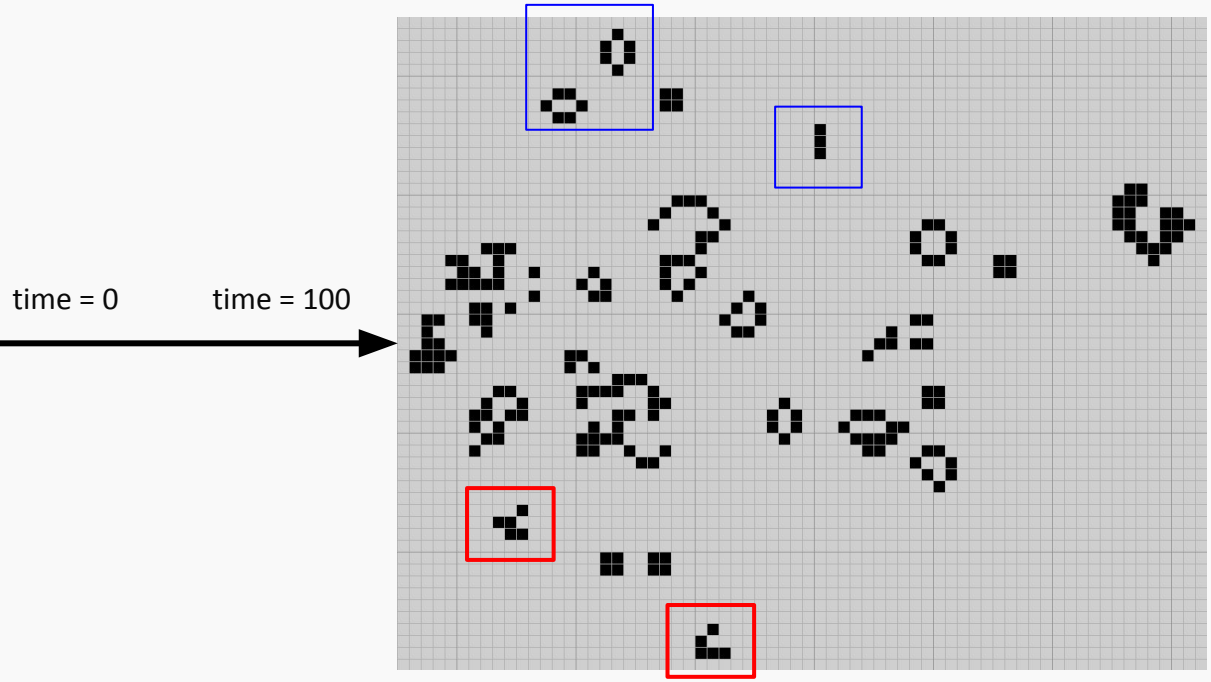

Figure 5: Gliders, still lifes, and blinkers emerging from Fig. 4 random soup

Current Methodology

- **Finite Deterministic Configurations**
  - Evans describes the use of geometric initial configurations, such as rectangles and circles, that a rule will “sculpt” into a bug [2] [6]
  - Commonly used and leverages configurations that resemble the geometry of bugs

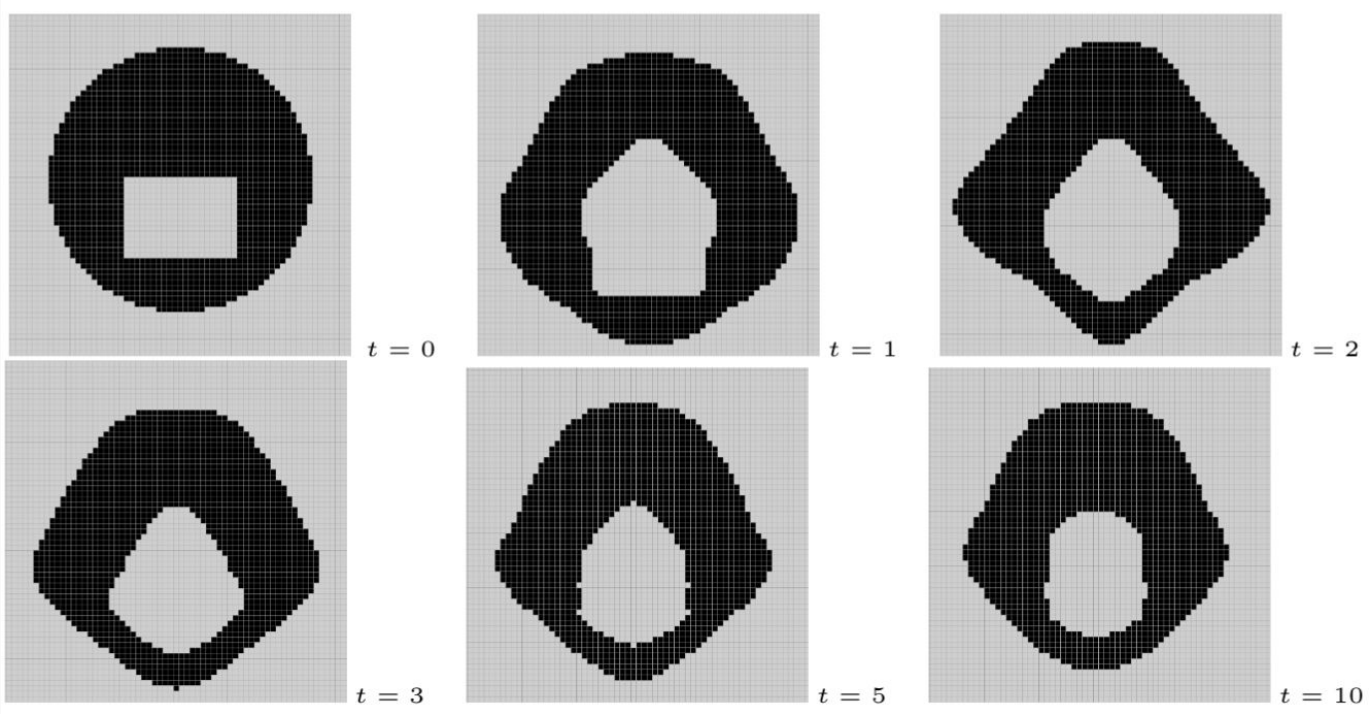

Figure 6: Range 25 geometric initial configuration from time 0 to 10
Circle: radius=24, y-setback=7, Rectangle: length=21, width=15

Proposed Scripting Design

A set of Lua scripts has been developed to automate the creation and simulation of geometric initial configurations onto the Golly grid

The most notable script aims to:

- **Create and place configurations**, based on user-defined parameters, such as:
  - Shape of live and dead sites: circle, rectangle, ellipse
  - Dimensions of shapes: radii, length/width, major/minor axes
  - Vertical “setback” of dead sites: vertical distance from center of live site to center of dead site

- **Run a simulation of every configuration** created to:
  - Detect surviving patterns, particularly bugs
  - Sort configurations into CSV files, based on their behavior after the simulation has been run, such as patterns that die off

Classifying Patterns

Wolfram's Framework [7]:

- **Class 1 (Homogenous States)**: Dead patterns with no live cells after simulation - logged in "not_survive.csv"
- **Class 2 (Periodic Structures):** Surviving patterns with no vertical displacement are classified as "still lifes" and logged in "still.csv" - those with displacement are logged in "survive.csv"
- **Class 3 (Chaotic Aperiodic Behavior):** Patterns that exceed the iteration limit - logged in "timeout.csv"
- **Class 4 (Complex Localized Structures):** Any intricate structures detected during exploration - logged in "survive.csv"

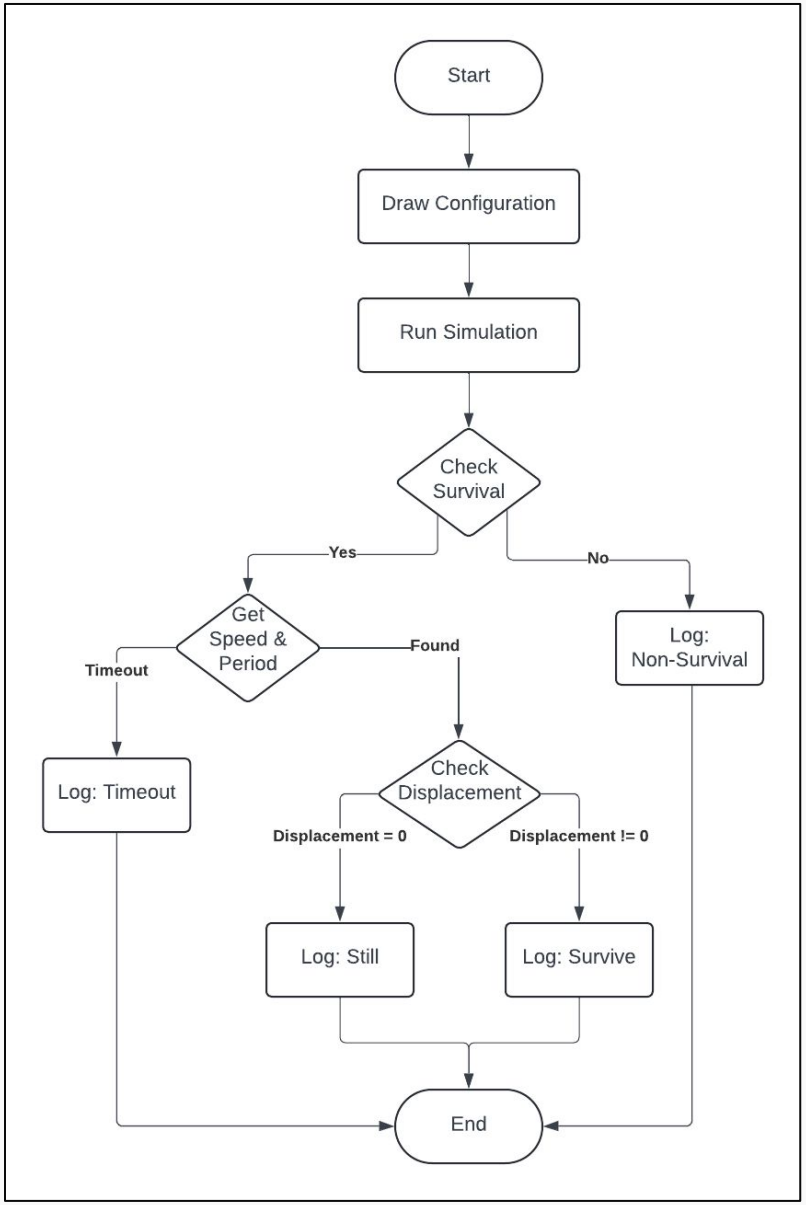

Figure 7: Flowchart of pattern classification

Experimental Results

The following user-defined parameters were utilized, as requested by our Golly-integrated script

- **Number of timesteps before classification:** 60
- **Rule for current grid:** R25,C0,M1,S720..1258,B720..978,NM
- **Max. iterations for timeout:** 1,000
- **Shape of live sites:** C (circle)
- **Bounds of radii:** 20, 25
- **Bounds of y-setback:** 5, 15
- **Shape of dead sites:** R (rectangle)
- **Bounds of length/width:** 17,22

Experimental Results

Experiment yielded 2,376 configurations created, simulated, and classified.

	A	B	C	D	E	F	G	H	I	J	K
1	Shape of Live Cells	Dimensions of Live Shape	Y Setback	Shape of Dead Cells	Dimensions of Dead Shape	Period	dy	Population	Bound Box Wd	Bound Box Ht	Hash Value
2	C	21	6	R	17 17	1	-3	1361	49	49	-565013023
3	C	21	7	R	17 17	1	-3	1361	49	49	-565013023
4	C	21	7	R	19 17	1	-3	1361	49	49	-565013023
5	C	21	8	R	17 17	1	-3	1361	49	49	-565013023
6	C	21	8	R	17 18	1	-3	1361	49	49	-565013023
7	C	21	9	R	17 17	1	-3	1361	49	49	-565013023
8	C	21	9	R	18 17	1	-3	1361	49	49	-565013023
9	C	21	9	R	19 17	1	-3	1361	49	49	-565013023
10	C	21	10	R	17 18	1	-3	1361	49	49	-565013023
11	C	21	10	R	17 19	1	-3	1361	49	49	-565013023
12	C	21	10	R	19 17	1	-3	1361	49	49	-565013023
13	C	21	10	R	21 17	1	-3	1361	49	49	-565013023
14	C	21	11	R	17 17	1	-3	1361	49	49	-565013023
15	C	21	11	R	17 20	1	-3	1361	49	49	-565013023

Figure 8: Snapshot of output, rows 1-15

Visualization of Results

A separate Lua script allows us to visualize our generated configurations for interactive analysis.

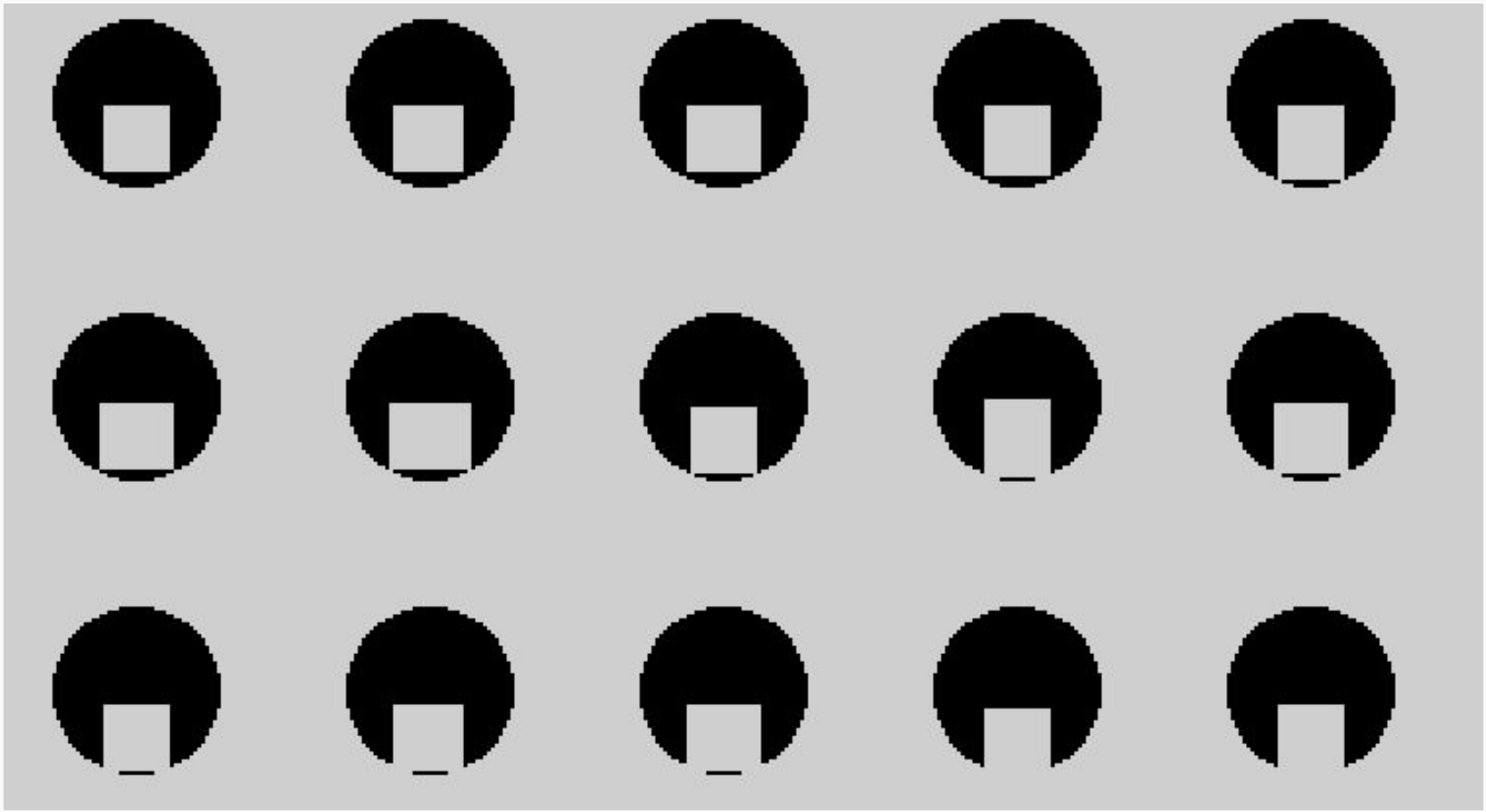

Figure 9: Configurations 6-20 from Fig. 8 output

Analysis

Data-Driven Exploration:

- Automating the creation, simulation, and sorting of geometric configurations produced large datasets
- Datasets allow for comprehensive analysis, supporting the formatting and validation of conjectures regarding LtL patterns

Key Insights:

- Understanding the sensitivity of bugs and other life-like patterns to specific initial configurations
- Identifying common traits in configurations that lead to stable or emergent behaviors

Conclusion & Future Work

- Introduced a suite of Lua scripts for automated exploration and classification of LtL CA patterns
- Developed a more systematic approach to classify patterns based on behavior and dynamics
- Enables targeted searches compared to random soups
- More efficient than manual creation of configurations

Figure 6: Range 100 disoriented alien bug, period of 58,775

Future Work

- Enhance detection & classification algorithm for asymmetric patterns or those with extremely large periods
- Parallelization strategies to improve runtime

References

- [1] M. Gardner, “Mathematical games - The fantastic combinations of John Conway’s new solitaire game ‘Life’,” Scientific American, Oct. 1970. [Online]. Available: https://www.scientificamerican.com/article/mathematical-games-1970-10/
- [2] K. M. Evans, “Larger than Life: it’s so nonlinear,” PhD Dissertation, University of Wisconsin-Madison, Madison, WI, 1996.
- [3] T. Rokicki and A. Trevorrow, “Golly: An open source, cross-platform application for exploring Conway’s Game of Life and other cellular automata,” 2024, version 4.3, accessed on August 11, 2024. [Online]. Available: https://golly.sourceforge.io
- [4] “Glider - LifeWiki.” Accessed: Sep. 04, 2024. [Online]. Available: https://conwaylife.com/wiki/Glider
- [5] A. P. Goucher, Mar. 2024. [Online]. Available: https://gitlab.com/apgoucher/apgmera
- [6] K. M. Evans, “Larger than Life: threshold-range scaling of Life’s coherent structures,” Physica D: Nonlinear Phenomena, vol. 183, no. 1, p. 45–67, Sep. 2003.
- [7] S. Wolfram, Cellular Automata and Complexity: Collected Papers. Boca Raton, FL: CRC Press, 1994. [Online]. Available: https://www.wolframscience.com/reference/

Questions?

brandon.ismalej.671@my.csun.edu

CSUN® | COMPUTER SCIENCE