Cellular Automata in the Triangular Grid

MohammadReza Saadat

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
February 2016
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Cem Tanova
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Nazim Mahmudov
Chair, Department of Mathematics and Computer Science

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Assoc. Prof. Dr. Benedek Nagy
Supervisor

Examining Committee

1. Prof. Dr. Rashad Aliyev

2. Prof. Dr. Rza Bashirov

3. Assoc. Prof. Dr. Benedek Nagy

ABSTRACT

Cellular automata are parallel computing devices working on a discrete timescale. Each cell of a regular grid has a finite number of states and the state in the next time instant depends only on the actual state of the cell itself and the states of its neighbor cells. When every cell could have exactly two states, they can be identified as “live” and “dead” states.

“Game of Life” is a very popular type of cellular automata on the square grid based on the 8 neighborhood of the cells. There are various initial configuration that leads to periodic or growing or moving patterns, etc. Triangular Grid which also called an isometric grid, is a grid generated by tiling the plane regularly with equilateral triangles. In the thesis, life like cellular automata are analyzed in the triangular gird based on 3-neighborhood relation that is the next state of a cell depends only on its actual state and the states of its closest neighbor cells.

There are 2 sets of conditions. The first set is called “Birth” or “B” which shows the number of live cells needed in the neighborhood of a dead cell to make it alive. The second set is called “Stay alive” or “S” which shows the number of live cells needed in the neighborhood of a live cell to keep it alive. Various B/S models are analyzed. While some can generate patterns like snowflake or etc, some can be used for noise removal in image processing.

Keywords: Cellular Automata, Game of Life, Triangular Grid

ÖZ

Hücresel otomatlar ayrık zaman çizelgesi üzerinde çalışan paralel hesaplama araçlarıdır. Düzenli kılavuzun her hücresinin sonlu sayıda ‘durum’ u vardır ve gelecek zamandaki durum ancak hücrenin ve komşu hücrelerin şu anki durumlarına bağlıdır. Hücrenin sadece iki durumu bulunduğunda, bunlar ‘canlı’ ve ‘ölü’ olarak adlandırılırlar.

‘Hayat Oyunu’ kare kılavuz üzerinde, hücrelerin 8 komşuluğuna dayalı çok popüler bir hücresel otomat çeşididir. Dönemli, büyüyen, veya hareket eden kalıpları yaratan değişik başlangıç ayarlamaları bulunur. Ügensel kılavuz, ya da farklı bir adıyla ‘eş’ kılavuz, düzlemi düzenli şekilde eşkenar üçgenlerle döşeyerek elde edilir. Bu tezde 3’lü komşuluk bağıntısına dayalı üçgensel kılavuz yaşam tarzı hücresel otomatlar analiz edilir. Öyle ki hücrenin bir sonraki durumu sadece kendisinin ve en yakınındaki komşu hücrelerin şu anki durumuna bağlıdır.

Toplam 2 sonuç kümesi vardır. İlk kümenin adı ‘Doğum’ ya da kısaca ‘D’ dir ve bu küme ölü hücreyi canlandırmak için komşuluğunda ihtiyaç duyulan canlı hücre sayısını gösterir. İkinci kümenin adı ise ‘Hayatta Kal’ veya kısaca ‘K’ olmakla beraber, bu küme de canlı hücreyi canlı tutmak için gerekli olan komşuluğundaki canlı hücre sayısını temsil eder. Farklı D/K modelleri analiz edilir. Bunların bazıları kartanesi gibi desenler üretirken, bazıları da görüntü işleme alanında gürültü gidermek için kullanılır.

Anahtar Kelimeler: Hücresel Otomatlar, Hayat Oyunu, Ügensel kılavuz

To My Wife

ACKNOWLEDGMENT

I would first like to thank my thesis advisor Assoc. Prof. Dr. Benedek Nagy. The door to Prof. Nagy office was always open whenever I ran into a trouble spot or had a question about my research or writing. His ideas, experiences, and passions have truly inspired and enrich my growth as a student. He steered me in the right the direction whenever he thought I needed it. Without his kind supports it was impossible to finish the thesis.

I would like to thank Prof. Dr. Rza Bashirov and Prof. Dr. Rashad Aliyev for their kind comments and suggestions which encouraged me to continue the research.

Finally, I must express my very profound gratitude to my parents and to my lovely wife Bahareh for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Cellular automatons come in many shapes. One of the most important properties of the cellular automata is the grid type. A one-dimensional line is the simplest "grid". There are square, triangular, pentagonal and hexagonal grids in two dimensions.

Game of Life is one of the most famous cellular automatons which was introduced in 1970 by John Horton Conway. The automaton is designed on a square grid where each cell has only 2 states: dead or alive. Many life liked automatons has been studied since then. In this thesis we are going to study the cellular automaton on the triangular grid and try to find some general characteristic of different possible rules.

In literature review chapter we study the original game of life and its variations. The triangular grid is the base of the thesis and all researches here has been done on this type of grids. The coordinate system which has been used in this thesis is also described to help understating the grid and the meaning of the neighborhood.

In the triangular grid there are different possible neighborhoods. While the last researches have been done on 12 neighborhoods, in this thesis we have studied 3 neighborhood cases. In order to create the figures and find the properties of each case we have designed a computer program using Microsoft Visual Studio platform.

Chapter 2

LITERATURE REVIEW

2.1 Cellular Automaton

A cellular automaton consists of a regular grid of cells with finite number of dimensions. Each cell has a finite number of states, such as “on” and “off” (in case of 2 states it is called binary).

For each cell, there are a set of cells called its neighbors. An initial state (time t = 0) is where each cell has one state (for example “on” or “off”). A new generation (t + 1) is created according to some fixed rules that specify the new state of each cell based on the current state of the cell and its neighbors’ state. The rules are the same for each cell and do not change, and is simultaneously applied to the whole grid, (Wolfram, 1983).

The idea was introduced in the 1940s by Stanislaw Ulam and John von Neumann.

2.2 Conway's Game of Life

The Game of Life, also known just as Life, is a cellular automaton introduced in 1970 by the British mathematician John Horton Conway (Gardner M. , 1970).

The progress of the game is determined by its initial state, requiring no further input so it is a zero-player game. To interacts with this game one creates an initial pattern and observes how it generates the next states.

2.3 Rules

The “Game of Life” runs in a space of unlimited two-dimensional grid of orthogonal square cells. Each cell has two possible states, dead or alive and interacts with its eight neighbors that are adjacent diagonally, vertically, or horizontally.

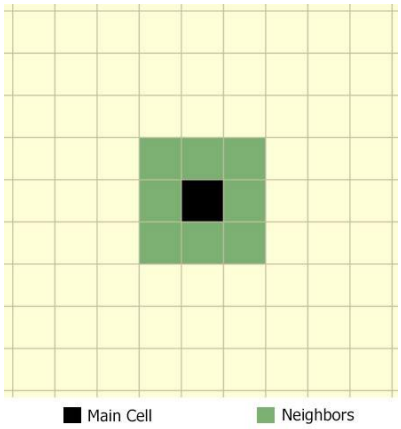


Figure 1: A cell and its neighbors

The game has simple rules which are as follow:

- Birth: a dead cell at time t with exactly 3 live neighbors will be alive at time t + 1.
- Death: a live cell at time t with less than 2 or more than 3 live neighbors will die at time t + 1.

The initial pattern is the system’s seed. Each generation is a pure function of the previous one. The generation t + 1 is created by applying the game’s rules simultaneously to every cell in the generation t, and this discrete moment is also known as a tick. The rules keep on to be applied repetitively to create next generations.

2.4 Patterns

The first interesting patterns in the game were discovered without using of computers.

There are many different types of patterns in the Game of Life include “still lifes”, “spaceships” and “oscillators”. Below are some commonly occurring examples of these three classes. Dead cells are shown in white and live cells are shown in black (Ewert, W, Dembski, W, & Marks, R. J, 2015).

2.4.1 Still Lifes

These are patterns which are stable and do not change.

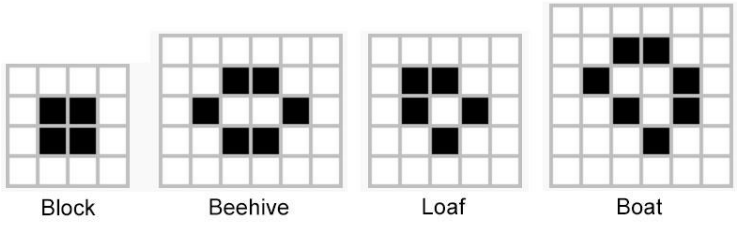


Figure 2: Still lifes patterns

2.4.2 Oscillators

These patterns repeat themselves in a specific number of steps which is called period.

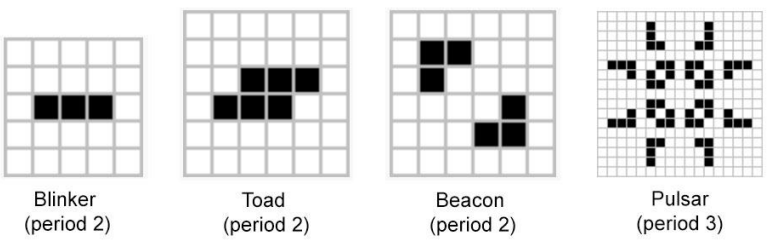


Figure 3: Oscillators patterns

2.4.3 Spaceships

These patterns move themselves on the grid (Rendell, 2002).

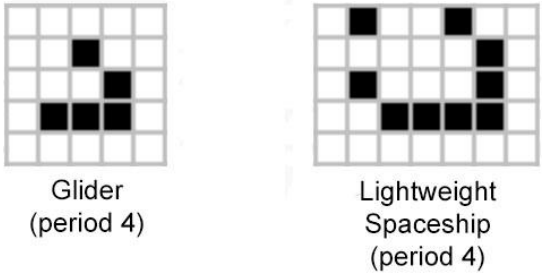


Figure 4: Spaceship patterns

2.5 Variations on the Game of Life

Since Game of Life's beginning, new cellular automata with similar rules have been developed. The standard game is symbolized as "B3/S23": A cell is "Born" from a dead cell if it has exactly 3 live neighbors and it "Stays alive" if 2 or 3 of its neighbors be alive; it dies if not. The first is number or list of numbers that is required for a dead cell to come to live. The second set shows the requirement for a live cell to keep alive on the next generation. For example "B5/S24" means "a dead cell comes to life if there are 5 live neighbors, and a live cell keep alive if there are 2 or 4 live neighbors".

There are also other variations of the game which are run in the grids other than the square grid.

2.5.1 HighLife

It is a variant of the original game with one more rule: a dead cell comes to life if it has 6 live cells. It was introduced in 1994 by Nathan Thompson and is symbolized as “B36/S23”. Most patterns act similarly in both Conway’s Game of Life and

HighLife, HighLife is best known for a pattern called "replicator" , that in fact makes copies of itself (an similar pattern has not been found in the standard Game) (Wong, 2008). After running the replicator for twelve generations, the result is two replicators and they will constantly reproduce themselves, all on a diagonal line.

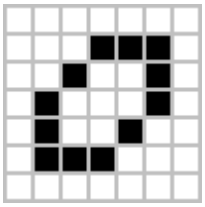


Figure 5: Replicator pattern for HighLife

2.5.2 Pentagonal Life

It is an alternative of Conway's Game of Life on Pentagonal Grid. By Pentagonal Grid we mean tiling of the plane where each piece is in the shape of a pentagon.

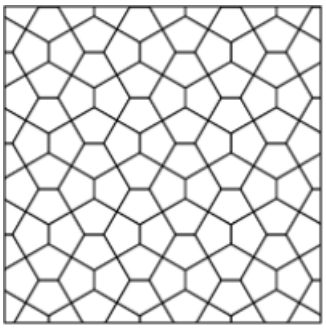


Figure 6: Pentagonal Grid

2.5.3 Hexagonal Life

It is an alternative of Conway's Game of Life on Hexagonal Grid. By Hexagonal Grid we mean tiling of the Euclidean plane, in which at each vertex three hexagons meet (Bays, A note on the Game of Life in hexagonal and pentagonal tessellations., 2005).

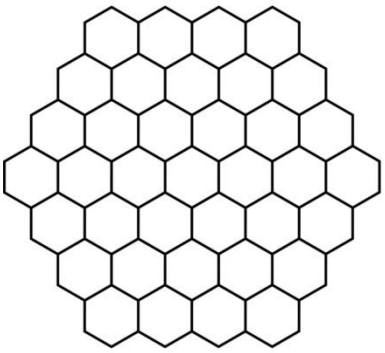


Figure 7: Hexagonal Grid

2.5.1 Triangular Life

It is an alternative of Conway's Game of Life on Triangular Grid. Triangular Grid which also called an isometric grid is a grid generated by tiling the plane regularly with equilateral triangles.

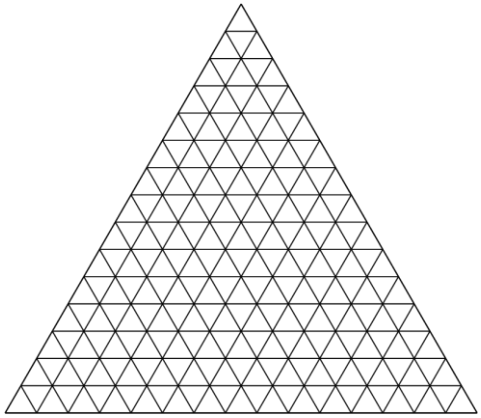


Figure 8: Triangular Grid

Chapter 3

LIFE ON TRIANGULAR GRID

3.1 Triangular Coordinate System

There are different types of coordinate systems which can be used for a triangular grid. In this paper and in order to write the computer program we have used a square like grid with one additional coordinate which has only two possible values “L” and “R”.

In this coordinate system every cell has 3 coordinates, “X”, “Y” and “Z” where $X, Y \in \mathbb{Z}$ and $Z \in \{L, R\}$.

Figure 9 shows the cells (0, 0, L) and (0, 0, R).

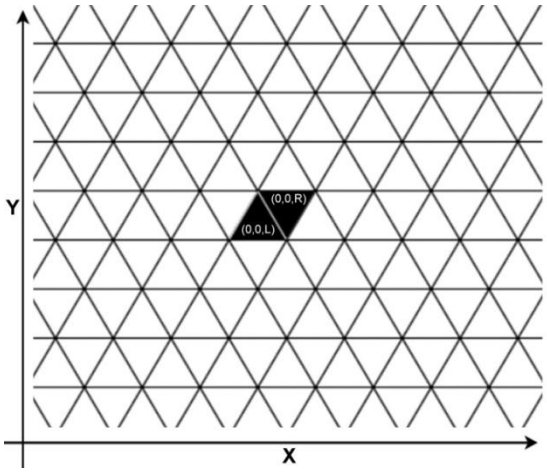


Figure 9: Cells (0, 0, L) and (0, 0, R)

Figure 10 shows the cells where Y = 0.

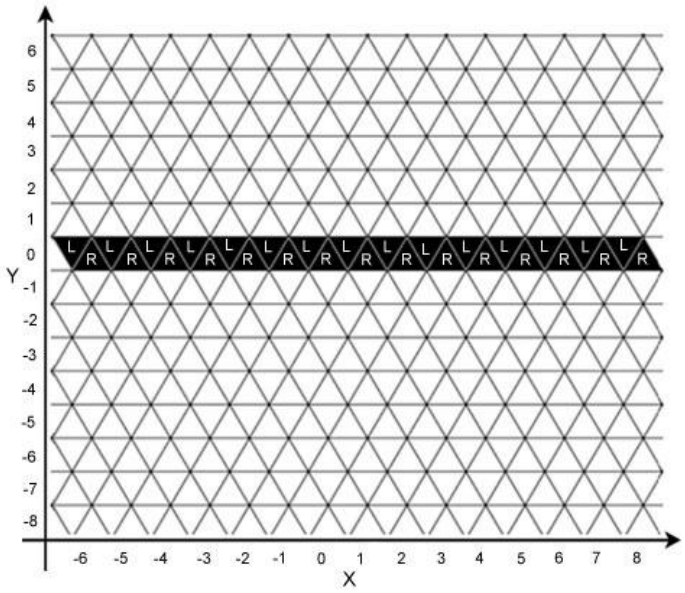


Figure 10: Y = 0

Figure 11 shows the cells where X = 0.

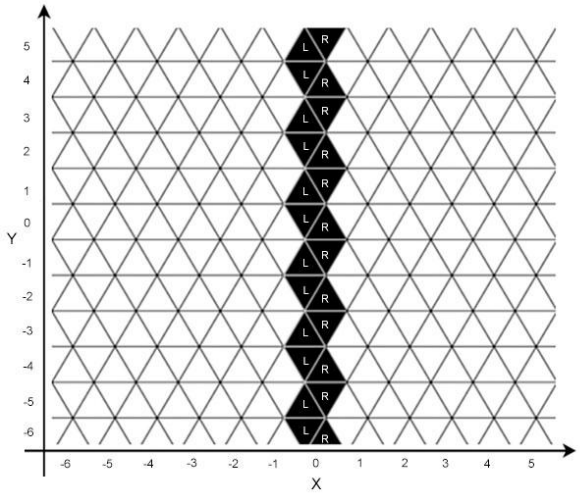


Figure 11: X = 0

3.2 Cell Types in 12 Neighbors Case

There are two types of cells: E and O cells. Each cell has 12 touching neighbors (Bays, Cellular automata in the triangular tessellation., 1994). The neighbors for each type are shown below.

Figure 12: Triangular grid cell types in 12 neighbors case

3.3 Cell Types in 3 Neighbors Case

Just like the 12 neighbors case we have 2 types of cell.

Figure 13: Triangular grid cell types in 3 neighbors case

3.4 Life on Triangular Grid with 3 effective neighbors

While the 12 neighbors case is already studied in the literature we are going to study the game of life in triangular grid with 3 effective neighbors which share an edge with the cell.

3.4.1 Starting State

It is a set of finitely many live cells on the triangular grid. Each cell has 2 possible states: dead or alive.

The black cell means it's alive otherwise the cell is dead.

3.4.2 The Game Rules

Just like the Conway's Game of Life, we have 2 sets of conditions. The first set is called "Birth" or "B" which shows the number of live cells needed in the neighborhood of a dead cell to make it alive. The second set is called "Stay alive" or "S" which shows the number of live cells needed in the neighborhood of a live cell to keep it alive.

Since we have 3 neighbors then it is clear that B and S are a subset of the set M where M = {0, 1, 2, 3}.

For each B and S we have 2^4 - 1 = 15 possible sets. The total possible rules would be B x S which equals to 225.

3.5 Definitions

Bounding Hexagon

It is the smallest Hexagon on the triangular grid which contains all the live cells of the pattern inside of itself. It is similar to the Embedded Hexagon which was defined by Benedek Nagy and Krisztina Barczi (Benedek Nagy, Krisztina Barczi, 2013).

Outer Hexagon

It is the smallest Hexagon on the triangular grid which contains the bounding hexagon inside of itself.

Figure 14 shows the pattern (▲), bounding hexagon (▲) and outer hexagon (▲) for the 3 sample patterns.

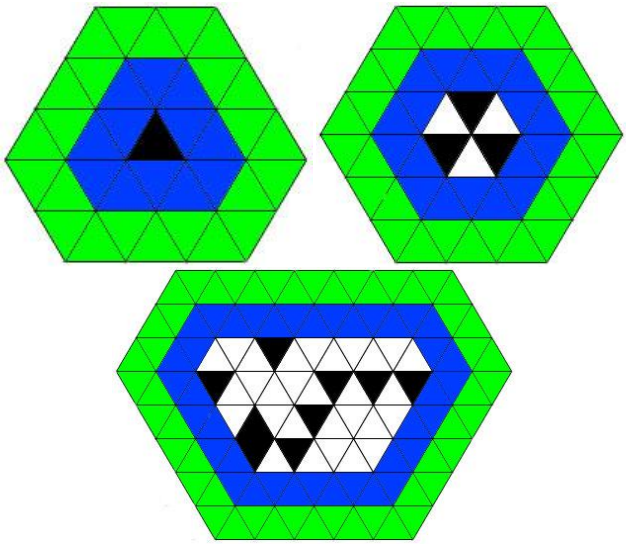


Figure 14: Bounding and Outer Hexagons

3.6 Theorems

Here we try to find some general characteristic of the B/S rules.

Theorem 1

Any B/S rule where $0 \in B$ and $3 \notin S$ leads to alternating states such that every second state is always consist of infinitely many live cells.

Proof:

Based on the definition, starting state is a set of finitely many live cells. If we consider the Outer Hexagon of the pattern, every cell outside of this hexagon is dead. Since the 0 is included on the set B so on the next step all these cells comes to life and we have infinitely many live cells which have 3 live neighbors. For the next generation since 3 is not included on the set S so all these cells would die and then we have infinitely many dead cells. This property is repeated by period 2. □

Theorem 2

Any B/S rule where $1 \in B$ leads to unbounded grow.

Proof:

If we consider the Bounding Hexagon of the pattern, there would be 1 or more boundary cells which share an edge or a vertex with the Bounding Hexagon.

If at least 1 boundary cell shares an edge with the Bounding Hexagon then at the next step since the 1 is included on set B so the neighbor cells comes to life and it's clear that 1 of the neighbors is on the last step Bounding Hexagon which means the new Bounding Hexagon has got bigger.

If none of the boundary cells share an edge with the Bounding Hexagon then at the next step since the 1 is included on set B so the neighbor cells comes to life but all the neighbors are still inside of the last step Bounding Hexagon so the new Bounding Hexagon would be the same size as the last one but in this step we have boundary cells which share an edge with Bounding Hexagon so on the next step the new Bounding Hexagon would be bigger.

The unlimited increase of the Bounding Hexagon means the automata leads to unbounded grow. □

Proposition

In any B/S rule where S = M, each live cell will live forever.

Proof:

It is a trivial consequence of the definition: if the survive condition is not restricted to a limited set but includes all cases then no cell can die. □

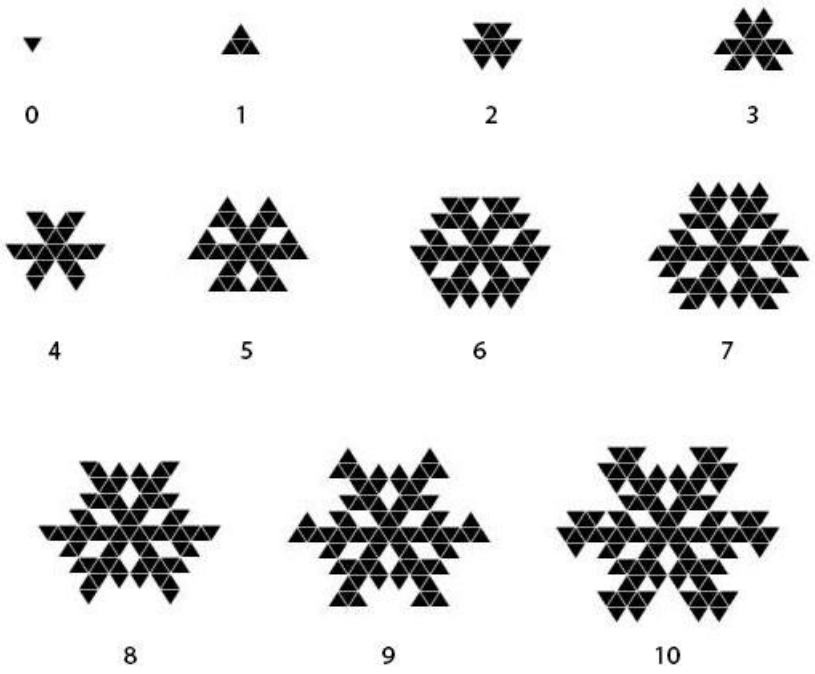
3.7 Case Study

After some general results we are going to study some of these possible rules.

3.7.1 Case "B1/S0123"

In this situation since S equals to M any live cell live forever. There is no stable or periodic shape. It usually makes the shape of snowflakes. It will never stop and it goes to infinity. Regardless of the start cells type ("E" or "O") it generates similar patterns.

Figure 15 presents sample 1-1 where the starting state is a single "O" type cell and its first 21 generations.



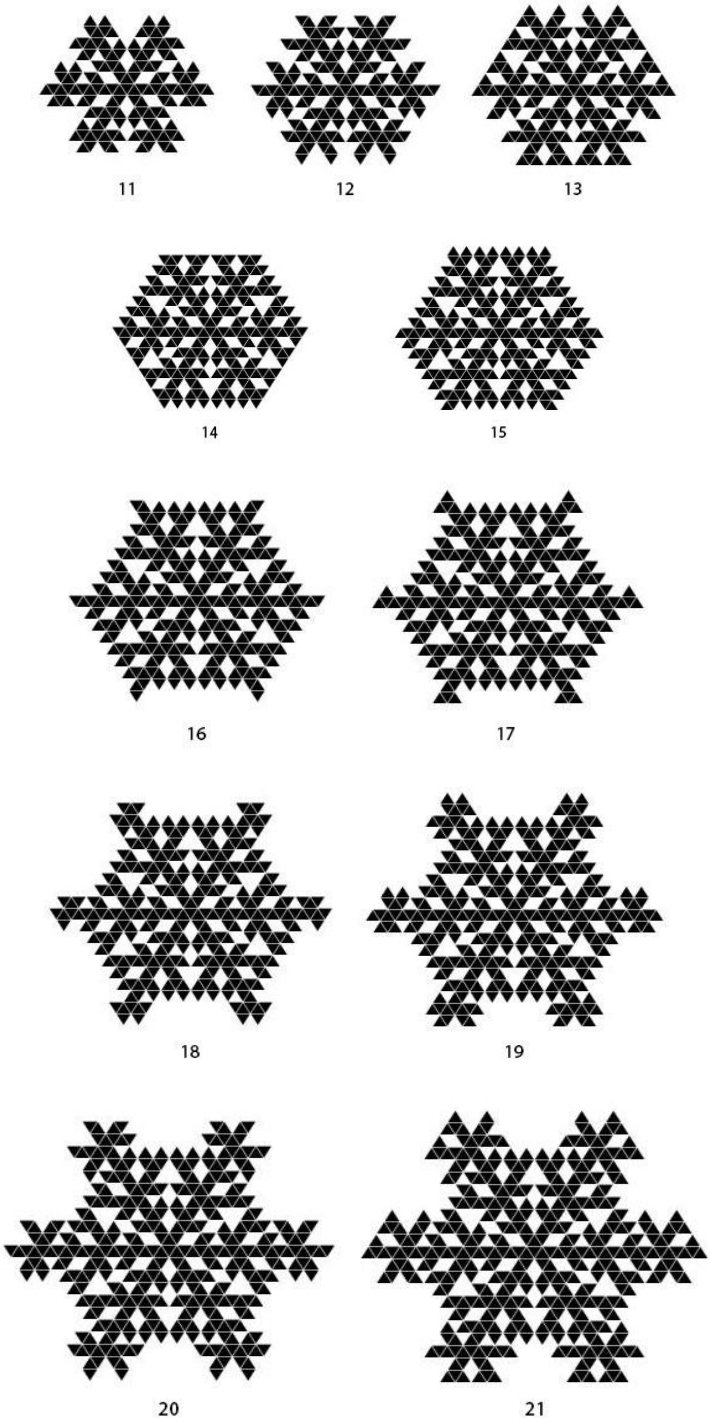


Figure 15: Case “B1/S0123”, sample 1-1

As it can be seen in the figure 15, the pattern generates 6 branch lines. In this sample all branches are the same with thickness of 1 cell.

Figure 16 presents the number of live cells in each step. The chart shows a strictly monotone increasing sequence of live cells as it was expected.

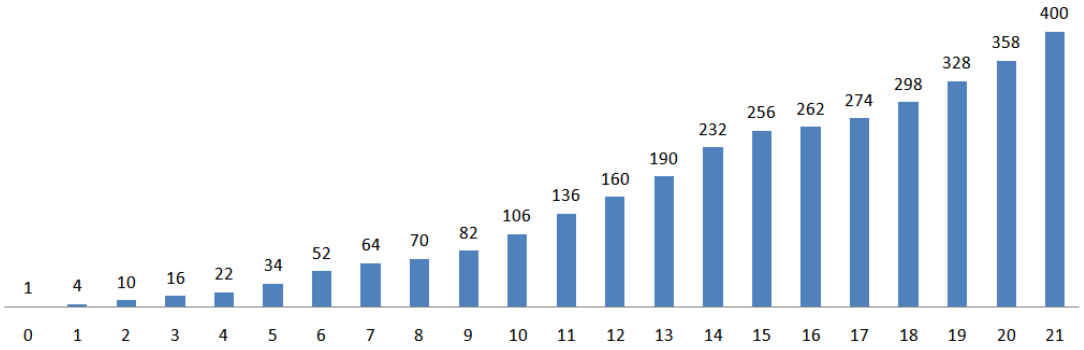


Figure 16: Case “B1/S0123”, sample 1, population stat

Figure 17 presents a real snowflake and 20th generation of sample 1 for the comparison.

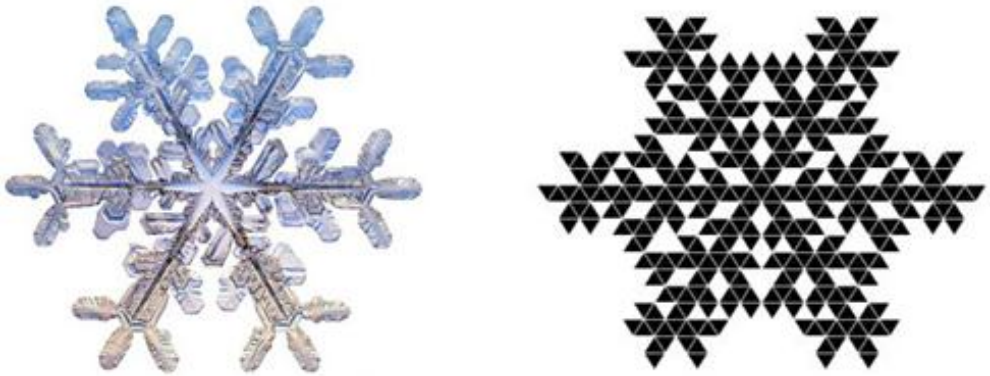


Figure 17: real snowflake comparing generation 20 of sample 1

Next we test the rule for a sample where the starting state is more than 1 single live cell. In sample1- 2 the starting state is consisting of 2 adjacent cells with different types. Figure 18 presents some of its first 21 generations.

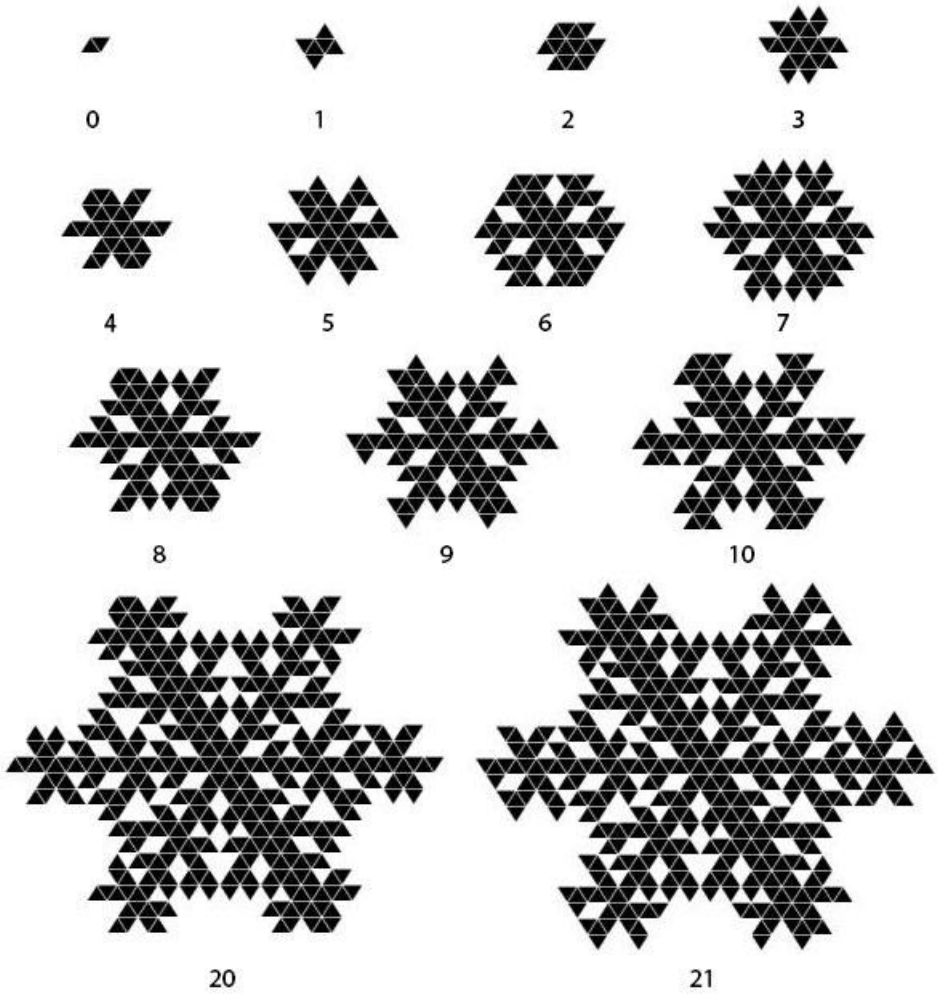


Figure 18: Case “B1/S0123”, sample 1-2

As it can be seen in the Figure 18, the pattern generates 6 branch lines. In this sample we have 4 branch lines with thickness of 1 and 2 branch lines with thickness of 2. Figure 19 presents the number of live cells in each step. The chart shows a strictly monotone increasing sequence of live cells.

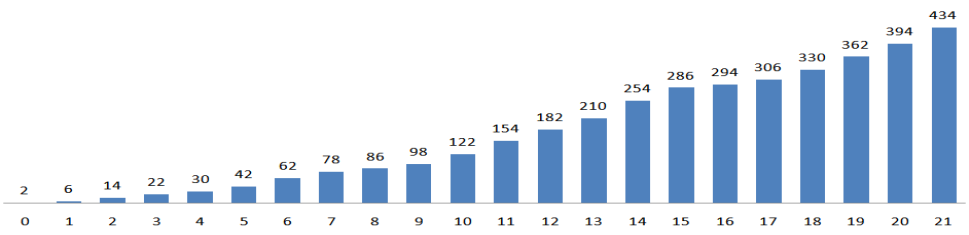


Figure 19: Case “B1/S0123”, sample 2, population stat

In sample 1-3 the starting state is a no isometric pattern. Figure 20 presents some of its first 21 generations.

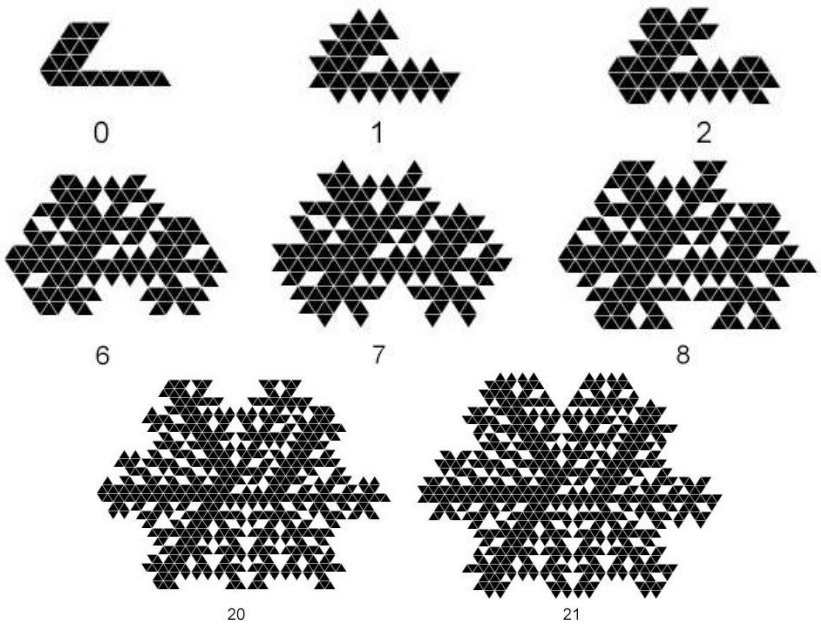


Figure 20: Case “B1/S0123”, sample 1-3

Figure 21 presents the number of live cells in each step. The chart shows a strictly monotone increasing sequence of live cells.

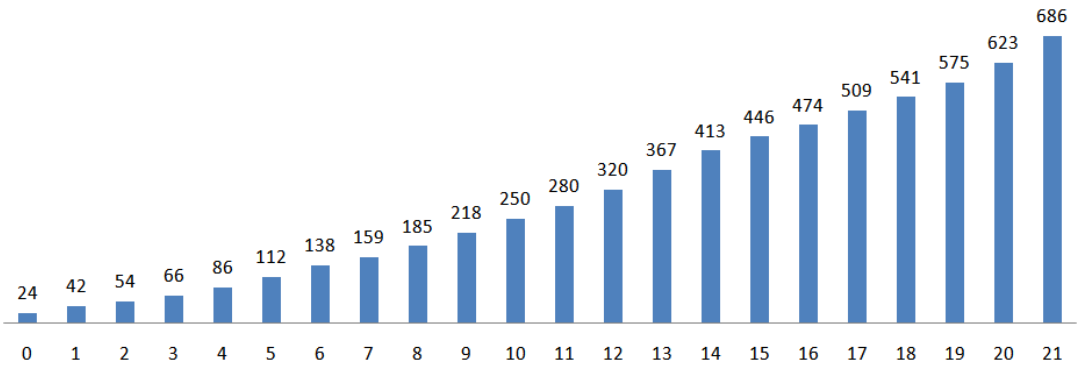


Figure 21: Case “B1/S0123”, sample 3, population stat

In sample 1-4 the starting state is consisting of 2 separate cells with different types. Figure 22 presents some of its first 21 generations.

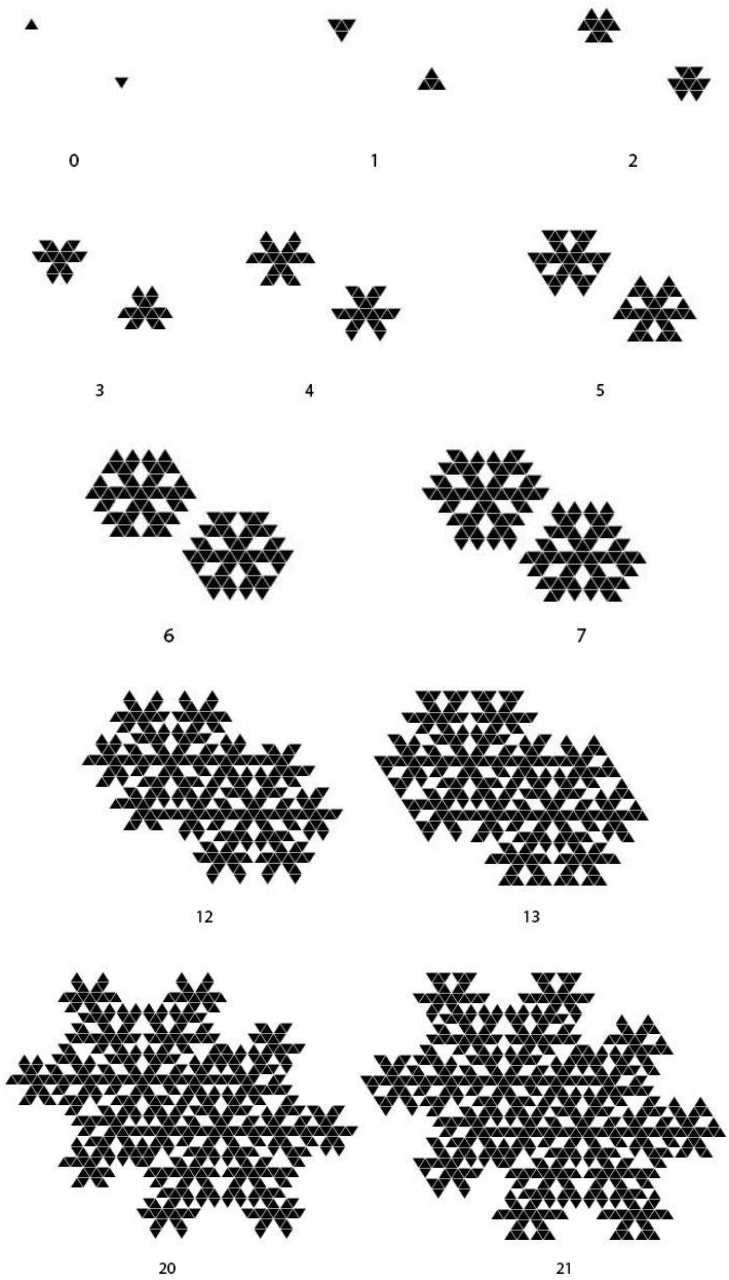


Figure 22: Case “B1/S0123”, sample 1-4

Figure 23 presents the number of live cells in each step. The chart shows a strictly monotone increasing sequence of live cells.

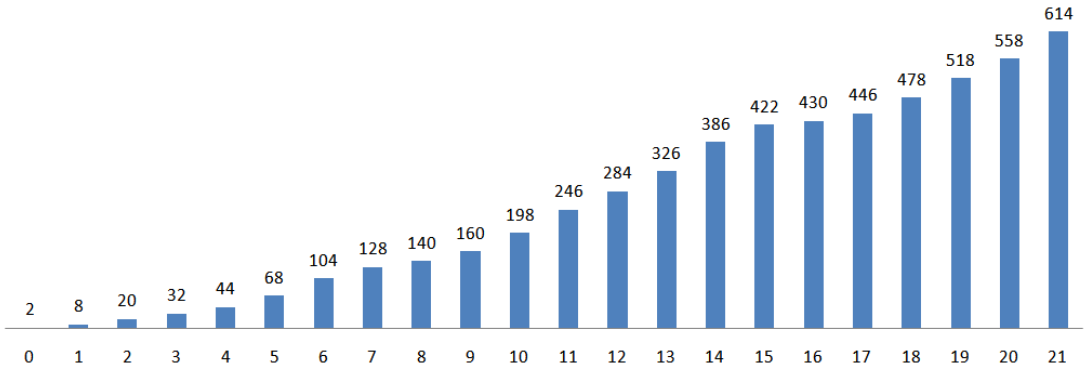


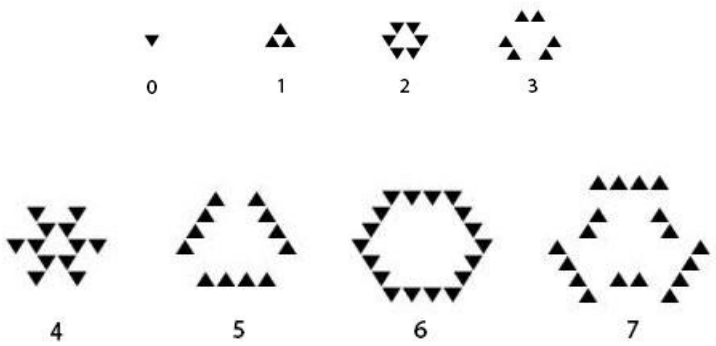
Figure 23: Case “B1/S0123”, sample 4, population stat

3.7.2 Case “B1/S123”

Since the birth condition is 1 so there is no stable shape. Also there is no periodic shape and it goes to infinity. However comparing it to the previous version, S does not includes “0”, thus live cells may not live forever.

It generates different patterns if the starting state includes only one type of cells or both of them.

In sample 2-1 the starting state is consisting of a single “O” type cell. Figure 24 presents its first 21 generations. In each state we have only one type of cells. It starts from “O” type and after that “E” and it repeats again.



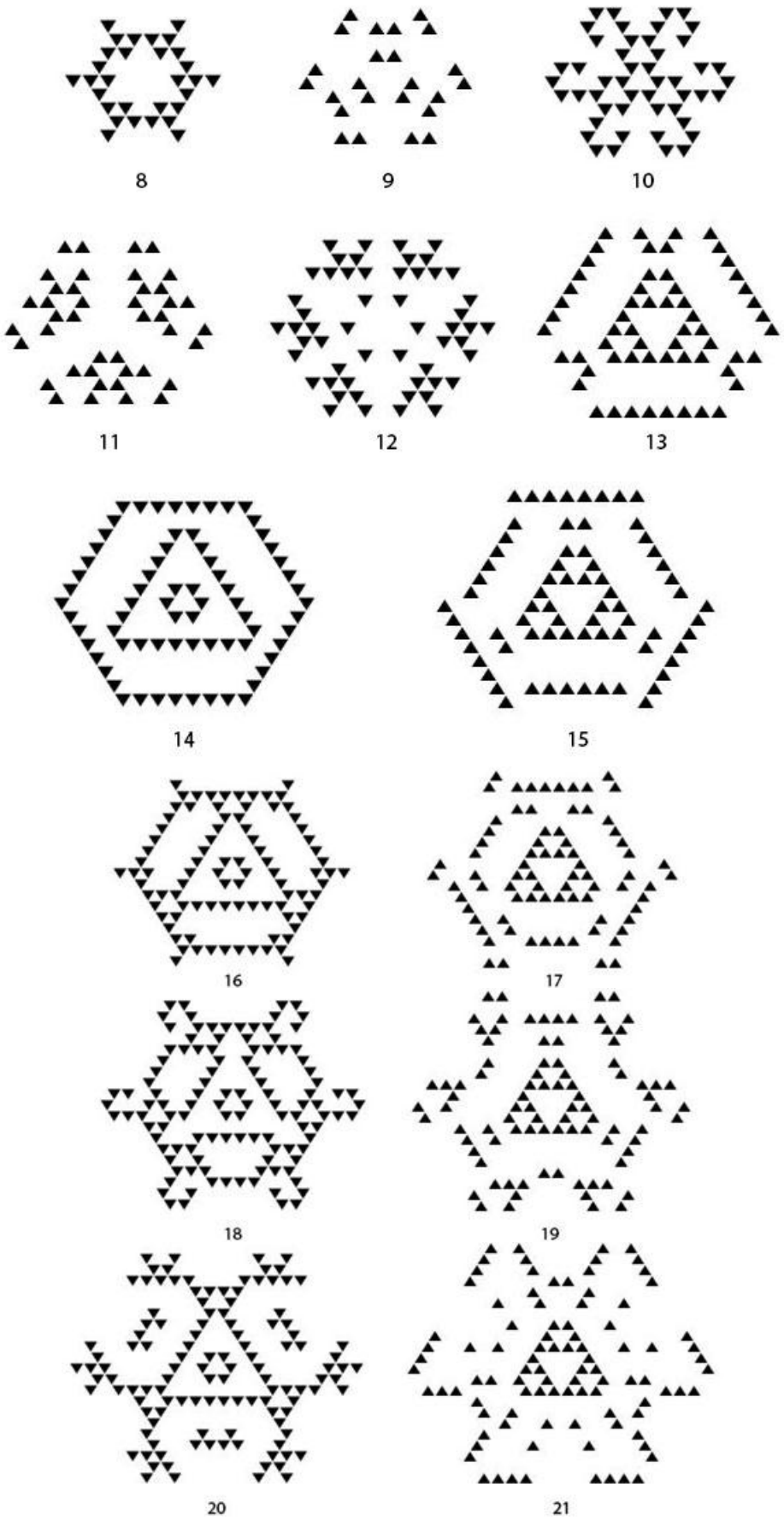


Figure 24: Case “B1/S123”, sample 2-1

Figure 25 presents the number of live cells in each step. The number of live cells on each step can be less, equals to or more than the last steps.

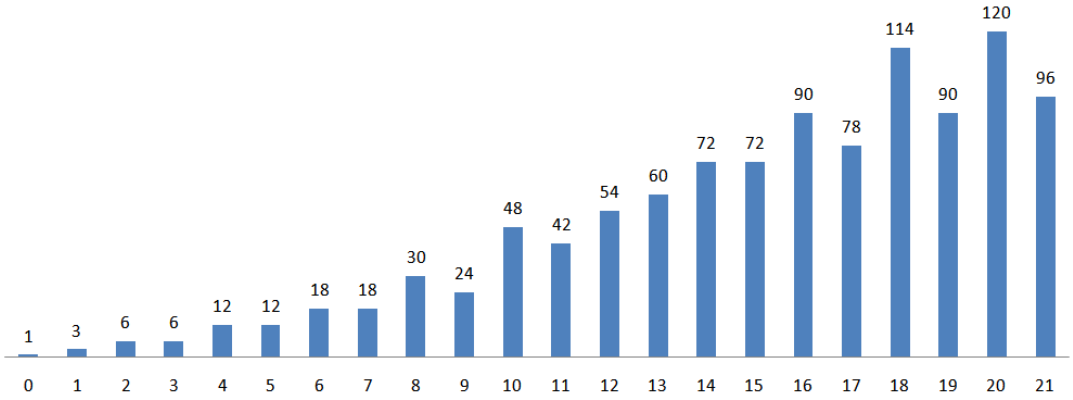


Figure 25: Case “B1/S123”, sample 2-1, population stat

In sample 2-2 the starting state is consisting of 2 adjacent cells with different types. Figure 26 presents some of its first 21 generations.

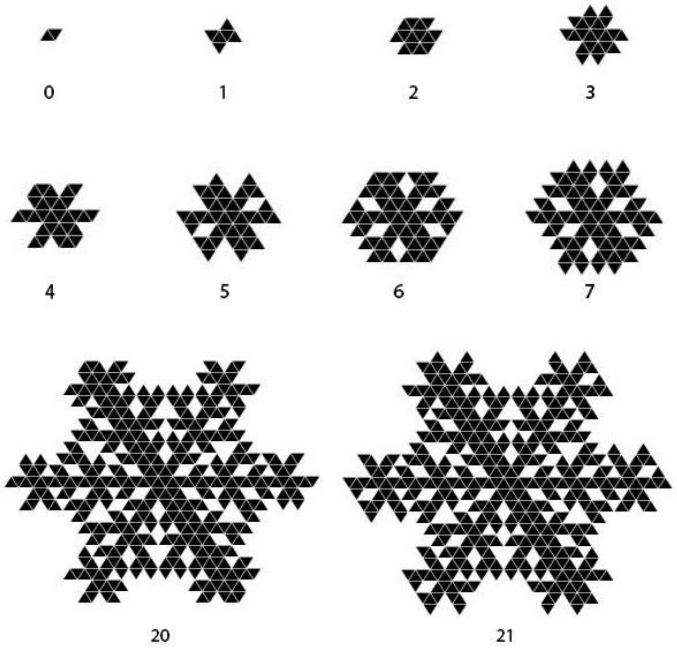


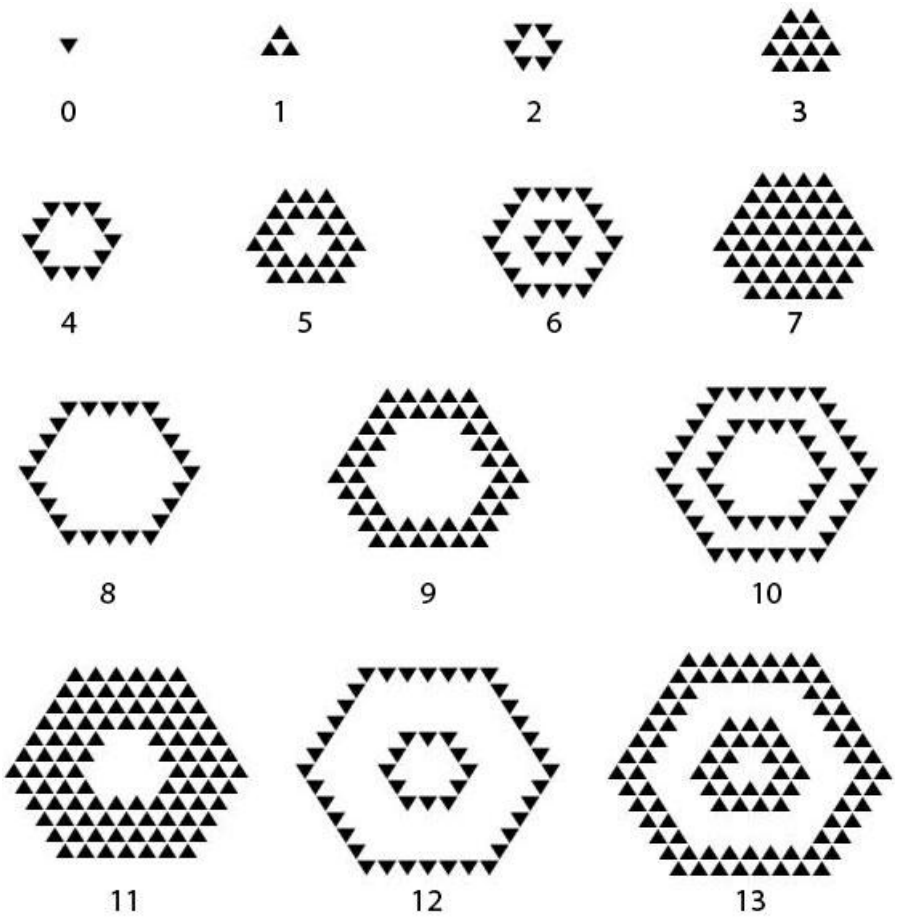
Figure 26: Case “B1/S123”, sample 2-2

Surprisingly, we got back exactly the same snowflake form as the case “B1S/0123” sample 1-2 and any live cell will live forever although the Zero is not included in the set S.

3.7.3 Case “B12/S123”

Since “1” is included in birth condition so there is no stable pattern. Also there is no periodic pattern and it goes to infinity.

If we start with one type of cells the method generates interesting water’s wave patterns. In sample 3-1 the starting state is consisting of a single “O” type cell. Figure 27 presents its first 21 generations.



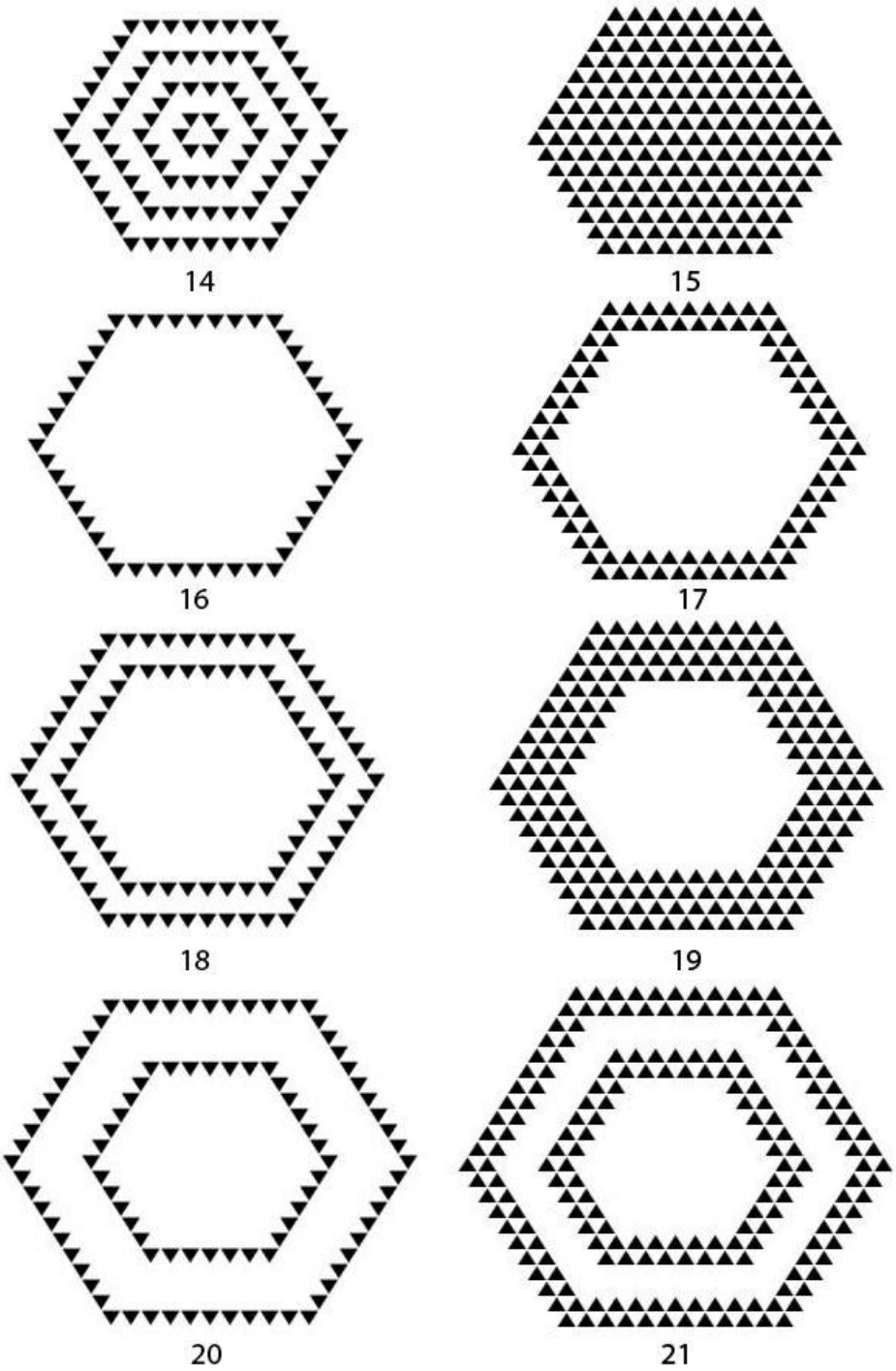


Figure 27: Case “B12/S123”, sample 3-1

In each step we only have one type of cells. Figure 28 shows the number of live cells in each step. The number of live cells on each step can be less, equals to or more than the last steps.

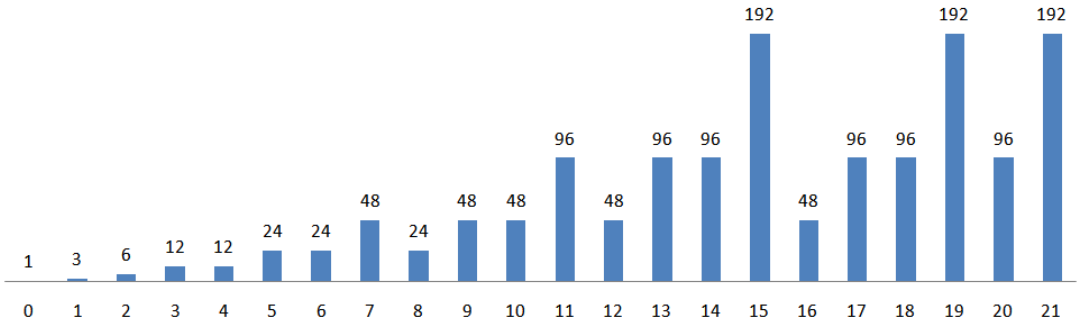
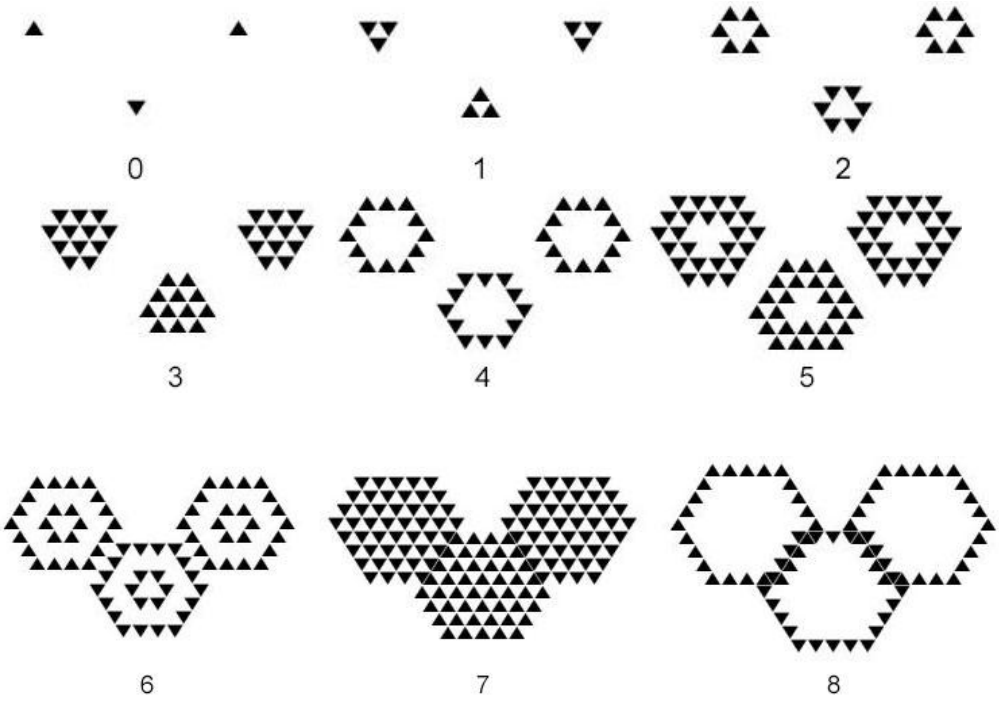


Figure 28: Case “B12/S123”, sample 3-1, population stat

This method has completely different behavior when we have both types of cells in starting state. In such cases it fills the whole space easily and also sometimes we may have finitely many holes.

In sample 3-2 the starting state is consisting of 3 separate cells. There are 2 cells with type “O” and 1 cell with type “E”. Figure 29 presents some of its first 21 generations.



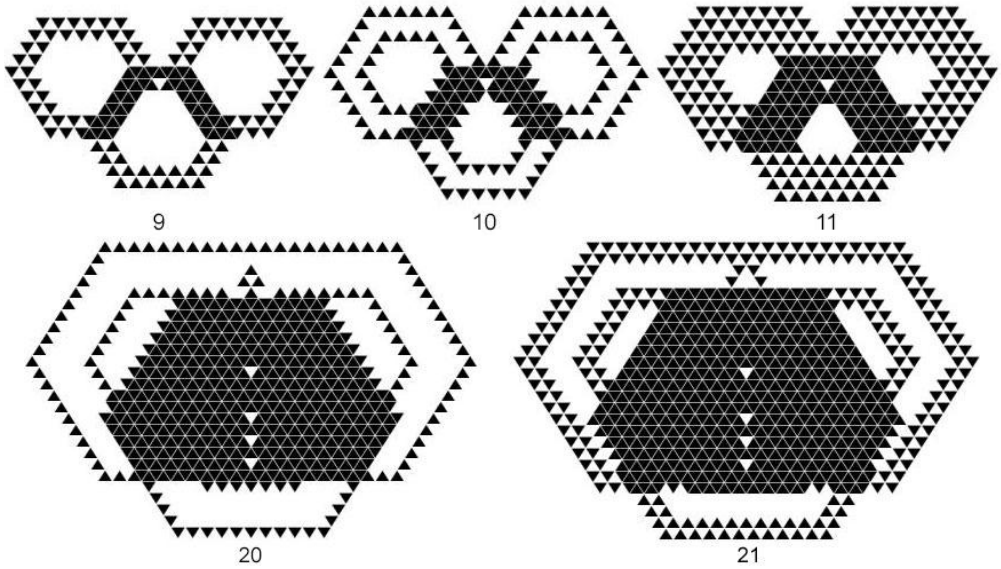


Figure 29: Case “B12/S123”, sample 3-2

Figure 30 shows the number of live cells in each step. The number of live cells on each step can be less, equals to or more than the last steps.

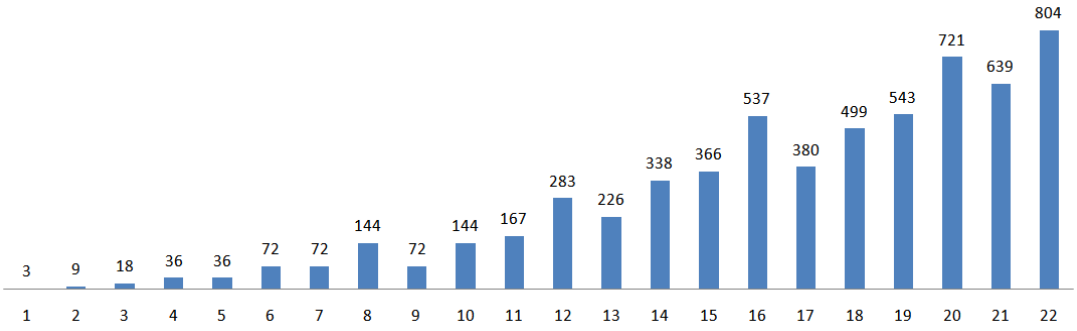


Figure 30: Case “B12/S123”, sample 3-2, population stat

3.7.4 Case “B2/S0123”

In this case any starting pattern will transfer to a stable pattern soon and there is no infinite situation. Also since S equals to M so any live cell live forever.

This pattern is a good tool for filling small spaces between cells. Sample 4-1 transfers to a stable pattern in one generation.

Figure 31: Case “B2/S0123”, sample 4-1

In sample 4-2 we have we have 2 lines with distance 1. They create a stable pattern and automaton stops on 0 generation.

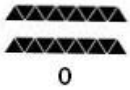


Figure 32: Case “B2/S0123”, sample 4-2

In sample 4-3 we add one single live cell to the left end of the sample 4-2. As it can be seen on the figure 33, the pattern starts to feel the empty spaces between 2 lines step by step and then stops on generation 11.

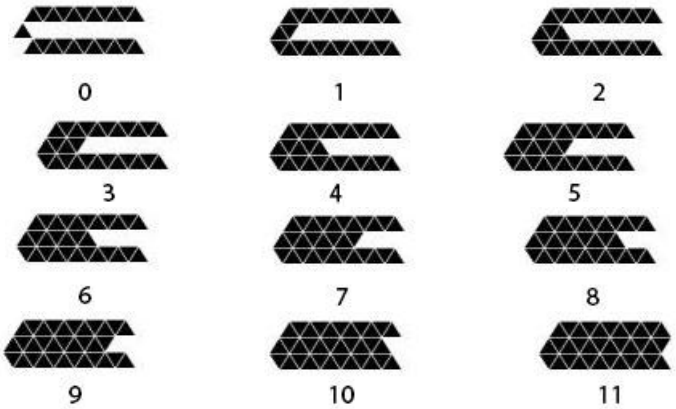


Figure 33: Case “B2/S0123”, sample 4-3

This rule can’t be used to fill the holes with more than 1 cell in width like sample 4-4.

Figure 34: Case “B2/S0123”, sample 4-4

This rule can be used for image recovery. In sample 4-5 we have a set of live cells which should represent the character “Z”. We also have randomly deleted some of the live cells.

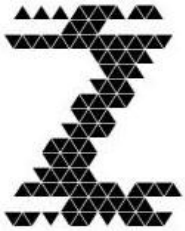


Figure 35: Case “B2/S0123”, sample 4-5

Now we apply the rule. Figure 36 shows the result.

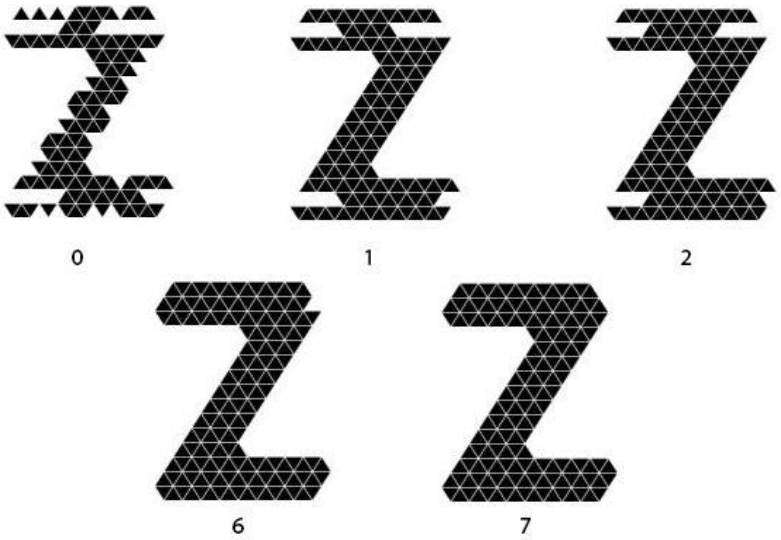


Figure 36: Case “B2/S0123”, sample 4-5

While this method is good at image repairing but it keeps the single live cells on the image which is not good for noise filtering. Below example shows this problem. In sample 4-6 the pattern should represent the word “Hi”.

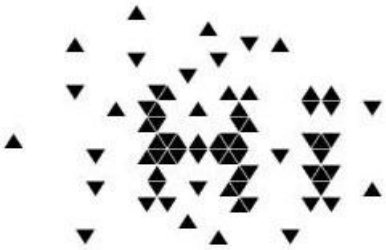


Figure 37: Case “B2/S0123”, sample 4-6

Adding the rule to Figure 37 creates Figure 38 in 1 generation which is a stable pattern.

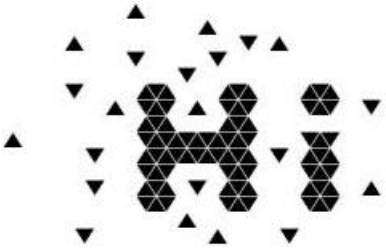


Figure 38: Case “B2/S0123”, sample 4-6

As it can be seen in Figure 38 although the rule has repaired the word but it couldn’t remove the noises around the word.

This rule can’t filter inner noises. By inner noises we mean the dead cells which are surrounded by 3 live cells. Figure 39 shows an example.

Figure 39: Case “B2/S0123”, sample 4-7

In sample 4-8 we test the rule for a noisy hexagon. Figure 40 shows the result.

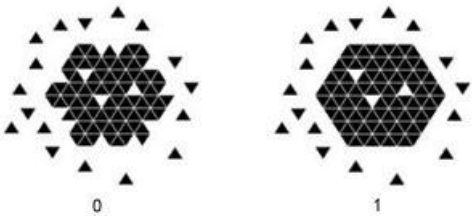


Figure 40: Case “B2/S0123”, sample 4-8

The method repaired the shape but couldn’t remove the inner and outer noises.

3.7.5 Case “B2/S123”

This case is similar to the case “B2/S0123” but since “0” is not included in the “S” so any live cells needs at least one live cells in its neighborhood to keep alive.

With this method we have stable shapes or periodic shapes but no infinite grow. This method gives us an important ability which is “Noise Filtering”.

Sample 5.1 should represent the word “Hi”.

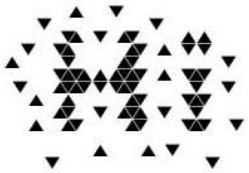


Figure 41: Case “B2/S123”, sample 5-1

Adding the rule to Figure 41 will result to Figure 42 in 1 generation which is stable.

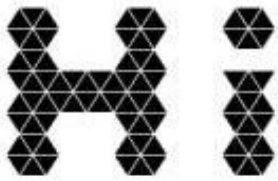


Figure 42: Case “B2/S123”, sample 5-1

As it can be seen in the image the method has removed the noises and also repaired the main shape which is the word “Hi”.

This method can’t filter inner noises. In sample 5-2 we apply the method to a shape which should represent a filled hexagon. Figure 43 shows the staring state.

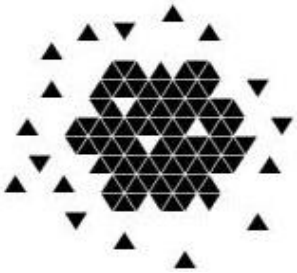


Figure 43: Case “B2/S123”, sample 5-2

We apply the method “B2/S123” to the Figure 43. Figure 44 shows the result.

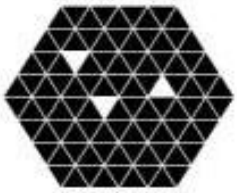


Figure 44: Case “B2/S123”, sample 5-2

The method could repair the shape and also removed the outer noises but still couldn’t fill the inner noises.

3.7.6 Case “B23/S123”

With this method we may have stable shapes or periodic shapes but not infinite grow. This method can be used to repair the shapes, removing the outer noises and filling the inner noises at the same time.

First example is Figure 45 which should represent the shape of a filled hexagon.

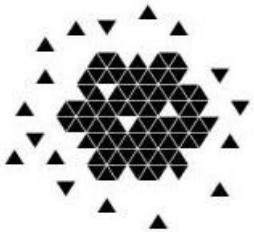


Figure 45: Case “B23/S123”, sample 6-1, Starting State

Now we apply the method. Below is the result.

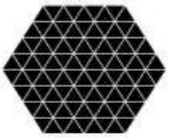


Figure 46: Case “B23/S123”, Sample 6-1, Generation 1

Using this method also has some disadvantages. Figure 47 should represent a horizontal line.

Figure 47: Case “B23/S123”, Sample 6-2, Starting State

Applying the method will completely delete the pattern in 6 generations. Figure 48 shows the steps.

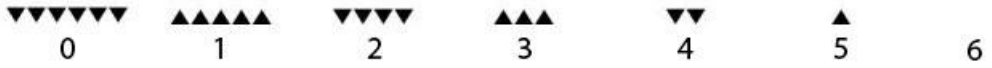


Figure 48: Case “B23/S123”, Sample 6-2

3.8 Patten Recovery

Using combination of two or more methods can help recovering the patterns and filtering the noises. One good way is to apply methods “B2/S0123” first then applying method “B23/S123” to the middle state. By middle state we mean the result of method “B2/S0123”.

Below is an example where we use 2 methods. It should represent the word “Ai”.

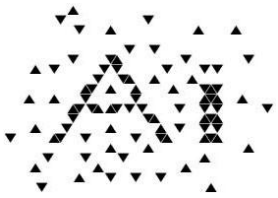


Figure 49: Case “B2/S0123/” + “B23/S123”, Sample 7-1, Starting State

First we apply the method “B2/S0123”. Below is the result in 1 generation.

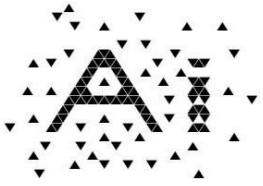


Figure 50: Case “B2/S0123/” + “B23/S123”, Sample 7-1, Middle State

Now we apply the method “B23/S123” to the middle state. Below is the result in 2 generations.

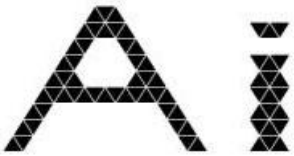


Figure 51: Case “B2/S0123/” + “B23/S123”, Sample 7-1, Final State

If we only use the method “B23/S123” at the starting state, the result would be as follow in 3 generations.

Figure 52: Case “B23/S123”, Sample 7-1, Final State

As can be seen in the Figure 52 only applying the method “B23/S123” removes dot of the character “i”.

For the next example we test the combination on “Eastern Mediterranean University” Logo. Figure 53 shows the original pattern.

Figure 53: Sample 7-2, Original Pattern

In figure 54 we applied some noises to the original pattern and used it as the starting state.

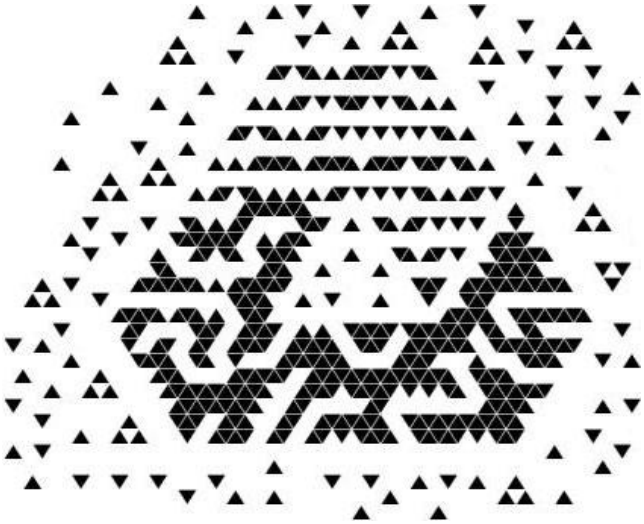


Figure 54: Case “B2/S0123/” + “B23/S123”, Sample 7-2, Starting State

First we apply the method “B2/S0123”. Figure 55 shows the result in 35 generations.

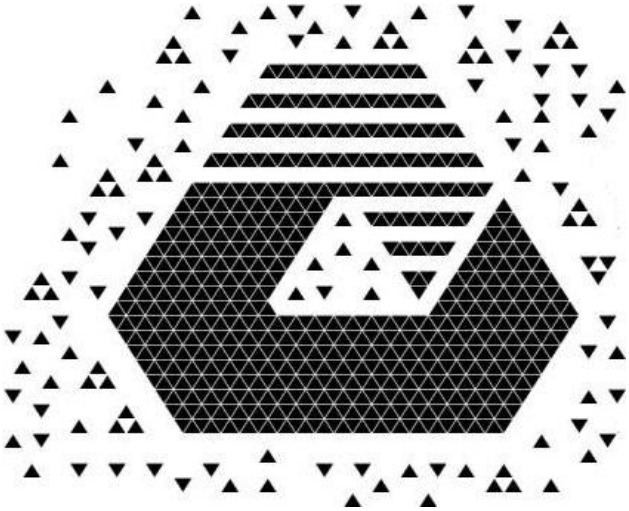


Figure 55: Case “B2/S0123/” + “B23/S123”, Sample 7-2, Middle State

Now we apply the method “B23/S123” to the middle state. Figure 56 shows the result in 2 generations.

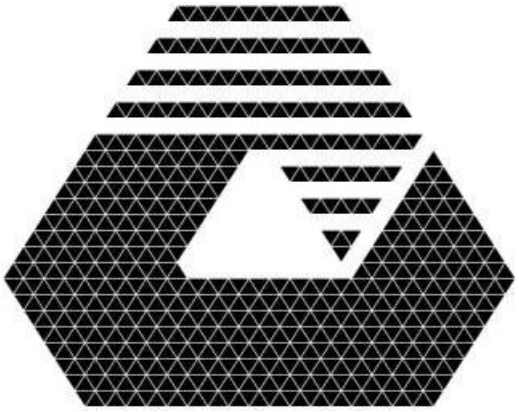


Figure 56: Case “B2/S0123/” + “B23/S123”, Sample 7-2, Final State

Figure 56 is almost the same as the original pattern (figure 53) with only 2 different cells.

Chapter 4

CONCLUSION AND RECOMMENDATION

4.1 Conclusions

While using cellular automaton on a triangular grid with 3 neighborhood may seems to be simple at the first step but it can generate complicated patterns. We studied different B/S rules with simple starting patterns and the automaton generated different patterns which some of them are similar to natural structures like snowflakes or water waves.

The generated patterns are a function of the B/S rule and the starting state. For example the same series of patterns was generated with 2 different B/S rules when using specific starting state.

We also have found that some of the B/S rules or their combinations can be used in image processing. Some of them are good for image recovery and some works well for noise filtering. In order to use this ability the images should be presented on a triangular grid.

4.2 Recommendations

We studied some of the possible B/S rules but there are still many to study. One can try to test the other B/S rules and find their abilities for pattern generating or image processing.

One also may study this automaton on triangular grid using fuzzy logic where the state of each is a number between 0 and 1. In this case the rules should also be updated based on the fuzzy numbers. It would help to using image processing ability of the automaton for RGB pictures.

REFERENCES

Bays, C. (1994). Cellular automata in the triangular tessellation. *Complex Systems*, 8(2), 127.

Bays, C. (2005). A note on the game of life in hexagonal and pentagonal tessellations. *Complex Systems*, 15(3), 245-252.

Ewert, W., Dembski, W., & Marks, R. J. (2015). Algorithmic specified complexity in the Game of Life. Systems, Man, and Cybernetics: Systems, IEEE Transactions on, 45(4), 584-594.

Gardner, M. (1970). Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223(4), 120-123.

Nagy, B., & Barczi, K. (2013). Isoperimetrically optimal polygons in the triangular grid with Jordan-type neighbourhood on the boundary. *International Journal of Computer Mathematics*, 90(8), 1629-1652.

Rendell, P. (2002). Turing universality of the game of life. In collision-based computing (pp. 513-539). Springer London.

Tamás Herendi, Benedek Nagy: Parallel approach of algorithms, Typotex, Budapest, 2014.

Variations on the game of life. (2016, February 12). Retrieved from "http://cs.stanford.edu/people/eroberts/courses/soco/projects/2008-09/modeling-natural-systems/gameOfLife2.html"

Wolfram, S. (1983). Statistical mechanics of cellular automata. Reviews of modern physics, 55(3), 601.