Implementing Cellular Automata

Nicholas Marshall, Mary McCann, and Jacob Mazur

April 26, 2022

1 Overview and Background

A cellular automata is a regular array of cells. The input to each cell is the state of several of its neighboring cells and the output is specified by a transition function.

The goal of this project is to build a software emulator for cellular automata (CA). The program runs in a terminal and provides a visual representation of the automaton being emulated.

Going into this project we knew we would need to determine how to read in a text representation of a CA, store the automaton state in memory, and simulate the behavior from the provided state. We created solutions to each of these challenges. These solutions are how we implemented our program, and they are detailed below:

1.1 Challenge 1

First, we needed to determine how to read in a text representation of a CA. We decided that the best way to do this would be to use a JSON file. In the file there is a dictionary that includes the dimension of the CA, the length, the states, the outputs, and the transition rules. From there, our program takes in the JSON file and implements the rules listed within the JSON file.

1.2 Challenge 2

Second we needed to decide how to store the automaton state in memory. We decided to make the board itself a class whose parameters are its dimension, length (the length in every dimension is equal), and the transition rules which it gets from the input JSON file.

1.3 Challenge 3

Finally we needed to determine how to simulate the behavior from the provided JSON file. We decided to use an nparray to represent the board and used transition rules which hold information that the board needs. It includes the neighbors output, the current state, the new state, and new output. Using the this information from the JSON file, the board can implement each cellular automata. The progression of the cellular automata can both be seen as a progres

2 Implementation

2.1 Instructions for Execution

In order to execute the code, use the test files we have provided to implement different dimensions of boards. For example, run `./test_1d.py` to see a 1D CA. If you would like to experiment with it, you can alternatively use the python environment to try out various inputs. If you wish to download the code, here is our github.

2.2 Photos

Here are some photos of our successful implementation of the program (there are four total photos on different pages).

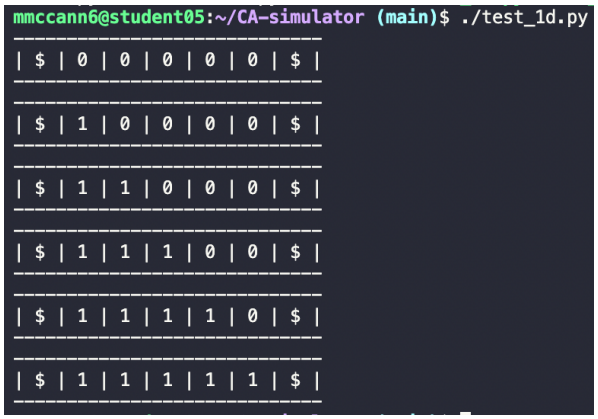

Figure 1: Six iterations of a simple 1D automaton that fills in 1s from left to right.

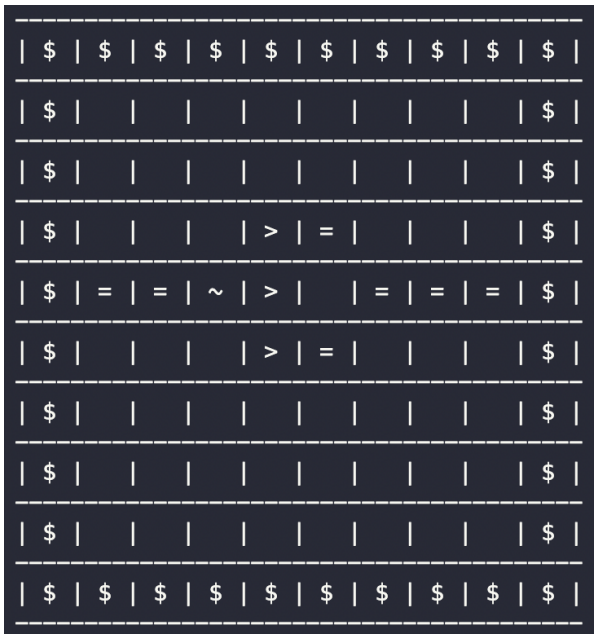

Figure 2: Wireworld.

```
>>> print(board)
-----------------------------------------
| $ | $ | $ | $ | $ | $ | $ | $ | $ | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   | X | X | X |   |   | X |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   | X |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   | X |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ | $ | $ | $ | $ | $ | $ | $ | $ | $ |
-----------------------------------------
>>> board.iterate()
>>> print(board)
-----------------------------------------
| $ | $ | $ | $ | $ | $ | $ | $ | $ | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   | X |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   | X |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   | X |   |   | X | X | X | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ | $ | $ | $ | $ | $ | $ | $ | $ | $ |
-----------------------------------------
```

Figure 3: Game of Life: blinker

```
>>> print(board)
-----------------------------------------
| $ | $ | $ | $ | $ | $ | $ | $ | $ | $ |
-----------------------------------------
| $ |   | X |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   | X |   |   |   |   |   | $ |
-----------------------------------------
| $ | X | X | X |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ | $ | $ | $ | $ | $ | $ | $ | $ | $ |
-----------------------------------------
>>> board.iterate()
>>> print(board)
-----------------------------------------
| $ | $ | $ | $ | $ | $ | $ | $ | $ | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ | X |   | X |   |   |   |   |   | $ |
-----------------------------------------
| $ |   | X | X |   |   |   |   |   | $ |
-----------------------------------------
| $ |   | X |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ |   |   |   |   |   |   |   |   | $ |
-----------------------------------------
| $ | $ | $ | $ | $ | $ | $ | $ | $ | $ |
-----------------------------------------
```

Figure 4: Game of Life: glider