INSTITUT FÜR INFORMATIK
der Ludwig-Maximilians-Universität München

JOHN HORTON CONWAY'S GAME OF LIFE

An overview and examples

Raphael Hirte

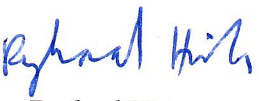
Bachelorarbeit

Betreuer Prof. Dr. François Bry
Mentoren Nemi Pelgrom

Abgabe am 30.03.2022

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst habe und keine anderen als die angegebenen Hilfsmittel verwendet habe.



München, den 30.03.2022

Raphael Hirte
11165412

Abstract

John Horton Conway's Game of Life is the most well known cellular automaton. It takes place on a grid of square cells which have one of the two states "dead" and "alive". Then the following generations of the population of cells are determined by specific rules. Depending on the initial configuration, there can be different possible outcomes; Still Lifes which do not change, Oscillators which return to their original form after a fixed number of generations, or Spaceships, that move across the grid. This Bachelor thesis gives an overview about the Game and its variations, and shows examples of the most important patterns. In order to do this, the relevant literature was collected, analyzed, and the key messages from each paper were extracted. Although the Game of Life originally comes from mathematics, it is relevant to people from many different areas, because it can be used in various fields of application, such as science, biology or cryptography. In this thesis, the reader can find an overview of interesting aspects of the Game.

Zusammenfassung

John Horton Conways Game of Life ist der bekannteste zelluläre Automat. Es findet auf einem Gitter aus quadratischen Zellen statt, die einen der beiden Zustände ”tot” und ”lebendig” haben. Dann werden die folgenden Generationen der Zellpopulation durch bestimmte Regeln bestimmt. Abhängig von der anfänglichen Konfiguration, kann es verschiedene mögliche Ergebnisse geben; Stillleben, die sich nicht ändern, Oszillatoren, die nach einer festgelegten Anzahl von Generationen zu ihrer ursprünglichen Form zurückkehren, oder Raumschiffe, die sich über das Gitter bewegen. Diese Bachelorarbeit gibt einen Überblick über das Spiel und seine Variationen und zeigt Beispiele der wichtigsten Strukturen. Dazu wurde die relevante Literatur gesammelt, analysiert und dann die Kernaussagen aus jedem Paper zusammengestellt. Obwohl das Game of Life ursprünglich aus der Mathematik stammt, ist es für Menschen aus vielen verschiedenen Bereichen relevant, da es in verschiedenen Anwendungsfeldern wie Wissenschaft, Biologie oder Kryptografie eingesetzt werden kann. In dieser Arbeit findet der Leser einen Überblick über interessante Aspekte des Spiels.

Acknowledgments

At first, I want to thank Prof. Dr. François Bry for his supervision during the writing of my Bachelor thesis. He gave me some very important inspirations, that improved the research process for the thesis.

Of course, I need to thank Nemi Pelgrom for her great help and advice in our weekly Zoom meetings, where she answered all my questions. Without her, it would not have been possible to write this thesis.

I also want to thank my brother Konstantin Weißenow, who supported me in the process of choosing this topic.

Finally, I would like to say thank you to my boss Dr. Julius Heintz, who agreed to let me work part-time for one more semester, to give me enough time to finish my studies and this thesis.

Contents

CHAPTER 1

Introduction

There are few topics in science that can be used in as many fields of application as cellular automata. From mathematics to computer science, biology, music and even traffic analysis, there is almost no problem that can not be described by a cellular automaton. A cellular automaton is a model built from a set of sites that all have a specific state. The states of the sites are then changed (or not changed) by certain rules. While the first cellular automaton which was built by John von Neumann in 1966 was very complex with its 29 states, there are much simpler ones which have only two states. Despite this simplicity, many of them show a very complex behavior, and some have even been proven to be universal (or Turing-complete) which means that a Universal Turing Machine can be built with them. Therefore, everything that can be computed can be computed with them. Most cellular automata are one and two-dimensional, but there are also higher dimensional ones.

The most well known cellular automaton is the Game of Life, a two-dimensional cellular automaton which has been invented by John Horton Conway in 1970. It is a simulation that describes the evolution of a population of cells across a two-dimensional grid of squares by three (or four, depending on the wording) simple rules. Every cell has one of the two states "dead" and "alive", and the state changes of the cells depend on the number of living neighbor cells. When it was first published by Martin Gardner in the journal Scientific American in 1970, it was praised to have "fantastic combinations" [16]. Since its invention, thousands of patterns have been found by many people, and there seems to be no end in sight. Even after more than 50 years of history, interesting new patterns are still discovered.

This thesis gives an overview about the Game of Life. Chapter 2 starts with some related work, both about the Game of Life and other interesting cellular automata. Then the mathematical background of cellular automata will be explained (chapter 3), focusing on two-dimensional ones. In the main chapter 4, the rules of the Game of Life will be displayed, and examples of the most important patterns will be shown. Among others, there are "Still Lifes" which do not change at all, "Oscillators" which oscillate between different structures and return to its original form after a fixed number of generations, and "Spaceships" which move across the grid. The latter are responsible for many of the countless computing possibilities of the Game of Life, because they can encode data. Some of these and other mathematical results on the Game will be shown in this section, too. Later in the chapter, some of the many possible variations of the Game will be presented. Afterwards,

several fields of application of the automaton will be pointed out (chapter 5). Some of these, such as biology and cryptography, turned out to be very interesting. Lastly, there will be a timeline of the major events in the history of the Game of Life, starting in 1970 with its invention, to the latest discoveries (chapter 6).

CHAPTER 2

Related Work

In this chapter, some other work that is similar will be presented. The first part contains articles and books that are specifically about the Game of life, while the second part also covers other cellular automata.

2.1 Game-of-Life-related Work

There are a lot of scientific articles about the Game of Life. Central, of course, is the first article, in which the Game of Life was published. This was the article "MATHEMATICAL GAMES The fantastic combinations of John Conway's new solitaire game 'life'" which was published in 1970 by Martin Gardner in the journal Scientific American [16]. This was the start of the still ongoing history of the Game of Life.

Another important work is the book "Game of Life Cellular Automata" by Andrew Adamatzky [1]. This is a large collection of many important articles about the Game of Life from 35 different authors all across the world. If someone wants to know more about the Game, this book will be a good starting point

2.2 Not-Game-of-Life-related Work

Apart from the Game of Life, there are many other forms and versions of cellular automata. In the following, I will list some of them which are either well known or from different fields of application, to make it as varied as possible. The first cellular automaton was created by John von Neumann in 1966 [37]. It is a very complex automaton with 29 different states based on a Universal Turing Machine that can reproduce itself. This was simplified by Christopher G. Langton in 1984, when he created a cellular automaton with only eight states which gives up the universality, but still can run a program (Langton's Loops) that can reproduce itself [23]. Langton also created the automaton Langton's ant in 1986 which simulates the movement of an ant across the grid [24, Chapter 11.2.]. The rules are very simple for the ant, as it only uses two states, and only one cell changes its state every generation, but it has been proven to be universal [15]. Another interesting cellular automaton is a model introduced by Kai Nagel and Michael Schreckenberg in 1992 [36]. It is a one

dimensional automaton that can simulate freeway traffic. In 1997, Felix Gers, Hugo Garis and Michael Corkin introduced a cellular automaton named CoDi (Collect and Distribute) that can simulate Spiking Neural Networks [18]. Just recently, in 2020, a group of Romanian scientists developed an enhanced cellular automaton with autonomous agents that can simulate the spread of Covid-19 between people inside rooms [9]. It is interesting to see, that unlike the Game of Life, all of these automata have been developed for a specific purpose.

CHAPTER 3

Background

The Game of Life is a two-dimensional cellular automaton. This chapter gives a brief background about what cellular automata, and in particular two-dimensional cellular automata are.

3.1 Cellular Automata

The most simple ones are one-dimensional cellular automata. A one-dimensional cellular automaton is built from a line of sites [53, Chapter Cellular Automata]. Every site has one of k different values from 0 to $k-1$. The cites are labeled a_i, while i is the position of the cite on the line. The value of each site a_i is updated regularly by a given set of rules ϕ depending on a neighborhood of cells around it. The application of the rules is defined as $a_i^{(t+1)} = \phi[a_{i-r}^{(t)}, a_{i-r+1}^{(t)}, ..., a_{i+r}^{(t)}]$ at each time t. The value r is the number of neighbors that are considered. The complexity of the behavior of the automaton has no connection to the selection of the values, so even an automaton with $k = 2$ and $r = 1$ can be very complex. In this case, the sites have one of the two different values 0 and 1, and only the one, closest neighbor on each side of the site is considered for the application of the rules. So the rules depend on the following three sites: a_{i-1}, a_i and a_{i+1}.

One-dimensional cellular automata are often displayed in two dimensions by adding the time t as the second dimension. So the development of the automaton can be viewed from top to bottom (Figure 3.1).

This definition also works for n-dimensional cellular automata, with the difference, that it is not a line, but a n-dimensional space.

3.2 Two-dimensional Cellular Automata

To build a two-dimensional space, where the definition can be applied, there are several possibilities [39, Chapter 1]. The most common way is choosing a plane of squares. But triangles, pentagons or hexagons are also possible.

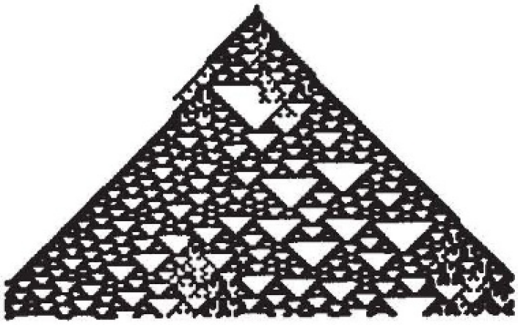


Figure 3.1: The figure shows the development of a one-dimensional cellular automaton in two dimensions, with time as the second dimension [53].

Once the shape has been chosen, there are still two possibilities for neighborhoods for square-planes (Figure 3.2):

1. The von Neumann neighborhood: Only four neighbors are considered.
2. The Moore neighborhood: All eight neighbors are considered.

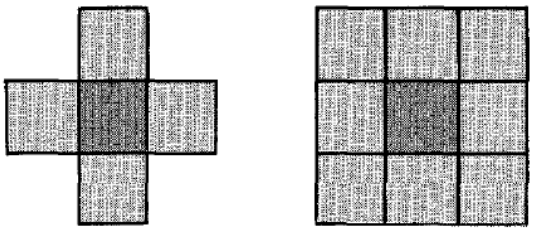


Figure 3.2: The figure on the left shows the von Neumann neighborhood, where only four neighbors are considered. The figure on the right shows the Moore neighborhood, with all eight neighbors. [39]

There is a very large number of possible rules, even for square planes, there are 2^{32} rules for von Neumann neighborhoods and 2^{512} for Moore neighborhoods. A possible rule in a von Neumann neighborhood with squares that have values 0 and 1 can be "If the values of the neighbors are as following, then the value of the site is changed: $top = 0, left = 1, right = 0, bottom = 1$. In all other cases, the value is not changed.". But these rules can be reduced by adding further conditions.

The rules of a totalistic cellular automaton depend only on the sum of the values in the neighborhood. This reduces the number of rules to $2^5 = 32$ for von Neumann neighborhoods and $2^9 = 512$ for Moore neighborhoods.

Another possibility are the outer totalistic cellular automata, where the rules depend on the sum of the values in the neighborhood and the value of the site itself. This leads us to the Game of Life, which is an outer totalistic cellular automaton using Moore neighborhoods.

CHAPTER 4

Game of Life

The Game of life is a two-dimensional cellular automaton invented by the British mathematician John Horton Conway in 1970 [16]. Due to its name it is sometimes referred to as a game, but actually it is more precisely a simulation, because the only input required by the "player" is the initial configuration of the automaton [6, Chapter 2.1]. From this point on it develops on its own, sometimes terminating, and sometimes not, depending on the input. This chapter describes the rules of the automaton and provides some examples of common structures which can be achieved in the Game of Life. Then mathematical results on the Game of Life are presented. After that, some possible variations of the original, intended rules, and other parts of the game will be explained. For the figures in 4.1 and 4.2 (except Figure 4.21 and Figure 4.22) the software Golly was used, a common implementation of the Game of Life [46].

4.1 Rules

According to John Conway, who invented it, the Game of Life is played on an infinite grid of square cells [16]. He suggested using a large checkerboard. Each cell can have two states: alive (1) and dead (0). It is important to notice, that each single cell has eight neighboring cells, making it a Moore neighborhood (Figure 4.1).

The simulation is started with an initial configuration of dead and alive cells, which is given by the player. From this point on, there is nothing the player can do to influence the progress of the game. From now on, the following generations of cells are built by the following simple rules (Figure 4.2):

1. Survival rule: Every living cell with two or three living neighbors will stay alive in the next generation.

2. Death rule: Every living cell with less than two or more than three living neighbors will die. [1]

[1] Sometimes, the Death rule is split into two rules [6, Chapter 2.2]: the death-by-isolation rule (Every living cell with less than two neighbors will die) and the death-by-overcrowding rule (Every living cell with more than three neighbors will die).

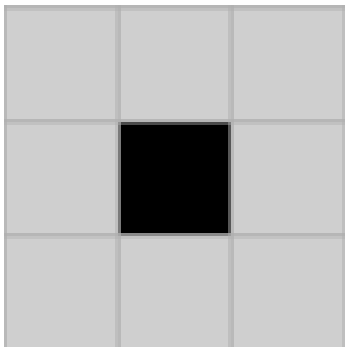


Figure 4.1: The black square depicts a living cell, and the white squares are its dead neighboring cells.

3. Birth rule: Every dead cell with exactly three living neighbors will be reborn in the next generation.

The application of these rules must be simultaneously, to make sure, that the state changes don't influence each other in the same move.

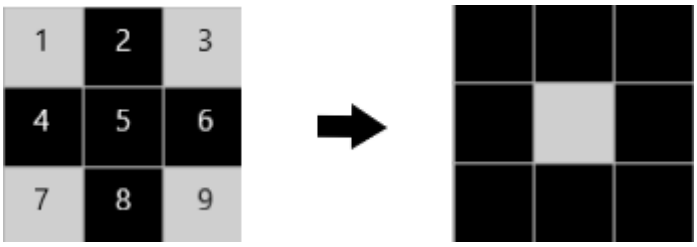


Figure 4.2: This figure shows an initial configuration, where all three rules must be applied. The cells on the left side of this picture are numbered, so they can be referenced easier. Survival Rule: Cells 2, 4, 6 and 8 are alive and have three living neighbors, so they will stay alive. Death Rule: Cell 5 is alive and has four living neighbors, so it will die. Birth Rule: Cells 1, 3, 7 and 9 are dead, but have exactly three living neighbors, so they will be reborn.

John Conway did not choose these rules arbitrarily. He experimented for a long time to find rules that make the behavior of an initial configuration unpredictable. In order to do this, he tried to meet the following three criteria:

1. There should be no initial configuration for which it is simple to prove that it can grow without limit.
2. There should be initial configurations that seem to grow without limit.
3. There should be simple initial configurations that after some time terminate in one of three ways:
   (a) Vanish completely
   (b) Turn to a stable configuration that does not change
   (c) Enter an oscillating phase that repeats in an endless cycle

4.2 Examples

The application of the rules on initial configurations can lead to numerous different results, some of them terminating, and some not [16]. These are the most common ways, in which a configuration can end:

1. The living cells form a stable figure that stays the same forever, if not disturbed. This is called "Still Life".
2. Configurations that oscillate between different forms with a certain period. This is called "Oscillator".
3. The object moves across the board eternally. These objects are called "Spaceships".
4. All cells die.

Combinations of them are also possible. Some outcomes, such as the so called "Glider Gun", do not fit in these categories. They will be explained later in this section. Moving objects can also collide to form new structures [42, Chapter 1.3]. This section shows the most common patterns, starting with the most simple ones, slowly leading to more complex structures.

4.2.1 Still Lifes

The most simple outcome is the so called "Still Life" [2]. A Still Life is a stable figure which does not change [16]. If a Still Life is the only object in an initial configuration, there will be no change at all. But they can also evolve from other configurations.

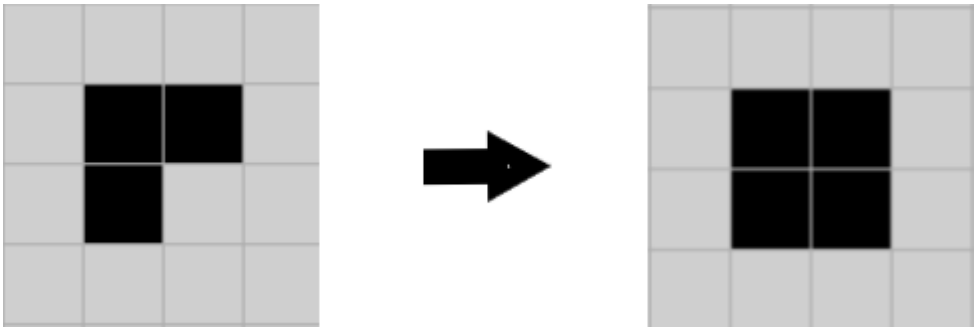


Figure 4.3: Block: The three living cells on the left side have two neighbors each, so they will stay alive in the next generation on the right side. The dead cell on the bottom right side has exactly three living neighbors, so it will be reborn.

The smallest possible Still Life is a 2x2 Block (Figure 4.3). One way to achieve it is an initial configuration of two living cells in a row, accompanied by a single cell next to them. This first example will be explained very detailed to make it easier to understand. The following ones will be explained in a shorter way. In this configuration each living cell has exactly two living neighbors, so according to the Survival rule, they stay alive for the next generation. The dead cell between the living ones has exactly three living neighbors, so according to the Birth rule, it will be reborn in the next generation. In generation two,

[2]The term "Still Life" comes from art, and has nothing to do with life, so its plural form is "Still Lifes" and not "Still Lives", as one would suggest [33].

all four cells have three living neighbors, so they stay alive. Additionally, there is no dead cell that has more than two neighbors, so there will be no rebirths in the third generation. The third generation is the same as the second, so there can be no more change in future generations, and the Block stays the same forever.

Another common Still Life is the Beehive (Figure 4.4). It can be reached after two steps by a line of four living cells in a row.

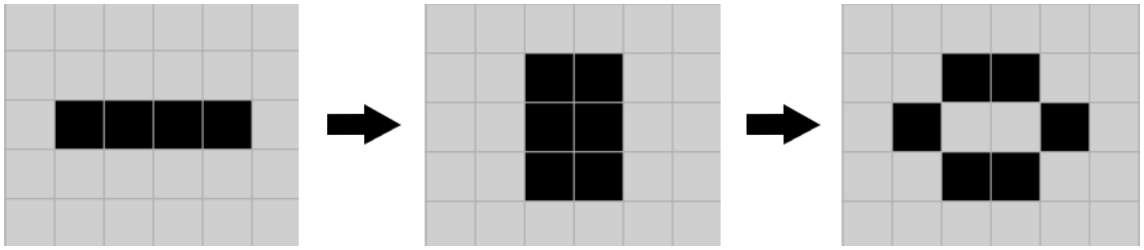


Figure 4.4: Beehive: The pattern on the left forms the Beehive on the right after two steps.

A circle of four Beehives is called a Honeyfarm. It can be achieved after 14 generations by seven cells in a row (Figure 4.5).

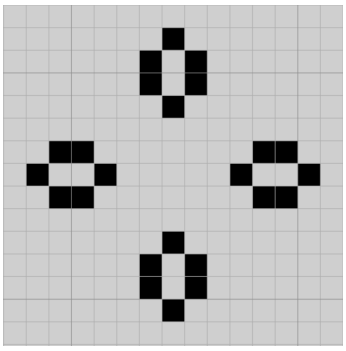


Figure 4.5: Honeyfarm: It is a circle of four Beehives.

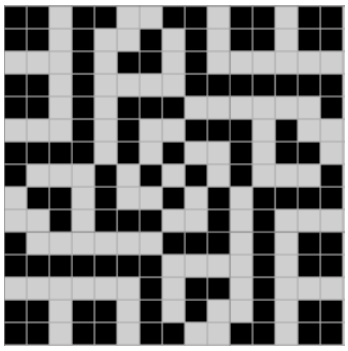


Figure 4.6: This Still Life is of maximum density. It is not possible to add more living cells without destroying it. [6, Chapter 3.5]

It is not possible to construct Still Lifes of any size, because of overcrowding (Death rule). There is always a maximum density of living cells which can't be surpassed [6, Chapter 3]. Figure 4.6 shows an example of a maximum density Still Life in a board of 15x15 cells.

4.2.2 Oscillators

Another interesting structure is the Oscillator. An Oscillator is a figure that changes its appearance only to return to its original form after a fixed number of generations [16]. This number is called the period of the Oscillator.

Figure 4.7: Blinker: This Oscillator changes between a horizontal and a vertical row of three cells.

The four most common Oscillators are the Blinker, the Beacon, the Toad and the Pulsar. Those appear very frequently as results of random initial configurations [14].

The most simple one is the "Blinker" (Figure 4.7). Again, unlike the following ones, this one is explained very detailed. It is a horizontal row of three living cells. The cell in the middle has two living neighbors, so it will stay alive (Survival rule). But the two outer cells have only one living neighbor each. So they will die (Death Rule). But the two vertical neighbors of the cell in the middle have three living neighbors, so they will be reborn. Now it is a vertical row of three living cells. After application of the same rules, the figure becomes the same horizontal row of three cells in the second generation, as in the beginning. So the process starts again and again and again. The Blinker is an Oscillator of period two. Oscillators of period two are also called Alternators [1, Chapter 3.4].

As mentioned before, three other common Oscillators are the Beacon, the Toad and the Pulsar (Figure 4.8) [16]. The Beacon and the Toad have a period of two. The Pulsar is a little bit bigger and has a period of three.

Bigger Oscillators can be formed out of smaller ones, so another common Oscillator is the "Traffic Lights" (Figure 4.9). These are four blinkers in a circle.

Another interesting Oscillator is a figure consisting of two 3x3 blocks touching each other in one corner [1, Chapter 3.6]. This one repeats itself after 8 generations, so it has a period of eight (Figure 4.10).

4.2.3 Spaceships

The third type of patterns is called the Spaceship. A Spaceship is a figure, that moves across the board continuously [16]. Just like an Oscillator, a Spaceship has a period. But it can also be characterized by its speed. The fastest possible speed is called the Speed of Light, which is the speed, a chess king moves across the board. There is no object in the Game of Life that can move with such speed. It has been proven, that the fastest speed is a fourth the speed of light diagonally, and half the speed of light vertically.

The most common Spaceship is called the Glider (Figure 4.11). A Glider is a small Spaceship consisting of five cells. It moves across the field diagonally with a fourth the speed of light with a period of four. Gliders are a common result of random initial configurations [14]. They can also be used to simulate an input for more complex patterns, as I will explain later.

Three other common Spaceships are the Lightweight, Midweight and Heavyweight

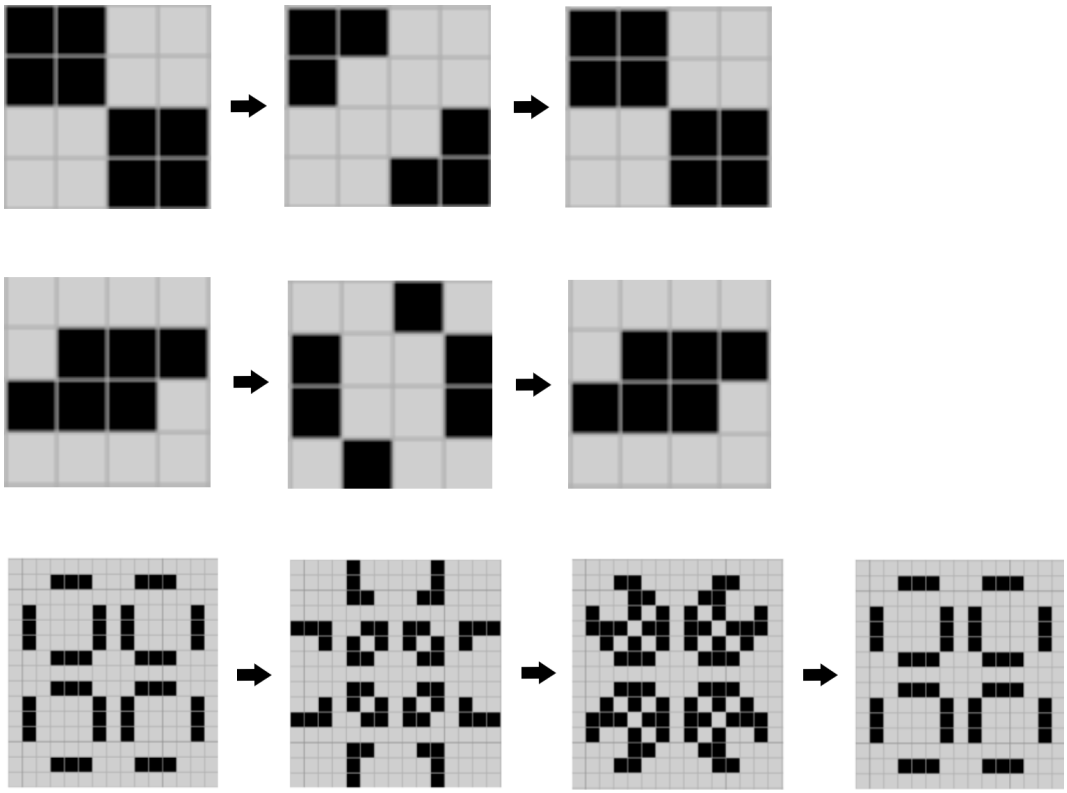


Figure 4.8: The first Oscillator is the Beacon. The second one is the Toad. Both have period two. The third one is the Pulsar with period three.

Spaceships [1, Chapter 1.1]. All of them are moving with half the speed of light horizontally and have a period of four (Figure 4.12).

There are many other Spaceships of almost all sizes. The largest Spaceship which has been found so far is the Caterpillar [38]. It is much too big to include a picture of it, because it consists of 11,967,399 cells. It has a period of 270.

4.2.4 Vanishing Patterns

There are many patterns that vanish completely after a few generations. The most simple ones are of course all single cells or a pair of two, because none of these can have two or three neighbors. So all cells die in the first generation according to the Death Rule.

The same happens to a diagonal row of three cells after two generations, because the cell in the middle has two neighbors in the first generation [16]. But the death of the other two cells makes it die in the second generation. This also means, that any diagonal row of cells vanishes, because in every generation the two cells on the outside die. In the end, there is only one cell left, that will die, too.

Objects can also vanish because of a collision with another object. This can happen, when Gliders collide with other patterns or with each other.

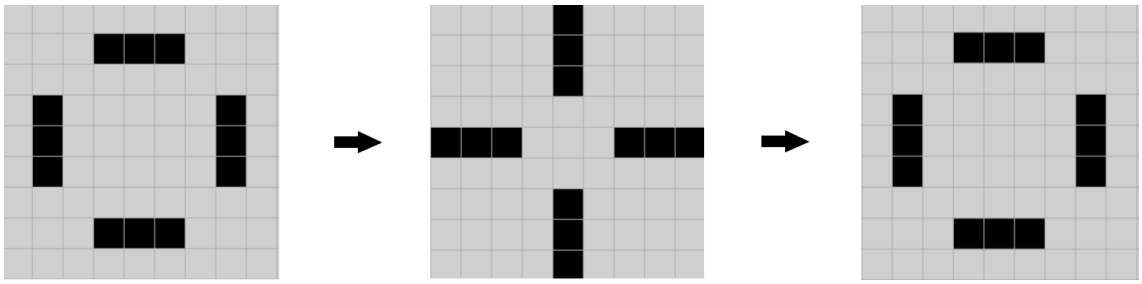


Figure 4.9: Traffic Lights: This Oscillator consists of four Beehives.

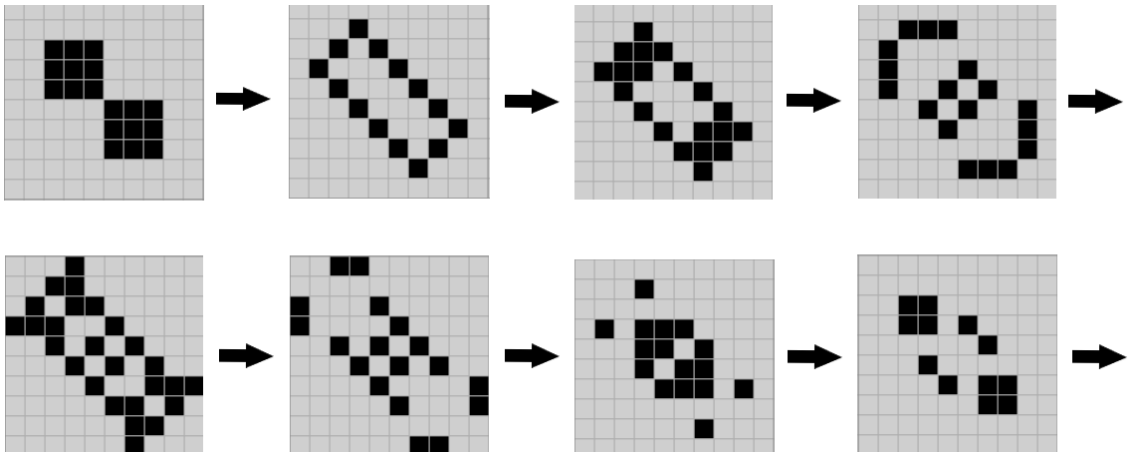


Figure 4.10: This Oscillator has a period of eight.

4.2.5 More Complex Patterns

Some patterns do not fit in the categories explained above. These can be combinations of objects of other categories or more complex patterns.

A very interesting pattern is the Glider Gun[3] [42, Chapter 1.1]. It is an object that shoots out a Glider and returns to its original form every 30 generations (Figure 4.13). Its existence is a proof, that it is possible to generate infinite growth in the Game of Life [16]. There are other guns as well, but this Glider Gun is by far the most well known [28].

The Glider Gun consists of four parts. A block on the left, a left part of the gun, a right part of the gun and a block on the right. The two inner parts of the gun move to the inside and collide. This collision produces a Glider. The parts now move to the outside, where they collide with the blocks. The blocks reflect the gun parts, which move back to the inside, where they produce another glider. This happens on and on. It is important to notice, that all the Gliders move in the same direction, so a constant stream of Gliders is generated. This Glider stream can simulate an input in a bigger pattern.

There are also objects that can collide with other objects and then return to their original form. This can be used to consume other objects. These objects are called Eaters.

[3]It is sometimes called Gosper Glider Gun after its inventor Bill Gosper [30].

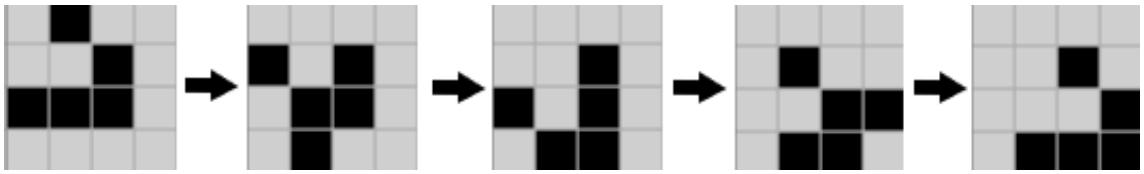


Figure 4.11: Glider: This small Spaceship moves with a fourth the speed of light at a period of four.

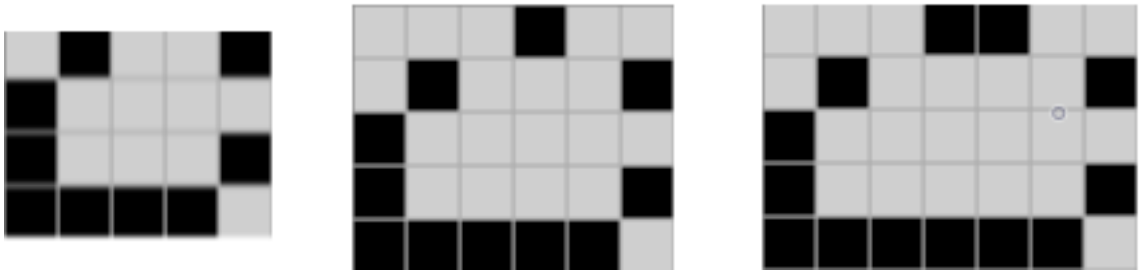


Figure 4.12: The figure shows from left to right the Lightweight, Midweight and Heavyweight Spaceship.

The most common Eater is the Stopper [42, Chapter 1.3.2]. A Stopper is a small Eater that can consume Gliders hitting it in the right angle (Figure 4.14). Because it returns to its original form after eating the Glider, it can be used to stop a stream of Gliders generated by a Glider Gun. When another living cell is added at the bottom left, the Stopper destroys itself completely.

Another Eater that takes a bit longer (13 generations) is shown in figure 4.15. This is still enough to stop the stream from a Glider Gun, which has 30 generations between each Glider. It can be used to detect Gliders, because there is one specific cell (red in the Figure) that comes to life only if a Glider hits the Eater. So this cell can simulate an output.

With these patterns, it is possible to create Logical Gates in the Game of Life [42]. To simulate an input to a Logical Gate, Glider streams from a Glider Gun can be used. A moving Glider stream means True. If the stream is interrupted by a Stopper, the input is False. The output is simulated by the Eater from Figure 4.15. When the specific cell (red in the Figure) is activated, the output is True, otherwise False.

The AND gate is formed by two Glider Guns that simulate the inputs A and B, and an inverted Glider Gun, a Stopper, to prevent Glider Streams to move outside the gate and an Eater for the output (Figure 4.16).

There are four possible cases:

1. When both A and B are False, there are no Glider streams from the input Glider Guns. The Glider stream from the other Glider Gun hits the Stopper, and not the output Eater. So the specific cell stays dead, and the output is False.

2. When A is True and B is False, there is only a Glider stream from the A Glider Gun. This stream collides with the one from the other Glider Gun, and the Gliders destroy each other. The output Eater is still not hit, so the output is still False.

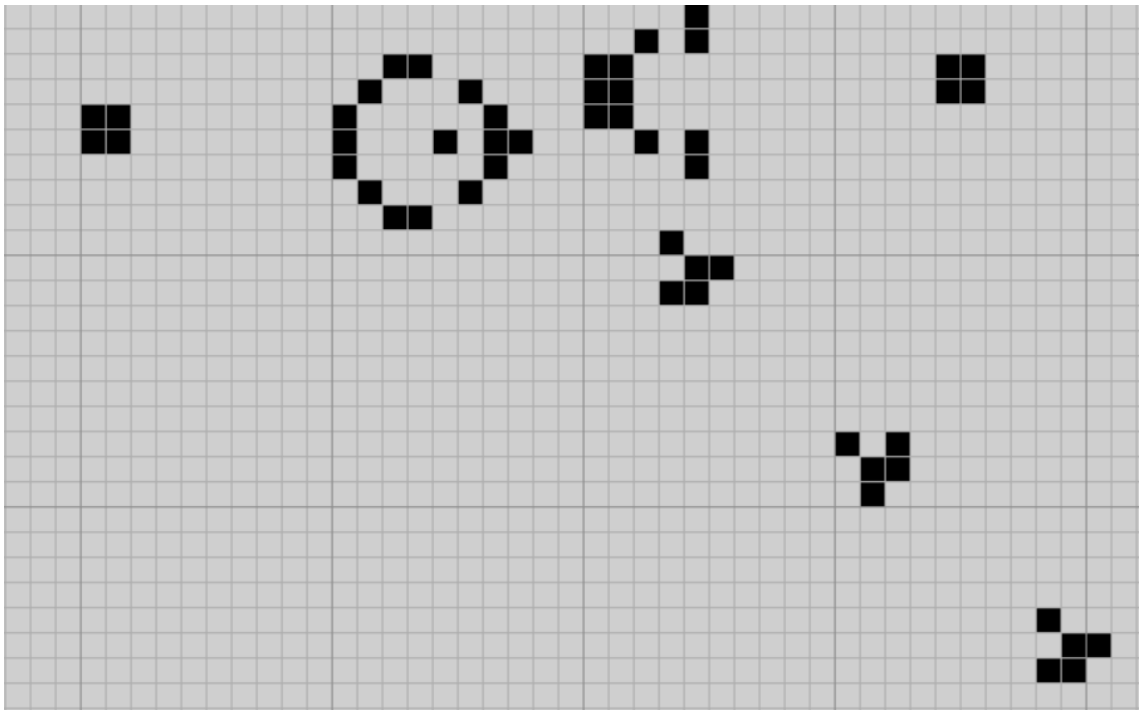


Figure 4.13: The Glider Gun shoots out a Glider every 30 generations.

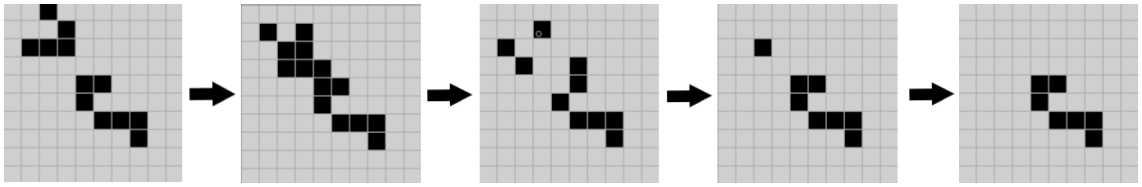


Figure 4.14: The Stopper consumes a Glider and returns to its original form.

3. When A is False and B is True, the collision occurs between the B Glider stream, and the other Gun's stream. So the output stays false.
4. But when A and B are both True, the A Glider stream collides with the other Gun's stream, and the A stream is free to hit the output Eater. The specific cell is activated, and the output is True, as it should be. All four cases are shown in Figure 4.17.

The OR gate is more complex, because it needs one more Gun, that shoots parallel to the input Guns. So it consists of a parallel Glider Gun, the two input Guns A and B, an inverted Gun, a Stopper and the output Eater.

Similar to the AND gate, there are also four cases:

1. When A and B are both False, there are only the streams from the parallel and the inverted Gun which collide with each other without hitting the output Eater. So the Output is false.
2. When A is True and B is False, the A stream collides with the stream from the inverted Gun, and the stream from the parallel gun hits the output Eater, making the output True.

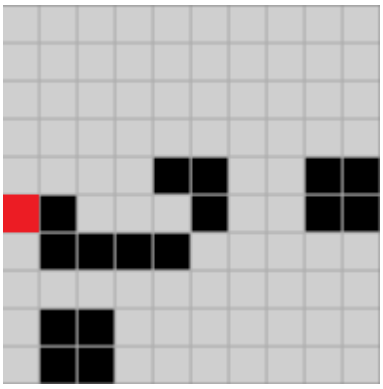


Figure 4.15: This Eater needs 13 generations to eat a glider. The red cell is not part of the Eater. It is only alive, when a Glider collides with it.

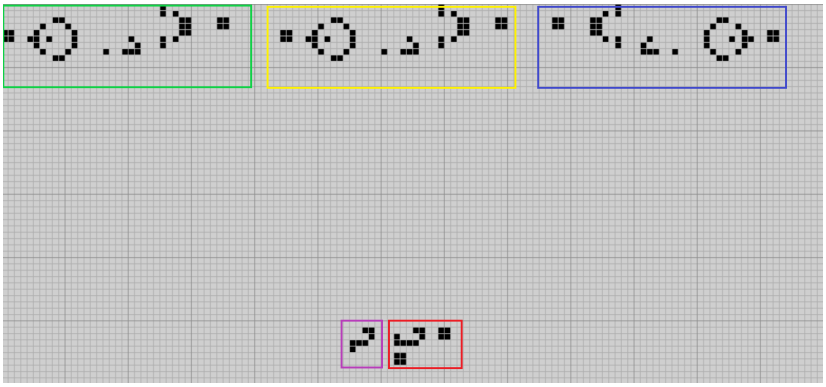


Figure 4.16: This pattern simulates an AND Gate. The green part is the B Glider Gun, the red part is the A Glider Gun and the blue part is the inverted Glider Gun. The purple part is the Stopper and the red part is the output Eater.

3. When A is False and B is True, the B stream collides with the inverted stream, similar to above, so the parallel gun hits the output Eater, again. The output is True.
4. When A and B are both True, the A stream collides with the inverted stream, the B stream hits the Stopper, and the parallel stream hits the output Eater. It is still true. This case is shown in Figure 4.18.

The NOT gate only needs one input, so it is the smallest gate. It consists of the A input Gun, an inverted Gun and the output Eater. Unlike the other gates, the input gun must shoot from right to left.

There are only two cases for this gate:

1. When A is False, there is no stream from the A gun, and the stream from the inverted gun hits the output Eater, making the output True. This case is shown in Figure 4.19.
2. When A is True, both streams collide with each other, so the output Eater is not hit. The output is False.

The outputs from the gates can be used to replace the inputs of other gates, so they can be combined with each other. This means, that every logical function can be implemented

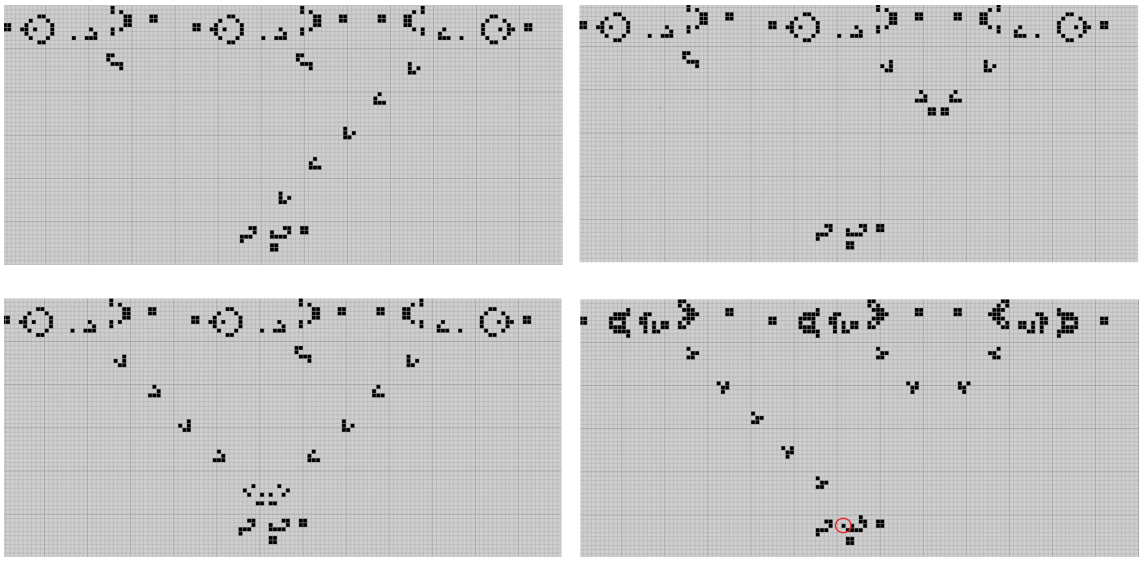


Figure 4.17: Top left: A False, B False. The output Eater is not hit, so the output is False. Top right: A True, B False. The output stays False. Bottom left: A False, B True. The output is still False. Bottom Right: A True, B True: The Glider Stream from the A Glider Gun hits the output Eater, and the specific cell (circled in red) is alive. So the output is true.

in the Game of Life. It is important to notice, that the input from the Not gate needs to be inverted before it can be combined with the other gates.

Other interesting patterns are the so-called Puffer Trains. These are patterns similar to Spaceships, but they leave behind a ”trail of smoke” [16].

One example is the Blinker Puffer 1 [26]. It has a period of eight and leaves behind a trail of Blinkers (Figure 4.20).

One of the most fascinating patterns in the Game of Life is the OCTA Metapixel (Figure 4.21) [12, Page 240-242]. This is a 2048x2048 cell wide object that simulates a large cell which can have the two states ”On” and ”Off”. By making many of these Metapixels communicate with each other by Glider Streams, it is even possible to simulate the Game of Life inside the Game of Life (Figure 4.22). Due to the fact, that every pattern in the Game of Life can be built with OTCA Metapixels, another, bigger OTCA Metapixel can be built, leading to a Game of Life inside the Game of Life inside the Game of Life. Theoretically, this could be repeated again and again. Not only the standard rules are possible to be simulated, but in fact any outer totalistic rule [11]. This leads to even more possibilities to imagine.

4.3 Mathematical Results

The Game of Life is not only fascinating in terms of the varied patterns, which can be built. Its mathematical characteristics are also interesting. In the following, the most important mathematical results about the Game of Life will be presented.

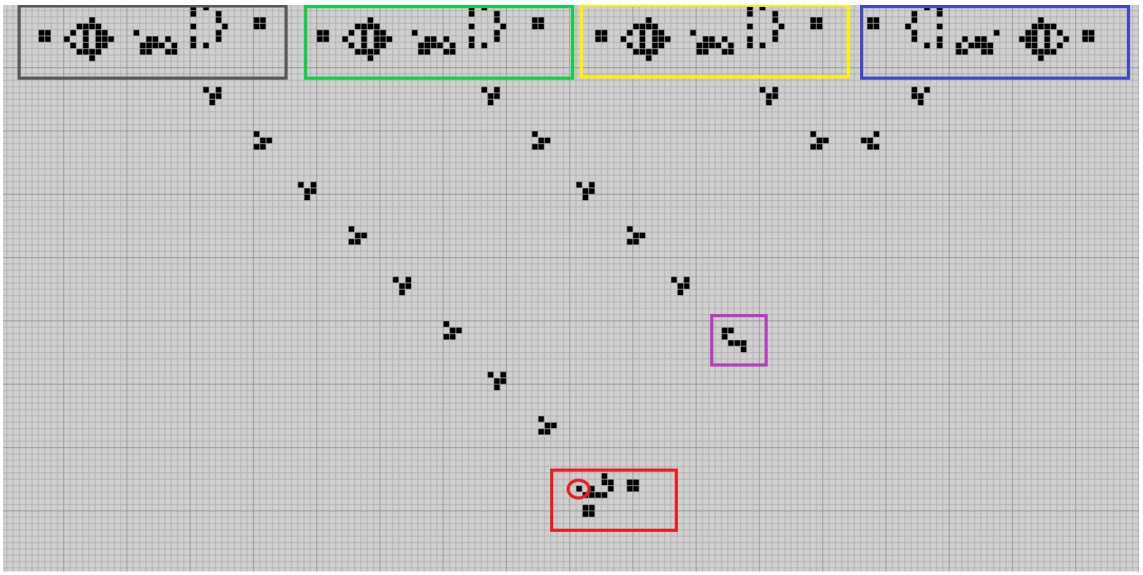


Figure 4.18: The OR gate: The gray part is the parallel Gun, the green part is the B Gun, the yellow part is the A Gun and the blue part is the inverted Gun. The purple part is the Stopper and the red part is the output Eater. Both, the A and B Guns are active, so A and B are True. The parallel Gun hits the output Eater and the specific cell is alive. So the output is True.

4.3.1 Unpredictability

In his book "Winning Ways for Your Mathematical Plays Vol. 2" from 1982[4], John Conway himself proved some important findings about his Game of Life [3, Chapter 25]. At first, he showed, that it is not possible to predict the outcome of an initial configuration. He did this by presenting the histories of many similar initial patterns that all evolved completely different. Some disappeared completely, while others turned into Still Lifes or Oscillators. A few even sent out Gliders. The last thing he needed was a pattern that could grow without limit, so the discovery of the Glider Gun finished his proof. So he confirmed his initial intention to make the Game of Life unpredictable (Chapter 4.1).

4.3.2 Gardens of Eden

In the same book he also proved that there are patterns that can only exist as an initial configuration, but have no predecessor. He called them Gardens of Eden. He assumed that in a $(5n-2) \times (5n-2)$ square there is a pattern that has no parent, if n is large enough. He showed this explicitly for $n = 465163200$, which means that there is a Garden of Eden in a $232581600 \times 2325816000$ square. Later, this has been largely reduced to 1400×1400.

4.3.3 Turing-universality

Besides these rather short proofs, this book also includes an 18-page-long proof that the Game of Life is universal (or Turing-complete). Due to its length, the components of his proof will only be sketched briefly. Conway composed his proof, by showing that all parts

[4]This was later split into 4 parts, so the relevant part was shifted to Vol. 4.

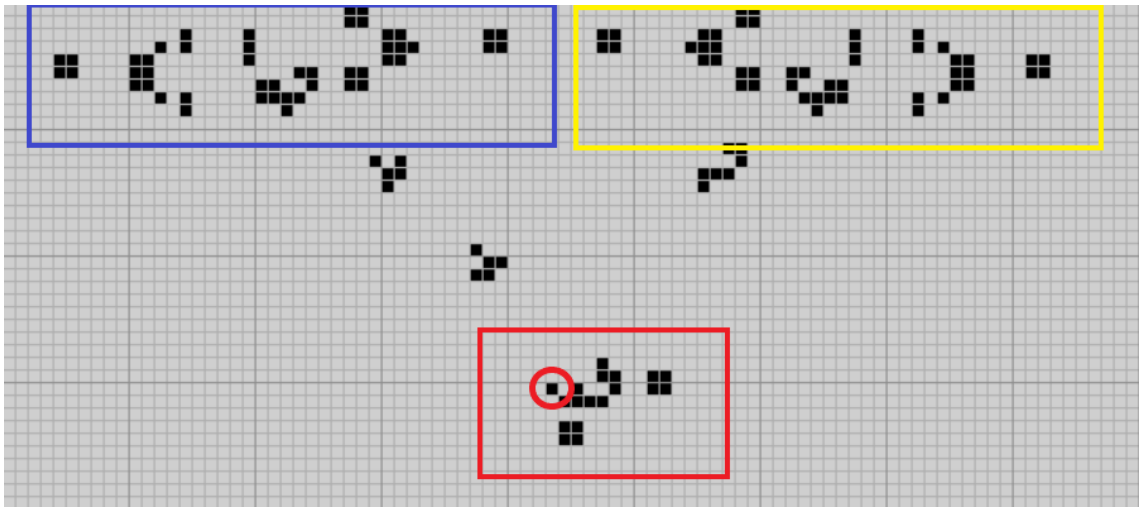


Figure 4.19: The NOT gate: The blue part is the inverted gun and the yellow part is the A input Gun. The red part is the output Eater. The A gun is not active, so A is False. The stream from the inverted Gun hits the output Eater, and the specific cell is activated. So the output is True.

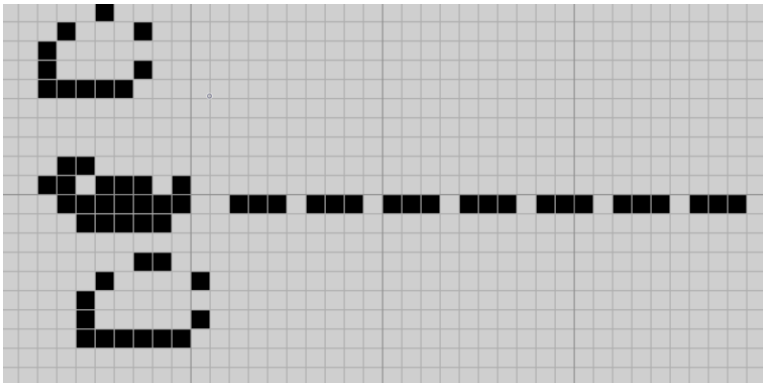


Figure 4.20: Blinker Puffer 1: This Puffer Train leaves behind a trail of Blinkers.

needed to build a Turing Machine can be built in the Game of Life. Nevertheless, he did not finish creating this Turing Machine. These are the components included in his proof:

1. Clock: This can be simulated by a Glider Stream from a Glider Gun.
2. Logical Gates: These can be built (Chapter 4.2.5).
3. Stopping of Glider Streams: This can be done by Eaters (Chapter 4.2.5).
4. Creating new Glider Streams: A Glider Gun can be built by letting Gliders collide.
5. Kickback Reactions: A Glider Stream can be forced to reverse its direction and travel back in the opposite direction close to the original Stream.
6. Crossing of Glider Streams: Glider Streams can be thinned, to enable other Glider Streams to slip through the resulting gap.

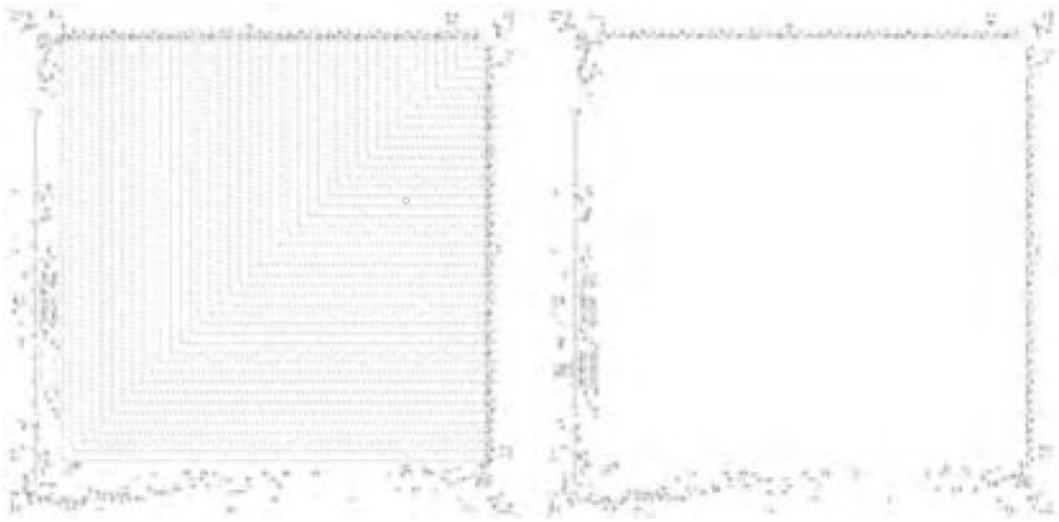


Figure 4.21: OTCA Metapixel: This large pixel can have the two states ”On” (left) and ”Off” (right) [12, Page 241].

7. Copying the information from a Glider Stream: Kickback Reactions can be used to achieve this.
8. Auxiliary Storage: This can be simulated by Blocks, whose distance from the Turing Machine determines the stored number.
9. Moving Blocks: Can be done by certain Glider-Block-collisions.
10. Creating Glider Streams close to others: Glider Guns can be programmed to emit Gliders at certain times. So two other Guns can be used to shift some Gliders of a Stream.

Afterwards, he showed that it is possible for the Machine to destroy itself by precisely sent Gliders.

Then he summarized all this to his proof that a Universal Turing Machine can be built in the Game of Life. This means that the Game of Life is Universal.

4.3.4 Self-replication

John Conway also showed that it is possible to build a pattern that can reproduce itself by creating an exact copy of itself and then destroying itself. Examples for self-replicating patterns were created by Dave Greene and Adam P. Goucher (Chapter 6.2) [31][25].

4.4 Variations

Though the rules and circumstances of the Game of Life are clearly defined, it is possible to vary it in many different ways. They are too many to mention, so in this chapter only some of them are described. These were chosen, because on the one hand they are similar

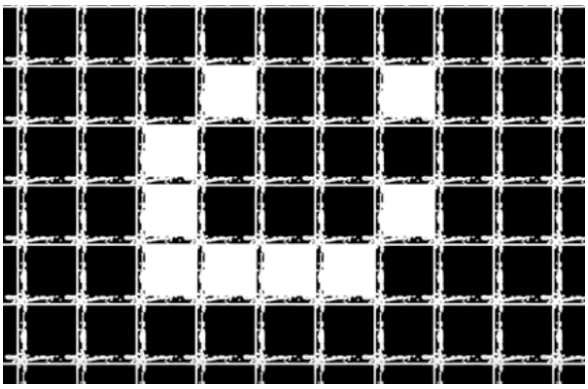


Figure 4.22: This is the Game of Life implemented in the Game of Life using the OTCA Metapixel [7].

enough to the original game, that the connection can be easily seen, and on the other hand, they are interesting enough to make them worth mentioning.

4.4.1 Rule Changes

One of these ways of varying the Game is by changing the rules.

The most simple way to do this is varying the number of neighbors that are required for a cell to change its state. This is done by Larger Than Life [13, Page vii]. Larger Than Life is not just one, but a family of variations, because it is kept more general than the original Game of Life.

The rules of Larger Than Life are the following:

1. Survival rule: Every living cell with δ_1 to δ_2 living neighbors, will stay alive in the next generation, with $1 \leq \delta_1 \leq \delta_2$.
2. Birth rule: Every dead cell with β_1 to β_2 living neighbors will be reborn in the next generation, with $0 \leq \beta_1 \leq \beta_2$.
3. Death rule: Every living cell that does not fulfill the Survival rule will be dead in the next generation.

This definition also covers the original Game of Life, when $\delta_1 = 3, \delta_2 = 4, \beta_1 = 3$ and $\beta_2 = 3$.

Another possibility is making state changes dependent to a certain probability.

This is done by the Stochastic Game of Life (SGL) [34]. This variation has two different versions: SGL1 and SGL2.

SGL1 is a one-dimensional version, so it relies only on the four nearest neighbors. The rules are the following:

1. Survival rule: Every living cell with four living neighbors will stay alive in the next generation. Every living cell with two living neighbors will stay alive with a probability of p_b. Every living cell with one or three living neighbors will stay alive with a probability of p_s.

2. Birth rule: Every dead cell with exactly three living neighbors will be reborn in the next generation. Every dead cell with exactly two living neighbors will be reborn with a probability of p_b.

This version has no specific Death Rule, but this functionality is included in the Survival Rule. A cell that does not survive dies.

SGL2 is two-dimensional, so the usual eight neighbors are considered. These are the rules:

1. Death Rule: Every living cell with zero, one or four or more living neighbors will die in the next generation.
2. Survival rule: Every living cell with two or three living neighbors will stay alive with a probability of p_s.
3. Birth rule: Every dead cell with exactly three living neighbors will be reborn in the next generation. Every dead cell with exactly two living neighbors will be reborn with a probability of p_b.

Unlike the original Game of Life, SGL is not deterministic, so the same initial configuration can lead to different results, when run several times. But for both versions, there exists one set of probabilities, where it is still deterministic. In SGL1 it is $p_b = 1$ and $p_s = 0$. In SGL2 it is $p_b = 0$ and $p_s = 1$. This is the original Game of Life.

4.4.2 Additional States

It is also possible to use more than the two states "dead" and "alive".

One possibility is the Cancer Cell Extension [10]. This version adds the third state "cancer". These are the rules:

1. Survival rule: Every living cell with two or three neighbors that are not dead will be not dead in the next generation. If there are more living cells than cancer cells around, it will be alive. Otherwise, it will be a cancer cell.
2. Birth rule: Every dead cell with exactly three not dead neighbors will be reborn in the next generation. If there are more living cells than cancer cells around, it will be alive. Otherwise, it will be a cancer cell.
3. Cancer Rule: Every cancer cell with one to three not dead neighbors will stay a cancer cell. A cancer cell can not become alive, except when it dies, and is reborn.

Figure 4.23 shows a Blinker in the Cancer Cell extension. It still works as an Oscillator, but with a cancer cell in the middle.

Another possibility is to include a second player with an additional state.

The Immigration Game does this [49]. It also has three states: "Dead", "Alive - Player 1" and "Alive - Player 2". It is also interesting, because it adds actual "playing" by making it possible to alter the game after it starts. Both players start with five living cells of themselves, that are placed alternately, one by one. Then the game starts. The rules are the same as in the original Game of Life, while both states "Alive - Player 1" and "Alive - Player 2" count as living cells. The only difference is in the Birth Rule, where the reborn cell

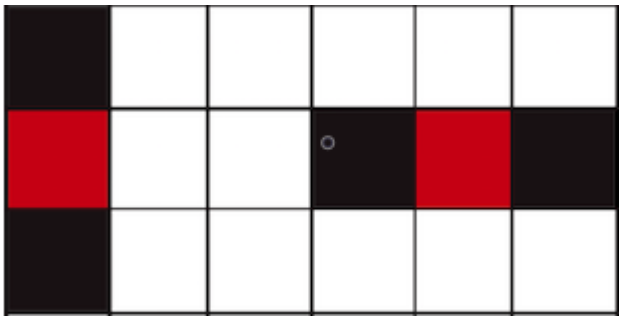


Figure 4.23: Blinker in the Cancer Cell Extension: The white cells are dead, the black ones are alive and the red ones are cancer cells. It oscillates between the left and the right pattern. [10]

receives the living state of the player who has the most neighboring cells. This is always possible, because of the odd number (three). Every ten generations, both players can place an additional living cell on a dead cell, starting with the one with the most living cells. If both players have the same number of living cells, this is suspended until one of them has more than the other. If one player has no more living cells on the grid, the other wins. If nobody has a living cell left, it is a draw.

4.4.3 Grid Changes

The original Game of Life is played on an infinite square grid. But this can be changed, too. Either by replacing the squares by other shapes or by not playing on a infinite, plane surface, but on topologies.

Instead of squares, triangles can be used [1, Chapter 17.2]. Here, each cell has twelve different neighbors (Figure 4.24, left). The original rules can still be used, but due to the different number of neighbors, many other possibilities are imaginable. The triangles can also have two different orientations. This can also be included in the rules.

Hexagons can be used, too (Figure 4.24, right).

Another possibility for changing the grid is leaving the shape of the cells untouched, but taking out the infinity of the grid. This needs to be done in every case, where the Game of Life is implemented (whether on a computer or in reality). But instead of simulating the infinity by using a very large board, there is another approach that makes cells that leave the board on one side enter it again on another. In this case, there is only a finite number of cells, but objects can move across the board in the same direction without end. This makes the board a topological space.

One example of that is Toridial Life, where the grid lies on a torus [17]. A torus is a topological space which looks similar to a donut. It can be constructed from a rectangle by ”gluing” two opposite sides together, which will form a tube. When the ends of the tube are glued together again, the result is a torus [17, Chapter 3.5]. Still Lifes and Oscillators that fit into the finite torus work without change, but Spaceships or especially objects with infinite growth like the Glider Gun can work only as long as they, or something they send out, collide with other objects (Figure 4.25)[17, Chapter 6.2].

It is possible to run the Game of Life on a Klein Bottle, but this does not add anything interesting [8].

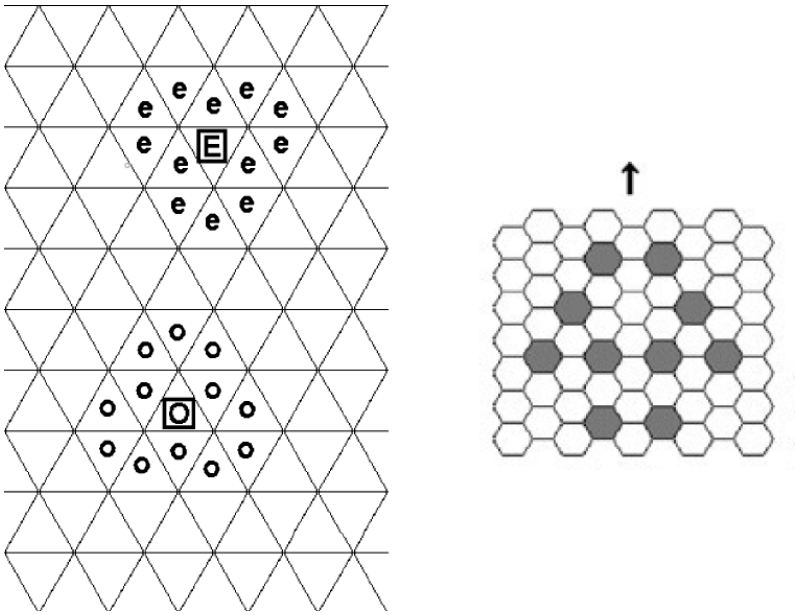


Figure 4.24: The left figure shows two cells of different orientation (E and O) with all their twelve neighbors in the Triangular Game of Life. The right one shows a Glider in the Hexagonal Game of Life[1, Chapter 17.2].

4.4.4 Higher Dimensions

The original Game of Life is a two-dimensional cellular automaton. But there are also three or higher dimensional versions.

In three dimensions, each cell has 26 neighbors [2]. This means, that there are 123201 possibilities for the rules.

The focus lies on the rule (5766)[5], because it is more similar to the original Game of Life than all the other possibilities:

1. Survival rule: Every living cell with five to seven living neighbors will stay alive in the next generation.
2. Death rule: Every living cell with less than five or more than seven living neighbors will die.
3. Birth rule: Every dead cell with exactly six living neighbors will be reborn in the next generation.

When projected to the original Game of Life, there are certain objects that have an equivalent in the two-dimensional version, but that does not work for all of them. Figure 4.26 shows a Glider and a Blinker which, when projected to the original version, form the same objects. But figure 4.27 shows a Still Life which, when projected to the two-dimensional version, can never be a Still Life, no matter to which plane it is projected. It will destroy itself due to overcrowding.

[5]This is the name of the rule. It includes the number of neighbors that are required in the Survival Rule (5 to 7 living neighbors) and the Birth Rule (6 to 6 living neighbors).

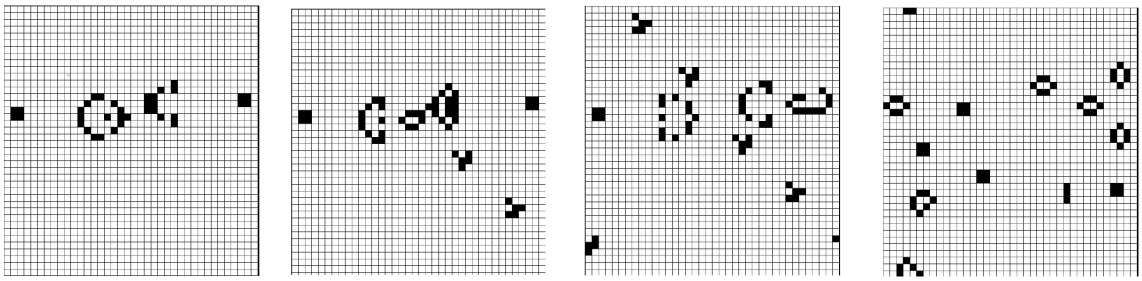


Figure 4.25: Glider Gun on a 40x38 torus: The left picture shows the Glider Gun before it started. The second picture shows the Gun, after two Gliders have been sent out. On the third picture, the Gliders hit the Gun, starting to destroy it. The fourth picture is the final configuration after the destruction of the Gun. It consists entirely of Still Lifes, except one Blinker. [17, Appendix A.5]

There are many objects, similar to objects in the original Game of Life that exist in the three-dimensional version, like Still Lifes, Oscillators or the Glider. But it has been proven that more complex objects like the Glider Gun can not be created [2, Pages 382-383].

There are also versions in four and five dimensions [44]. But these lead to two additional difficulties. They are very hard to compute and can not be visualized as easy as the two and three-dimensional ones. The visualization problem can be solved by projection to a three-dimensional subspace. Figure 4.28 shows two examples of objects in the four and five-dimensional versions.

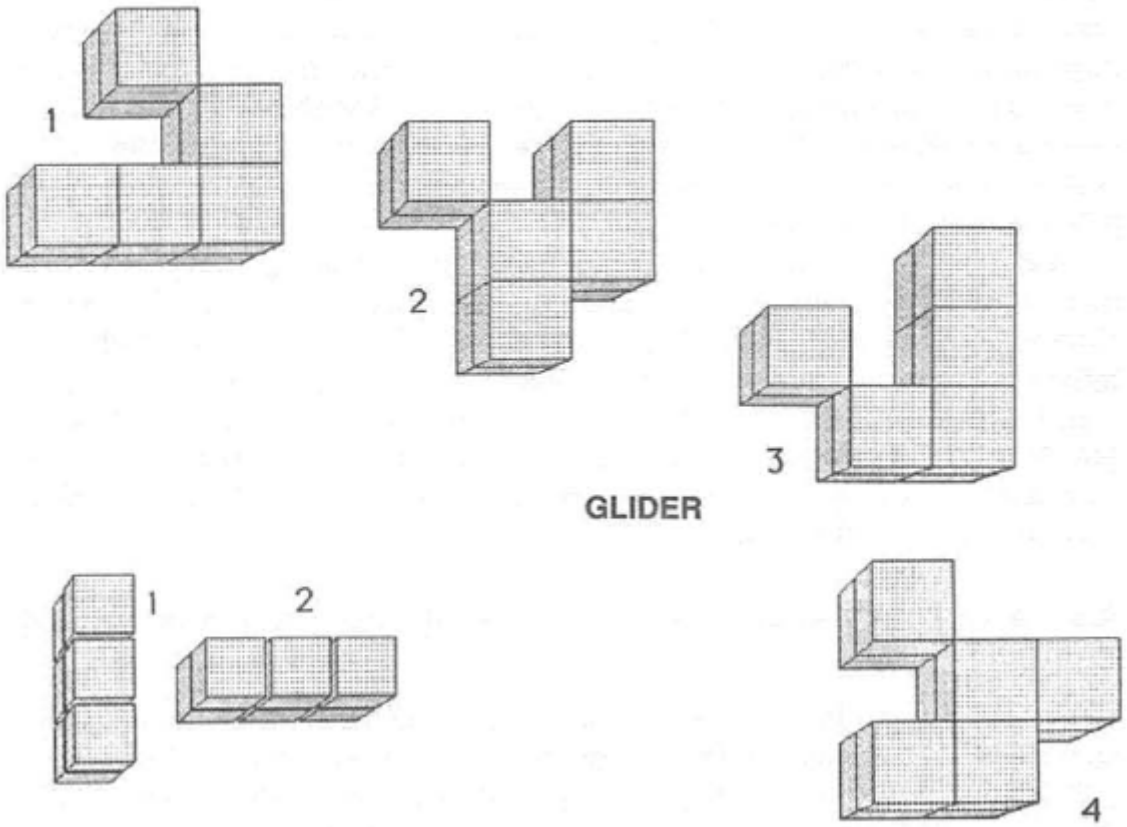


Figure 4.26: On top, there is a Glider in the three-dimensional Game of Life. On the bottom left, there is a Blinker. [2]

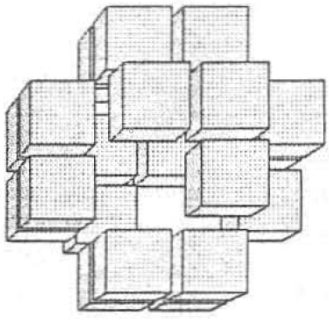
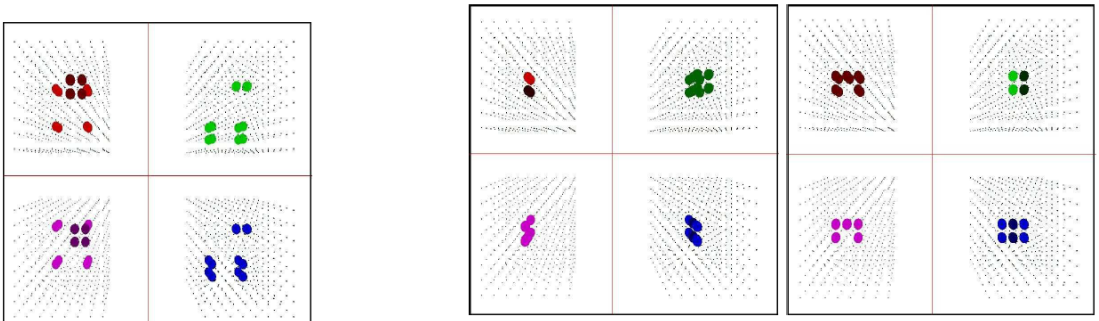


Figure 4.27: This is a Still Life in the three-dimensional Game of Life, that can never be a Still Life, when projected to the two-dimensional version. [2]



(a) State A (b) State B

Figure 4.28: On the left, there is a Still Life in four dimensions. On the right are the two states of a period-2-Oscillator in five dimensions. Both are projected to three-dimensional subspaces. [44]

CHAPTER 5

Fields of Application

The Game of Life comes into use in many areas. Some of these are presented, which are both well investigated and easy to explain. There are others, such as music [1, Pages 489-501] or quantum computing [4] which are either much too complex to explain in a short manner or too obscure and unexplored.

5.1 Science and Teaching

As seen before, there are many scientific works about the Game of Life. And the Game of Life is also used in university teaching. In this section, some additional examples from both areas are presented.

Despite all the other interesting findings, one of the most remarkable discoveries is, that the Game of Life is universal (or Turing-complete). This has been proven by John Conway himself in 1982 (Chapter 4.3.3) [41, Chapter 2.4]. It means, that a Universal Turing Machine can be built in the Game of Life, and thus, that everything that can be computed, can be computed in the Game of Life. A Universal Turing Machine is a Turing Machine that takes another Turing Machine and its initial tape as an input, and produces the same output as the input Turing Machine [41, Chapter 2.2.2]. For his PhD Thesis, John Rendell has built both a Turing Machine and a Universal Turing Machine in the Game of Life (Figure 5.1).

Recently, Mikhail Krechetov used the rules of the Game of Life on graphs with the vertices as cells [22]. This is an interesting view, because it is not fixed, that all cells have the same number of neighbors, depending on how the vertices are connected (Figure 5.2).

Another interesting approach is using neural networks to predict the outcome of n steps of the Game of Life. This has been done by Jacob M. Springer and Garrett T. Kenyon [43]. Unfortunately they discovered, that the training of the convolutional neural networks, they used, often fails, and that the prediction rarely converges. This confirms John Conways proof that the outcome of the Game is unpredicable (Chapter 4.3.1).

The Game of Life is also subject to university teaching in computer science. Melissa Gymrek from the Massachusetts Institute of Technology used it as the main topic of a lec-

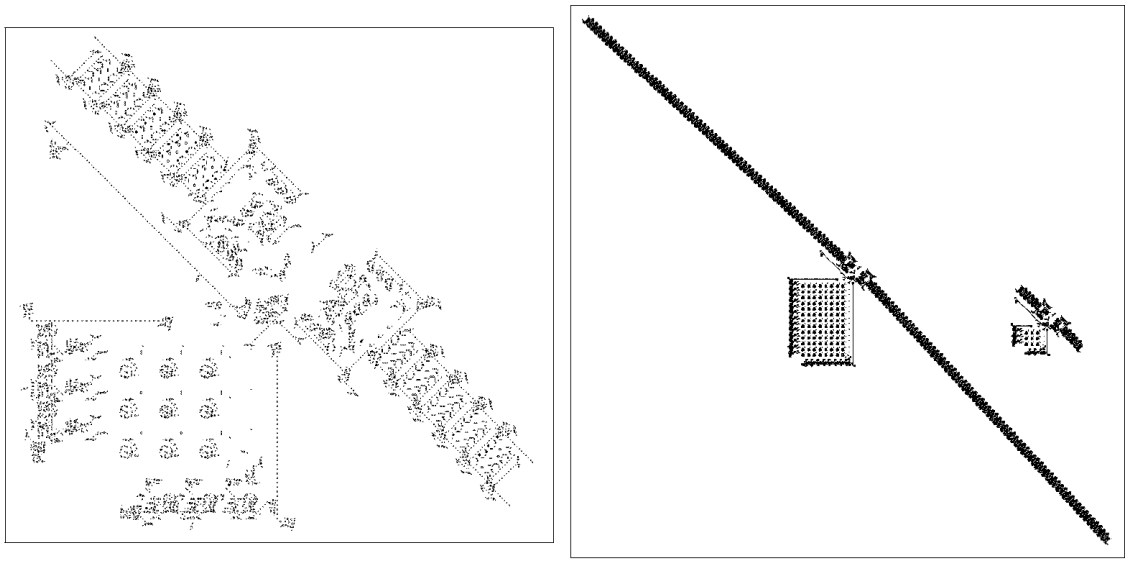


Figure 5.1: The left picture shows a Turing Machine built in the Game of Life. The right picture shows a Universal Turing Machine in the Game of Life, next to a normal Turing Machine, comparing their sizes. [41, Pages 56 and 116]

ture in 2010 [19]. Jarkko Kari from the University of Turrku chose it as an example for cellular automata in his tutorial in 2005 [21, Slides 3-24]. But there is also an example, where the Game of Life is not the main part of a course. Marco Block-Berlitz from Freie Universität Berlin used it in a course about object-oriented programming to explain the use of multidimensional arrays in Java [5, Slides 21-27].

5.2 Cryptography

The Game of Life can also be used as a tool in cryptography.

There is a surprisingly simple way to encrypt text in the Game of Life [20]. Every character from a text has its unique ASCII code. If this is turned to binary, all characters can be represented by seven binary bits. This can be turned to a Glider Stream. When there is a Glider in the Stream, it equals 1. When there is no Glider, it equals 0. Then we need to choose a key for the encryption. It could be a random number. The key is then turned to a Glider Stream as well. If the key is shorter than the original text, it can be repeated until the lengths match. Both Glider Streams are used as an input to an XOR Gate[1]. The output can be turned back to text, and forms the encrypted text. To decrypt it, the procedure is repeated, but with the encrypted Glider Stream and the key Glider Stream as inputs. When the output is converted back to text, it will be the original text.

It is also possible to encrypt grayscale images with the Game of Life [51]. This algorithm is much more complex, so it will be described very briefly. It uses the logistic map to confuse the image. After that, the image is turned into a grid of cells, where each pixel x_i is a cell. If $(x_i \times 10^{14})$ *mod* 3 is bigger than 0, the cell is alive. If not, it is dead. Then R

[1] An XOR gate can be built from the logical gates from Section 4.2.5 by combining a NOT gate, an OR gate and two AND gates as (NOT (A AND B)) AND (A OR B).

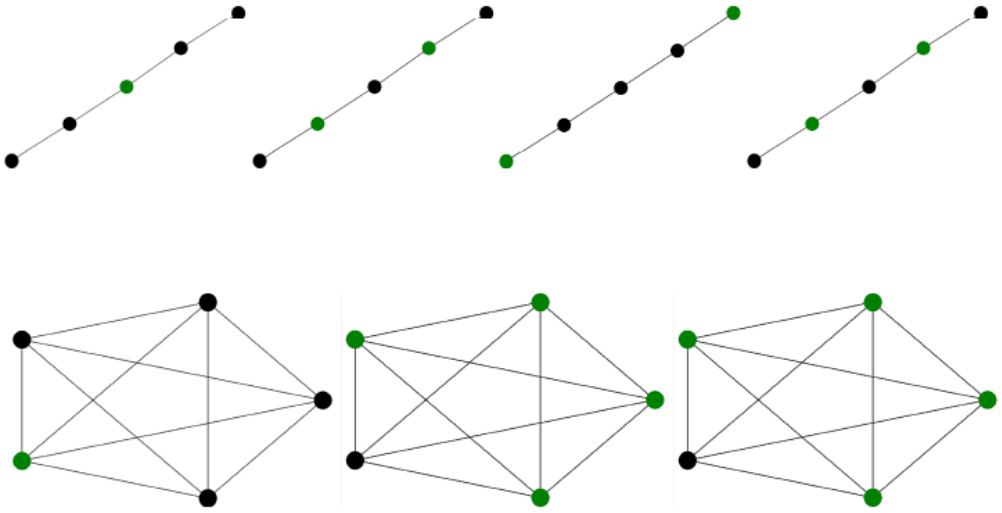


Figure 5.2: The upper figure shows the Game of Life on a line graph. This equals an oscillator, because it repeats after the last step. The lower figure shows another graph, which doesn't change after the second step. So it equals a Still Life. In both figures, dead vertices are black and living vertices are green. [22, Chapter 2.1]

generations of the Game of Life are produced. At the end, the image is diffused by using piecewise linear chaotic map (PWLCM).

5.3 Biology

The Game of Life can also be used as a model in biology.

Peter Turney from Ronin Institute has introduced a simulation of evolving entities which uses the immigration game (Section 4.4.2) [47]. He calls it Model-S (Model of Symbiosis). The simulation starts with a constant population of seeds. Every seed is represented by a binary matrix which represents its genome. Every seed has a special value, called the fitness. The fitness of a seed can be calculated by letting it compete in an immigration game (without the addition of cells every ten generations) against all other seeds in the population. The fitness is then the fraction of wins. So the fitness can be a value between 0 and 1. When the population is set, it starts evolving and reproducing by application of the following layers, from 4 to 1:

1. Layer 1: Uniform Asexual Layer: A parent (one of the seeds) is randomly chosen. Then the bits in the binary matrix are randomly flipped. The newly built seed replaces the least fit seed in the population. Then the fitness values for all seeds are updated.

2. Layer 2: Variable Asexual Layer: This layer is similar to layer 1, but there is a probability, that the newly built seed will grow or shrink by one row or column.

3. Layer 3: Sexual Layer: A parent is randomly chosen. Then it is searched for a mate whose matrix has a certain degree of similarity to the one of the first parent. If no mate is found, the parent moves on to layer 2. If a mate is found, a random horizontal

crossover point is chosen. Then the part above this point of the first parent and the part below this point of the second parent are combined. Then the child is passed on to layer 2.

4. Layer 4: Symbiotic Layer: A parent is randomly chosen. Then there is a certain probability for fusion or fission. If none of that happens, the parent is passed on to layer 3. If fission takes place, the matrix is cut in two at the sparsest point, and one part is discarded. Then the remaining part is passed on to layer 3. If fusion is chosen, a second seed is randomly chosen, both seeds are rotated randomly, then they are joined together with a column of zeros between each other. Then the new seed is passed on to layer 3. There is an option, that the fusion only takes place, if the fitness of the new seed is bigger than the fitness of both individual seeds.

In his recent work, Peter Turney showed, that the fitness of a cell correlates to the diversity of autopoietic structures[2] it creates in the Game of Life [48]. These structures correspond to biological cells. In biology, organisms with many different types of cells have evolutionary advantages against organisms with few. Therefore, two new values are added for each seed: the productivity and the diversity. The productivity is the number of autopoietic structures it generates, and the diversity the number of different structures. Experiments have shown, that the correlation between productivity and fitness as well as diversity and fitness is high and statistically significant.

These results show, that this model is very similar to the behavior of biological life in the following eight aspects:

1. Genomes (the seeds)
2. Phenomes (Layer 1 and 2)
3. Natural selection (the fact that the least fit cell is replaced in every generation)
4. Competition (the way, how the fitness is calculated)
5. Reproduction (Layer 1-3)
6. Mutualism (Layer 4)
7. Cells (the autopoietic structures)
8. Multicellular organisms (the seeds)

[2]Structures that reproduce themselves. It has been proven, that Blocks, Blinkers and Gliders are autopoietic, and it is likely, that all Still Lifes, Oscillators and Spaceships fulfill the definition.

CHAPTER 6

History

One of the most interesting things about the Game of Life is, that from its invention in 1970, there have constantly been new discoveries about it. This chapter gives an overview about its history.

6.1 Early History

The history of the Game of Life started in 1970, when Martin Gardner published the article "MATHEMATICAL GAMES The fantastic combinations of John Horton Conways new solitaire game 'Life'" [16] in the journal "Scientific American". Here he explained the rules of the game, and showed, what John Conway had already discovered. He had already discovered many of the most common structures, such as Still Lifes, Oscillators and Spaceships. It is interesting to note, that back then, he recommended using a large checkerboard or an oriental "go" board to play the Game of Life. He only sometimes used a PDP 7 computer[1] to explore the more complex patterns. He assumed, that there could be no pattern that could grow without limit. But he provided ideas for patterns which could prove him wrong, like the Glider Gun or Puffer Trains. He left it as a challenge for the readers of the journal to find these structures and offered a prize of 50$ for the first one to discover such a structure or prove, that it couldn't exist. This prize was won by William Gosper still in 1970 who discovered the Glider Gun [1, Chapter 1.2].

With growing interest, the mathematician Robert Wainwright started a hand-typed newsletter called "Lifeline" in which he presented new discoveries about the Game of Life [1, Chapter 2]. The invention of the Immigration Game by Don Woods was published here [49]. Wainwright also published some more challenges set by John Conway, including the Grandfather problem[2] [50], which has not been solved until 2016, when a user called mtve from the ConwayLife.com forum presented a pattern that solves it [35]. The newsletter ended in 1973, because with more than 1000 subscribers, it had become too big and time-consuming for the author to sustain it.

[1] A cheap but powerful minicomputer from 1964 [52]

[2] Is there a configuration which has a father but no grandfather?

6.2 Later Achievements

In the following years, many new patterns have been found:

- Another milestone was the discovery of John Conway, that the Game of Life is universal which he proved in 1982 (Chapter 4.3.3)[3, Chapter 25].
- In the late 90s and early 2000s there have been many discoveries in the field of Glider synthesis [29]. This means creating patterns by making Gliders collide. David Buckingham found syntheses for all Still Lifes and Oscillators up to 14 cells [27].
- In 2000, Paul Rendell created the first Turing Machine in the Game of Life [40].
- A very important point was in 2005[3], when the free software Golly was published [46]. This made it very easy for anyone, even without prior experience, to create patterns in the Game of Life and other cellular automata.
- Paul Rendell's Turing Machine was extended in 2010 to a Universal Turing Machine by himself, thus John Conways proof from 1982 was fully implemented [41].
- The first replicator, a pattern that can replicate itself, was found in 2013 by David Greene [31]. He called it Linear propagator.
- In 2018, Adam P. Goucher created a new metapixel, the 0E02 metapixel which can not only simulate the Game of Life and other cellular automata in the Game of Life, but is also a replicator [25]. There is no grid of cells. When a cell dies, it destroys itself, and when a cell is born, it is built completely new by its neighboring cells. With 18,650,000 cells, it is also the largest pattern by number of cells ever discovered.
- But there are still simple patterns that are discovered. Luka Okanishi found the first Oscillator of period 23, David Hilbert[4], in 2019. This pattern won the Pattern of the Year, a competition hosted by the ConwayLife.com community every year since 2010 [32].

Ultimately, in April 2020, John Horton Conway, the inventor of the Game of Life, died at the age of 82 years[45]. But this is not the end of the Game. There are still many exciting patterns to be discovered in the future.

[3]The year can be found in the program itself by clicking on Help and About Golly.
[4]This is actually the name of the pattern, which was named after the mathematician David Hilbert.

CHAPTER 7

Conclusion

As we have seen, the Game of Life has had an eventful history over more than 50 years which is not over, yet. There are still new patterns discovered. It can be used in numerous areas, just like university teaching (to explain object-oriented programming), cryptography (to encrypt text and images) or biology (to simulate biological life). Many fascinating variations such as the Immigration game for two players or higher dimensional versions exist which expand the topic by far. It is still interesting to see, what can be achieved by using these three simple rules.

Patterns of all imaginable sizes, from three to many millions of cells exist. Some populations can grow eternally (Glider Guns), while others keep their size and do not make any changes at all (Still Lifes). On the other hand, there are patterns that return to their initial state every few generations (Oscillators) or disappear completely. Spaceships, such as Gliders that move across the field, make it possible to encode data, which provides the base of the possibility to build a Universal Turing Machine. With the help of metapixels it is even possible, to simulate the Game of Life itself. Despite the simplicity of the rules, it is so hard to predict the outcome of an initial configuration, that even neural networks have difficulties with it.

Among all the other cellular automata, none has been explored as much as the Game of Life, because unlike many other cellular automata it is not limited to one specific purpose. Most likely John Conway had no idea, what an impact his invention would make, when he sent his findings to the Scientific American in 1970. And in the end, it is fair to say, that Martin Gardner was not exaggerating, when he described the combinations of the Game as fantastic.

Bibliography

[1] Andrew Adamatzky, *Game of life cellular automata*, vol. 1, Springer, 2010.

[2] Carter Bays et al., *Candidates for the game of life in three dimensions*, Complex Systems **1** (1987), no. 3, 373–400.

[3] Conway John H. Berlekamp, Elwyn R. and Richard K. Guy, *Winning ways for your mathematical plays*, A K Peters, 2004.

[4] D Bleh, T Calarco, and S Montangero, *Quantum game of life*, EPL (Europhysics Letters) **97** (2012), no. 2, 20012.

[5] Marco Block-Berlitz, *Proinformatik: Objektorientierte programmierung -tag 2*, http://www.inf.fu-berlin.de/lehre/SS09/PI03/docs/tag2.pdf, 2009, Accessed: 28.01.2022.

[6] Robert A Bosch, *Integer programming and conway's game of life*, SIAM review **41** (1999), no. 3, 594–604.

[7] Phillip Bradbury, *Life in life*, https://www.youtube.com/watch?v=xP5-iIeKXE8, 2012, Screenshot at 1:10, Accessed: 13.01.2022.

[8] *Game of life on a klein bottle*, https://breakingmathpodcast.app/gameoflife.html, 2018, Accessed: 16.01.2022.

[9] Monica Dascalu, Mihaela Malita, Adrian Barbilian, Eduard Franti, and Gheorghe M Stefan, *Enhanced cellular automata with autonomous agents for covid-19 pandemic modeling*, Rom. J. Inf. Sci. Technol **23** (2020), S15–S27.

[10] Erya Du, Dianhui Ke Faye Jackson, Trey Smith, Benjamin Riley, Carsten Sprunger, Katie Storey, and Jorn Zimmerling, *Extensions of conway's game of life*, (2018).

[11] Brice Due, *Otcametapixel*, https://otcametapixel.blogspot.com/, 2006, Accessed: 27.01.2022.

[12] Michele Emmer, *Imagine math 2: Between culture and mathematics*, Springer Milan, 2013.

[13] Kellie Michele Evans, *Larger than life: It's so nonlinear*, The University of Wisconsin-Madison, 1996.

[14] Achim Flammenkamp, *Top 100 of game-of-life ash objects*, https://wwwhomes.uni-bielefeld.de/achim/freq_top_life.html, 2004, Accessed: 02.01.2022.

[15] Anahi Gajardo, Andre Moreira, and Eric Goles, *Complexity of langton's ant*, Discrete Applied Mathematics **117** (2002), no. 1-3, 41–50.

[16] Martin Gardner, *Mathematical games: The fantastic combinations of john conway's new solitaire game "life"*, Scientific American **223** (1970), 120–123.

[17] Dan Garry and Andrea Schalk, *Conway's game of life*, (2010).

[18] Felix Gers, Hugo de Garis, and Michael Korkin, *Codi-1bit: A simplified cellular automata based neuron model*, European Conference on Artificial Evolution, Springer, 1997, pp. 315–333.

[19] Melissa Gymrek, *Conway's game of life*, Massachusetts Institute of Technology (2010).

[20] Majid Vafaei Jahan and Faezeh Khosrojerdi, *Text encryption based on glider in the game of life*, International Journal of Information Science **6** (2016), 20–27.

[21] Jarkko Kari, *Cellular automata: Tutorial*, Department of Mathematics, University of Turku, Finland TUCS (Turku Centre for Computer Science) (2005).

[22] Mikhail Krechetov, *Game of life on graphs*, arXiv preprint arXiv:2111.01780 (2021).

[23] Christopher G Langton, *Self-reproduction in cellular automata*, Physica D: Nonlinear Phenomena **10** (1984), no. 1-2, 135–144.

[24] _____, *Studying artificial life with cellular automata*, Physica D: Nonlinear Phenomena **22** (1986), no. 1-3, 120–149.

[25] *0e0p*, https://conwaylife.com/wiki/0E0P_metacell, Accessed: 03.02.2022.

[26] *Blinker puffer 1*, https://www.conwaylife.com/wiki/Blinker_puffer_1, Accessed: 13.01.2022.

[27] *David buckingham*, https://conwaylife.com/wiki/David_Buckingham, Accessed: 03.02.2022.

[28] *Glider gun*, https://conwaylife.com/wiki/Glider_gun, Accessed: 03.03.2022.

[29] *Glider synthesis*, https://conwaylife.com/wiki/Glider_synthesis, Accessed: 03.02.2022.

[30] *Gosper glider gun*, https://conwaylife.com/wiki/Gosper_glider_gun, Accessed: 03.03.2022.

[31] *Linear propagator*, https://conwaylife.com/wiki/Linear_propagator, Accessed: 03.02.2022.

[32] *Pattern of the year*, https://conwaylife.com/wiki/Pattern_of_the_Year, Accessed: 06.02.2022.

[33] *What is the plural of 'still life'?*, https://www.merriam-webster.com/words-at-play/still-life-compound-irregular-plurals, Accessed: 23.12.2021.

[34] Roberto A Monetti and Ezequiel V Albano, *On the emergence of large-scale complex behavior in the dynamics of a society of living individuals: the stochastic game of life*, Journal of theoretical biology **187** (1997), no. 2, 183–194.

[35] mtve, *Re: Thread for your unsure discoveries*, https://conwaylife.com/forums/viewtopic.php?t=&p=30646#p3064
1972, Forum entry from mtve on May, 5th 2016, 7:29 am, Accessed: 03.02.2022.

[36] Kai Nagel and Michael Schreckenberg, *A cellular automaton model for freeway traffic*, Journal de physique I **2** (1992), no. 12, 2221–2229.

[37] John von Neumann, *Theory of self-reproducing automata*, Edited by Arthur W. Burks (1966).

[38] Gabriel Nivasch, *The caterpillar spaceship*, https://www.gabrielnivasch.org/fun/life/caterpillar, 2005, Accessed: 02.01.2022.

[39] Norman H Packard and Stephen Wolfram, *Two-dimensional cellular automata*, Journal of Statistical physics **38** (1985), no. 5, 901–946.

[40] Paul Rendell, *A turing machine in conway's game life*, (2001).

[41] _____, *Turing machine universality of the game of life*, Ph.D. thesis, University of the West of England, 2014.

[42] Jean-Philippe Rennard, *Implementation of logical functions in the game of life*, Collision-based computing, Springer, 2002, pp. 491–512.

[43] Jacob M Springer and Garrett T Kenyon, *It's hard for neural networks to learn the game of life*, arXiv preprint arXiv:2009.01398 (2020).

[44] Abhishek Srivastava, Deepak Karunakaran, Shailesh Kumar, Vikram B Narayana, Veeresh Halakatti, and Shrisha Rao, *Higher dimensional games of life*, Research Gate (2015).

[45] SIGCAS Team, *Life turns fifty: computing history*, Acm Sigcas Computers and Society **49** (2021), no. 3, 10–10.

[46] Andrew Trevorrow and Tom Rokicki, *Golly*, http://golly.sourceforge.net/, 2021, Accessed: 16.01.2022.

[47] Peter D Turney, *Symbiosis promotes fitness improvements in the game of life*, Artificial Life **26** (2020), no. 3, 338–365.

[48] _____, *Evolution of autopoiesis and multicellularity in the game of life*, Artificial Life **27** (2021), no. 1, 26–43.

[49] Robert T. Wainwright, *Lifeline volume 2*, https://www.conwaylife.com/wiki/Lifeline_Volume_2, 1971, Accessed: 13.01.2022.

[50] _____, *Lifeline volume 6*, https://conwaylife.com/wiki/Lifeline_Volume_6, 1972, Accessed: 03.02.2022.

[51] Xingyuan Wang and Canqi Jin, *Image encryption using game of life permutation and pwlcm chaotic system*, Optics Communications **285** (2012), no. 4, 412–417.

[52] *Pdp-7*, https://en.wikipedia.org/wiki/PDP-7, Accessed: 03.02.2022.

[53] Stephen Wolfram, *Cellular automata as models of complexity*, Nature **311** (1984), no. 5985, 419–424.