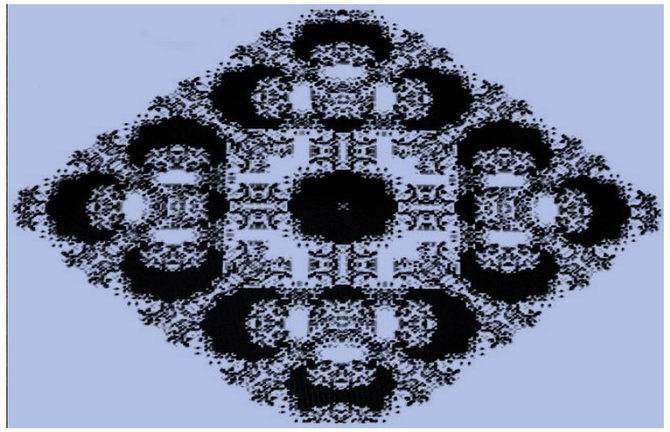
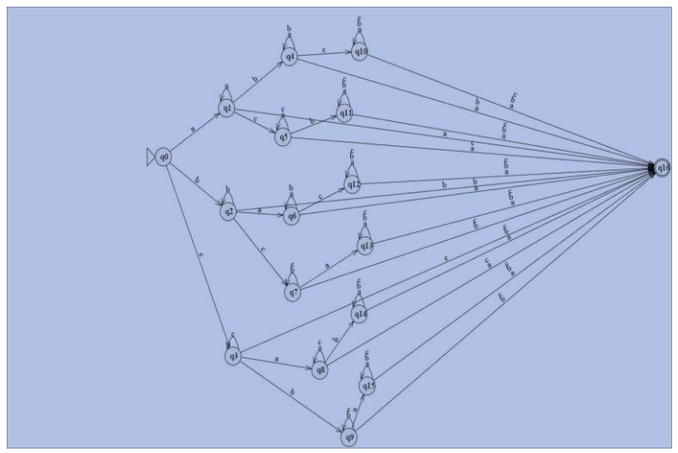
# CELLULAR AUTOMATON

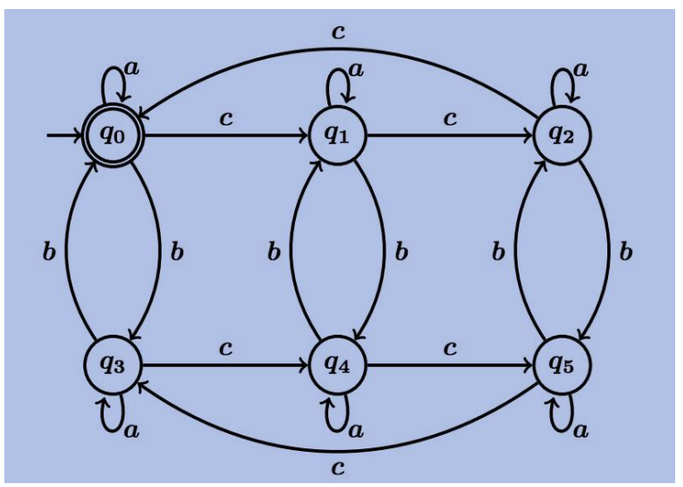
COMPLEXITY SCIENCE – MODULE 3

NINO PHILIP RAMONES | GITHUB
**2020 - 05616**

JULY 3, 2023







# INTRODUCTION

The concept of **cellular automaton** (**CA**) is built on the idea of a system of many objects that have **varying states** over time, contrary to such agent-based models where an element generally exists in only one state and remained as the same type of object over time. In principle, CA is a **model of a system of cell objects** defined by a **grid** where the cells are contained. Each cell has their own **state**, which can be typically finite but in practice, 0 or 1 is used to represent "alive" or "dead" or "on" and "off" states. Each cell has also a defined **neighborhood**, which is a list of adjacent cells. In real world, CA comes into play into the study of different **crystal fractals** where in CA, a set of simple **set of rules** are defined that would allow the **replication** of such pattern like reproduction.

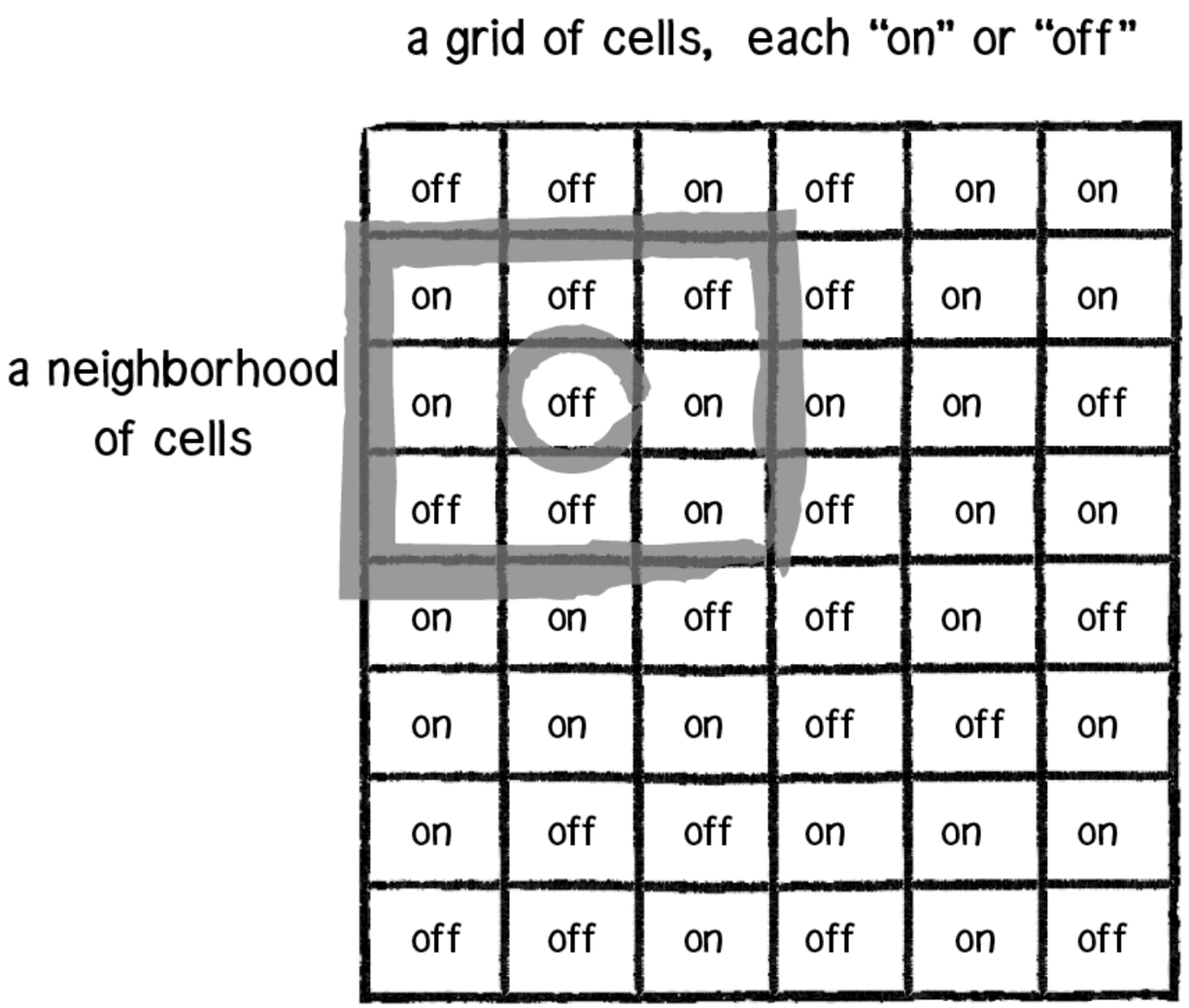


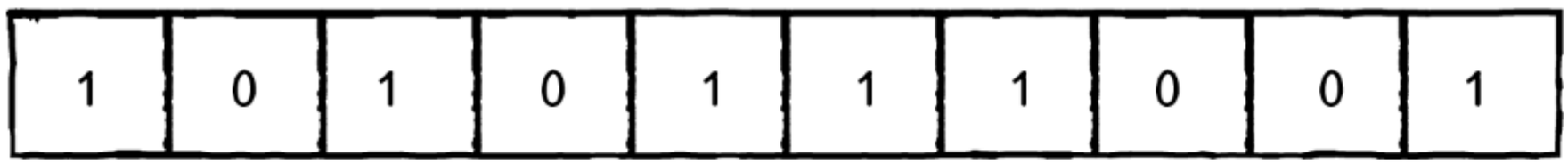

Figure 1. A neighborhood of cells in the cellular automaton theory. Figure referenced from The Nature of Code.

# KEY ELEMENTS

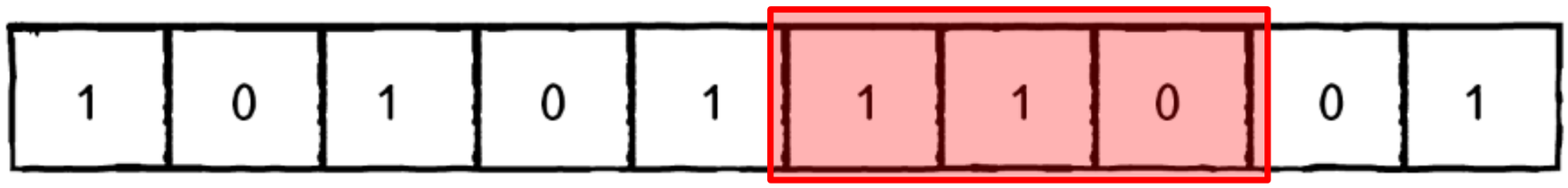
As mentioned, the three key elements of a CA as proposed from Wolfram are

**Grid.** This can be represented by a simple one-dimensional line or array of cells

**States.** Let us say, 0 or 1 are the only allowed states for our system, the simplest perhaps!




**Neigborhood.** The simplest neighborhood for any given cell would be the cell itself and its two adjacent neighbors, i.e.



# CELL GENERATION AND STATES

In CA, **cell generation** or frame count delves into the concept of time about the CA living over a period of time. These can be interpreted as the states of individual cells per generation such that the array of states above it will define the **succeeding states** that follows it below!

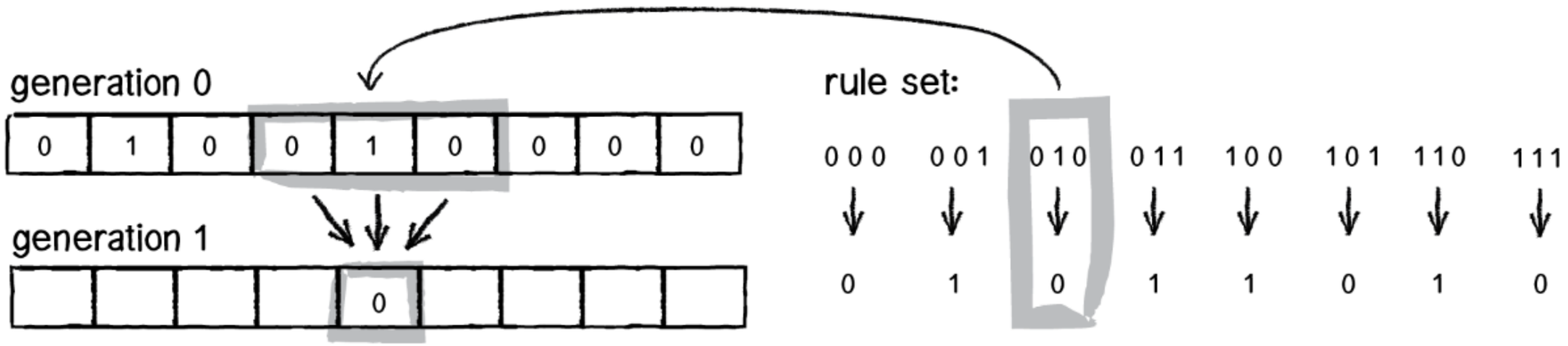



Figure 2. Rule set configuration that dictates the succeeding states in a cellular automaton.

Simply speaking, the first row in generation 0 shows the eight states a **neighborhood** can be in. For every three cells we consider, we can **configure the states**, for example, in a way that can be defined using a 3-bit number, resulting to a total number of $2^3$ or 8 states! In practice, the **standard model** initializes all cells at generation 0 to zero except 1 for the middle cell. Generation 1 states will follow the **rule set** shown above, take the neighborhoods, and the next generation rows will follow, resulting to some unique patterns when stacked all together!

# RULE 30

From the previous slide, a **set of rules** was set that will define the **new states of a generat**ion. Since a cell and its two neighbors from a neighborhood yields 8 possible patterns, then $2^{2^3}$ or **256 rules** exist that will decide whether a cell will be 0 or 1 in the next generation. In elementary CA, **rule 30** is given by (from Cellular automaton)

**Rule 30 cellular automaton**
**(binary 00011110 = decimal 30)**

| current pattern | 111 | 110 | 101 | 100 | 011 | 010 | 001 | 000 |
|---|---|---|---|---|---|---|---|---|
| new state for center cell | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

which is simply given by the binary 00011110. Notice that the rightmost 3-bit is 000, for which it follows the **gray code** for binary numbers, i.e the two successive values differ in only one bit!

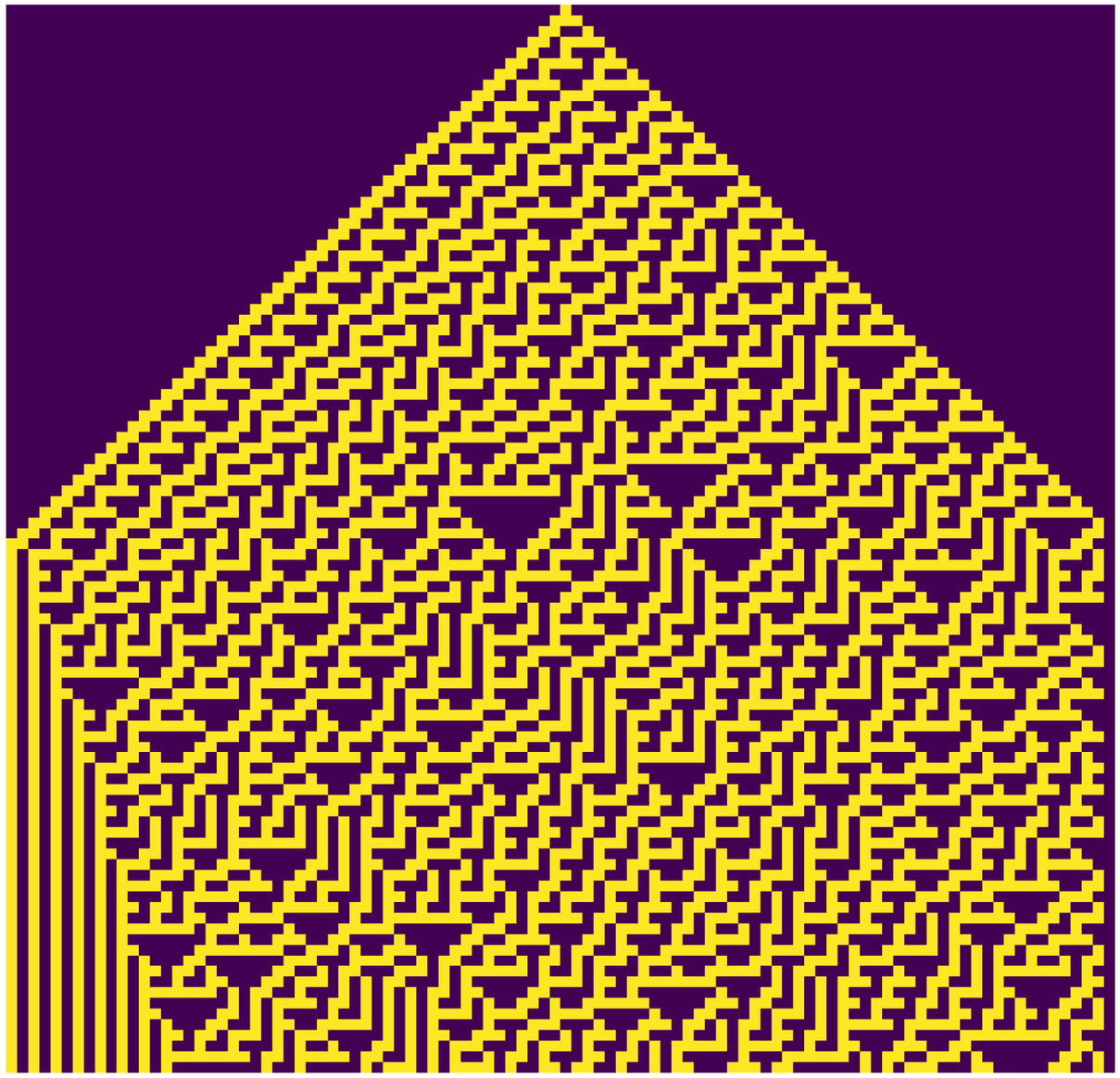


Figure 3. Rule 30 after 100 generations or time steps.

# RULE 150

We also try another possible rule out of all the possible 256 rules to exist in the **Wolfram code** for cellular automaton, rule 150 shows **symmetry** along the vertical axis. Along with rule 30, rule 150 exhibited a development of patterns that tend towards **chaotic, seemingly infinite growth**. For rule 150, the binary basis is 10010110 for the new state of the center cell, similar to the **convention** followed for rule 30.

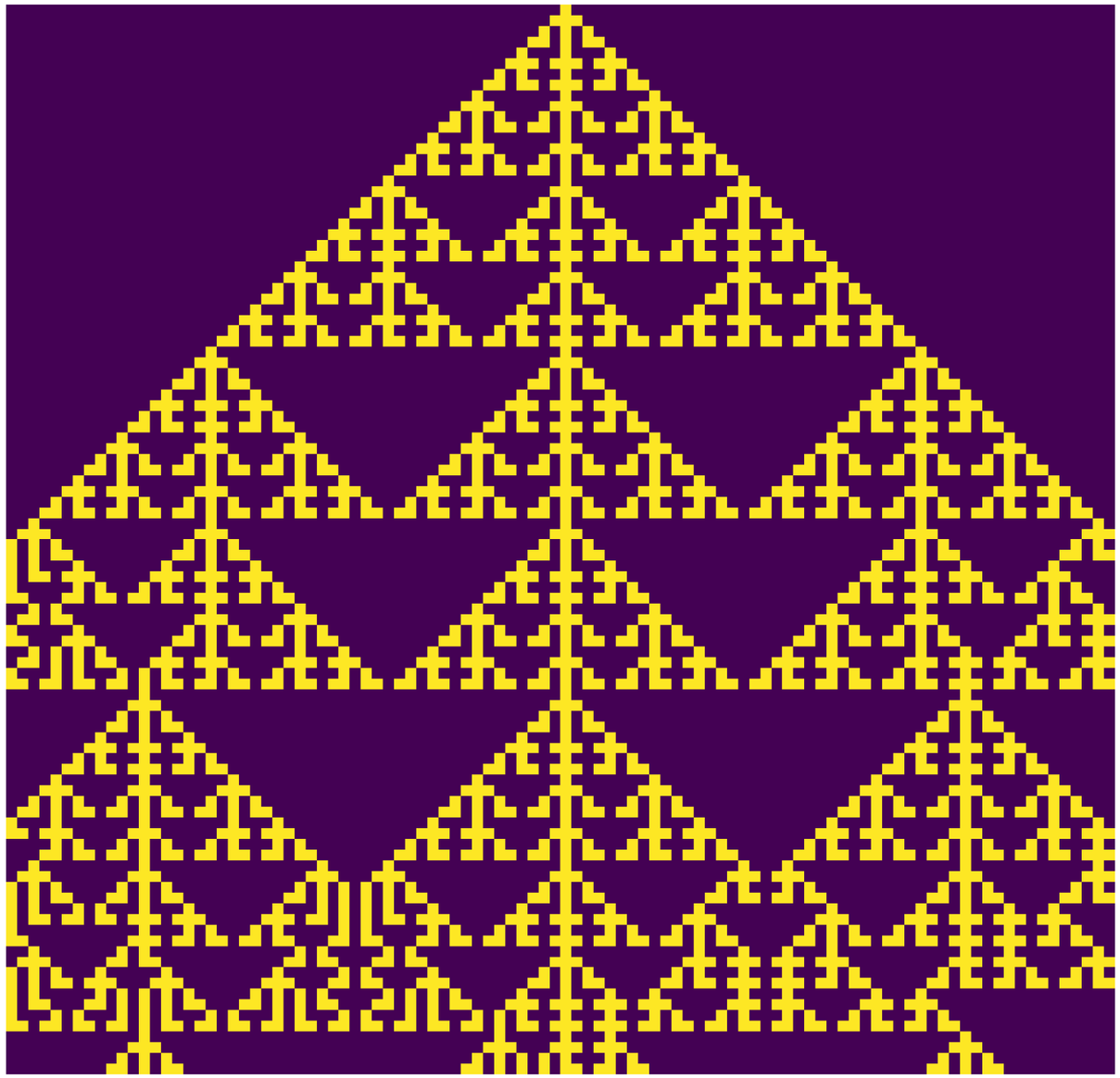


Figure 4. Rule 150 after 100 generations or time steps.

# REAL WORLD APPLICATIONS

The application of CA in real world experiments are endless as it transcends on the field of natural sciences. Some of the applications that are worth noting for are:

Figure 5. Cone snail pattern from Cellular automaton – Wikipedia.

**Biology.** Recurring patterns in nature are visible regularities of form found in living organisms. CA is exhibited on these patterns such as that of a shell pattern on a cone snail shell. Moving wave patterns on the skin of cephalopods can also be simulated with a two-state, two-dimensional cellular automata.

**Physics.** CA also applies to the simulation of fluid dynamics, that touches the concept of the Ising model.

# REFLECTION

I was able to enjoy the activity as it gave us freedom to explore some of the interesting applications and theory behind the cellular automata theory. The activity can be further improved by delving into the different class behaviors exhibited by different CA rules. Overall, I would give myself a score of **90/100** !

# REFERENCES | GITHUB

1. A.B. Downey, Think Complexity, https://github.com/AllenDowney/ThinkComplexity2
2. Elementary Cellular Automaton -- from Wolfram MathWorld
3. The Nature of Code
4. Wolfram's classification - LifeWiki (conwaylife.com)