Reporte Semestral

Frías Mercado Carlos Elliot

November 12, 2017

3CM6

Computing Selected Topics

Prof: Genaro Juárez Martínez

Contents

1 Juego de la Vida

1.1 Descripción del programa

Este programa permite simular el algoritmo conocido como Juego de la Vida de Conway. Contiene células las cuales se representan en el tablero a través de un cuadrito de un color a elegir, estando algunas vivas y otras muertas, además se puede controlar la velocidad con la que

cada iteración va cambiando. Se puede iniciar la simulación desde un archivo precargado, dibujada por el usuario o aleatoriamente. El tamaño máximo del tablero (o cantidad máxima de células) es de 1000 y el mínimo es de 10.

1.2 Pruebas del funcionamiento

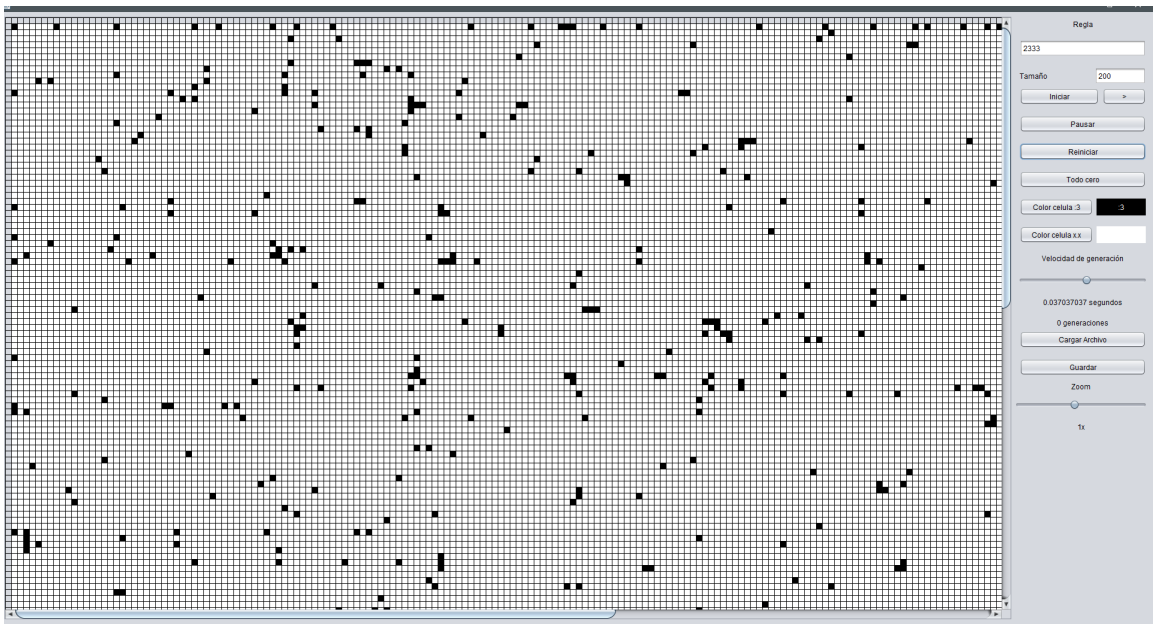


Figura 1: Inicialización del simulador con valores aleatorios

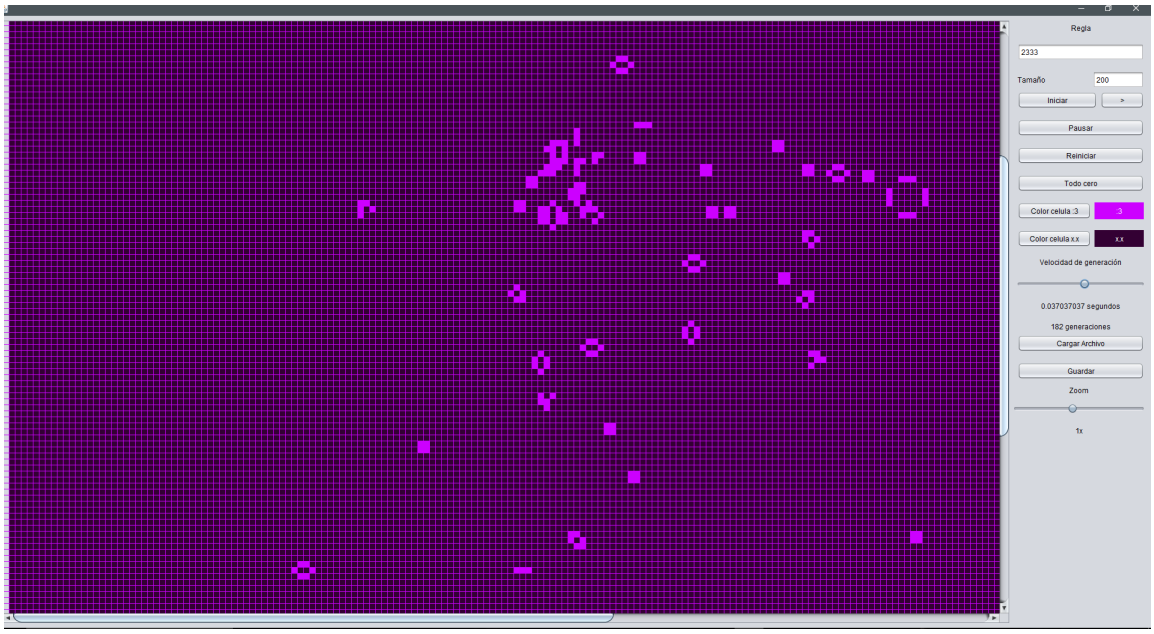


Figura 2: Cambio de colores del programa

Figura 3: Patrón introducido por el usuario

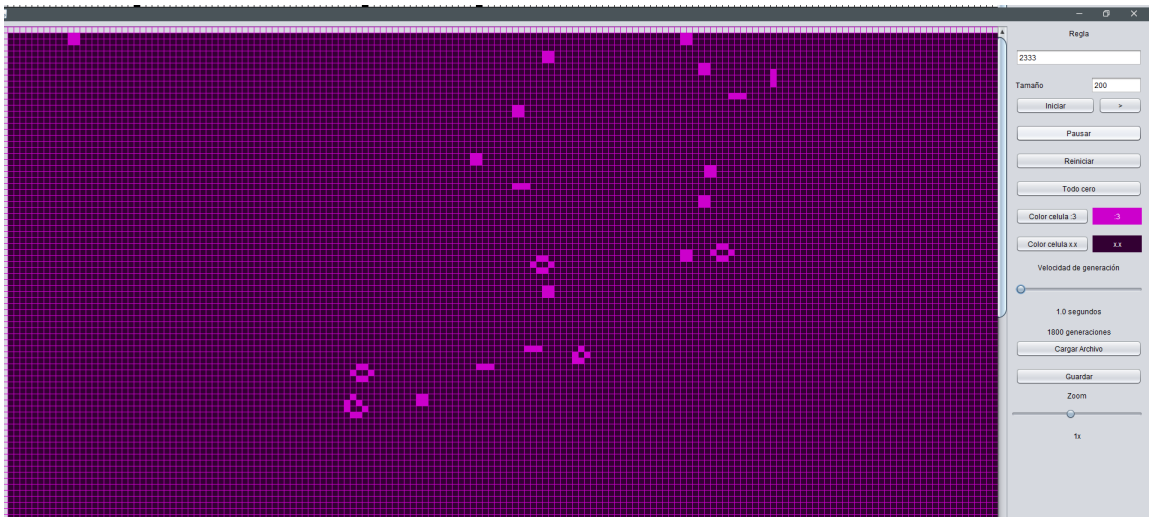


Figura 4: Patrón del usuario tras 1800 Iteraciones

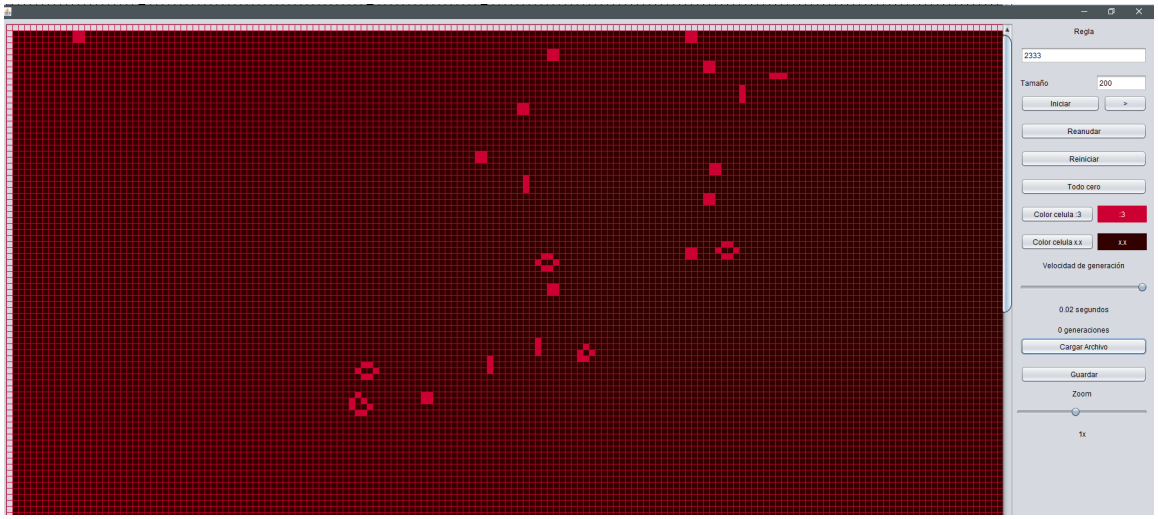


Figura 5: Formato guardado en un archivo y luego recargado

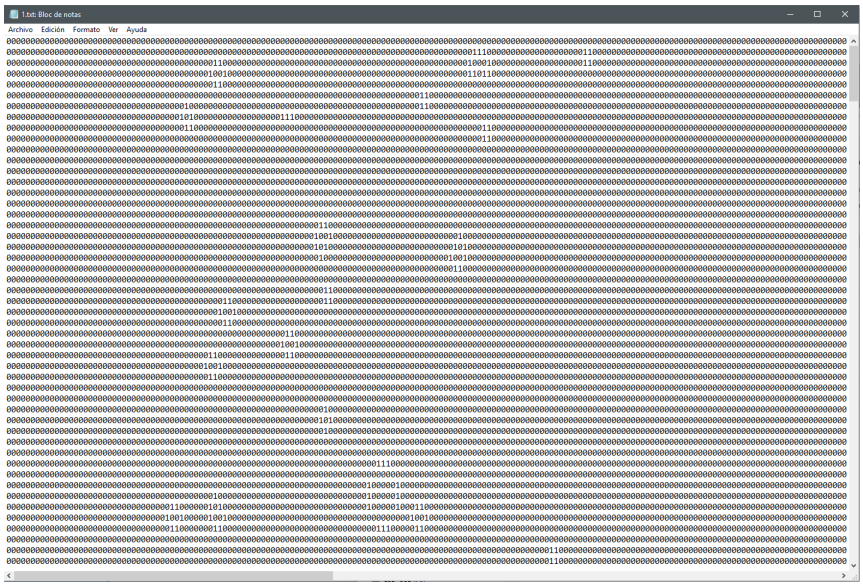


Figura 6: Aspecto del archivo guardado

1.3 Codigo fuente

1.3.1 Clase Universo

```
/*  * To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.  */
package gol;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseWheelEvent;
import java.awt.event.MouseWheelListener;
import javax.swing.JPanel;
import logica.Dios;
import logica.Vida;
/**  *  * @author Maku  */
class Universo extends JPanel  {
 int[][] m;
   int tam, click, tamCelula = 10, tiempo; //tam = num de celulas
```

```
   private Graphics2D g2d;
 Color viva,  muerta;
int[] regla = new int[4];
Dios gg = new Dios();
private int zoom = 0;
private static final double ZOOM_AMOUNT = 1.1;
    Universo ()             {                              }
Universo(int[] regla, int tam, int tiempo, Color vi, Color mu)
{
  this.regla = regla;
     this.tam = tam;
          setPreferredSize(new Dimension(tam*tamCelula,tam*tamCelula)); //x, y
          m = gg.creaVida(tam);
     this.tiempo = tiempo;
 viva = vi;
  muerta = mu;
  this .addMouseListener(new MouseListener()
{
  @Override
                       public void mouseClicked(MouseEvent e)
                         {
                       int x, y;
Point p = e.getPoint();
y = (int) Math.floor((float)p.getX()/tamCelula);
x = (int) Math.floor((float)p.getY()/tamCelula);
if(x < tam && y < tam)
{
if(m[x][y] == 1)
{
m[x][y] = 0;
}
else
{
m[x][y] = 1;
}
click = 1;
repaint();
}
}

@Override
public void mousePressed(MouseEvent e) {
//throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.
}
```

```
@Override
public void mouseReleased(MouseEvent e) {
// throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.
}

@Override
public void mouseEntered(MouseEvent e) {
//  throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.
}

@Override
public void mouseExited(MouseEvent e) {
//  throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.
}                                                    });

Universo(int[] regla, int tam, int tiempo, int cero, Color vi, Color mu)
{
this.regla = regla;
this.tam = tam;
setPreferredSize(new Dimension(tam*tamCelula,tam*tamCelula)); //x, y
m = gg.destruyeVida(tam);
this.tiempo = tiempo;
viva = vi;
muerta = mu;
this .addMouseListener(new MouseListener() {
@Override
public void mouseClicked(MouseEvent e)
{
int x, y;
Point p = e.getPoint();
y = (int) Math.floor((float)p.getX()/tamCelula);
x = (int) Math.floor((float)p.getY()/tamCelula);
 if(x < tam && y < tam)
{
if(m[x][y] == 1)
{
m[x][y] = 0;
}
else
{
m[x][y] = 1;
}
click = 1;
```

```
repaint();
}
}

@Override
public void mousePressed(MouseEvent e) {
//throw new UnsupportedOperationException("Not supported yet."); //To change body of ge

@Override
public void mouseReleased(MouseEvent e) {
// throw new UnsupportedOperationException("Not supported yet."); //To change body of g

@Override
public void mouseEntered(MouseEvent e) {
//  throw new UnsupportedOperationException("Not supported yet."); //To change body of

@Override
public void mouseExited(MouseEvent e) {
throw new UnsupportedOperationException("Not supported yet."); //To change body of gene
}

Universo(int[] regla, int tam, int tiempo, int[][] car, Color vi, Color mu)
{
this.regla = regla;
this.tam = tam-2;

System.out.println(this.tam);
setPreferredSize(new Dimension(tam*tamCelula,tam*tamCelula)); //x, y
m = car;
this.tiempo = tiempo;
viva = vi;
click = 1;
muerta = mu;
this .addMouseListener(new MouseListener() {
@Override
public void mouseClicked(MouseEvent e)
{
int x, y;
Point p = e.getPoint();
y = (int) Math.floor((float)p.getX()/tamCelula);
x = (int) Math.floor((float)p.getY()/tamCelula);
if(x < tam && y < tam)
{
if(m[x][y] == 1)
{
m[x][y] = 0;
```

```
}
else
{

m[x][y] = 1;
}
click = 1;
repaint();
}
}

@Override
public void mousePressed(MouseEvent e) {
//throw new UnsupportedOperationException("Not supported yet."); //To change body of ge

@Override
public void mouseReleased(MouseEvent e) {
// throw new UnsupportedOperationException("Not supported yet."); //To change body of g

@Override
public void mouseEntered(MouseEvent e) {
//  throw new UnsupportedOperationException("Not supported yet."); //To change body of

@Override
public void mouseExited(MouseEvent e) {
//  throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.

});
}
@Override
public void paintComponent(Graphics g)
{             super.paintComponent(g);
if(click == 1)
{
click = 0;
// repaint();
}
else
      {
m = analiza(m, regla);
}
g2d = (Graphics2D) g;
for(int i = 1; i <= tam; i++)
{
for(int j = 1; j <= tam; j++)
```

```
if(m[i][j] == 1)
{
g2d.setColor(viva);
g2d.fillRect(tamCelula*j, tamCelula*i, tamCelula, tamCelula); //x, y, tamx, tamy
g2d.drawRect(tamCelula*j, tamCelula*i, tamCelula, tamCelula); //x, y, tamx, tamy //x, y
}                              else
{
g2d.setColor(muerta);
g2d.fillRect(tamCelula*j, tamCelula*i, tamCelula, tamCelula); //x, y, tamx, tamy
g2d.drawRect(tamCelula*j, tamCelula*i, tamCelula, tamCelula); //x, y, tamx, tamy //x, y
}
}
}                         // repaint();
try
{
//System.out.println("tiempo " +tiempo);
Thread.sleep(tiempo);
 }
catch (Exception e) {}
     } //paint component
public int[][] analiza(int[][]m, int[] regla)
        {
Vida v = new Vida();
//System.out.println("hay"+regla[0]);
return v.existe(m,regla, tam);
}
 public int[][] getMatriz()
{
return m;
}
}//class
```

1.3.2 Clase Dios

```
/*  * To change this license header, choose License Headers in Project Properties.  * T
import java.util.Random;
/**  *  * @author Maku  */
public class Dios
{
public Dios(){}
public int[][] creaVida(int tam)
{
int[][] inicio = new int[tam + 2][tam + 2];
Random r = new Random();
for(int i = 1; i <= tam; i++)
```

```
{
for(int j = 1; j <= tam; j++)
{
inicio[i][j] = r.nextInt(3);

 }
}
return inicio;
        }
public int[][] destruyeVida(int tam)
        {

int[][] fin = new int[tam + 2][tam + 2];
              for(int i = 1; i <= tam; i++)
            {                    for(int j = 1; j <= tam; j++)
              {                        fin[i][j] = 0;
            }
           }
    return fin;
 }
   }
```

1.3.3 Clase Escritor

```
/*  * To change this license header, choose License Headers in Project Properties.  * T
| Templates  * and open the template in the editor.  */ package logica;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import static java.lang.Character.getNumericValue;
/**  *  * @author Maku  */
public class Escritor
{
public  Escritor()
{                      }
public int escribe(int[][] m)
 {
try                     {

PrintWriter writer = new PrintWriter("1.txt", "UTF-8");

for(int i = 0; i < m.length; i++)
```

```
{
for(int j = 0; j < m.length; j++)
 {
writer.print(m[i][j]);
 }
writer.println("");
}
 writer.close();
   return 1;
}
 catch (IOException e)
   {
return 0;

}
}
public int[][] lee(File f) throws FileNotFoundException, IOException
   {
int[][] x;
 int i = 0, j = 0, tam = 0;
String c = "", d;
if(f!=null)
{
FileReader archivos=new FileReader(f);
BufferedReader lee=new BufferedReader(archivos);
//c+= lee.readLine();
while((d = lee.readLine()) !=null)
{
tam++;
c += d;                                    /
/System.out.println("c es "+c);
}
lee.close();
x = new int[tam][tam];
for(int k = 0; k < c.length(); k++)
{
String car = "";
car += c.charAt(k);
System.out.print("["+car+"]");
if(j == tam )
{
System.out.println("");
i++;
j = 0;
k--;
}
```

```
else
{
x[i][j] = Integer.parseInt(car);
j++;
}
}//while2
return x;
}
  //if
  else
                    {
x = new int[10][10];
for(i = 0; i < 10; i++)
{
for(j = 0; j < 10; j++)
{
x[i][j] = 0;
}
}
return x;
}
}//funcion lee
}
```

1.3.4 Clase Vida

```
/*  * To change this license header, choose License Headers in Project Properties.  *
/**  *  * @author Maku  */
   import java.io.*; import java.util.*; public class Vida
{
public Vida()                   {                                    }

public int[][] existe (int[][] m, int[] regla, int tam)
{                                  int fila, colum, vec;
fila = tam;                                 colum = fila;
//Random vida = new Random();
//Integer[][] m = new Integer[fila+2][colum+2];
int[][] aux    = new int[fila+2][colum+2];
//  Integer[][][] saves = new Integer[fila+2][colum+2][10]; //[fila][columna][indice]
//   Integer[][] saves2 = new Integer[fila+2][colum+2]; //[fila][columna][indice]

for(int i = 1; i <= fila; i++) //analizar vida
{
for(int j = 1; j <= colum; j++)
{
```

```
vec = 0;
vec += m[i-1][j-1] + m[i-1][j] + m[i-1][j+1]; //las de arriba
vec += m[i][j-1]       + m[i][j+1]; //las de al lado
vec += m[i+1][j-1] + m[i+1][j] + m[i+1][j+1]; //las de abajo
if(m[i][j] == 1) //[fila][columna]  Si está viva entonces:
{

if(vec > regla[0] && vec < regla[1]) //morir de aislamiento :c  Smin
        {
          aux[i][j] = 0;
        }
else
   {
    }
}
else //si está muerta
{
if(vec  >= regla[2] && vec <= regla[3]) //nacer *3* //Tiene 3 vecinos B
     aux[i][j] = 1;
   }
else
  {
    aux[i][j] = 0;
  }
} //else muerta
}//for columa

} //for fila analizar vida
return aux;
}//main         }//clase
```

2 Analizador

2.1 Descripción del programa

Este programa se encarga de meterse dentro del juego de la vida (únicamente con 10 iteraciones) y trata de encontrar dentro de él los 3 tipos principales de elementos que se pueden formar (Glider, Still Life o Flip Flop). Lo eraliza diviendo todo el universo de células vivas y muertas en pequeñas secciones de 3x3 dentro de las cuales analiza el patrón de comportamiento obtenido al sumar un arreglo tridimensional de cada iteración de células vivas.

2.2 Pruebas del funcionamiento

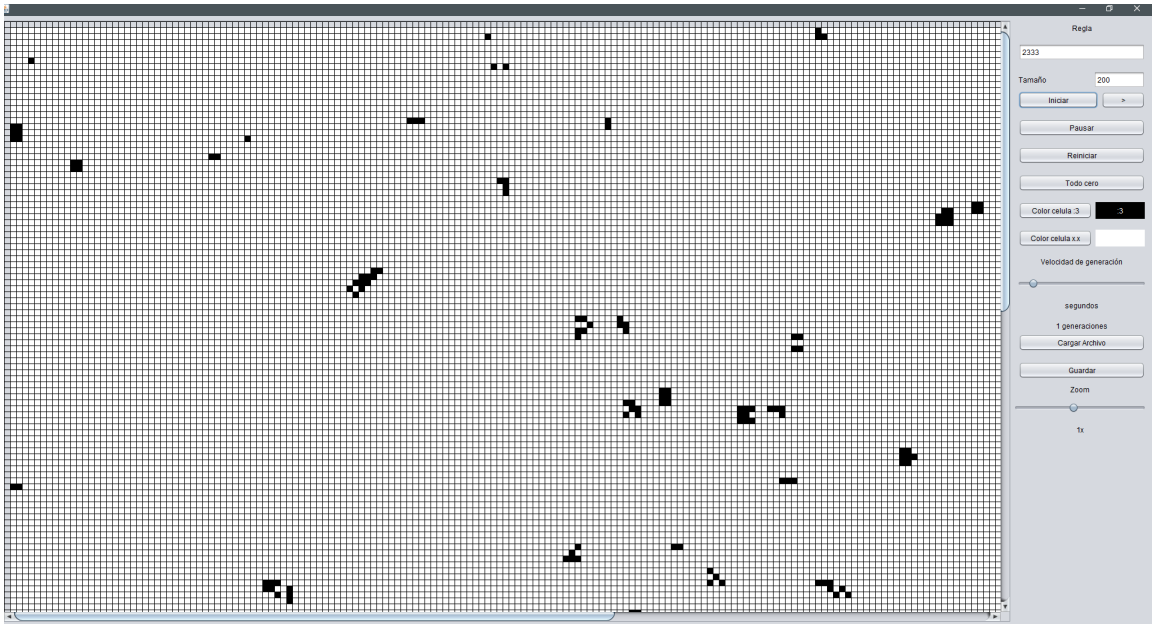


Figura 7: Inicio del simulador y analizador aleatoriamente.

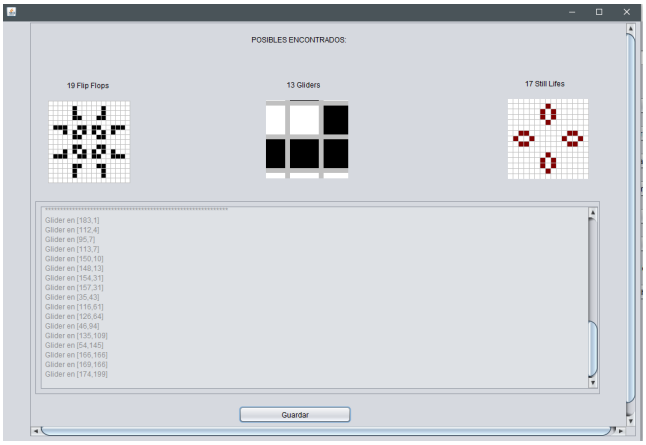


Figura 8: Análisis final generado tras las 10 iteraciones.

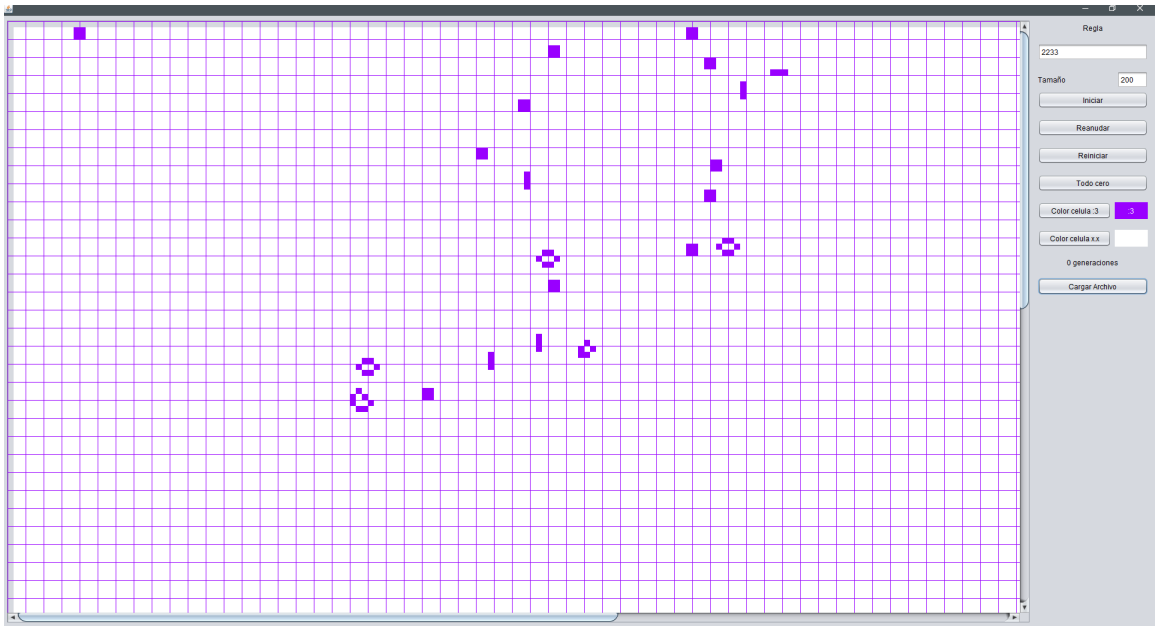


Figura 9: Archivo generado en el programa anterior, cargado dentro de éste y con nuevos colores.

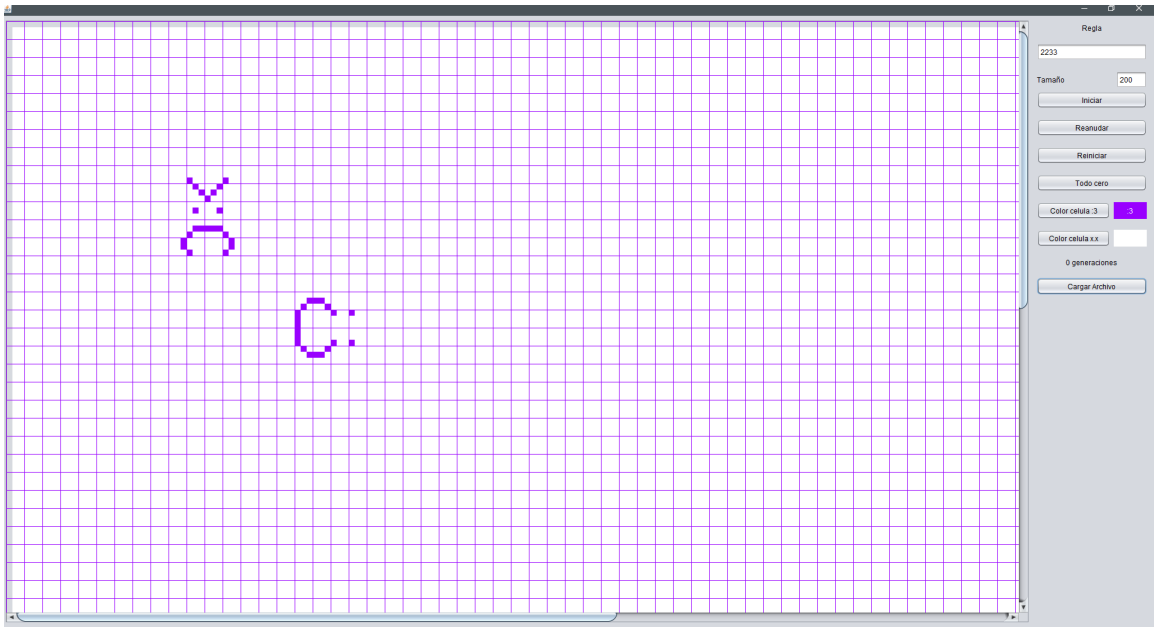


Figura 10: Sistema configurado por el usuario.

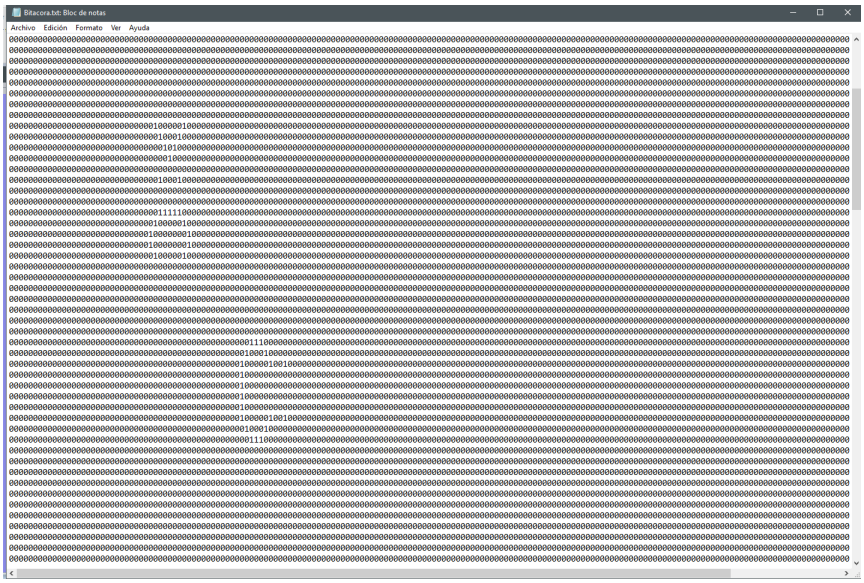


Figura 11: Archivo generado al guardar el analizador.

2.3 Codigo fuente

2.3.1 Clase Escritor

```
/*
* To change this license header, choose License Headers in Project Prope
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/
package logica;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import static java.lang.Character.getNumericValue;
/**  *  *
@author Maku  */
public class Escritor
{
public  Escritor()
{
}
public int escribe(int[][] m, int ff, int gl, int sl)
{
```

```
try
{
PrintWriter writer = new PrintWriter("Bitacora.txt", "UTF-8");
for(int i = 0; i < m.length; i++)
{
for(int j = 0; j < m.length; j++)
{
if(m[i][j] == 1
)
{
writer.print(m[i][j]);
}
else
{
writer.print(0);
}
}
writer.println("");
}
writer.println("Con 10 iteraciones existen");
writer.println("" + ff + " posibles Flip Flops");
writer.println(" " + gl + " posibles Gliders");
writer.println(" " + sl + " posibles Still Lifes");
writer.close();
return 1;
}
catch (IOException e)
{
return 0;
}
}
public int[][] lee(File f) throws FileNotFoundException, IOException
{
int[][] x;
int i = 0, j = 0, tam = 0;
String c = "", d;
if(f!=null)
{
FileReader archivos=new FileReader(f);
BufferedReader lee=new BufferedReader(archivos);
//c+= lee.readLine();
while((d = lee.readLine()) !=null)
{
tam++;
c += d;
//System.out.println("c es "+c);
```

```
}
lee.close();
x = new int[tam][tam];
for(int k = 0; k < c.length(); k++)
{
String car = "";
car += c.charAt(k);
System.out.print("["+car+"]");
if(j == tam )
{
System.out.println("");
i++;
j = 0;
k--;
}
else
{
x[i][j] = Integer.parseInt(car);
j++;
}
}//while2
return x;
}    //if
else
{
x = new int[10][10];
for(i = 0; i < 10; i++)
{
for(j = 0; j < 10; j++)
{
x[i][j] = 0;
}
}
return x;
}
}//funcion lee
}
```

2.3.2 Clase Divisor

```
/* * To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/
package logica;
```

```
import analizador.biatcora;
/** * * @author Maku */
public class Divisor
{
int[][][] saves;
int[][] saves2, ini; int sl = 0, gl = 0, ff = 0;
int[][] mini = new int[3][3];
int i0 = 1, i, j0 = 1, j, fil = 1, col = 1, tamSecc = 3, x = 0, y = 0, tam, gens; //mm,
Analiza an = new Analiza();
public Divisor(int[][][] saves, int tam, int gens, int[][] in)
{
this.saves = saves;
this.tam = tam;
this.gens = gens;
this.ini = in;
saves2 = new int[saves.length][saves.length];
}
public void analisis() {
for(int ii = 1; ii <= tam; ii++)
{
for(int jj = 1; jj <= tam; jj++)
{
saves2[ii][jj] = 0; //inicializo matriz de analisis
}
}
for(int ii = 1; ii <= tam; ii++) {
for(int jj = 1; jj <= tam; jj++) {
for(int kk = 0; kk < gens; kk++) {
saves2[ii][jj] += saves[ii][jj][kk]; //lleno mi matriz de analisis
}
}
}
for(int li = 1; li <= tam; li++)
{ for(int lj = 1; lj <= tam; lj++)
{
System.out.print(saves2[li][lj]);
}
System.out.print("\n"); } //imprimo matriz de analisis
while(j0 <= tam)
{
if(i0 <= tam + 1) {
for(i = i0; i < (fil*tamSecc+1); i++) //dividiendo la matriz
{
if(i > tam) { break; }
for(j = j0 ; j < (col*tamSecc+1); j++)
{ if(j > tam) { break; } //
```

```
System.out.print(i +","+ j +" "+ saves2[i][j]+" ");
mini[x][y] = saves2[i][j];
y++;//columnas } //
System.out.print("|\n");
y = 0;
x++;
if(x > 2) { //
System.out.println("Mini tiene: \n");
/* for(int a = 0; a < 3; a++)
{
for(int b = 0; b < 3; b++)
System.out.print(mini[a][b]);
System.out.print("\n"); }*/
if(an.FF(mini) == 1)
{
ff++;
System.out.println("POSIBLE FLIP FLOP ENCONTRADO\n");
}
if(an.SL(mini) == 1)
{
sl++;
System.out.println("POSIBLE STILL LIFE ENCONTRADO\n");
}
if(an.GL(mini) == 1)
{
gl++;
System.out.println("POSIBLE GLIDER ENCONTRADO\n");
}
x = 0;
}
}
i0 = fil*tamSecc+1;
j0 = 1;
fil++; //
System.out.println("----------------------");
} else { i0 = 1; j0 = col*tamSecc+1; col++; }
}//while analisis
biatcora b = new biatcora(ff, gl, sl, ini);
b.setVisible(true);
}//metodo analisis
}
```

2.3.3 Clase Analiza

```
/*  * To change this license header, choose License Headers in Project P
* To change this template file, choose Tools | Templates
```

```
* and open the template in the editor.  */
package logica;
/**  *  * @author Maku  */
public class Analiza
{
public Analiza()
{
}
public int FF(int[][] matriz)
{
int[] FFC = {0, 0, 0}; //{valor, i, j}
int max = 0;
int[] maxes = {0, 0, 0, 0};
while(max < 3)
{
for(int i = 0; i < 3; i++)
{
for(int j = 0; j < 3; j++)
{
if(matriz[i][j] > FFC[0] && matriz[i][j] != maxes[max])
{
FFC[0] = matriz[i][j];
FFC[1] = i;
FFC[2] = j;
}
}
} //for de encontrar centro
if(FFC[0] < 4)
{
return 0;
}
if(FFC[1] == FFC[2]) //esta en el centro {valor, i, j}
{
if(FFC[0] > matriz[0][1] && ( (matriz[0][1] == matriz[2][1]) && (matriz[
{
return 1;
}
}
else //está en los lados
{
if(FFC[1] == 1 && FFC[2] == 0) //lado izquierdo en medio
{
if((matriz[1][0] > matriz[0][0]) && (matriz[1][0] > matriz[1][1]) && (ma
{
return 1;
}
```

```
}
if(FFC[1] == 1 && FFC[2] == 2) //lado derecho en medio
{
if(matriz[1][2] > matriz[0][2] && (matriz[0][2] == matriz[2][2]) && (mat
{
return 1;
}
}
if(FFC[1] == 0 && FFC[2] == 1) //arriba en medio
{
if(matriz[0][1] > matriz[0][2] && (matriz[0][2] == matriz[0][0]) && (mat
//evaluo lados
{
return 1;
}
}
if(FFC[2] == 1 && FFC[1] == 2) //lado derecho en medio
{
if(matriz[2][1] > matriz[2][0] && (matriz[2][0] == matriz[2][2]) && (mat
evaluo lados
{
return 1;
}
}
}//else
max++;
maxes[max] = FFC[0];
}

return 0;
}

public int SL(int[][] m)
{
int vida = 3;
if(m[0][0] > vida) // esquinita de arriba izr //cuadrado y lados
{
if((m[0][1] > vida && m[1][0] > vida && m[1][1] > vida) || (m[0][1] > vi
m[0][2] > vida) || (m[1][0] > vida && m[1][2] > vida))
{
return 1;
}
}
if(m[0][2] > vida) // esquinita de arriba dere
{
if(m[0][1] > vida && m[1][2] > vida && m[1][1] > vida)
```

```
{
return 1;
}
}
if(m[2][0] > vida) // esquinita de abajo izr
{
if(m[1][0] > vida && m[2][1] > vida && m[1][1] > vida)
{
return 1;
}
}
if(m[2][2] > vida) // esquinita de abajo der
{
if((m[2][1] > vida && m[1][2] > vida && m[1][1] > vida) ||
(m[2][0] > vida &&  m[2][1] > vida) || (m[1][2] > vida && m[0][2] > vida
{
return 1;
}
}
return 0;
}//still life

public int GL(int[][] m)
{
int tam = 2;
if(m[0][0] >= tam) // esquinita de arriba izr
{
if(m[1][0] >= m[0][0] && m[0][1] >= m[0][0] || (m[1][1] >= m[0][0] && (m
{
if(m[2][1] >= m[1][0] && m[1][2] >= m[0][1] || m[2][2] >= m[1][1] && (m[
{
return 1;
}
}
}
if(m[0][2] >= tam) // esquinita de arriba der
{
if(m[0][1] >= m[0][2] && m[1][2] >= m[0][2] || (m[1][1] >= m[0][2] && (m
{
if(m[1][0] >= m[0][1] && m[2][1] >= m[1][2] || m[2][0] >= m[1][1] && (m[
{
return 1;
}
}
}
if(m[2][0] >= tam) // esquinita de abajo izq
```

```
{
if(m[1][0] >= m[2][0] && m[2][1] >= m[2][0] || (m[1][1] >= m[2][0] && (m
{
if(m[0][1] >= m[1][0] && m[1][2] >= m[2][1] || m[0][2] >= m[1][1] && (m[
{
return 1;
}
}
}
if(m[2][2] >= tam) // esquinita de abajo der
{

if(m[2][1] >= m[2][2] && m[1][2] >= m[2][2] || (m[1][1] >= m[2][2] && (m
{
if(m[1][0] >= m[2][1] && m[0][1] >= m[1][2] || m[0][0] >= m[1][1] && (m[
[2]))
{
return 1;
}
}
}
return 0;
}//glider
}
```

3 Juego de la Vida con regla 7722 y 2233

3.1 Descripción del programa

Este programa permite simular el algoritmo conocido como Juego de la Vida de Conway. Contiene células las cuales se representan en el tablero a través de un cuadrito de un color a elegir, estando algunas vivas y otras muertas. El tamaño máximo del tablero (o cantidad máxima de células) es de 1000 y el mínimo es de 10. El programa cuenta el número de células vivas por cada generación, de la cuál s epuede definir la densidad de población de la inicial y las reporta, además de que de dicho número de células saca un promedio o media y la varianza.

3.2 Pruebas del funcionamiento

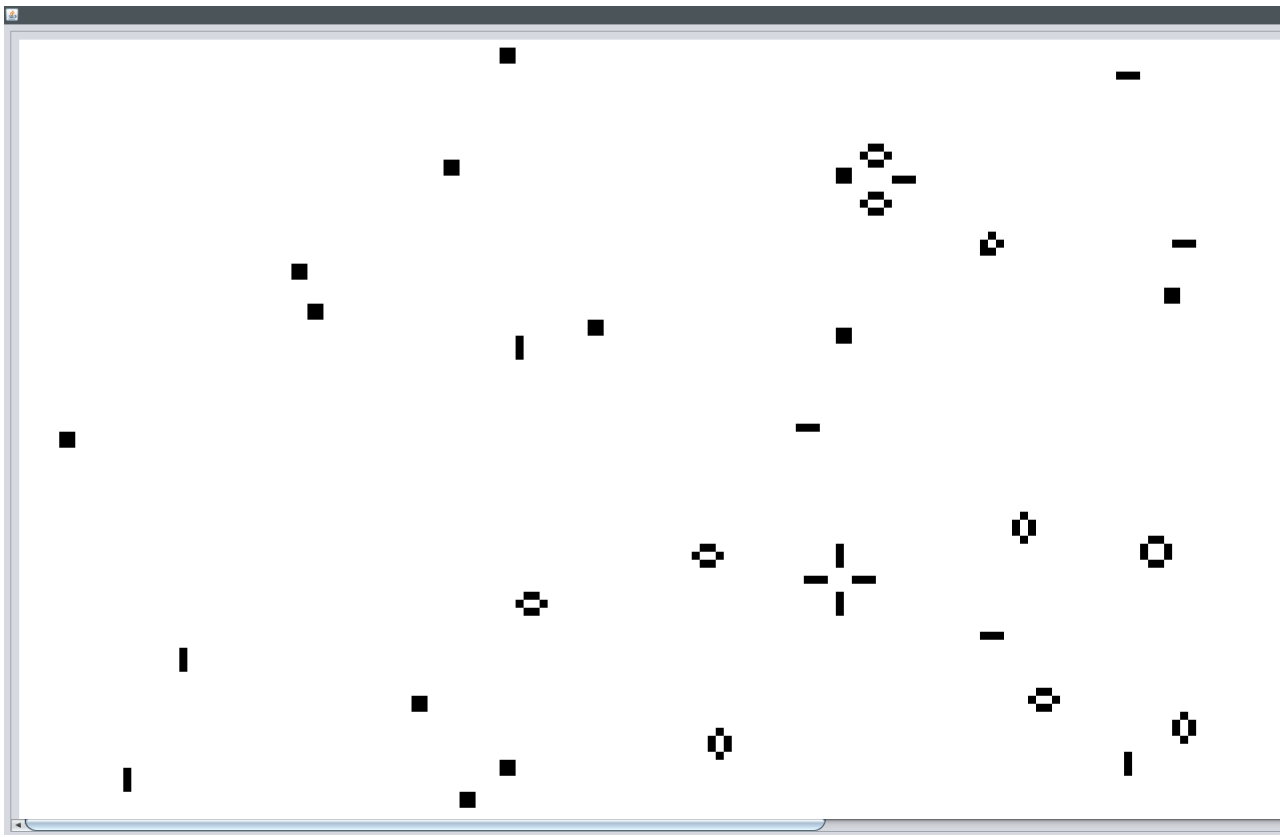


Figura 12: Inicialización del simulador con valores aleatorios y una densidad del 5% en la regla 2333.

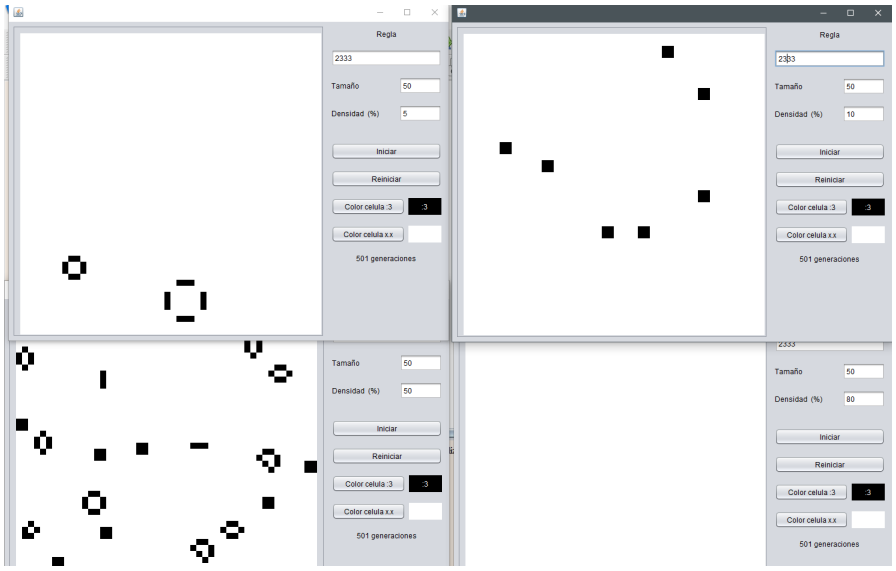


Figura 13: Simulador con varias densidades

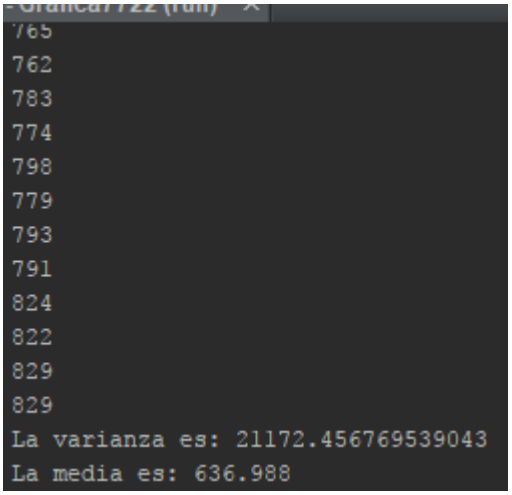


Figura 14: Resultado de ejecución del primer análisis con 5% de población y regla 2333.

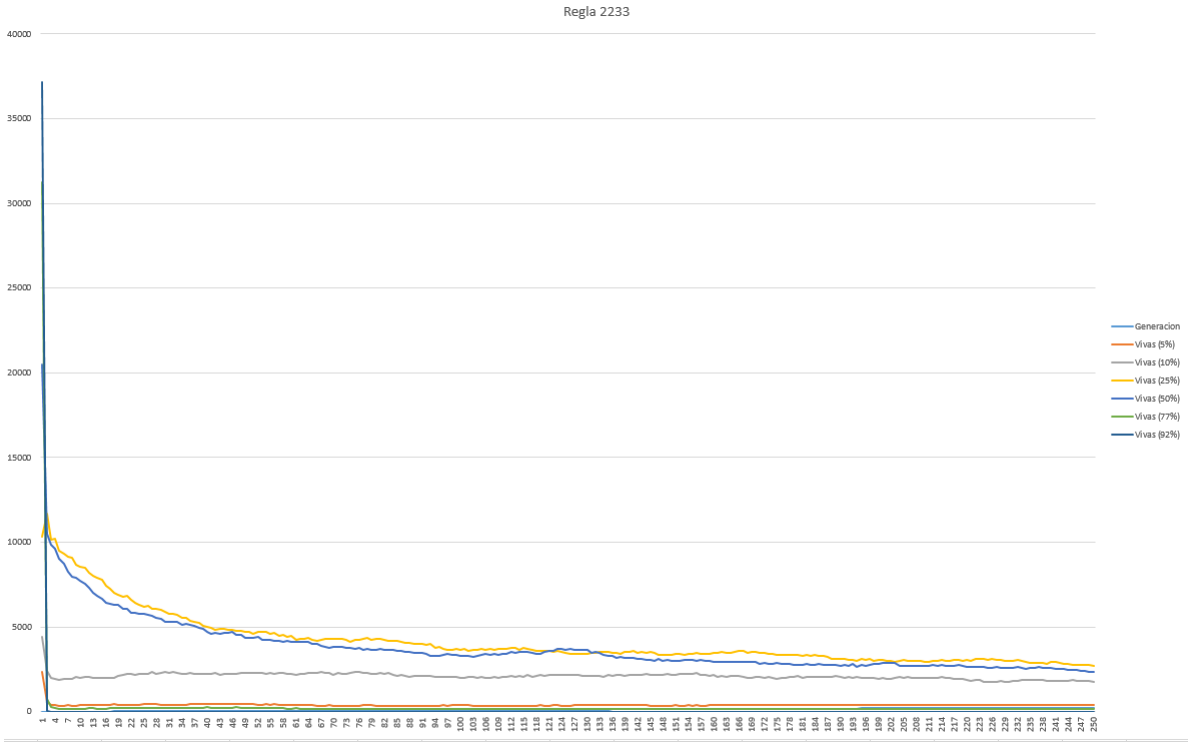


Figura 15: Gráfica de densidades para la regla 2333.

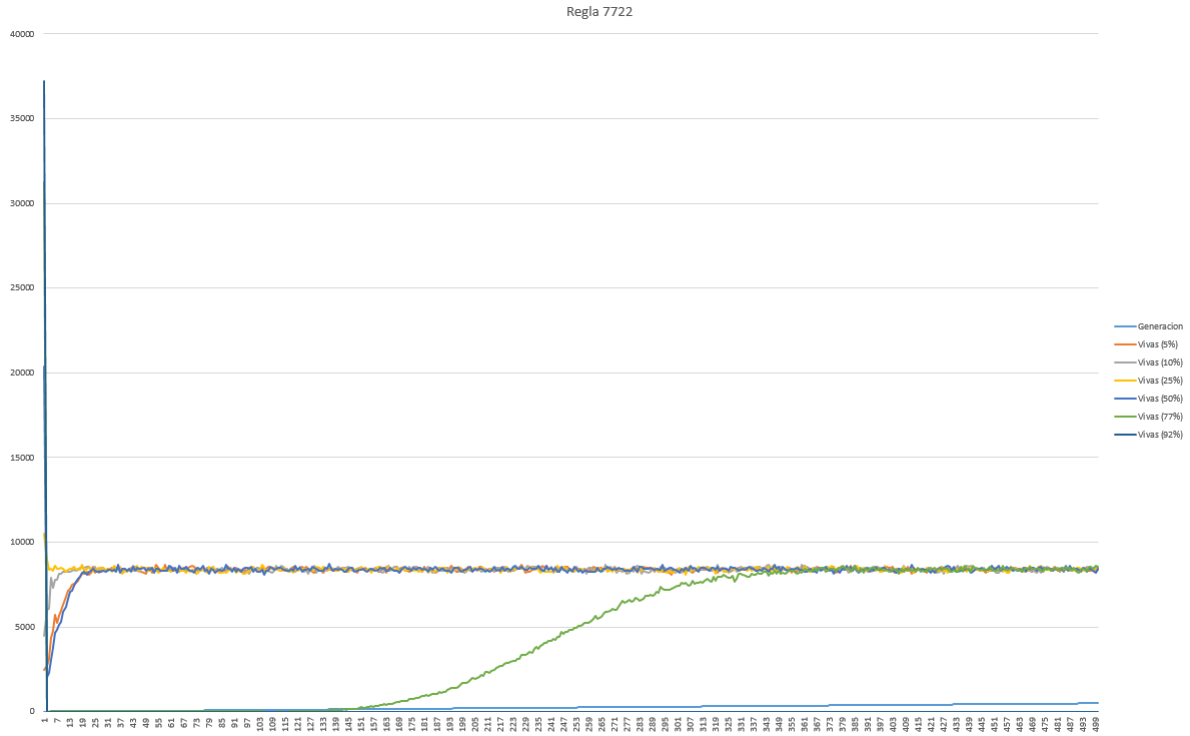


Figura 16: Gráfica de densidades de ala regla 7722.

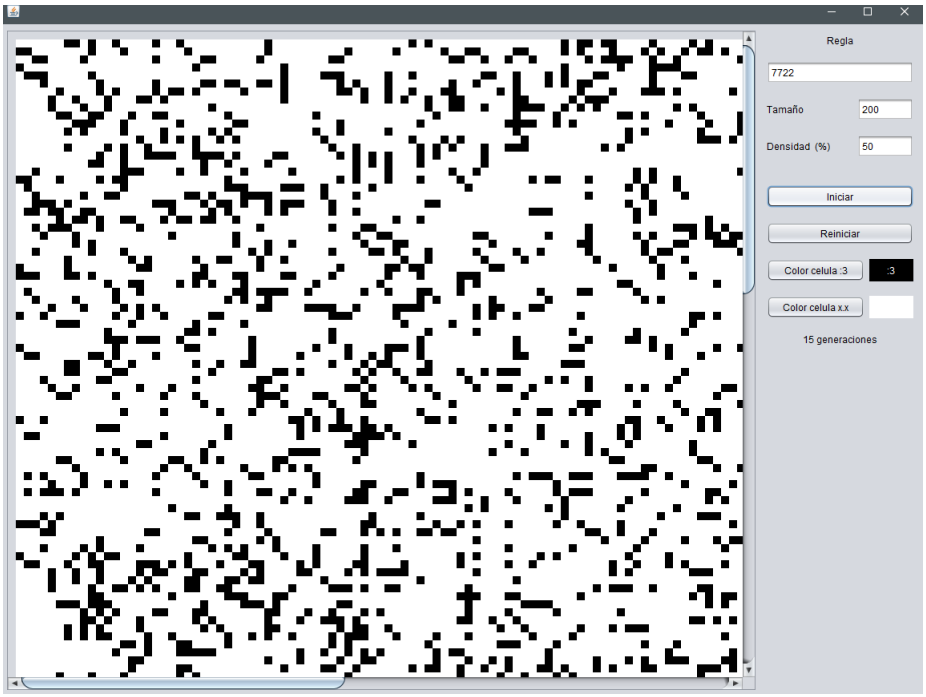


Figura 17: Cálculo con la regla 7722 y densidad de población de 50%

7722	Vivas (5%)	Vivas (10%)	Vivas (25%)	Vivas (50%)	Vivas (77%)	Vivas (92%)
Varianza	337419.01	67432.44	19529.42	683176.1	1.509	2774529
Media	8310.17	8378.266	8396.354	8328.246	4409.11	74.492
2233	Vivas (5%)	Vivas (10%)	Vivas (25%)	Vivas (50%)	Vivas (77%)	Vivas (92%)
Varianza	16450.28	42458.56	2525320.7	3176063.1	3860195.8	5528473.6
Media	399.96	2101.38	4165.11	3811.87	301.528	148.876

Figura 18: Resultado de la varianza y media de cada regla y densidad.

3.3 Codigo fuente

3.3.1 Clase Varianza

```
/*  * To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/ package logica;
/**  *
* @author Maku
*/ public class Varianza
```

```
 {
    int[] arrayVi;
    public Varianza(int[] arrayVi)
        {
            this.arrayVi = arrayVi;
        }
    public void calcula()
          {
            double acMedia = 0, acMedia2 = 0, varianza, x;
            int n = 0;
            for(int i = 0; i < arrayVi.length; i++)
                {
                  x = (double) arrayVi[i];
                  acMedia = acMedia + x;
                  acMedia2 = acMedia2 + x * x;
                  n++;
                }
            varianza =  acMedia2 / (n - 1) - (acMedia * acMedia) / (n * (n - 1));
            System.out.println("La varianza es: " + varianza);
            System.out.println("La media es: " + (acMedia / n));
                    }
}
```

3.3.2 Clase Generaciones

```
 */ package logica;
/**  *  * @author Maku
*/ public class Generaciones
 {
    int[] todas;
    public Generaciones(int[] todas)
        {
            this.todas = todas;
        }
    public void informa()
        {
            for(int i = 0; i < todas.length; i++)
                {
                    System.out.println(todas[i]);
                }
        }
}
```

3.3.3 Clase Dios

```
package logica;
```

```
import java.util.Random;
/**  *  * @author Maku  */
public class Dios
{
public Dios(){}
public int[][] creaVida(int tam, int densidad)
{
int[][] inicio = new int[tam + 2][tam + 2];
Random r = new Random();
for(int i = 1; i <= tam; i++)
{
for(int j = 1; j <= tam; j++)
{
inicio[i][j] = r.nextInt(100);
if(inicio[i][j] <=  densidad)
{
inicio[i][j] = 1;
}
else
{
inicio[i][j] = 0;
}
} //for 2
}
return inicio;
}
public int[][] destruyeVida(int tam)
{
int[][] fin = new int[tam + 2][tam + 2];
for(int i = 1; i <= tam; i++)
{
for(int j = 1; j <= tam; j++)
{
fin[i][j] = 0;
}
}
return fin;
}
}
```

4 AC - Reglas

4.1 Descripción del programa

Este programa permite simular las 256 reglas de producción de los autómatas celulares, permitiendo a su vez guardar o cargar archivos, cambiar colores, la velocidad y tamaño del anillo a evaluar por generación por cada iteración.

4.2 Pruebas del funcionamiento

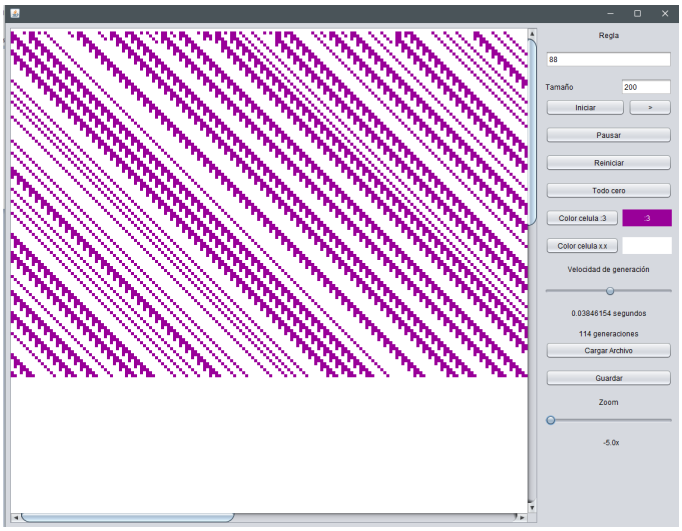


Figura 19: Inicialización del simulador con valores aleatorios

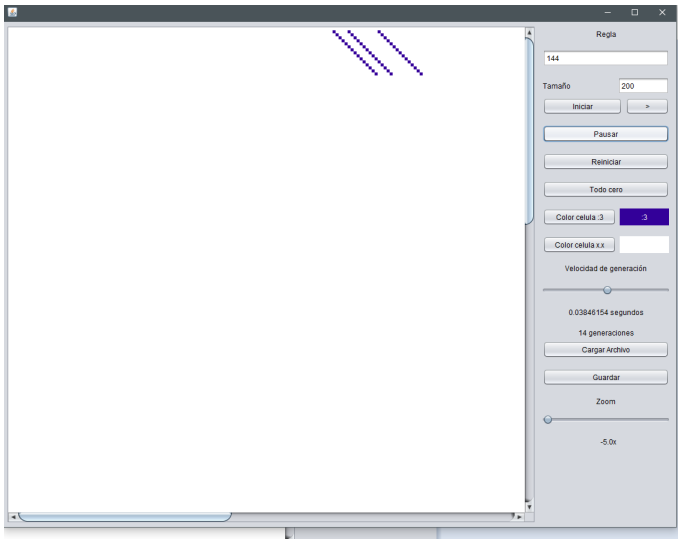


Figura 20: Cambio de colores del programa y patrón introducido por el usuario

Figura 21: Aspecto del Archivo Guardado

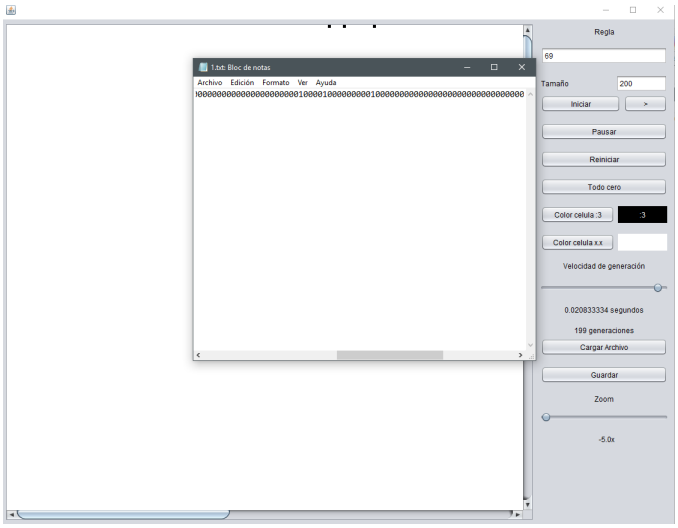


Figura 22: Archivo Cargado

4.3 Codigo fuente

4.3.1 Clase Universo

```
/*  * To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.  */
package gol;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseWheelEvent;
import java.awt.event.MouseWheelListener;
import javax.swing.JPanel;
import logica.Dios;
import logica.Vida;
/**  *  * @author Maku  */
class Universo extends JPanel  {
 int[][] m;
   int tam, click, tamCelula = 10, tiempo; //tam = num de celulas
   private Graphics2D g2d;
 Color viva,  muerta;
```

```
int[] regla = new int[4];
Dios gg = new Dios();
private int zoom = 0;
private static final double ZOOM_AMOUNT = 1.1;
    Universo ()             {                               }
Universo(int[] regla, int tam, int tiempo, Color vi, Color mu)
{
  this.regla = regla;
     this.tam = tam;
          setPreferredSize(new Dimension(tam*tamCelula,tam*tamCelula)); //x, y
          m = gg.creaVida(tam);
     this.tiempo = tiempo;
 viva = vi;
  muerta = mu;
  this .addMouseListener(new MouseListener()
{
  @Override
                       public void mouseClicked(MouseEvent e)
                         {
                       int x, y;
Point p = e.getPoint();
y = (int) Math.floor((float)p.getX()/tamCelula);
x = (int) Math.floor((float)p.getY()/tamCelula);
if(x < tam && y < tam)
{
if(m[x][y] == 1)
{
m[x][y] = 0;
}
else
{
m[x][y] = 1;
}
click = 1;
repaint();
}
}

@Override
public void mousePressed(MouseEvent e) {
//throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.
}

@Override
public void mouseReleased(MouseEvent e) {
```

```
// throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.
}

@Override
public void mouseEntered(MouseEvent e) {
//  throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.
}

@Override
public void mouseExited(MouseEvent e) {
//  throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.
}                                                    });

Universo(int[] regla, int tam, int tiempo, int cero, Color vi, Color mu)
{
this.regla = regla;
this.tam = tam;
setPreferredSize(new Dimension(tam*tamCelula,tam*tamCelula)); //x, y
m = gg.destruyeVida(tam);
this.tiempo = tiempo;
viva = vi;
muerta = mu;
this .addMouseListener(new MouseListener() {
@Override
public void mouseClicked(MouseEvent e)
{
int x, y;
Point p = e.getPoint();
y = (int) Math.floor((float)p.getX()/tamCelula);
x = (int) Math.floor((float)p.getY()/tamCelula);
 if(x < tam && y < tam)
{
if(m[x][y] == 1)
{
m[x][y] = 0;
}
else
{
m[x][y] = 1;
}
click = 1;
repaint();
}
```

```
}

@Override
public void mousePressed(MouseEvent e) {
//throw new UnsupportedOperationException("Not supported yet."); //To change body of ge

@Override
public void mouseReleased(MouseEvent e) {
// throw new UnsupportedOperationException("Not supported yet."); //To change body of g

@Override
public void mouseEntered(MouseEvent e) {
//  throw new UnsupportedOperationException("Not supported yet."); //To change body of

@Override
public void mouseExited(MouseEvent e) {
throw new UnsupportedOperationException("Not supported yet."); //To change body of gene
}

Universo(int[] regla, int tam, int tiempo, int[][] car, Color vi, Color mu)
{
this.regla = regla;
this.tam = tam-2;

System.out.println(this.tam);
setPreferredSize(new Dimension(tam*tamCelula,tam*tamCelula)); //x, y
m = car;
this.tiempo = tiempo;
viva = vi;
click = 1;
muerta = mu;
this .addMouseListener(new MouseListener() {
@Override
public void mouseClicked(MouseEvent e)
{
int x, y;
Point p = e.getPoint();
y = (int) Math.floor((float)p.getX()/tamCelula);
x = (int) Math.floor((float)p.getY()/tamCelula);
if(x < tam && y < tam)
{
if(m[x][y] == 1)
{
m[x][y] = 0;
}
else
```

```
{

m[x][y] = 1;
}
click = 1;
repaint();
}
}

@Override
public void mousePressed(MouseEvent e) {
//throw new UnsupportedOperationException("Not supported yet."); //To change body of ge

@Override
public void mouseReleased(MouseEvent e) {
// throw new UnsupportedOperationException("Not supported yet."); //To change body of g

@Override
public void mouseEntered(MouseEvent e) {
//  throw new UnsupportedOperationException("Not supported yet."); //To change body of

@Override
public void mouseExited(MouseEvent e) {
//  throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.

});
}
@Override
public void paintComponent(Graphics g)
{             super.paintComponent(g);
if(click == 1)
{
click = 0;
// repaint();
}
else
      {
m = analiza(m, regla);
}
g2d = (Graphics2D) g;
for(int i = 1; i <= tam; i++)
{
for(int j = 1; j <= tam; j++)
if(m[i][j] == 1)
{
```

```
g2d.setColor(viva);
g2d.fillRect(tamCelula*j, tamCelula*i, tamCelula, tamCelula); //x, y, tamx, tamy
g2d.drawRect(tamCelula*j, tamCelula*i, tamCelula, tamCelula); //x, y, tamx, tamy //x, y
}                              else
{
g2d.setColor(muerta);
g2d.fillRect(tamCelula*j, tamCelula*i, tamCelula, tamCelula); //x, y, tamx, tamy
g2d.drawRect(tamCelula*j, tamCelula*i, tamCelula, tamCelula); //x, y, tamx, tamy //x, y
}
}
}                         // repaint();
try
{
//System.out.println("tiempo " +tiempo);
Thread.sleep(tiempo);
 }
catch (Exception e) {}
     } //paint component
public int[][] analiza(int[][]m, int[] regla)
        {
Vida v = new Vida();
//System.out.println("hay"+regla[0]);
return v.existe(m,regla, tam);
}
 public int[][] getMatriz()
{
return m;
}
}//class
```

4.3.2 Clase FrameTodo

```
private void BtnStepActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_BtnStepActionPerformed
// TODO add your handling code here:
if(autoriza() == 1)
{
if(gens == 0 && arshivo == 0)
     {
        ScrollM.revalidate();
        tam = Integer.parseInt(txtTamUniv.getText()) ;
                un=newUniverso(regla,tam,getTiempo(), cViva, cMuerta);
                ScrollM.getViewport().add(un);
                ScrollM.setVisible(true);
                repaint();
                gens++;
```

```
     }
    else
     {
        arshivo = 0;
         repaint();
         gens++;
          }
 un.setTiempo(tiempo);
 un.setZoom(zoom);
 lbGens.setText(gens + " generaciones");
 BtnPausar.setText("Reanudar");
}
}
//GEN-LAST:event_BtnStepActionPerformed
private void BtnReiniciarActionPerformed(java.awt.event.ActionEvent evt)
{
//GEN-FIRST:event_BtnReiniciarActionPerformed
// TODO add your handling code here:
//ScrollM.removeAll();
if(autoriza() == 1)
{
 ScrollM.revalidate();
//   ScrollM.repaint();
gens = 0;
lbGens.setText("0 generaciones");
tam = Integer.parseInt(txtTamUniv.getText()) ;
un = new Universo(regla, tam,getTiempo(), cViva, cMuerta);
ScrollM.getViewport().add(un);
    ScrollM.setVisible(true);
BtnPausar.setText("Pausar");
 timer.stop();
     repaint();
un.setTiempo(tiempo);
un.setZoom(zoom);
}
}//GEN-LAST:event_BtnReiniciarActionPerformed
private void VelocidadSliderStateChanged(javax.swing.event.ChangeEvent e
{
//GEN-FIRST:event_VelocidadSliderStateChanged
// TODO add your handling code here:
jLabelTiempo.setText(Float.toString((float) (1.0 / VelocidadSlider.getVa
tiempo = 1000/VelocidadSlider.getValue();
setTiempo(tiempo);
}
//GEN-LAST:event_VelocidadSliderStateChanged
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
```

```
{//GEN-FIRST:event_jButton1ActionPerformed

//INICIAR
jButton1.setText("Iniciar");
if(autoriza() == 1)
 {
            if(timer.isRunning())
                     {
                         System.out.println(":3");
                     }
                    else
                {
//
ScrollM.removeAll();
                    ScrollM.revalidate();
                    gens = 0;
                    lbGens.setText("0 generaciones");
                    tam = Integer.parseInt(txtTamUniv.getText()) ;
        un = new Universo(regla, tam,getTiempo(), cViva, cMuerta);

                                          ScrollM.getViewport().add(un);
                    ScrollM.setVisible(true);
                    timer.start();
                    repaint();
                 }
}
//autoriza
}//GEN-LAST:event_jButton1ActionPerformed
    private void BtnPausarActionPerformed(java.awt.event.ActionEvent evt
{//GEN-FIRST:event_BtnPausarActionPerformed
// TODO add your handling code here:
if(timer.isRunning())
{
    timer.stop();
   BtnPausar.setText("Reanudar");
}
else
{
   if(BtnPausar.getText().equals("Reanudar"))
        {
            timer.restart();
            BtnPausar.setText("Pausar");
        }
}
}
```

5 Generador de Atractores

5.1 Descripción del programa

Este programa se encarga de generar el código del programa Mathematica para poder garficar el atractor generado por la regla dada del autómata celular.

5.2 Pruebas del funcionamiento

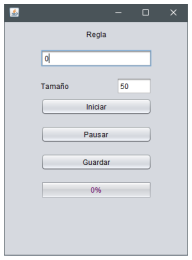


Figura 23: Inicio del simulador y analizador aleatoriamente.

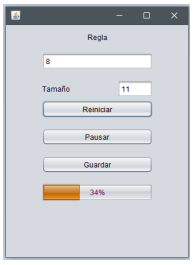


Figura 24: Ejecución del analizador

Figura 25: Análisis Terminado

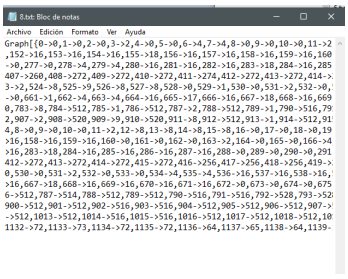


Figura 26: Archivo generado.

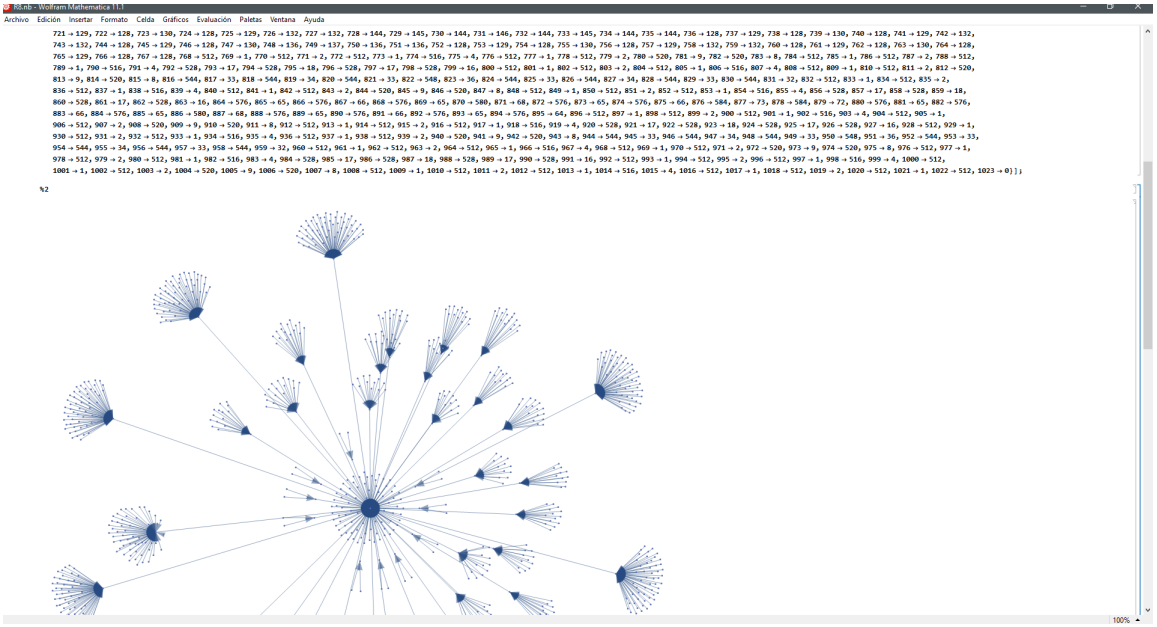
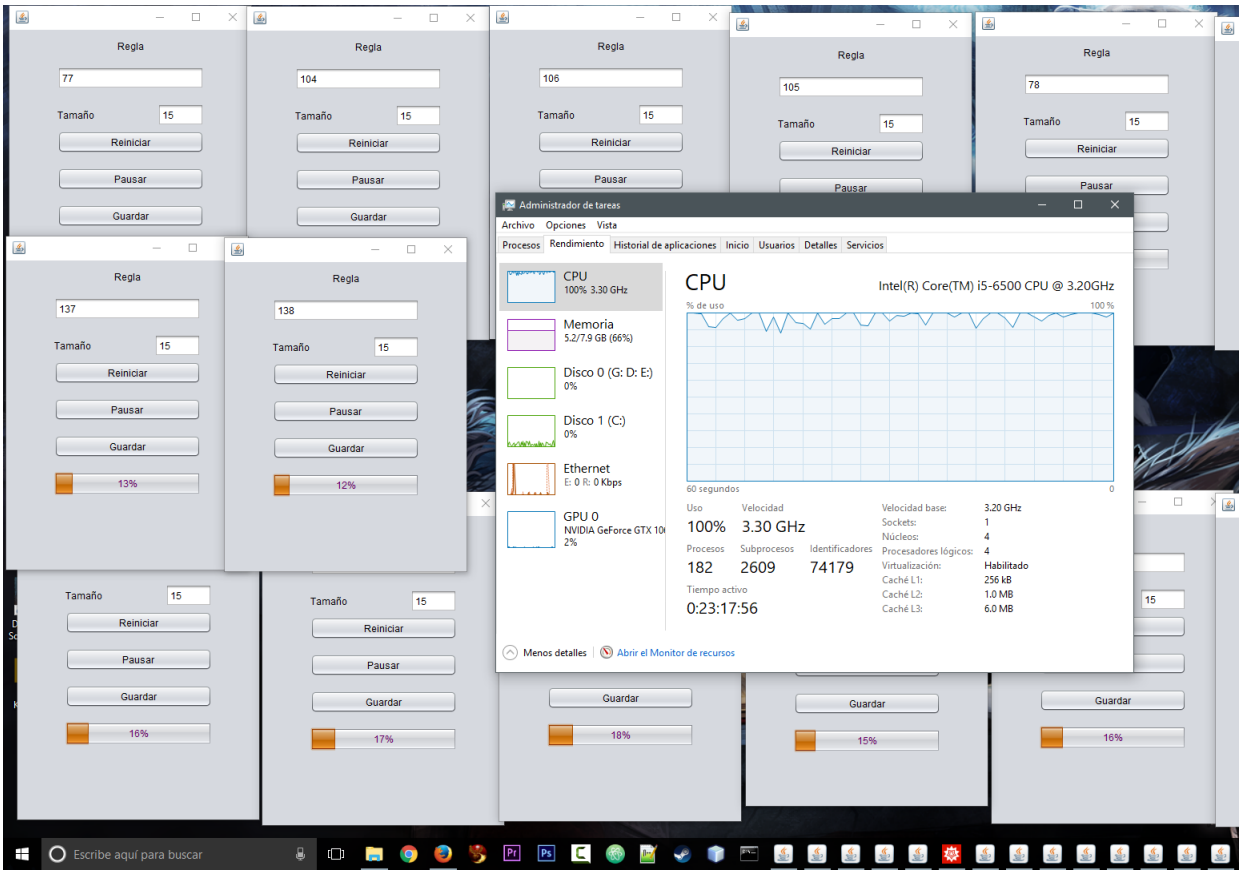


Figura 27: Graficación del atractor

Figura 27.5: Ejecución del programa x15



5.3 Codigo fuente

5.3.1 Clase Vida

```
Public int[] binario(int r, int tam)  //entero
    a binario
{
int[] b = new int[tam];
String bin = Integer.toBinaryString(r);
while(bin.length() < tam)
    {
        bin = "0" + bin;
    }
//  System.out.println("s" + bin);
for(int i = 0; i < tam; i++)
    {
        b[i] = Character.getNumericValue(bin.
            charAt(i));
```

```
    }
return b;
}
public int[] existe (int[] m, int regla, int
    tam) //analizador del automata y
    sustituidor
{
     int[] aux = new int[tam], sust = new int
         [8]; //aqui va el sustituidor
     String cadenita;
     int[] bin = binario(regla, 8);
     for(int i = 0; i < tam; i++) //aqui divido
          el arreglo en pedacitos de 3 para
         analizarlo
          {
if(i == 0) //uno el inicio con el final
{
cadenita = Integer.toString(m[tam - 1]);
cadenita += Integer.toString(m[i]) + Integer.
    toString(m[i + 1]);
}
else
{
if(i == (tam - 1))
{
cadenita = m[i - 1] + "" + m[i] + "" + m[0] +
    ""; //uno el final al inicio
}
else
{
cadenita = m[i - 1] + "" +m[i] + "" +m[i + 1]
    + "";
}
}
         //    System.out.println(cadenita);
switch(cadenita)
     {
          case "000":
              aux[i] = bin[7];
          break;
          case "001":
              aux[i] = bin[6];
          break;
          case "010":
              aux[i] = bin[5];
          break;
```

```
            case "011":
                aux[i] = bin[4];
            break;
            case "100":
                aux[i] = bin[3];
            break;
            case "101":
                aux[i] = bin[2];
            break;
            case "110":
                aux[i] = bin[1];
            break;
            case "111":
                aux[i] = bin[0];
            break;
            }
    }
        return aux;
    }//main
    public int decimal(int[] m)
    {
    String bin = "";
    int dec;
    for(int i = 0; i < m.length; i++)
    {
        bin += Integer.toString(m[i]);
    }
    dec = Integer.parseInt(bin, 2);
     //convierto binario a decimal
    return dec;
    }
```

5.3.2 Clase FrameTodo

```
if (src == timer)
{
Prog.setValue(nodoActual);
if(nodoActual == (int)Math.pow(2,tam)) //ya terminé
    {
        timer.stop();
        esc.escribe(regla, "}];");
        JOptionPane.showMessageDialog(null, "Análisis Terminado. Se ha creado un
    }
else
    {
```

```
        int[] mat = new int[tam]; //actual
        int[] mat2 = new int[tam]; //evaluado
        int nodoSiguiente; //evaluado en decimal
        String coma = ",";
        mat = v.binario(nodoActual, tam); //obtengo mi nodo actual en binario
        mat2 = v.existe(mat, regla, tam);
        nodoSiguiente = v.decimal(mat2);
        //System.out.println(nodoActual + " -> " + nodoSiguiente);
        if(nodoActual == (int)Math.pow(2,tam) - 1)
            {
                coma = "";
            }
        String txt = nodoActual + "->" + nodoSiguiente+coma;
        esc.escribe(regla, txt);
        nodoActual++;
    }
}
```

5.3.3 Clase Escritor

```
        public int escribe(int regla, String text)
            {
                    try (BufferedWriter bw = new BufferedWriter(new FileWriter(r
                            {
                               bw.write(text);
                               bw.close();
                               return 1;
                            }
                    catch (IOException e)
                            { e.printStackTrace(); return 0;}
               }
```

5.4 Clasificación de los atractores

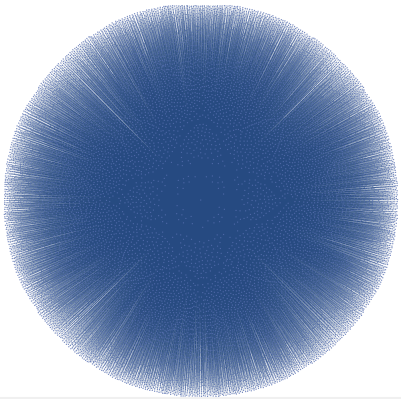
5.4.1 Clase I - Estáticos

1. Descripción

   (a) Éstos autómatas celulares presentan un comportamiento estático y homogéneo donde su atractor siempre se dirige a un punto y se mantiene estacionado ahí de manera infinita.

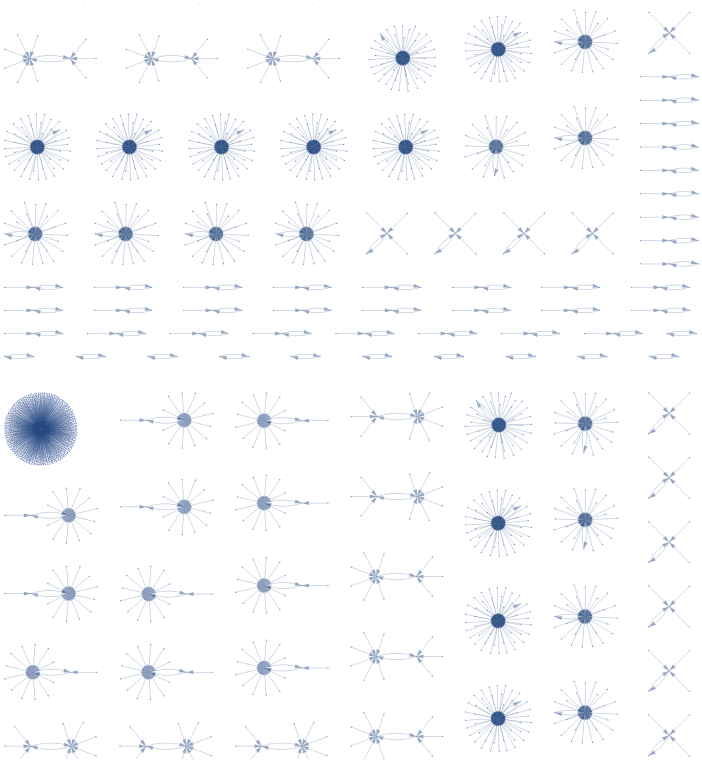
2. Atractores de automatas clase I

   (a) Regla 0



i.

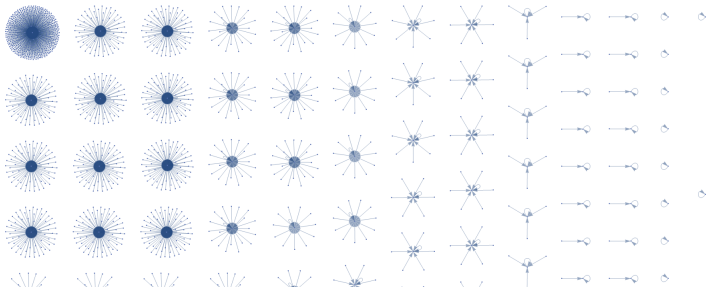
(b) Regla 1



i.

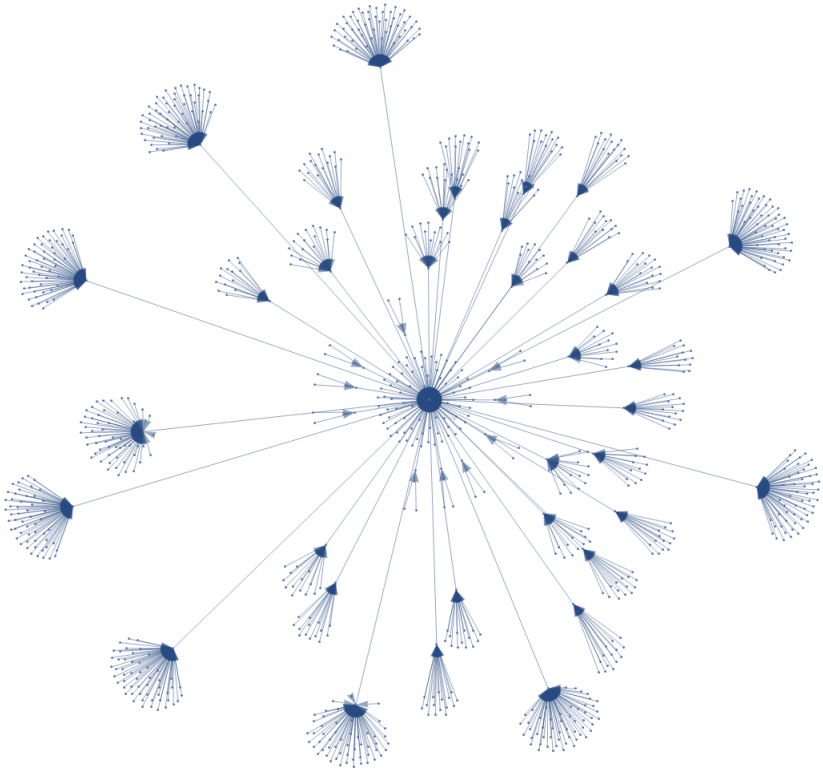
ii.

(c) Regla 4



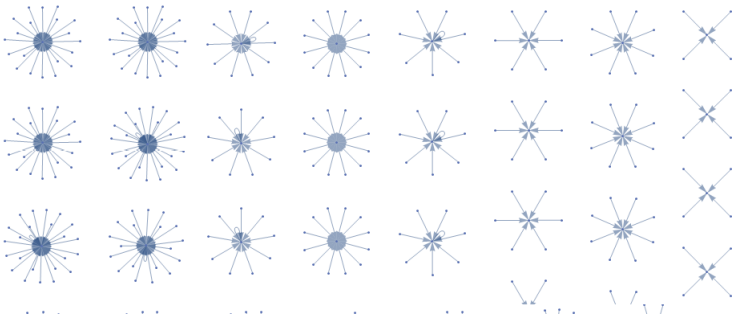
i.

(d) Regla 8

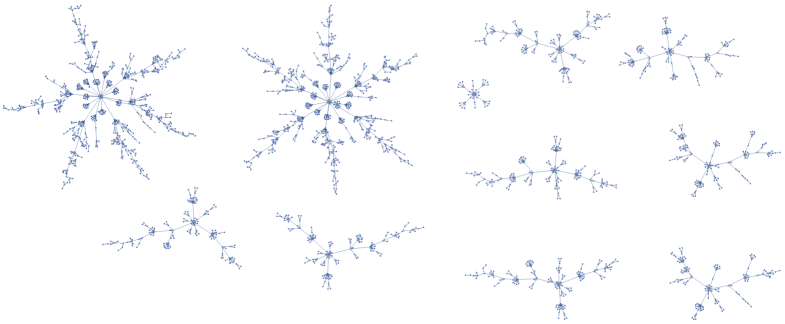


i.

(e) Regla 12

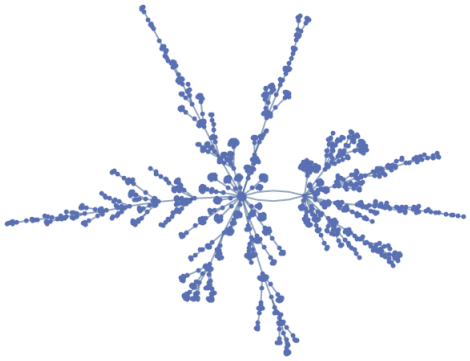
i. 

(f) Regla 13



i.

(g) Regla 28

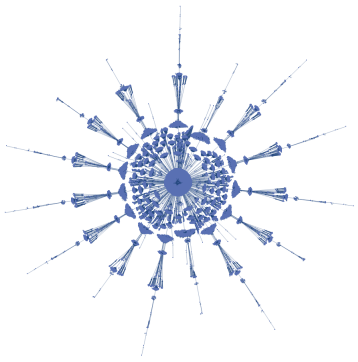


i.

(h) Regla 29

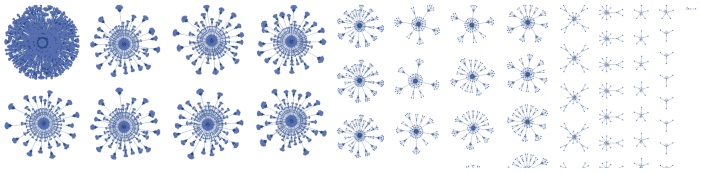
i. 

(i) Regla 32



i.

(j) Regla 36

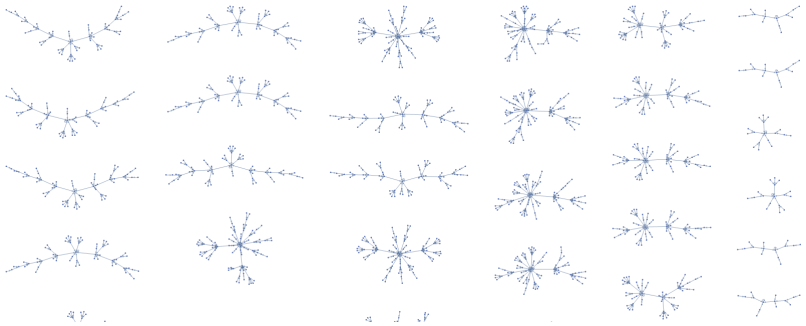


i.

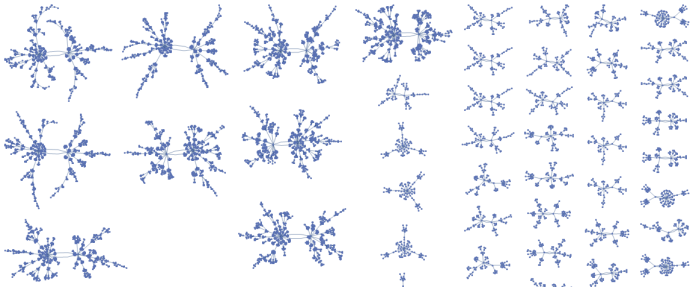
(k) regla 40

i.

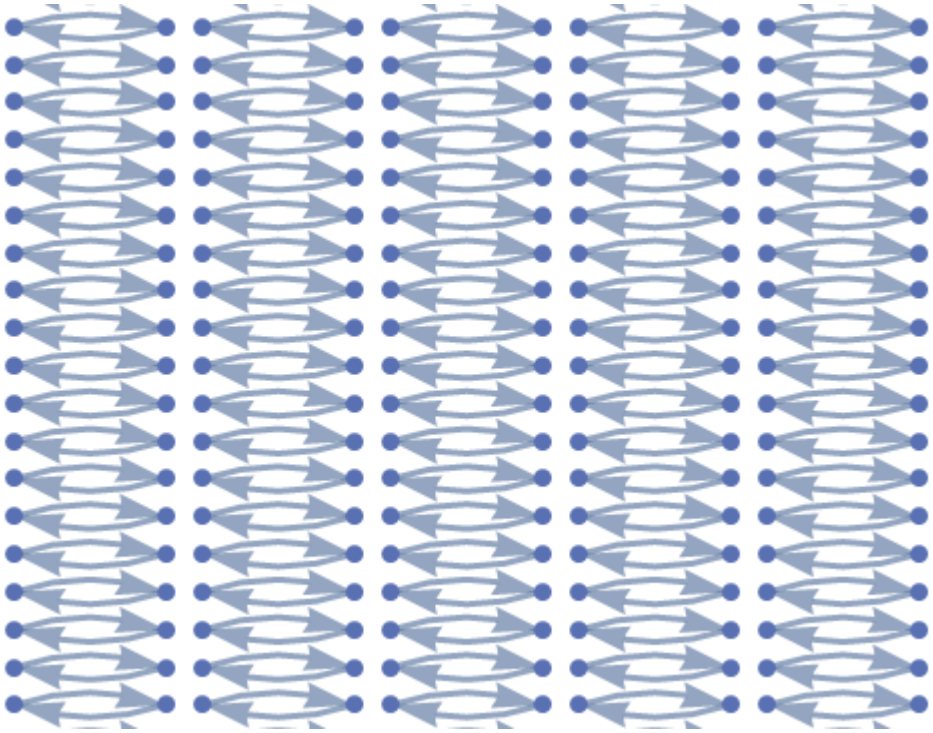
(l) Regla 44

i. 

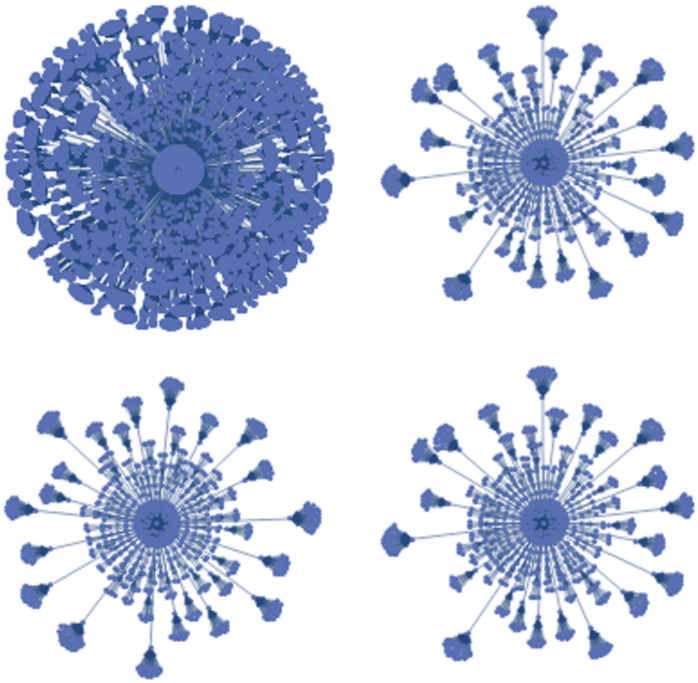
(m) Regla 50

i. 

(n) Regla 51

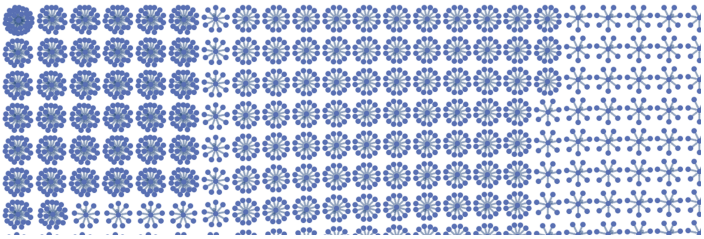
i. 

(o) Regla 72



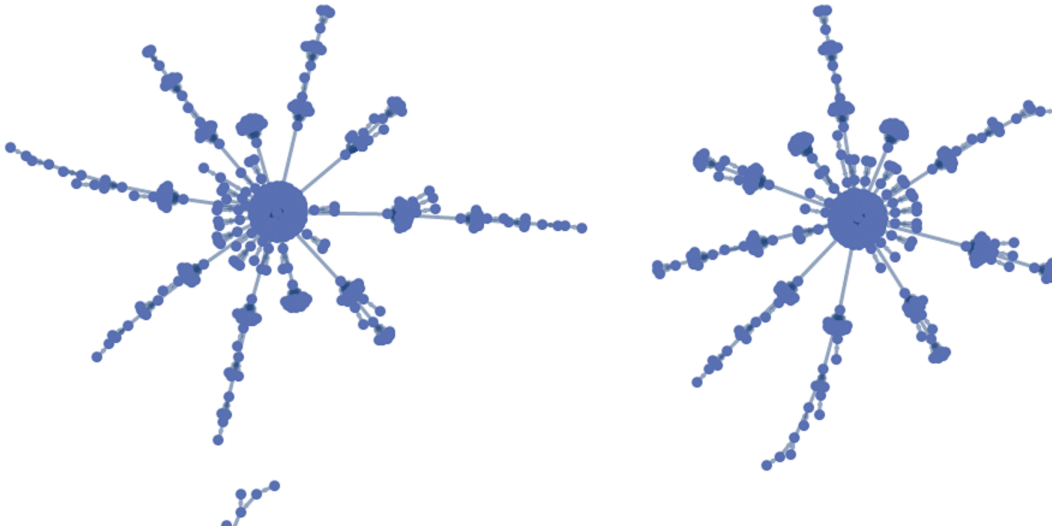
i.

(p) Regla 76



i.

(q) Regla 77

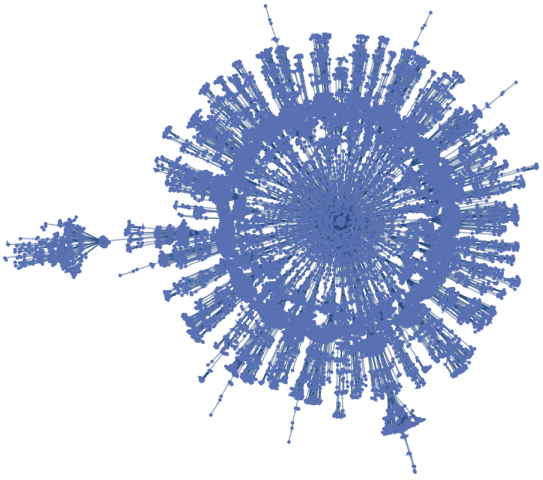


i.

(r) Regla 78

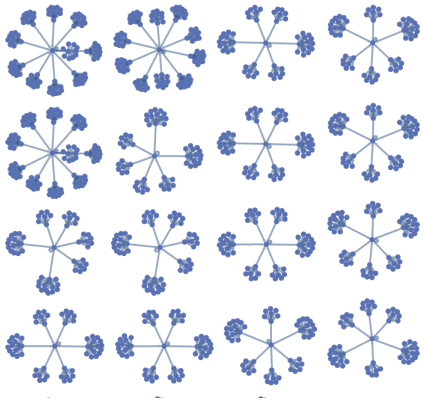
i.

(s) Regla 104



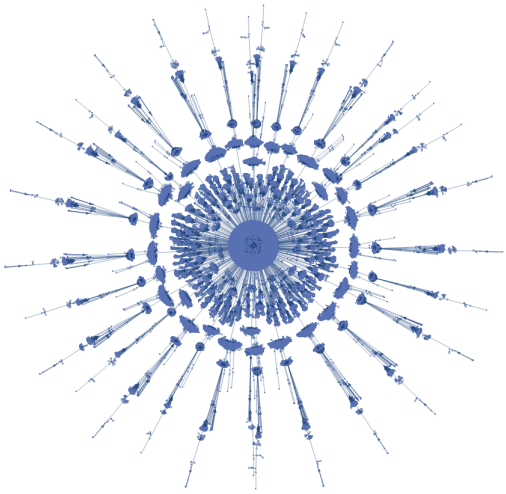
i.

(t) Regla 108



i.

(u) Regla 128

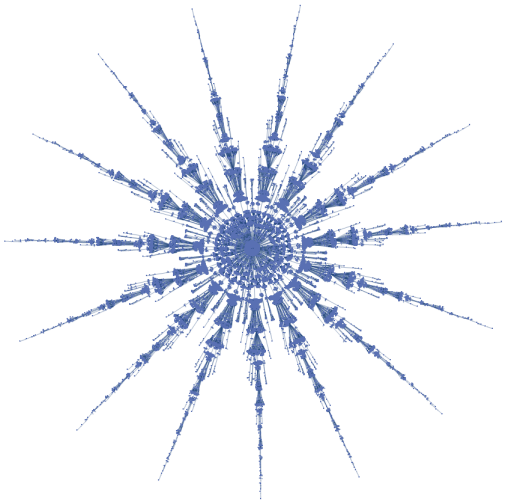


i.

(v) Regla 132

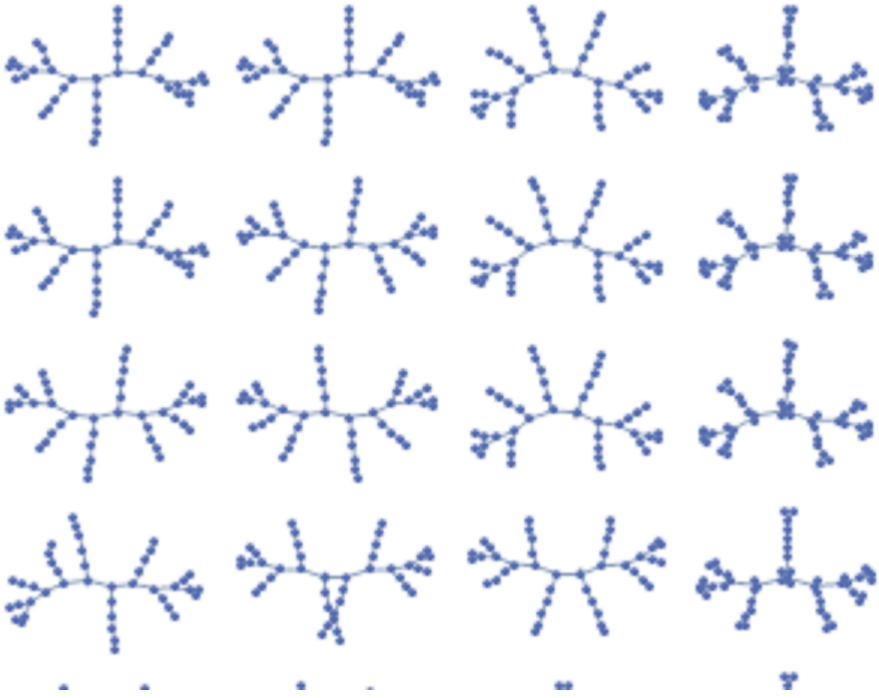
i.

(w) Regla 136



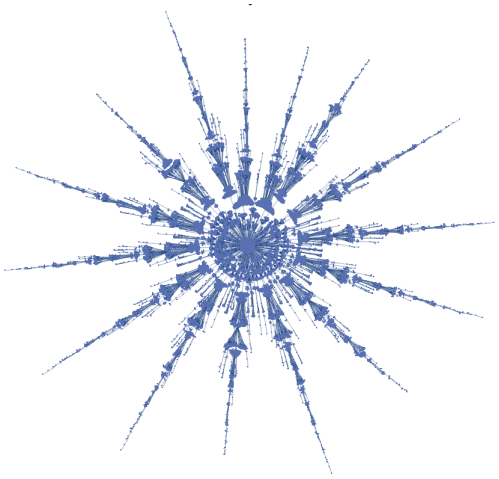
i.

(x) Regla 140



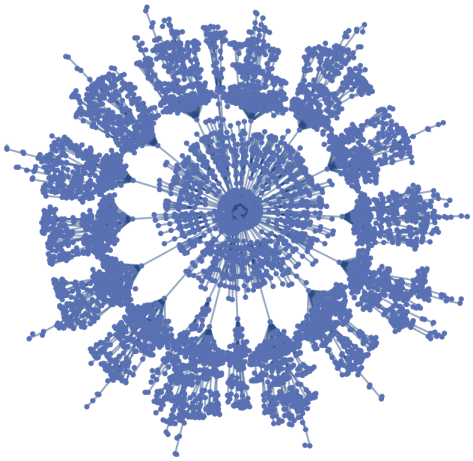
i.

(y) Regla 160



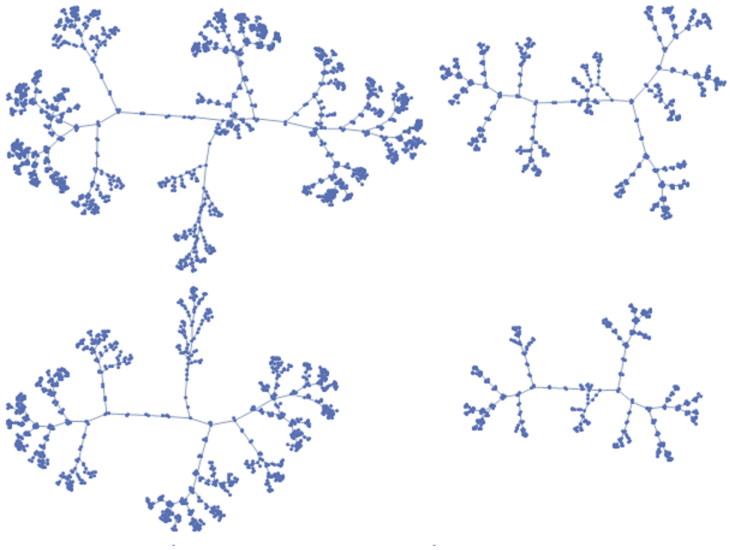
i.

(z) Regla 164



i.

() Regla 172

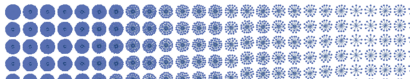


i.

() Regla 200

i.

() Regla 204



i.

() Regla 232

i.

5.4.2 Clase II - Periódicos

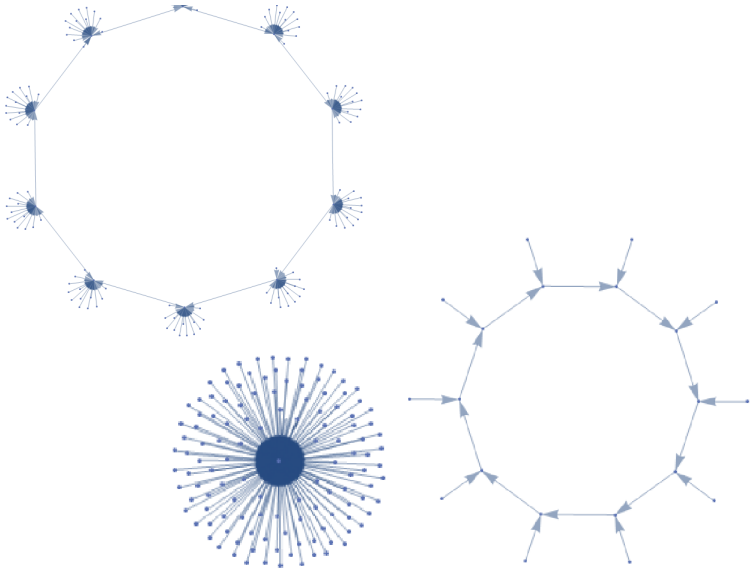
1. Descripción

   (a) El atractor o los atractores de éstas reglas siempre terminan en n estados cíclicos que se repiten uno tras otro indefindiamente, por más ramas y divisiones que tenga siempre caen en los estados loop.

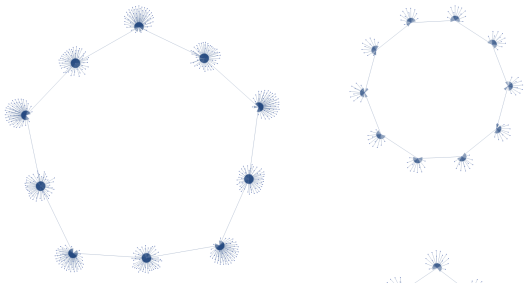
2. Reglas

   (a) Regla 2

   i.

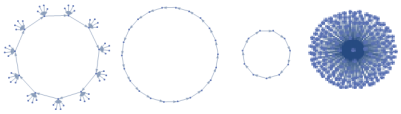


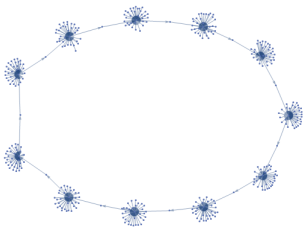
   ii.



   (b) Regla 3

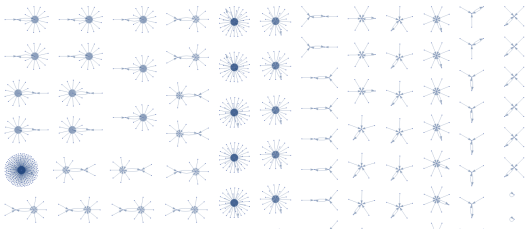
   i.





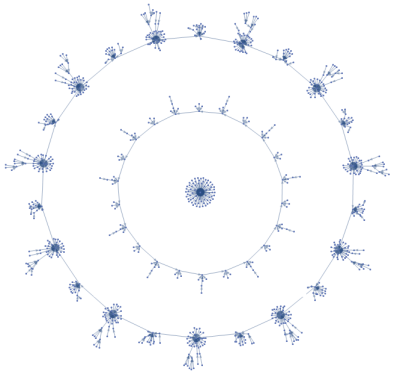
ii.

(c) Regla 5

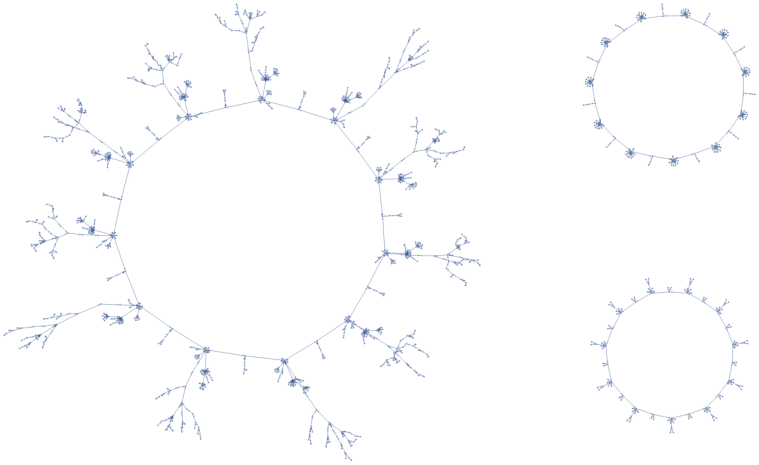


i.

(d) Regla 6



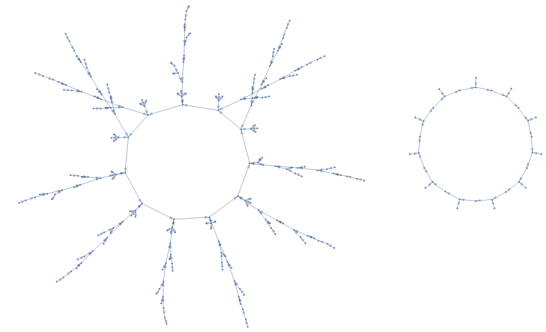
i.



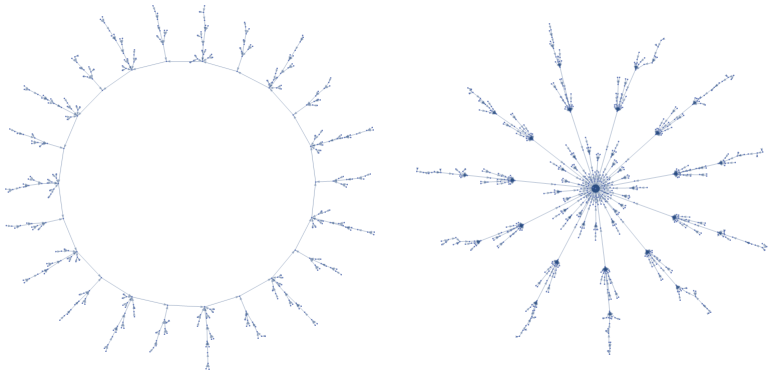
ii.

(e) Regla 7

i.

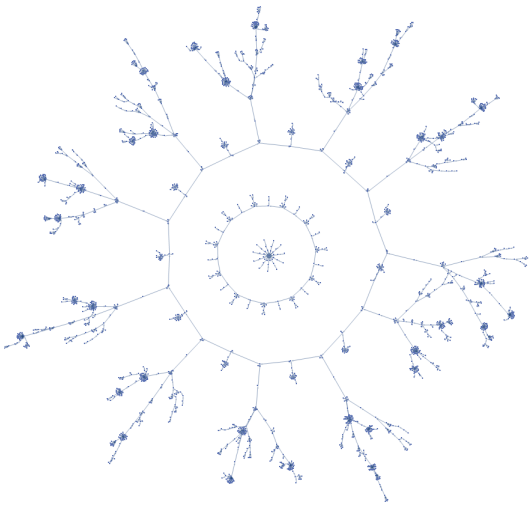


ii.



(f) Regla 9

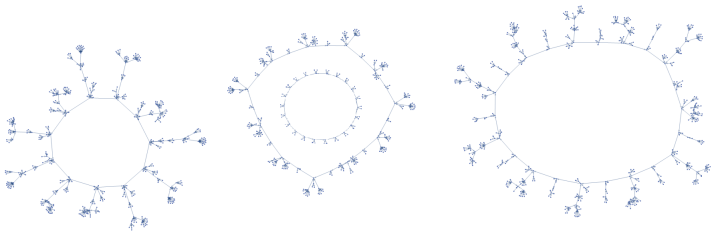
i.



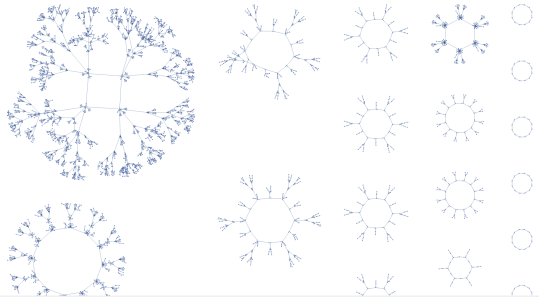
(g) Regla 10

i. 

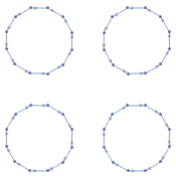
(h) Regla 11

i. 

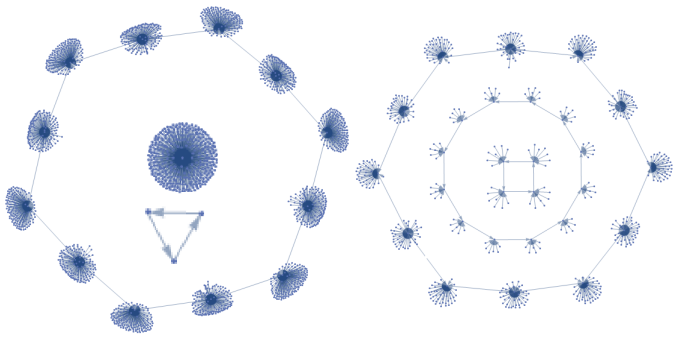
(i) Regla 14

i. 

(j) Regla 15

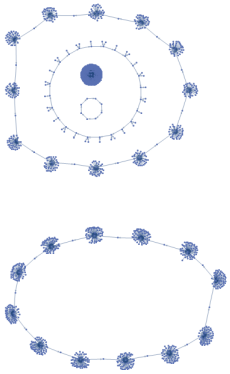
i. 

(k) Regla 16



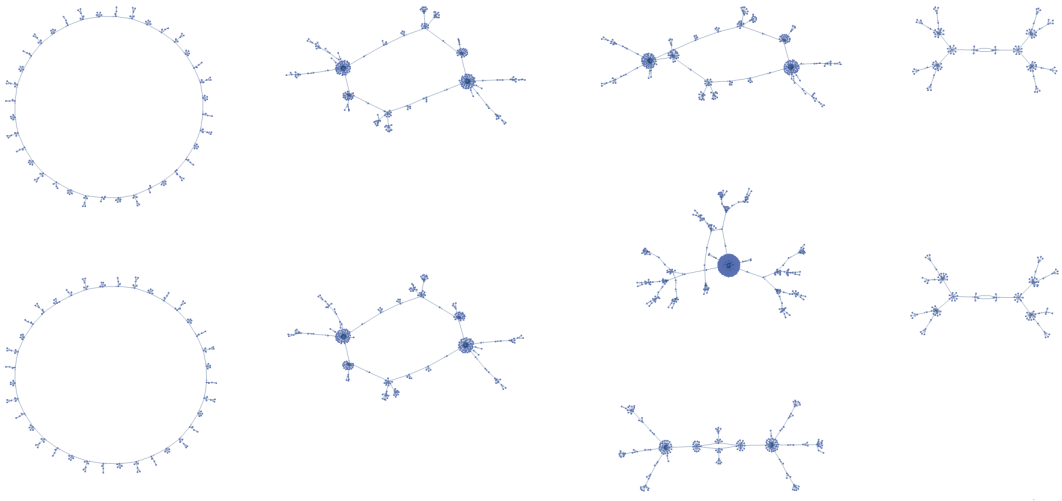
i.

(l) Regla 17



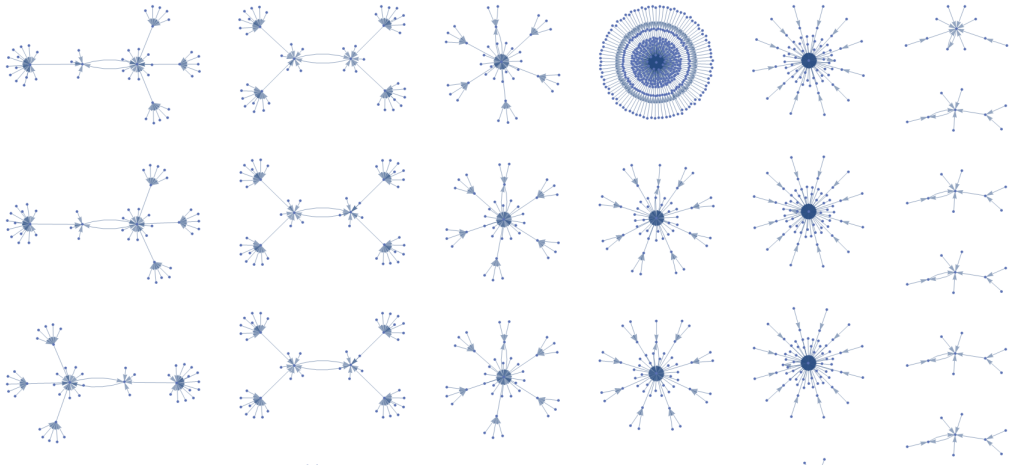
i.

(m) Regla 18



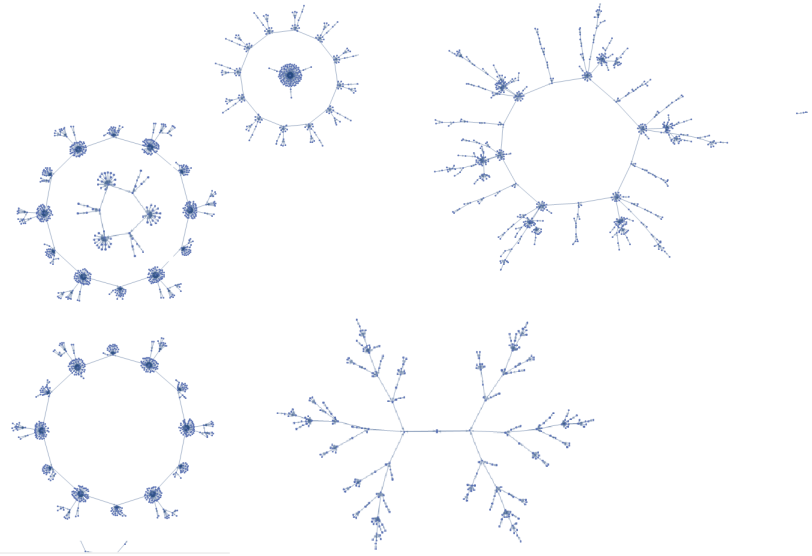
i.

(n) Regla 19



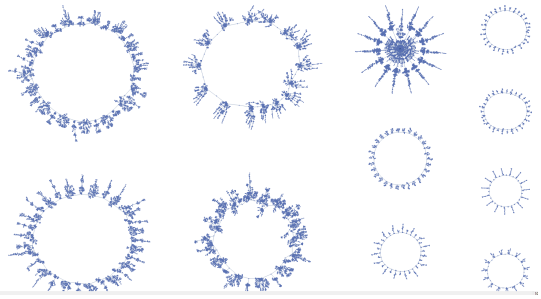
i.

(o) Regla 20



i.

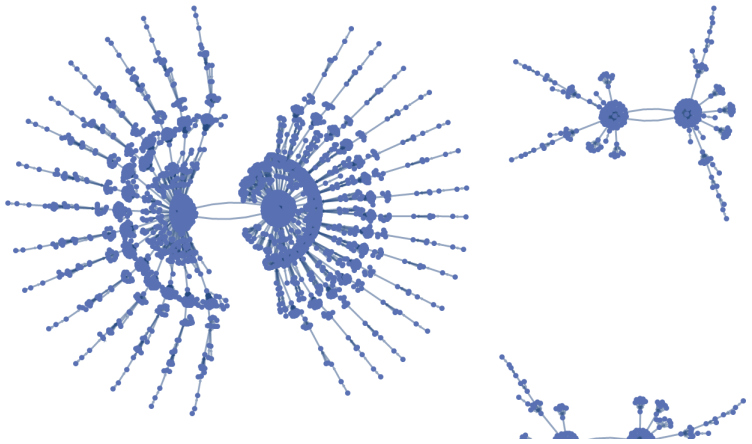
(p) Regla 21



i.

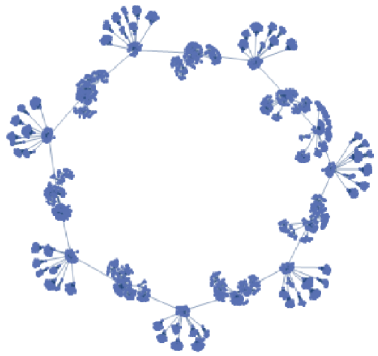
(q) Regla 23

i.

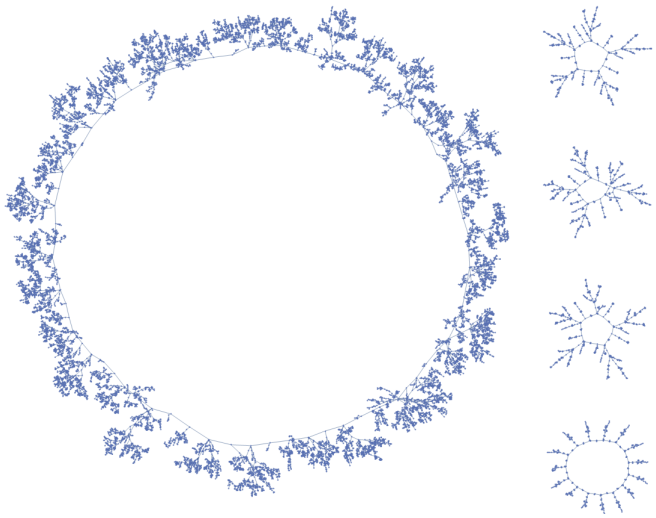


(r) Regla 24

i.

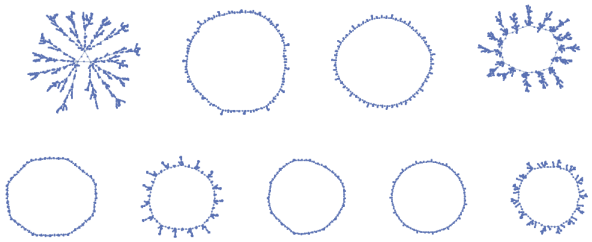


(s) Regla 25



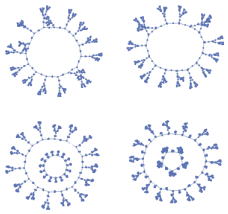
i.

(t) Regla 26



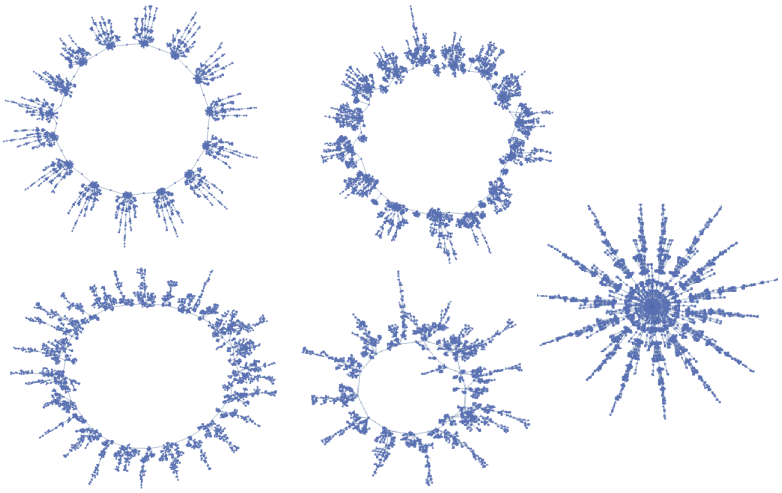
i.

(u) Regla 27



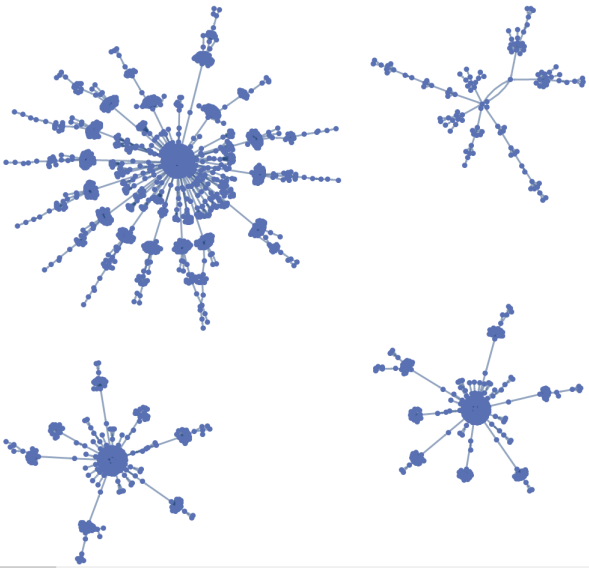
i.

(v) Regla 31



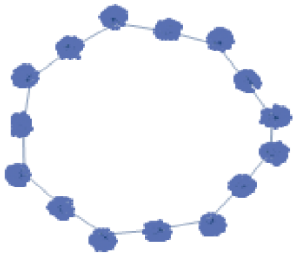
i.

(w) Regla 33



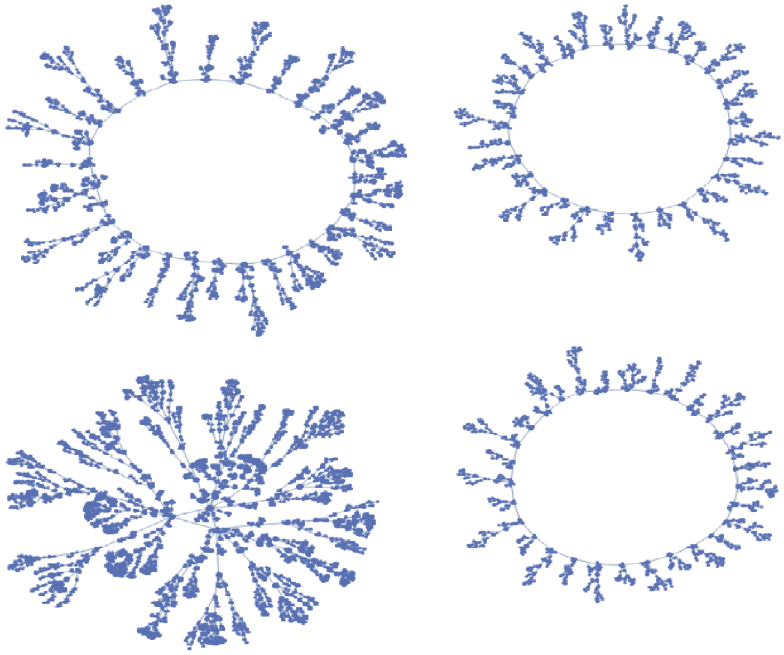
i.

(x) Regla 34



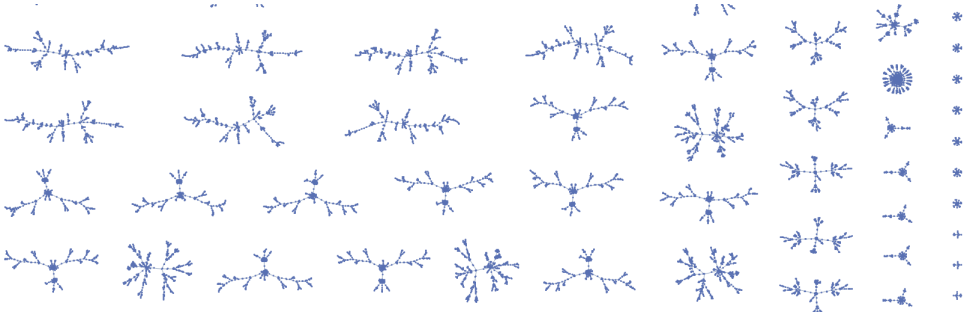
i.

(y) Regla 35



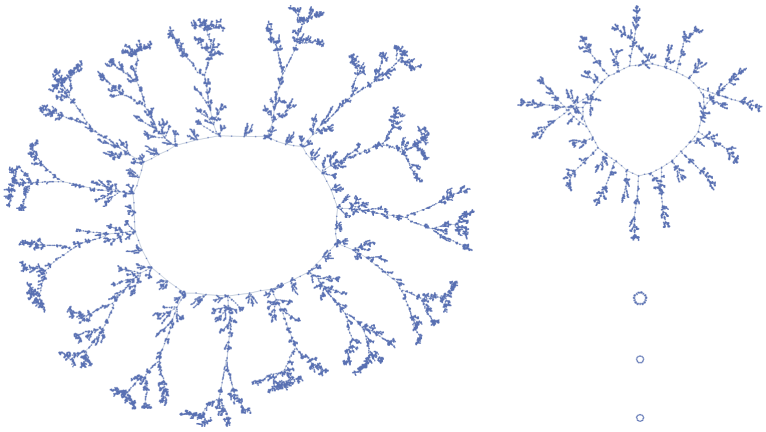
i.

(z) Regla 37



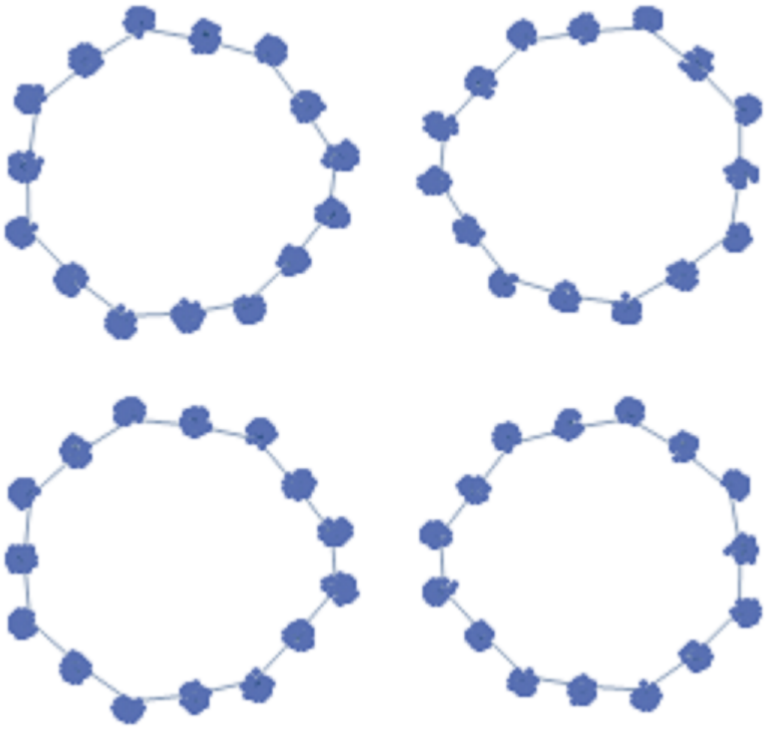
i.

() Regla 41



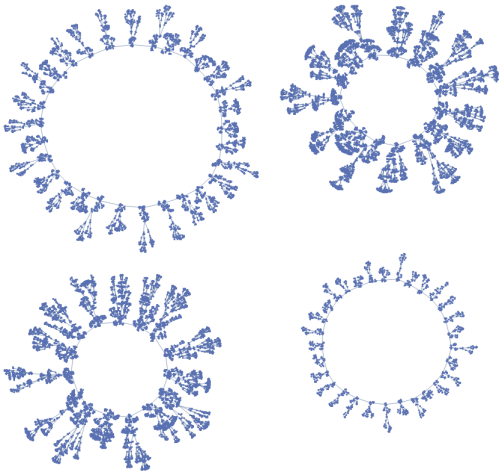
i.

() Regla 42



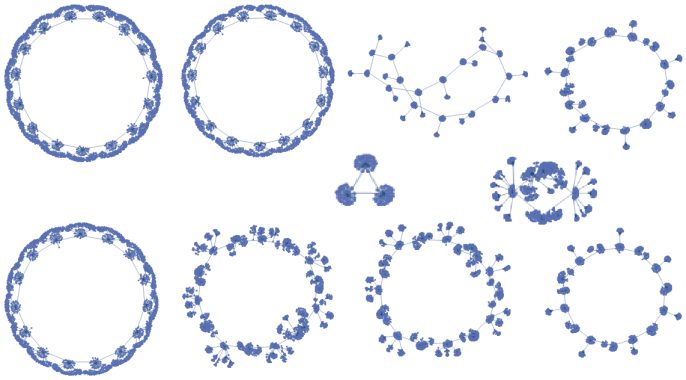
i.

() Regla 43



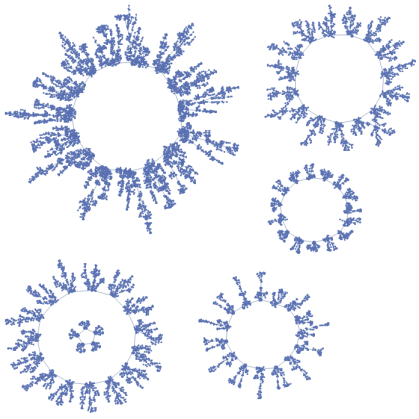
i.

() Regla 46



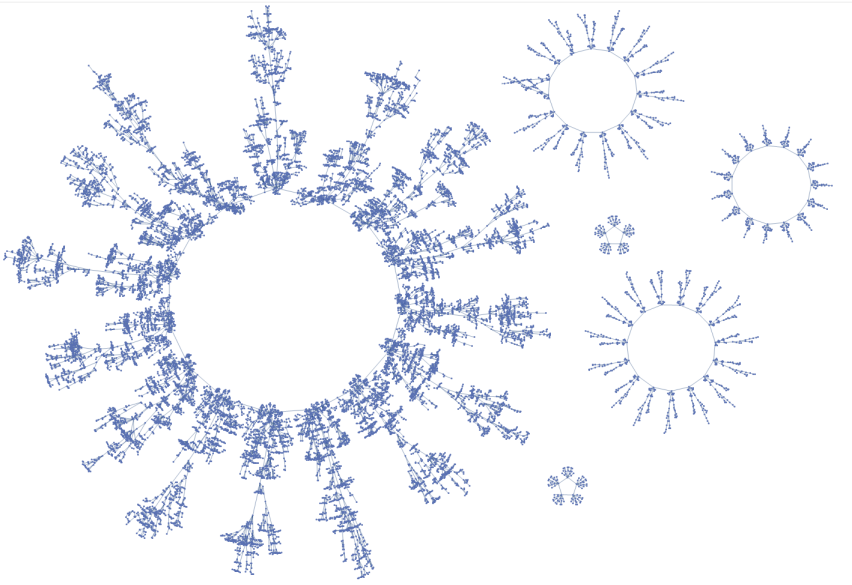
i.

() Regla 56



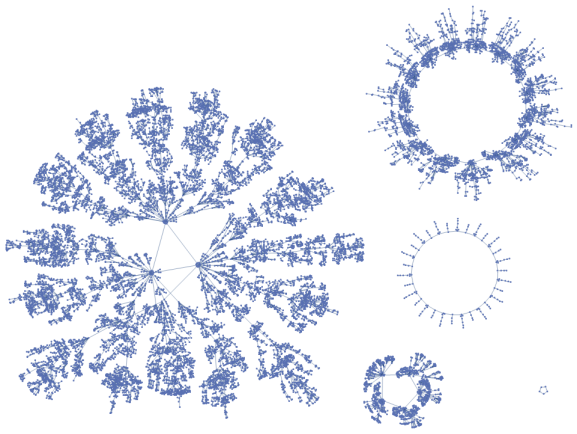
i.

() Regla 57

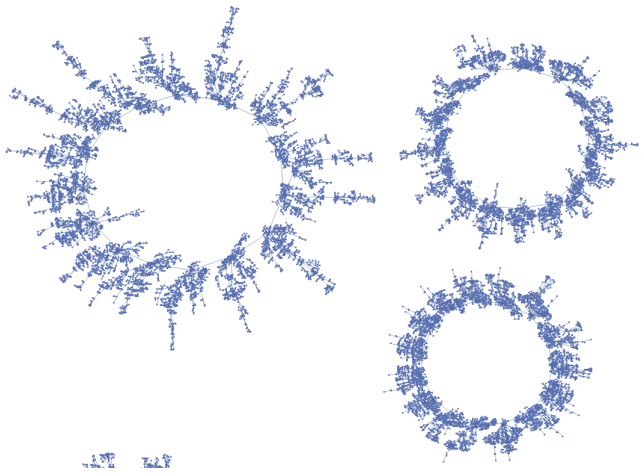


i.

() Regla 58

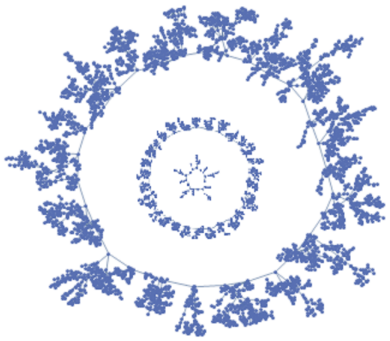


i.



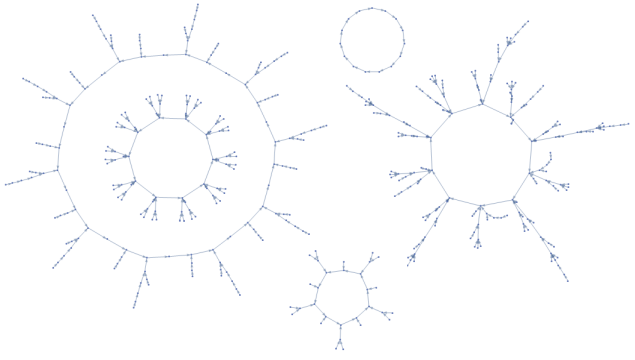
ii.

() Regla 74



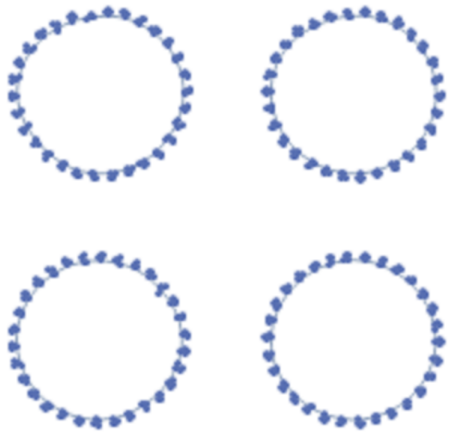
i.

() Regla 88



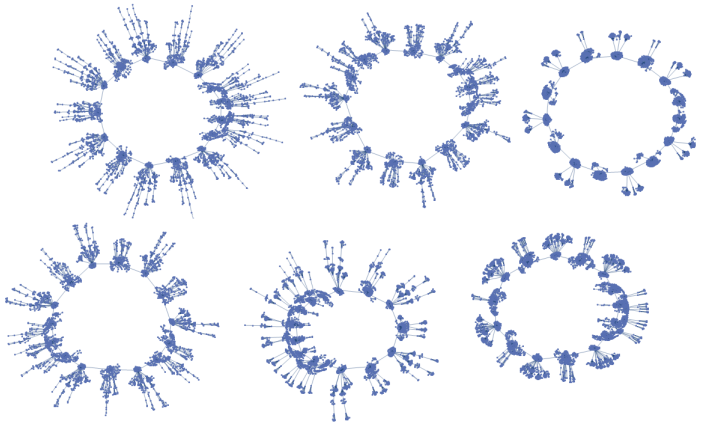
i.

() Regla 105



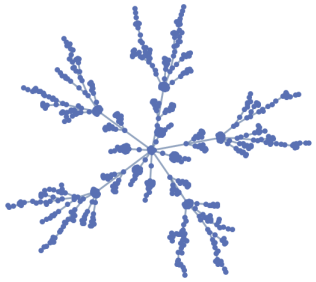
i.

() Regla 130



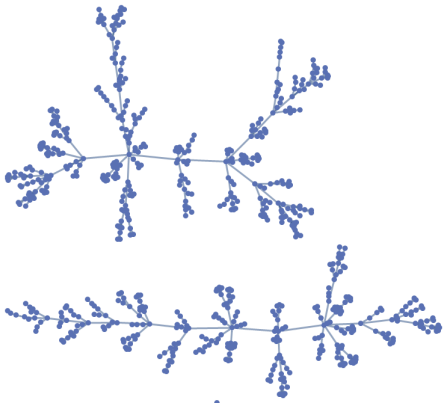
i.

() Regla 133



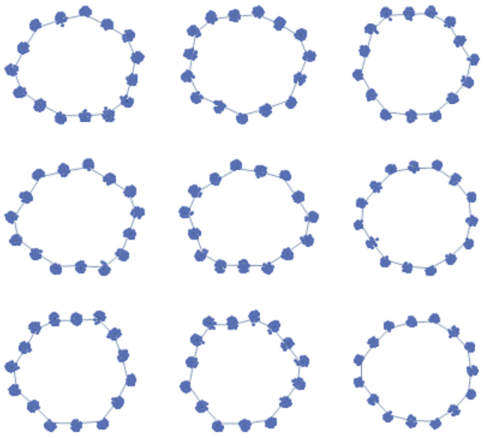
i.

() Regla 134



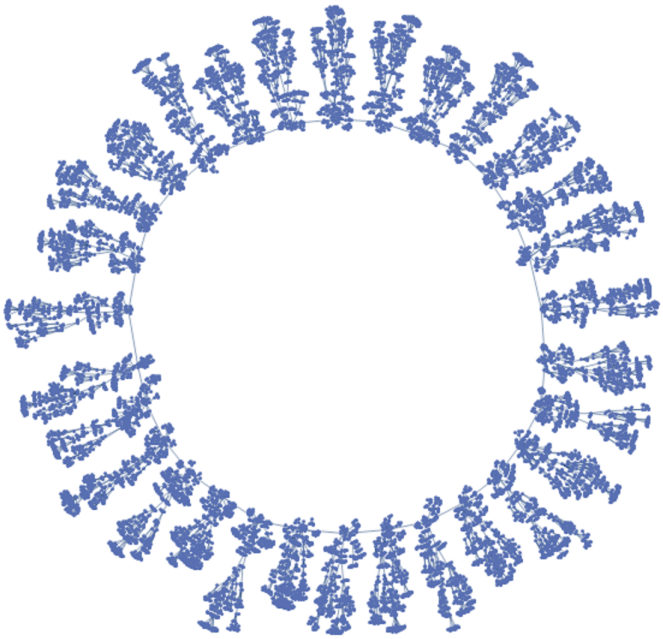
i.

() Regla 138



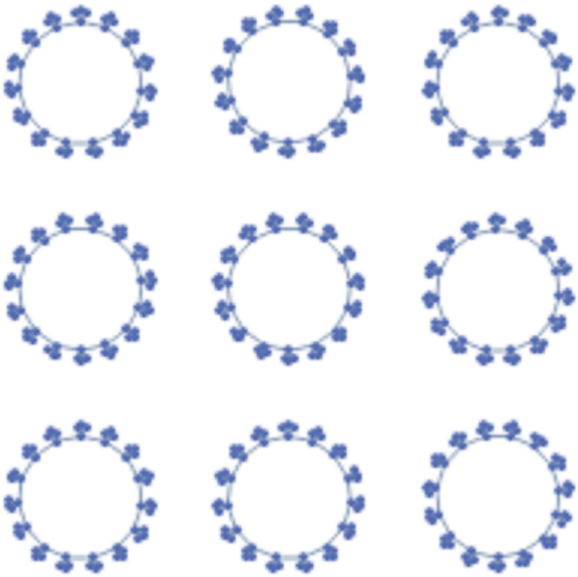
i.

() Regla 142



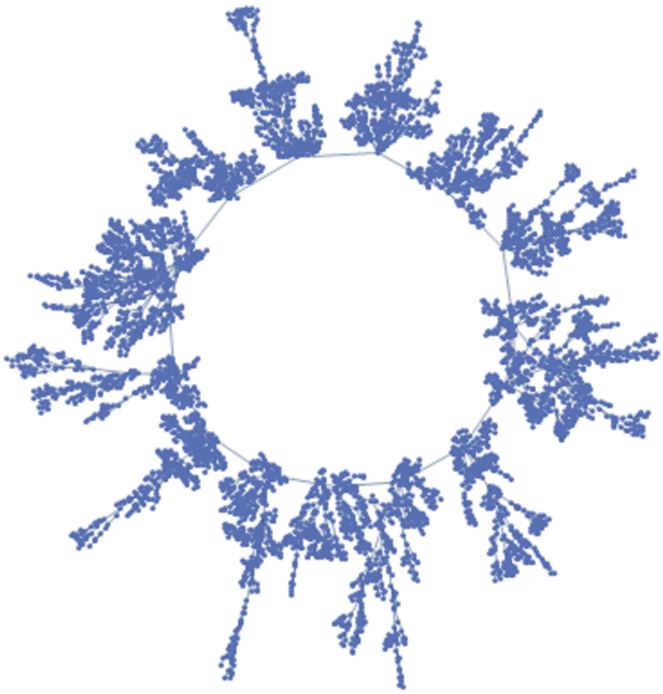
i.

() Regla 150



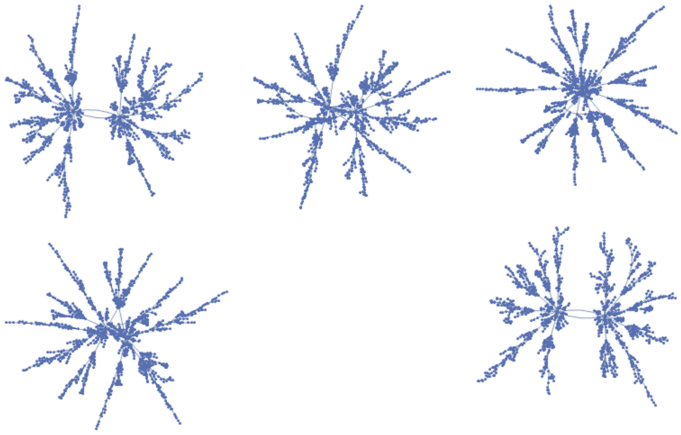
i.

() Regla 152



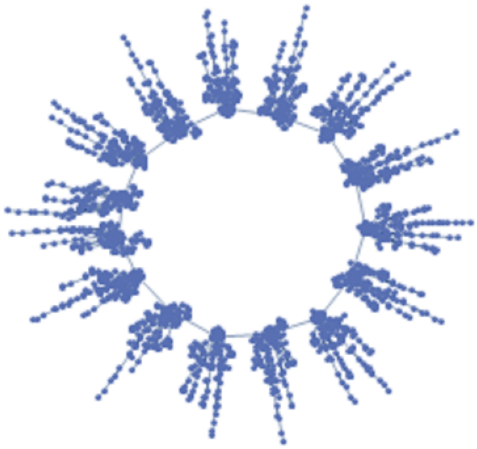
i.

() Regla 156



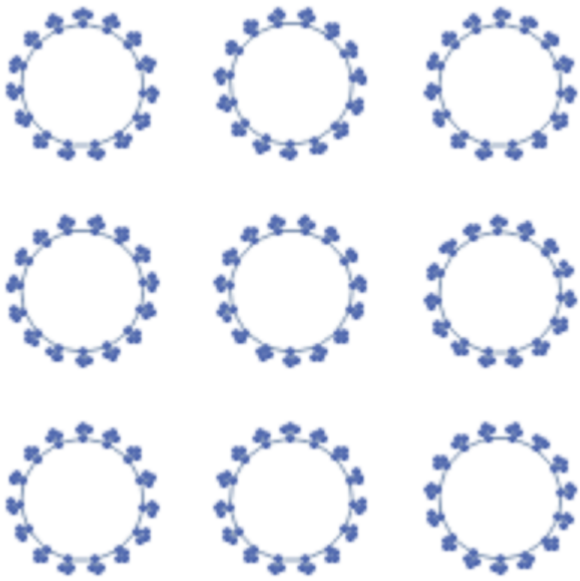
i.

() Regla 162



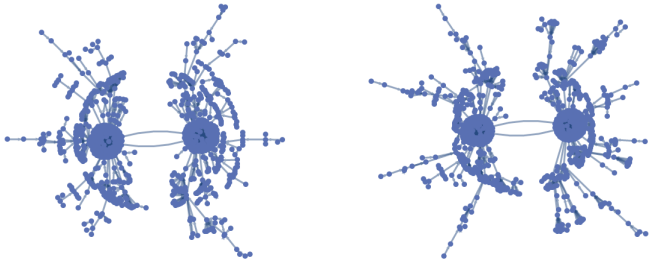
i.

() Regla 170



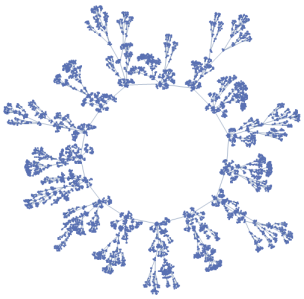
i.

() Regla 178



i.

() Regla 184



i.

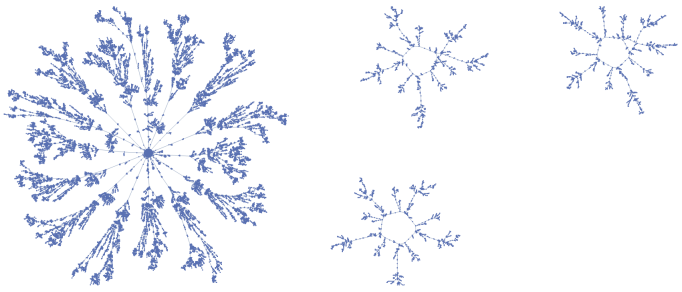
5.4.3 Clase III - Complejos

1. Descripción

   (a) Este tipo de autómatas presentan diversos atractores, los cuale spresentan comportamientos estáticos o cíclicos, llegando a cada uno de ellos con estados diferentes lo que al final produce cierto tipo de patrones combinados.

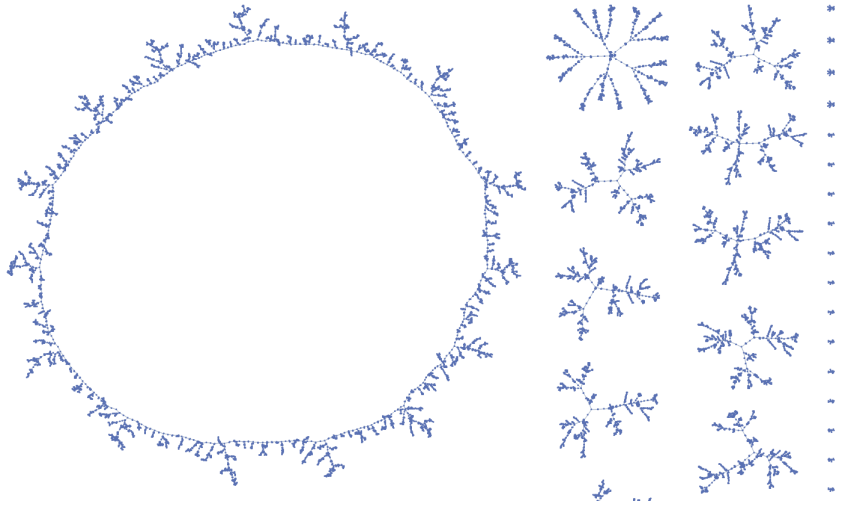
2. Reglas

   (a) Regla 22



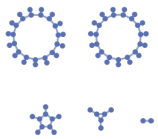
i.

   (b) Regla 54



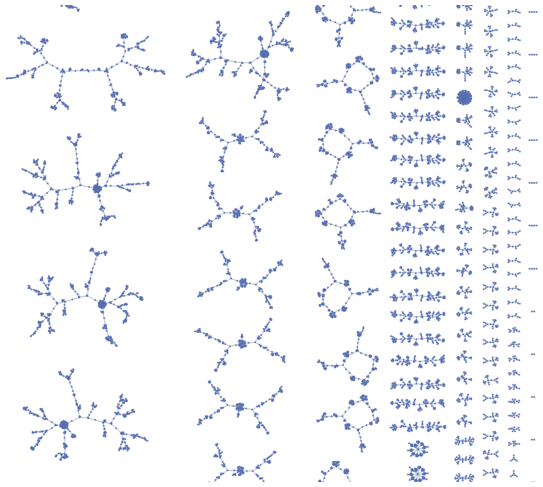
i.

(c) Regla 60



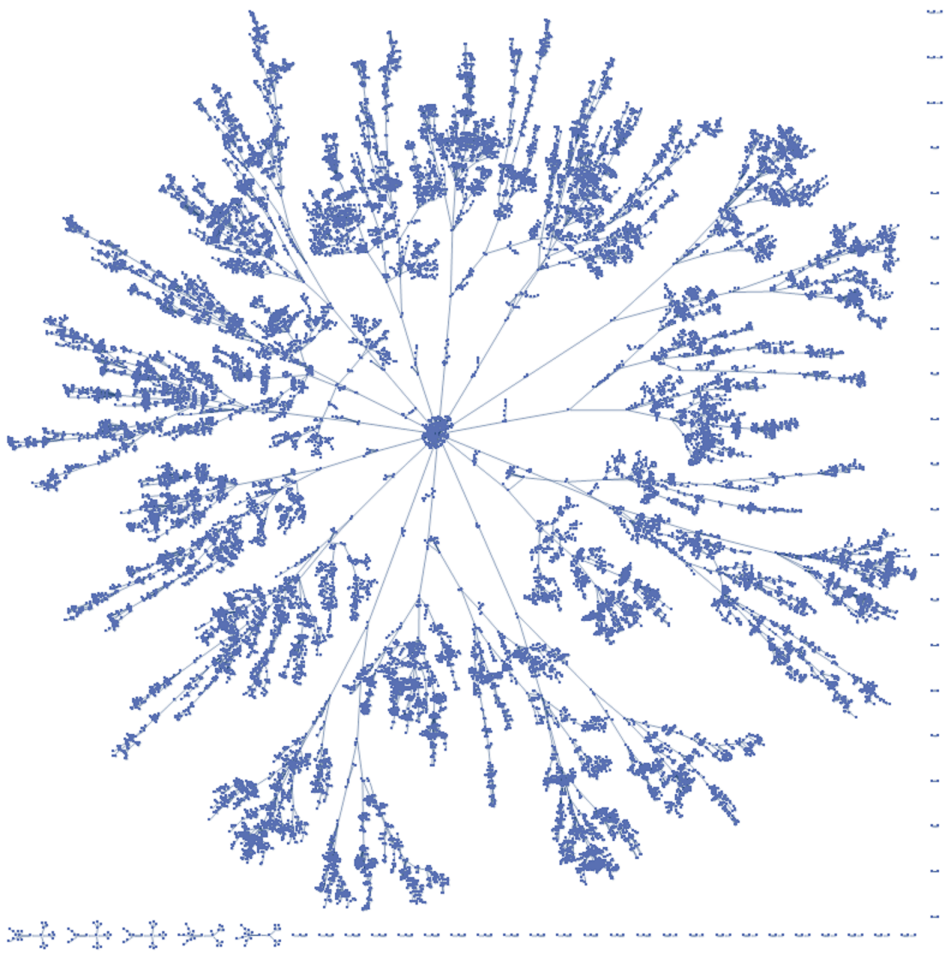
i.

(d) Regla 73



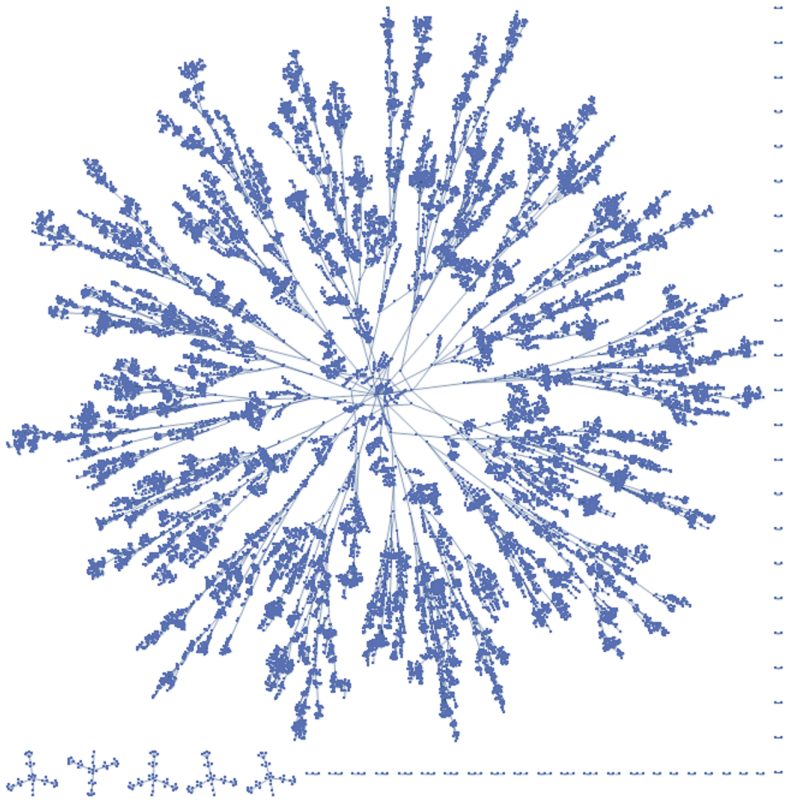
i. }

(e) Regla 146



i.

(f) Regla 161

i. 

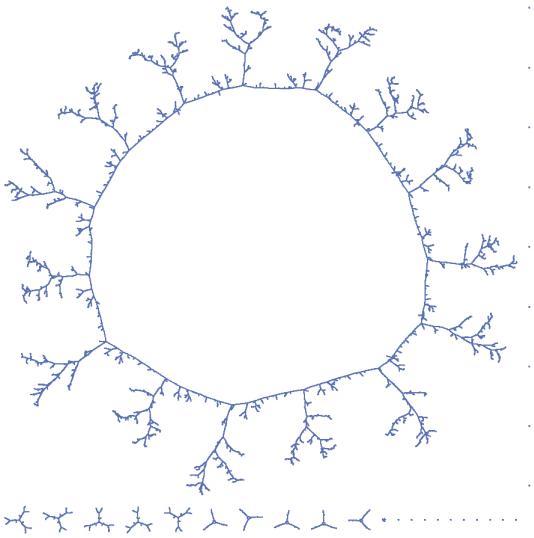
5.4.4 Clase IV - Caóticos

1. Descripción

   (a) Este tipo de autómatas celulares presenta combinaciones de los atractores de autómatas celulares clase I y clase II pero a su vez tiene ciclos muy muy largos lo que genera comportamientos erráticos pero con secciones en aparente orden, suelen tener raman tupdias cortas y girar en torno a un ciclo.

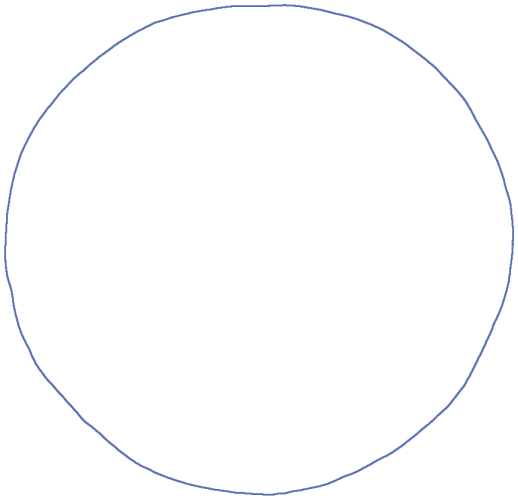
2. Reglas

   (a) Regla 30



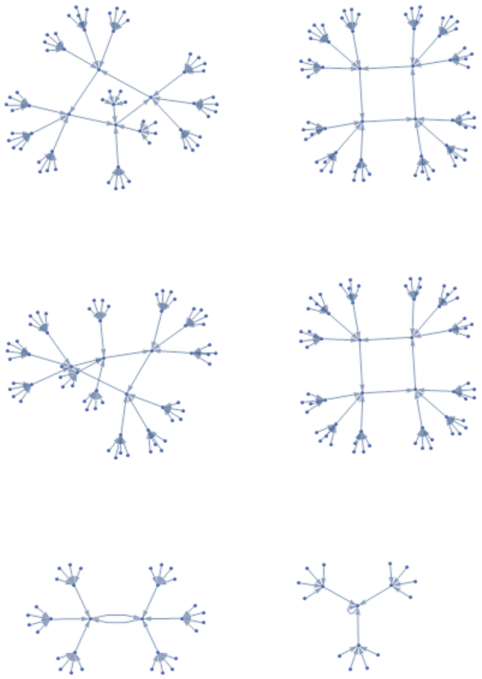
i.

(b) Regla 45



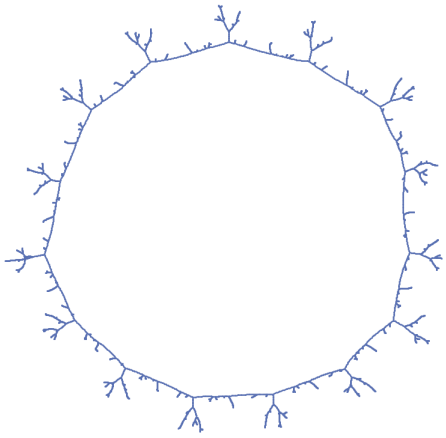
i.

(c) Regla 90

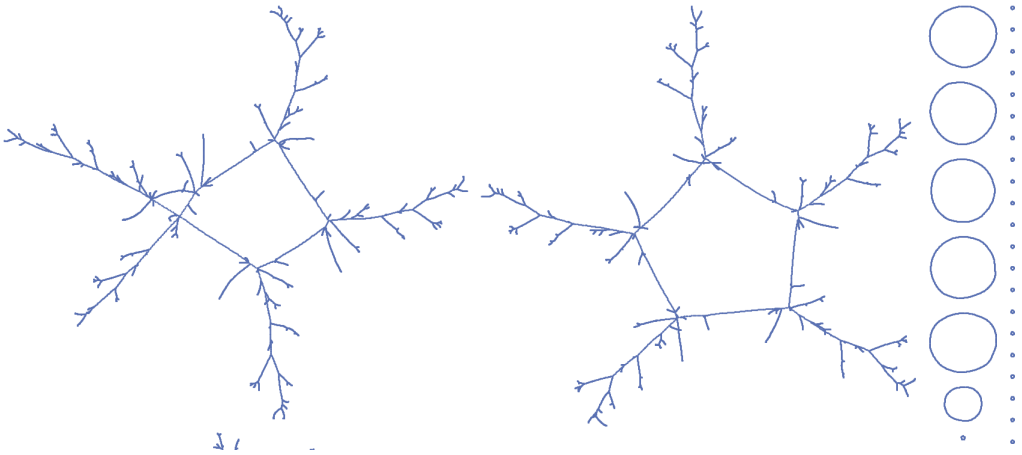


i.

(d) Regla 106

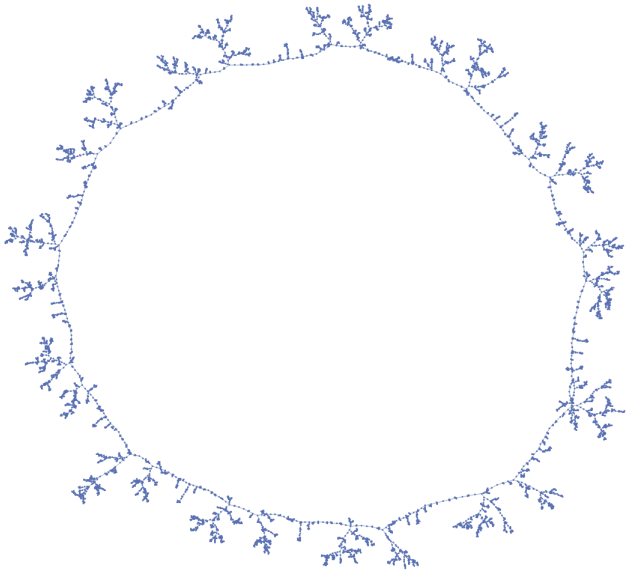


i.



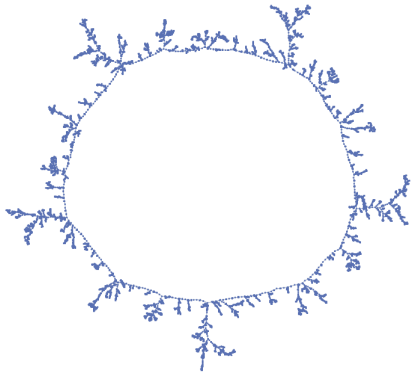
ii.

(e) Regla 110



i.

(f) Regla 137



i.

6 Atractor de autómata celular de 2 dimensiones

6.1 Descripción

Este programa permite calcular los valores decimales de producción para un espacio cuadrado de nxn donde se puede aplicar cualquier regla de la vida, siendo en este caso la regla 2333 de Game of Life y la regla 7722 de Diffusion Rule, pudiendo calcularse en espacios nxn planos o nxn toroidales.

6.2 Pruebas de Funcionamiento

Figura 26: Ejecución del programa

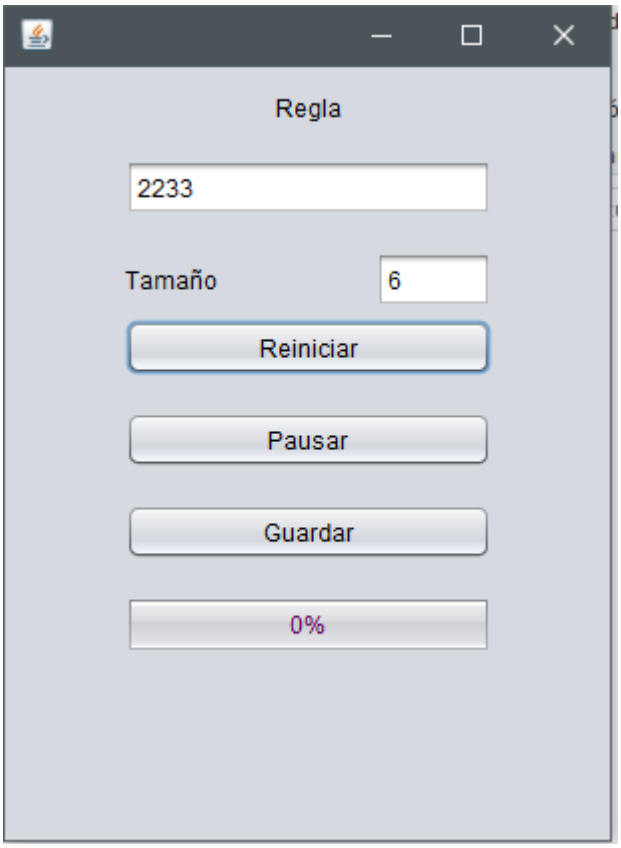
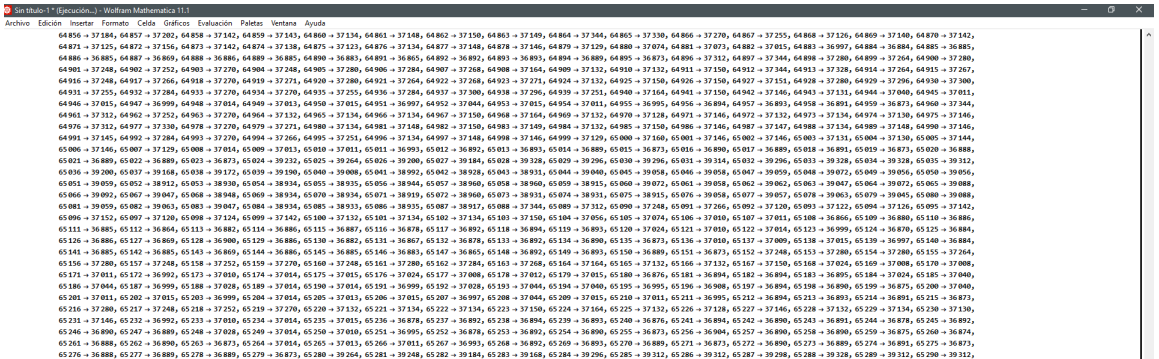


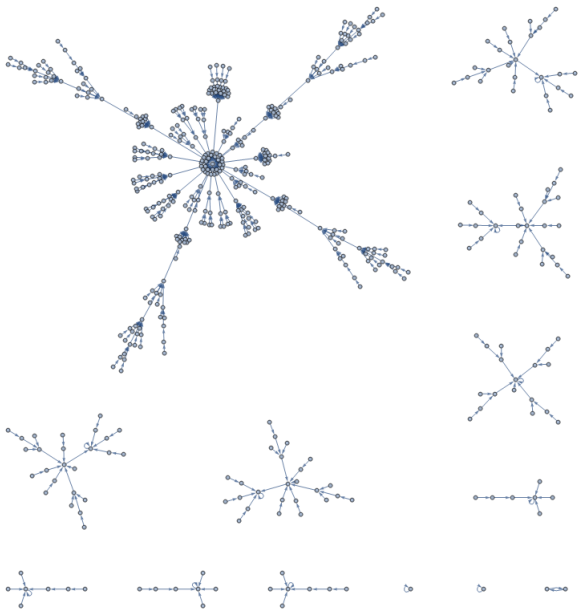
Figura 27: Archivo generado

Figura 29: Procesado en Mathematica

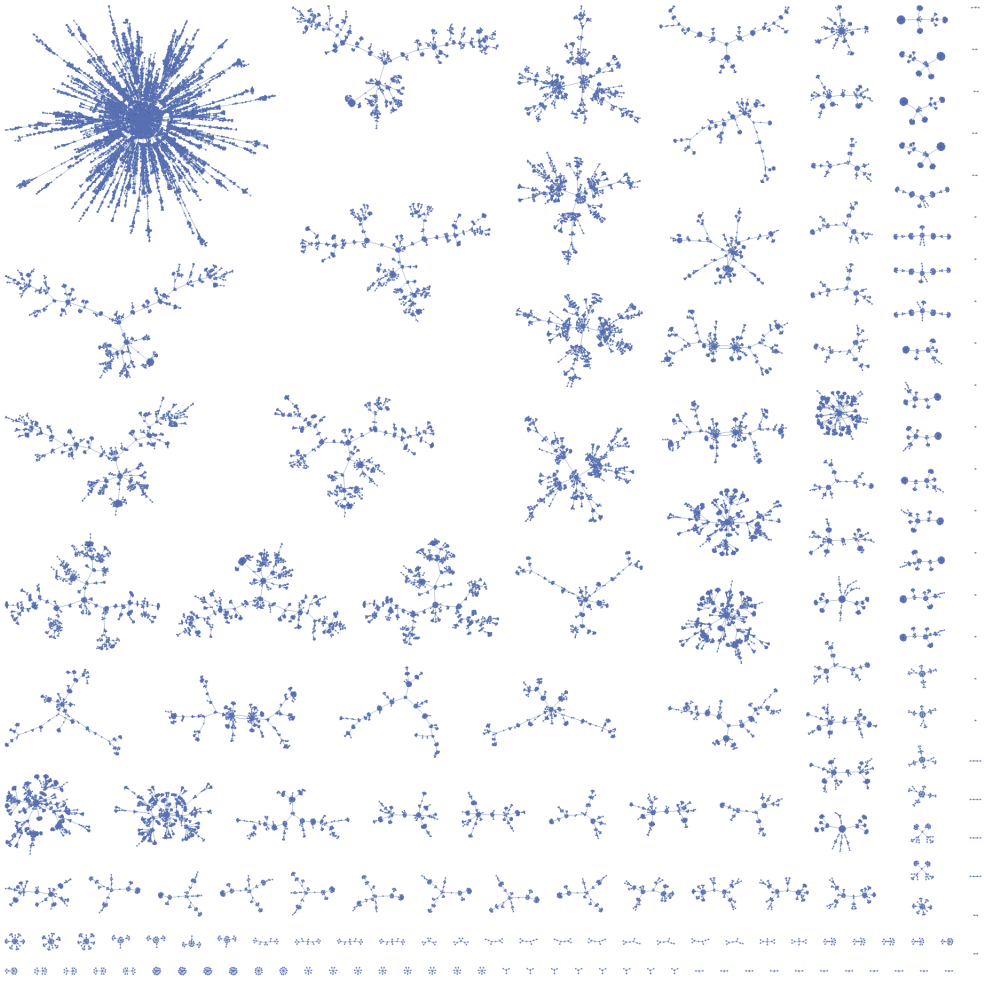


6.2.1 Atractores Generados

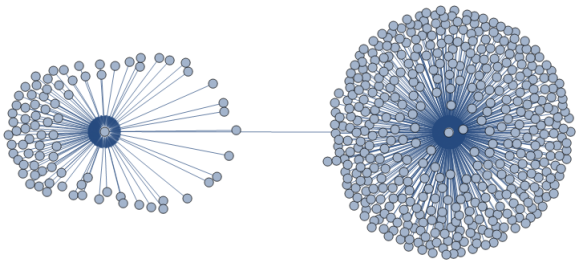
- Regla 2333 Game of Life de 3x3 plano



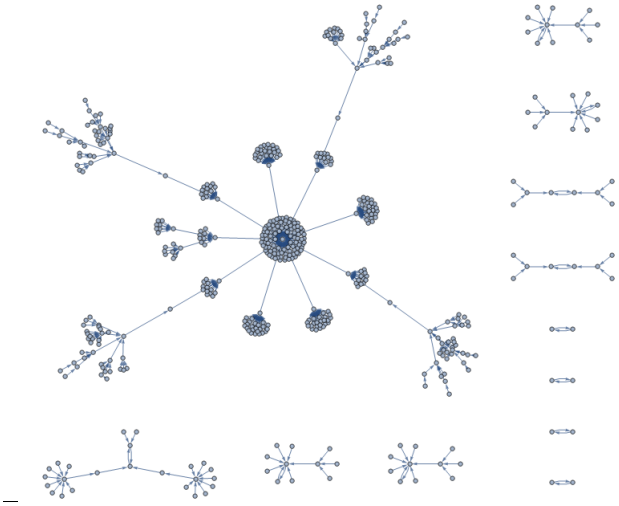
- Regla 2333 Game of Life de 4x4 plano



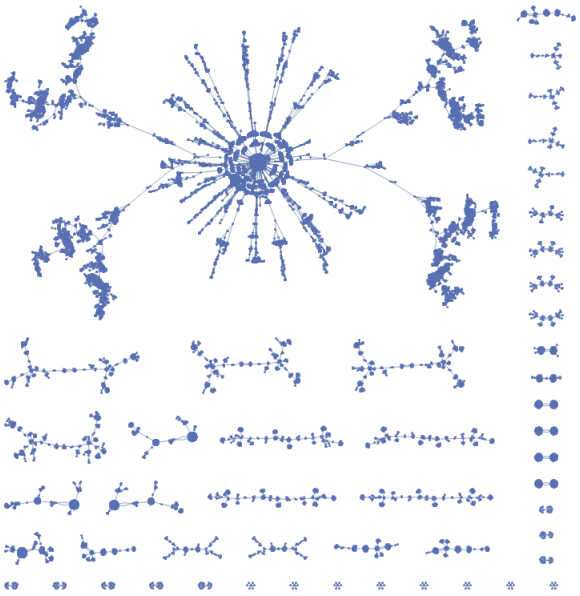
- Regla 2333 Game of Life de 3x3 Tororide



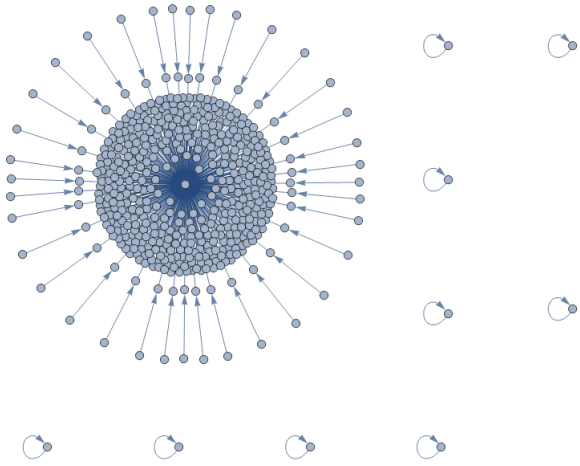
- Regla de difusión 7722 de 3x3 plano



- Regla de Difusión 7722 de 4x4 plano



- Regla de difusión 7722 de 3x3 Toroidal



_

6.3 Código Fuente

6.3.1 Clase Vida | Tororide

```
        /*  * To change this license header, choose License Headers in Project P
* To change this template file, choose Tools | Templates
* and open the template in the editor.  */ package logica;
        /**  *  * @author Maku  */
        import java.io.*; import java.util.*;
        public class Vida
        {
        public Vida()   {         }

        public int[] binario(int r, int tam)  //entero a binario
        {
        int[] b = new int[tam];
        String bin = Integer.toBinaryString(r);
        while(bin.length() < tam)
        {
        bin = "0" + bin;
        }

        //  System.out.println("s" + bin);
        for(int i = 0; i < tam; i++)
        {
        b[i] = Character.getNumericValue(bin.charAt(i));
        }
        return b;
        }
```

```
public int[] existe (int[] an,  int tam, int[] regla)
{
int vec, k = 0;
int[][] m  = new int[tam][tam];
int[][] aux = new int[tam][tam];
for(int i = 0; i < tam; i++)
{
for(int  j = 0; j < tam; j++)
{
m[i][j] = an[k]; //pasar el anillo a matriz
k++;
}
}
for(int i = 0; i < tam; i++) //analizar vida
{
for(int j = 0; j < tam; j++)
{
vec = 0;
if(i == 0 &&  j  >=  0)
{
if(i == 0 && j == 0)
{
vec += m[tam-1][tam-1] + m[tam-1][j] + m[tam-1][j+1]; //las de arriba
vec += m[i][tam-1]                    + m[i][j+1]; //las de al lado
vec += m[i+1][tam-1] + m[i+1][j] + m[i+1][j+1]; //las de abajo
}
{
if(i == 0 && j == (tam-1))
{
vec += m[tam-1][j-1] + m[tam-1][j] + m[tam-1][0]; //las de arriba
vec +=  m[i][j-1]                           + m[i][0]; //las de al lado
vec += m[i+1][j-1] + m[i+1][j] +              m[i+1][0]; //las de abajo
}
else
{
vec += m[tam-1][j-1] + m[tam-1][j] + m[tam-1][j+1]; //las de arriba
vec += m[i][j-1]                           + m[i][j+1]; //las de al lado
vec += m[i+1][j-1] + m[i+1][j] + m[i+1][j+1]; //las de abajo
}

}
}
else
{
if(i == tam-1 && j >= 0 )    //esquians inferiores a toroide
{
```

```
        if(i == tam-1 && j == 0)
        {
        vec += m[i-1][tam-1] + m[i-1][j] + m[i-1][j+1]; //las de arriba
        vec += m[i][tam-1]                + m[i][j+1]; //las de al lado
        vec += m[0][tam-1] + m[0][j] + m[0][j+1]; //las de abajo
        }
        else
        {
        if(i == j && j == tam -1)
        {
        vec += m[i-1][j-1] + m[i-1][j] + m[i-1][0]; //las de arriba
        vec += m[i][j-1]                          + m[i][0]; //las de al lado
        vec += m[0][j-1] + m[0][j] + m[0][0]; //las de abajo
        }
        {
        vec += m[i-1][j-1] + m[i-1][j] + m[i-1][j+1]; //las de arriba
        vec += m[i][j-1]                            + m[i][j+1]; //las de al lado
        vec += m[0][j-1] + m[0][j] + m[0][j+1]; //las de abajo
        }
}
else
{
        if(j == 0)
        {
        vec += m[i-1][tam-1] + m[i-1][j] + m[i-1][j+1]; //las de arriba
        vec += m[i][tam-1]
+ m[i][j+1]; //las de al lado
        vec += m[i+1][tam-1] + m[i+1][j] + m[i+1][j+1]; //las de abajo
        }
        else
        {
        if(j == tam - 1)
        {
        vec += m[i-1][j-1] + m[i-1][j] + m[i-1][0]; //las de arriba
        vec += m[i][j-1]
+ m[i][0]; //las de al lado
        vec += m[i+1][j-1] + m[i+1][j] + m[i+1][0]; //las de abajo

        }
        else
        {
        vec += m[i-1][j-1] + m[i-1][j] + m[i-1][j+1]; //las de arriba
        vec += m[i][j-1]
+ m[i][j+1]; //las de al lado
        vec += m[i+1][j-1] + m[i+1][j] + m[i+1][j+1]; //las de abajo
        }
```

```
        }

//  System.out.println("i = "+i+"  j =
        " + j );
        }
        }
        if(m[i][j] == 1) //[fila][columna]  Si está viva entonces:
        {
        if(vec >= regla[0] && vec <= regla[1]) //
{
aux[i][j] = 1;
        }
        else
        {
        aux[i][j] = 0;
        }
        }

        else //si está muerta
        {
if(vec >= regla[2] && vec <= regla[3])
        {
        aux[i][j] = 1;
        }
        else
        {
        aux[i][j] = 0;
        }
        } //else muerta
        }//for columa

        } //for fila analizar vida
        k = 0;
        for(int i = 0; i < tam; i++)
        {
        for(int  j = 0; j < tam; j++)
        {
        an[k] = aux[i][j]; //pasar el anillo a matriz
        k++;
        }
        }
        return an;
        }//main
        public int decimal(int[] m)
        {
        String bin = "";
```

```
int dec;
for(int i = 0; i < m.length; i++)
{
//convierto binario a decimal
return dec;
}
}
}//clase
```

6.3.2 Clase Vida | Plano

```
 public int[] existe (int[] an,  int tam, int[] regla)
{
int vec, k = 0;
int[][] m  = new int[tam+2][tam+2];
int[][] aux = new int[tam+2][tam+2];
for(int i = 1; i <= tam; i++)
{
for(int  j = 1; j <= tam; j++)
{
m[i][j] = an[k]; //pasar el anillo a matriz
k++;
}
}
for(int i = 1; i <= tam; i++) //analizar vida
{
for(int j = 1; j <= tam; j++)
{
vec = 0;
vec += m[i-1][j-1] + m[i-1][j] + m[i-1][j+1]; //las de arriba
vec += m[i][j-1]                              + m[i][j+1]; //las de al lado
vec += m[i+1][j-1] + m[i+1][j] + m[i+1][j+1]; //las de abajo
if(m[i][j] == 1) //[fila][columna]  Si está viva entonces:
{
if(vec >= regla[0] && vec <= regla[1]) //
{
aux[i][j] = 1;
}
else
{
aux[i][j] = 0;
}
}
else //si está muerta
{
if(vec >= regla[2] && vec <= regla[3])
```

```
{
aux[i][j] = 1;
}

else
{
aux[i][j] = 0;

}
}//else muerta
}//for columa
}//for fila analizar vida
k = 0;
for(int i = 1; i <= tam; i++)

{
for(int  j = 1; j <= tam; j++)
{
an[k] = aux[i][j]; //pasar el anillo a matriz
k++;
}
}
                            return an;
      }//main
```