CELLULAR AUTOMATA AND THE GAME OF LIFE

KAITLYN ZHANG

Abstract. This paper will introduce Conway's game of Life, a celebrated cellular automaton. We define cellular automata and delve into the rules and properties of the game of Life, and show that the game of Life is Turing complete. Therefore, a computer can be constructed out of the game of Life, albeit a slow and primitive one. We create a universal Turing machine from the game of Life, depicting that the game of Life can compute any computational problem. We conclude by connecting the game of Life with other practical and interdisciplinary applications.

Contents

1. Introduction to the Game of Life 1
1.1. Cellular automata 1
1.2. von Neumann's Universal Constructor 2
1.3. Rules to the Game of Life 3
1.4. Patterns of the Game of Life 4
2. Turing Machines 10
2.1. Defining Turing Machines 10
2.2. Universal Turing Machines 11
3. Game of Life is Turing-complete 11
4. Game of Life is Irreversible 13
5. Applications of the Game of Life 13
5.1. Game of Life and Circuits 13
5.2. Biological applications of the Game of Life 14
5.3. Brief digression: Philosophy of the Game of Life 14
6. Conclusion 14
References 14

1. Introduction to the Game of Life

1.1. Cellular automata.

Date: December 11, 2022.

Definition. Cellular automata are models of computation involving cells that may be empty or filled at any given point in time, and that change whether they are empty of filled based on their neighbors. They are used as a framework for investigating the behavior of complex, extended systems. Cellular automata begin with an initial configuration of the parts of the system and a set of rules describing how each part changes over time, based on parts nearby. The behavior of the automata is then observed over time. The concept of cellular automata was conceived by John von Neumann in the 1940s, toward designing a theoretical self-replicating machine.

In general, cellular automata are characterized by a k-dimensional set of cells, where k is a positive integer, a positive number of states that can describe each cell, and a set of rules describing how the state of each cell changes over time, based on the states of nearby cells.

Applications. Cellular automata can be used to model nature, such as fluid flow, ecosystems, traffic, city growth, insect colonies, and crystals. They can also build machines, i.e. a configuration of states.

1.2. **von Neumann's Universal Constructor.** An example of a cellular automaton is von Neumann's Universal Constructor, a predecessor to the game of Life. von Neumann's constructor was the first to connect cellular automata with computation, and his cellular automaton aimed to implement the self-replication found in natural systems. von Neumann distilled two requirements for self-replicating machines:

(1) The machine should be a universal constructor: given a description for any machine, it will search until it locates the proper parts, and then construct that machine. In particular, given a description of itself, it will construct a copy of itself.
(2) The machine should contain a description copying machine capable of making a copy of the description of the universal constructor.

Interestingly, note a system of nature that qualifies as a self-replicating machine by the above requirements: DNA. DNA is a genetic program that encodes the instructions for making all the enzymes and structural proteins that the cell needs to function, and is a repository of genetic data which is duplicated and given to the new cell, at the same time.

Although von Neumann was able to construct a self-replicating pattern in his universal constructor, the constructor has 29 possible states per cell and

complicated state transitions. Conway's game of LIFE later provided an easier solution.

1.3. **Rules to the Game of Life.** The game of LIFE is a cellular automaton created by John H. Conway that resides in an (optionally) infinite 2D lattice of sites. At each point in time, the fate of each site is dependent on its eight nearest neighbors[1] and parallel updating according to a set of rules.

There are two states in the game of LIFE: a cell can be alive, in which it is filled, or dead, in which it is empty.

The rules. At each time step, the following rules are applied to every cell based on the states of its current eight neighbors.

(1) A cell in the *alive* state will stay alive in the next generation if it has two or three alive neighbors, otherwise it will be *dead*.
(2) A cell in the *dead* or empty state will be in the *alive* state in the next generation if it has exactly three alive neighbors, otherwise it will stay in the *dead* state.

Keep in mind that each cell has exactly eight neighbors.

These two rules describe the evolution of the grid in the game of LIFE, and can be distilled into four named categories:

- *Birth*: A cell that is *dead* at time t will be alive at time $t+1$ if exactly 3 of its eight neighbors were alive at time t.
- *Death*: A cell can die by:
  - *Overcrowding*: If a cell is alive at time $t+1$ and 4 or more of its neighbors are also alive at time t, the cell will be dead at time $t+1$.
  - *Exposure*: If a live cell at time t has only 1 live neighbor or no live neighbors, it will be dead at time $t+1$.
- *Survival*: A cell survives from time t to time $t+1$ if and only if 2 or 3 of its neighbors are alive at time t.

From these rules and the initial configuration, the game board continuously evolves, playing the game by itself. The game of LIFE is a zero-player game. Many "life-forms" can be created with this game, and we will now transition into evaluating the types of patterns the game of LIFE can create. Later,

[1]Up, down, left, right, left top diagonal, right top diagonal, left bottom diagonal, right bottom diagonal

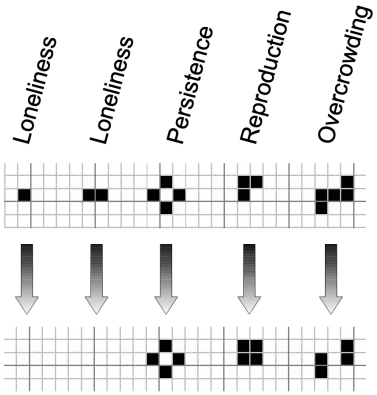


Figure 1.1. Rules to the game of LIFE and how they play out in the grid

we will show that the game of LIFE can be used to solve any computational problem a computer can solve.

1.4. **Patterns of the Game of Life.**

Definition 1.1 (Pattern)**.** A unique arrangement of living cells.

If started with random initial conditions, the game of LIFE generates and destroys a multitude of different objects during the evolution process until it reaches the final steady state. By probability, the final steady state consists mostly of a population of five different objects. These objects are commonly known as the oscillatory blinker and the still lifes block, boat, beehive, and burloaf. Detailed analyses of these patterns as well as other well known ones are provided below.

We begin with introducing the main categories of patterns.

Still life. A still life pattern is one that does not change in time, a stable, finite and nonempty pattern.

Examples of still lifes include the block, the beehive, the boat, the ship, and the loaf, amongst others (see Figure 1.2).

Oscillator. Oscillating patterns, also called *Periodic Life Forms*, continuously alternate between two or more unique arrangements, oscillating periodically.

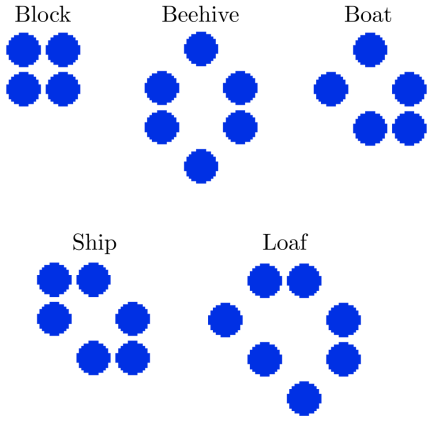


Figure 1.2. Examples of still life patterns

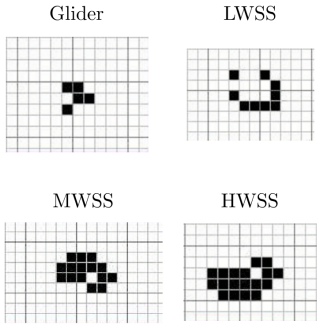


Figure 1.3. Examples of oscillators

Examples of oscillators with period 2 include the blinker and the toad (see Figure 1.3). Oscillators of many more periods exist, such as the glider, spaceship, and gun. These patterns are more complex, and thus we will describe them in greater detail in the following sections.

Spaceship. Firstly, let us take a birds-eye view at these patterns. A spaceship is an oscillator that moves across space over time. Types of spaceships include *gliders*, *Light Weight Space Ships (LWSS)*, *Medium Weight Space Ships (MWSS)*, and *Heavy Weight Space Ship (HWSS)*, amongst others.

Glider. A glider is a simple 5-cell pattern that repeats itself every 4 generations, but is offset diagonally by one cell. Gliders are the smallest and most common spaceship. The glider is said to travel with speed $\frac{c}{4}$ (Section 1.4.1 explains c). See Figure 1.4.

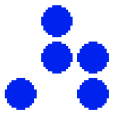


Figure 1.4. The glider

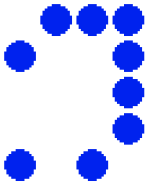


Figure 1.5. Light Weight Space Ship (LWSS)

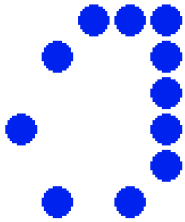


Figure 1.6. Medium Weight Space Ship (MWSS)

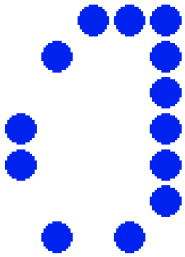


Figure 1.7. Heavy Weight Space Ship (HWSS)

Light Weight Space Ship. A spaceship that travels at speed $\frac{c}{2}$. Second most common spaceship. See Figure 1.5.

Medium Weight Space Ship. Third most common spaceship after the glider and lightweight spaceship. Travels at $\frac{c}{2}$ orthogonally. See Figure 1.6.

Heavy Weight Space Ship. Fourth most common spaceship after the glider, lightweight spaceship and middleweight spaceship. Travels at a speed of $\frac{c}{2}$ orthogonally. See Figure 1.7.

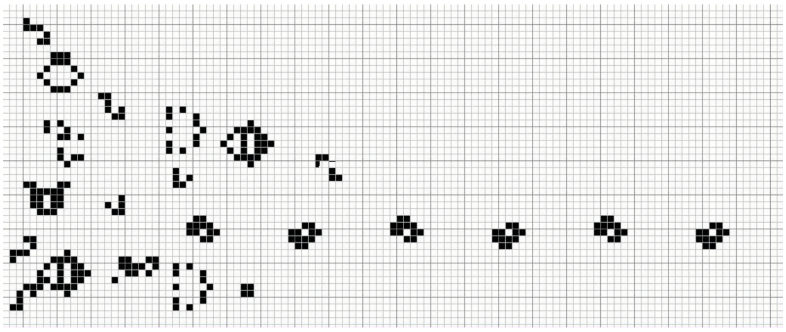


Figure 1.8. Fleet

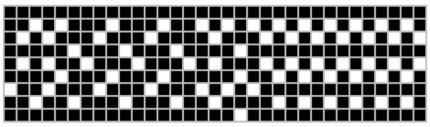


Figure 1.9. Garden of Eden

Fleet. A collection of space ships travelling along the same trajectory. See Figure 1.8.

Gun. A pattern that periodically generates a space ship along a fixed trajectory.

Garden of Eden. Does not belong to one of the categories described above, but is interesting and thus included. The Garden of Eden is a pattern that can only exist as initial pattern. In other words, no parent could possibly produce the pattern. See Figure 1.9.

We now have covered the main patterns in the game of LIFE, although there are many more that have been discovered.

1.4.1. *Stability of patterns generated by the Game of Life.* We will briefly discuss the stabilization of patterns. Firstly, stability is important because when we later build complex machines, it is ideal that they stay together. Thus, certain patterns must be stabilized to be useful, such as the Queen Bee shuttle. The Queen Bee shuttle is the smallest known oscillator with period greater than 15, and is the basis of the Gosper glider gun, the first known and most compact gun in the game of LIFE. The Queen Bee shuttle can be stabilized, as with other oscillators, by deleting the beehives as they are formed. See Figure 1.10.

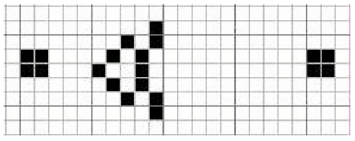


Figure 1.10. Stabilized Queen Bee shuttle

Infinite Game of Life patterns. It is not immediately obvious whether a given initial Life pattern can grow indefinitely, or whether any pattern at all can. This is because it simply is not easy to tell from looking at an initial game of LIFE configuration how exactly it will evolve. For example, let us evaluate what happens to a straight line of n live cells as a start configuration.

As seen, it is hard to evaluate from the starting configuration how the pat-

n	Ultimate pattern
1, 2	Fades immediately
3	Blinker
4	Becomes a Beehive at time 2
5	Traffic lights at time 6
6	Fades at $t = 12$
7	Makes a symmetric display before terminating in the Honey Farm
8	Gives 4 blocks and 4 beehives
14, 15	Vanish completely

Table 1. Straight line of n live cells after game of LIFE evolution

tern will ultimately evolve during the game of LIFE. In 1970, a pattern was discovered that proved that certain initial configurations can grow indefinitely throughout the game of LIFE. This pattern was the glider gun, which emits a new glider every 30 generations. Since the gliders are not destroyed, and the gun produces a new glider every 30 generations indefinitely, the pattern grows forever, and thus proves that there can exist initial Life patterns that grow infinitely.

Speed of the Game of Life. We will briefly discuss speed in the game of LIFE before concluding this introduction of the game of LIFE. In the game of LIFE, we can define the "speed of life" c as the maximum attainable speed of any moving object, a propagation rate of one step (horizontally, vertically, or diagonally) per generation. This is both the maximum rate at which information can travel and the upper bound on the speed of any pattern.

For example, the glider takes 4 generations to move one cell diagonally, and so it has a speed of $\frac{c}{4}$. The light weight spaceship moves one cell orthogonally every other generation, and so it has a speed of $\frac{c}{2}$.

We now prove that the maximum attainable speed of any moving object is either $\frac{c}{4}$ diagonally or $\frac{c}{2}$ orthogonally. No spaceships can move faster than the glider or light weight spaceship.

Theorem 1.2. *No spaceship can travel diagonally faster than* $\frac{c}{4}$.

Proof. Consider the grid below.

A				
	B	U	X	
	J	C	V	
		K	D	
				E

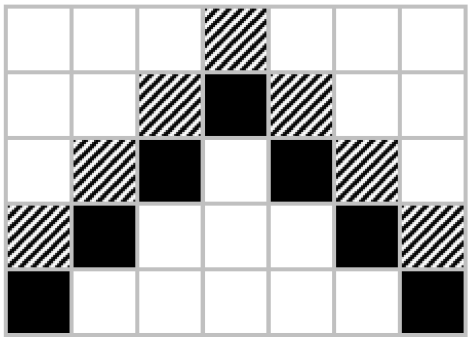
Say we have a spaceship on and to the left of the diagonal line defined by ABCDE on generation 0. Let us assume that the spaceship can travel up and to the right faster than $\frac{c}{4}$. Then, the cell X would be alive at generation 2.

Suppose this is true, and that X is alive at generation 2. Then, C, U, and V must be alive at generation 1. Then U and V must have had 3 alive neighbors in generation 0, and so B, C , D, J, and K must have been alive at time 0. Therefore, C must have had at least four live neighbors in generation 0, and so could not have survived to generation 1. But we needed C alive at generation 1, and so we have reached a contradiction.

Thus, if the spaceship is behind ABCDE at generation 0, it must be behind UV at generation 2, and therefore cannot travel faster than $\frac{c}{4}$. ■

Theorem 1.3. *No spaceship can travel orthogonally faster than* $\frac{c}{2}$.

Proof. Consider the grid below.



Let us say a spaceship is on and below the diagonal lines defined by the solid black squares in generation 0. Then, we can use the argument in the proof above to claim that the spaceship must be on or below the lines defined by the striped squares at generation 2. Therefore, it can move at most 1 square forward every 2 generations, and thus has maximum speed $\frac{c}{2}$. ■

This marks the end of the introduction of the game of Life. We now discuss Turing Machines and prove that the Game of Life is Turing-complete.

2. Turing Machines

Turing machines are a model of computation that describes an abstract machine that manipulates symbols on a strip of tape according to a table of rules. Although simple, Turing machines singled out as the next model of computation to explore due to their apparent ability to perform arbitrary algorithmic tasks. Moreover, Turing machines appear capable of implementing any computer algorithm. The Church-Turing Thesis states that "everything computable is computable by a Turing machine," which although is not proven, is nearly universally accepted.

Whilst there are many other models of computation that we can use, many of them more practical than Turing machines in terms of computing the particular computation more quickly, they cannot do anything a Turing machine cannot already do. Thus lies the appeal of Turing machines. We begin by defining a formal definition of Turing machines.

2.1. **Defining Turing Machines.**

Formally put, a Turing machine is an octuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \sqcup, q_f, q_r),$$

where Q is the set of states, Γ is a finite set of symbols called the *tape alphabet*, $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function, $q_0 \in Q$ is the initial state, $\sqcup \in \Gamma$ is a special symbol called the *blank*, $q_f \in Q$ is the accepting state,

and $q_r \in Q \setminus \{q_f\}$ is the rejecting state.

For our particular usage of exploring how the game of Life is Turing-complete, we distill the formal definition of Turing machines into the below criteria.

Firstly, a Turing machine has a finite set of states, rules for transitioning between states, an infinite sequence of cells known as a "tape," a set of symbols describing the possible contents of each cell in the tape, the ability to read and write the symbol in a single cell, and the ability to move along the tape to access different cells [Ren10].

Each state transition rule specifies the current state, the symbol read from the current cell, the state to move to, the direction to move the machine along the tape, and the symbol to write to the current cell [Ren10].

On each time step, the Turing machine reads the symbol from the current cell on the tape, finds the appropriate transition rule that matches the current state and the just symbol read, changes to the specified new state, writes the specified symbol, and moves the machine in the specified direction [Ren10].

2.2. **Universal Turing Machines.** A universal Turing machine is a Turing machine that can simulate an arbitrary Turing machine with arbitrary input. The input of universal Turing machines consists of both the description of the machine to be simulated, or the specification of a Turing machine via specifying a set of states and state transitions, and the input to this specified Turing machine from its own tape. The output of a universal Turing machine is the result of having run the specified Turing Machine on the given input. Universal Turing machines can simulate any Turing machine, including itself.

The game of Life exhibits universality: in other words, we can construct a universal Turing machine from the game of Life. Thus, the game of Life can theoretically compute anything that can computed. We work out *why* in the next section.

3. Game of Life is Turing-complete

Definition 3.1 (Turing-complete)**.** A model of computation is *Turing-complete* if it has rules followed in a sequence that hold the same computational power as a Turing machine.

We can show that the game of Life is capable of universal computation by displaying that elements of the game of Life can create a universal Turing machine. Distilling the requirements for the construction of a universal Turing machine from previous sections, we recall that we need a method of storing information, a method to transmit information, and a method of reading and writing information. Moving and stable objects similar to the *glider* provide a method for the transmission of information and ability to read and write. A glider at a certain location at a certain time can be interpreted as 0 and 1 of binary code. Glider guns, streams of gliders, are thus similar to the wires in usual computers, wherein they transmit bits of information. When gliders collide, information is altered or processed.

We conclude that the game of Life is capable of universal computation because we can implement the three fundamental logic gates, AND, OR, and NOT, in the game of Life. For the inputs of these logic gates, thinned and coded glider streams are used. *Uncoded glider*streams are also necessary. Glider streams are timed and spaced such that particular gliders collide then vanish. The *eater* is also necessary for computing in game of Life, as the eater can destroy unnecessary gliders without destroying itself [Rec95].

For the storage medium required to construct a universal Turing machine, the *block* pattern of the game of Life can be used. Blocks can store information in that the distance of the block from a certain point can serve as the coded information. Fleets of gliders can move the block in any direction to increase or decrease the distance of the block. These processes can be used to read or write the information stored in the block's distance.

In summary, with the block, eater, and glider gun patterns of the game of Life, we can create a universal Turing machine, and thus the game of Life is capable of universal computation.

For more specification, we can directly construct a universal Turing machine in the game of Life. To collect the necessary pieces of states, transition rules, a tape, possible symbols for each cell in the tape, cell read and write access, and multiple-cell access, we use the following aspects of the game of Life. States, transition rules, and possible symbols [0, 1, 2] are inherently built into the design of the game of Life. Rendell's memory cell can be used to implement storage, wherein one output of a fanout is looped to form an 8-glider memory [Ren10]. To create the tape, we arrange two stacks into one

tape, such that to move the machine along the tape, pop one stack and push the other. Eight symbols of 3 bits per element are used to implement stacks, where bits are trapped gliders. Each stack element holds up to three gliders, trapped between two anti-parallel fleets of gliders. To push and pop stacks in the game of Life system, we can make gaps in the fleets and propagate signals using fanouts. Thus, we have collected all necessary parts to create a universal Turing machine.

3.0.1. *What the Game of Life's Turing-completeness means.* The game of Life's Turing-completeness and capability to sustain a universal Turing machine indicates that if decision can be made by a Turing machine, it can also be made by a cellular automaton.

Strengths and weaknesses of building a Turing machine in the Game of Life. The machine we describe is Rendell's machine [Ren10]. In terms of weaknesses: Rendell's machine has a non-infinite tape and is quite complicated to implement, and is therein nonideal. Contrarily, strengths include easy extensions of the length of the tape, and that the machine constructed can be extended to a universal Turing machine.

4. Game of Life is Irreversible

Definition 4.1 (Reversibility)**.** A reversible cellular automaton is a cellular automaton where every configuration has a unique predecessor.

Generally speaking, the game of Life is irreversible. Conducting a quick thought experiment gives a rough proof of this statement. For one, patterns can die out completely in the game of Life, which makes it impossible to reverse to the predecessor configurations, as many initial states can lead to the same end state of complete emptiness. From another angle, certain game of Life patterns such as the Garden of Eden have no predecessors at all.

However, particular patterns of the game of Life can be modified to create a reversible cellular automaton. For further exploration, [Mor22] details how gliders can exist in reversible cellular automata. Machine learning techniques can also be used to attempt to reverse the game of Life, although even if such techniques achieve a high degree of accuracy in terms of approximation, the game of Life is still irreversible.

5. Applications of the Game of Life

5.1. **Game of Life and Circuits.** Due to the game of Life's ability to approach the behavior of complex natural phenomena by utilizing the locality

of interconnected simple elements and implement universal Turing machines, work can be done into reproducing the game of Life circuit-level. Such designs branch into the development of electronic systems that can operate real-time and communicate with other biological systems, similar to the essential operation of the game of Life itself. [KFN+18] explores one application of the game of Life in memristor circuits.

5.2. **Biological applications of the Game of Life.** The game of Life has many biological applications, especially in terms of modeling, due to its original concept being based off of nature's population dynamics and interactions between different agents. One of the original goals of the game of Life was to model the unpredictability of actual populations in nature, such as in studying the evolution of ecological communities. The game of Life can create simulations modeling everything from ants to traffic, clouds to galaxies. The game of Life can also be applied to biological processes such as symbiopoeisis, such as the explorations done in this study ([Ada10]).

5.3. **Brief digression: Philosophy of the Game of Life.** A key concept of the game of Life regards the ability for complexity to arise from simplicity. In essence, the game of Life mimics the philosophy Darwin conceptualized when he was observing actual life forms on the Galapagos Islands, the theory of evolution wherein complex life forms originate from simpler ones. Such a revelation is philosophically profound, in that complexity tends to be associated with complexity rather than utter simplicity. Simultaneously, the game of Life reflects the inherent unpredictability of life, in that unpredictability is rampant even in the simple world of the game of Life. One of Life's most important lessons, whether it be the game of Life or actual life, is that uncertainty is the only certainty.

6. Conclusion

This paper focused on introducing the game of Lifeand its Turing-completeness. Many applications exist regarding the game of Life, from electronics to biology to philosophy, which should be interesting to explore for the reader, such as using the game of Lifeto model population dynamics.

References

[Ada10] Andrew Adamatzky. *Game of Life Cellular Automata*. 01 2010.

[KFN+18] Rafailia-Eleni Karamani, Iosif-Angelos Fyrigos, Vasileios Ntinas, Ioannis Vourkas, and Georgios Ch. Sirakoulis. Game of life in memristor cellular automata grid. In *CNNA 2018; The 16th International Workshop on Cellular Nanoscale Networks and their Applications*, pages 1–4, 2018.

[Mor22] Kenichi Morita. *Gliders in the Game of Life and in a Reversible Cellular Automaton*, pages 105–138. Springer International Publishing, Cham, 2022.

[Rec95] Andreas Rechsteiner. Complexity properties of the cellular automaton game of life. Master's thesis, Portland State University, 1995.

[Ren10] Paul Rendell. *A Simple Universal Turing Machine for the Game of Life Turing Machine*, pages 519–545. Springer London, London, 2010.

Euler Circle, Palo Alto, CA 94306
Email address: kaitlynz@ohs.stanford.edu