TEORíA DEL CAMPO PROMEDIO EN AUTÓMATAS CELULARES SIMILARES A "THE GAME OF LIFE"

Por
Genaro Juárez Martínez

PARA CUMPLIR LOS REQUERIMIENTOS NECESARIOS
PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS
DEL
CENTRO DE INVESTIGACIONES Y ESTUDIOS AVANZADOS DEL I.P.N
DISTRITO FEDERAL, MÉXICO
OCTUBRE 2000

CENTRO DE INVESTIGACIONES Y ESTUDIOS AVANZADOS DEL I.P.N
DEPARTAMENTO DE
INGENIERíA ELÉCTRICA

El comité certifica que han revisado el trabajo realizado y recomiendan a la Sección Computación aceptar esta tesis titulada "**Teoría del Campo Promedio en Autómatas Celulares Similares a "The Game of Life"**" por **Genaro Juárez Martínez**, cumpliendo los requerimientos para obtener el grado de **Maestro en Ciencias**.

Fecha: Octubre 2000

Asesor Externo: ______________________
Harold V. McIntosh

Asesor Interno: ______________________
Sergio V. Chapa Vergara

Comité Revisor: ______________________
Feliú Sagols Troncoso

____
Xiaoou Li

CENTRO DE INVESTIGACIONES Y ESTUDIOS AVANZADOS DEL I.P.N

Fecha: **Octubre 2000**

Autor: **Genaro Juárez Martínez**

Título: **Teoría del Campo Promedio en Autómatas Celulares Similares a "The Game of Life"**

Depto.: **Ingeniería Eléctrica**

Grado: **M. en C.** Convocatoria: **Octubre** Año: **2000**

El permiso es concedido a el Centro de Investigaciones y Estudios Avanzados del I.P.N para reproducir y circular copias sin fines comerciales a juicio de los arriba mencionados, para requerimientos individuales o de instituciones.

Firma del Autor

A mis padres

Samuel Juárez y Maria Luisa Martínez Yonca

y a mis hermanos

Erik Juárez Martínez, Fabian Juárez Martínez y

Georgina Verónica Juárez Martínez.

Contenido

Agradecimientos

En especial a mi papá Samuel Juárez por apoyarme en toda mi formación y sus valiosos consejos que me han servido para alcanzar cada una de las metas obtenidas, principalmente por enseñarme a ser constante en el trabajo, aceptar las responsabilidades y seguir adelante a pesar de las adversidades que pone la vida. A mi mamá Luisa Martínez Yonca por su cariño y amor, por consertime mucho y ayudarme en los momentos difíciles que he pasado. A mis hermanos Erik Juárez Martínez, Fabian Juárez Martínez y Georgina Verónica Juárez Martínez por todas las ondas en que hemos estado juntos y por ayudarme a que los momentos difíciles sean mas agradables. A mi gran amiga Amabely que me ha ayudado mucho con sus consejos y su forma de ser, su gran ayuda en los problemas por los que pasé y los momentos agradables que me ha hecho pasar.

Un agradecimiento muy especial a mi mejor maestro que he tenido en toda mi vida Harold V. McIntosh, por el todo el apoyo que me a dado en cada una de las tesis que se han elaborado, así como en cada uno de los trabajos que he realizado, por enseñarme lo que es realmente la investigación, por la enorme paciencia que me a tenido, por cada una de las seciones que eran siempre de primer nivel, por su amistad que me a brindado y que gracias a usted tuve los mejores momentos que he vivido cada verano en el Instituto de Ciencias, por enseñarme a aprender. Una persona muy especial a otro gran maestro Sergio V. Chapa Vergara por tener confianza en mi trabajo, por su gran apoyo en los momentos difíciles desde que lo conocí en la licenciatura, por haberme dado la oportunidad de conocer a McIntosh, por tenerme mucha paciencia en todo este tiempo, por sus enseñanzas tanto académicas como de la vida.

A Seck por las seciones que hemos tenido en los autómatas celulares desde 1996 en aquellas desveladas memoriables, a René por su ayuda en la revición de esta tesis, su amistad y sobre todo aquella ida a la Roca inborrable en toda mi vida, a Verónica, Clarissa y Lupita que siempre me han hechado porras y me han dado muy buenos consejos. A mi tío Ernesto Lopez por sus valisos consejos, su ayuda siempre disponible y su gran amistad y confianza, a mis abuelitas Guadalupe Lopez y Sofia Yonca por consentirme demasiado, a mis abuelitos Samuel Lopez y Genaro Lopez que apesar de ya no estar conmigo siempre tuvieron confianza en mi, a mi mejor amigo que siempre confió en mi Serafín.

Al maestro Cedillo que siempre nos ha apoyado y ayudado en la Sección cuando hemos necesitado algo y aunque no lo necesitemos, a Favio por ayudarnos con las nuevas técnologias mac-queras, sobre todo con sus consejos y formas de ver la programación en objetos, a las secretarias de la sección de computación que me ayudaron en el transcurso de mi estancia en la maestría, a otros maestros que me han ayudado como Guillermo Morales, Maximino, Pedrito y tantas otras personas como el oso, al Ing. Vadillo, Alfonso, Silvana, Trueba, Alan, Valadéz, Anguel, Noé, Jei, al Dr. Alex, al Dr. Oscar y toda la bola, a Clavel, Maricarmen y otras más que no recuerdo en este momento. A CONACyT por apoyarme con la beca de maestría y al CINVESTAV en general.

Distrito Federal, México — Genaro Juárez Martínez
Octubre 25, 2000

Resumen

Los autómatas celulares son sistemas dinámicos discretos que evolucionan a través del tiempo, pueden evolucionar en varias dimensiones, producir réplicas de sí mismos y soportar comportamientos complejos. En la presente tesis se realiza una caracterización probabilística del autómata celular en dos dimensiones conocido como "*The Game of Life*" propuesto por John Horton Conway, aplicando la teoría del campo promedio para explicar el comportamiento de los estados dentro del espacio de evoluciones a través del tiempo. Este tipo de análisis en particular no ha sido reportado por otros autores, aunque se han realizado investigaciones para establecer una clasificación de los autómatas celulares con la teoría del campo promedio y la teoría de estructura local realizada por Howard Andrew Gutowitz y una clasificación de los autómatas celulares en altas dimensiones que presentan comportamientos colectivos no triviales empleando la teoría del campo promedio y técnicas de Monte Carlo realizada por Hugues Chaté y Paul Manneville. Las investigaciones realizadas sobre *Life* en la actualidad carecen de una explicación rigurosa y formal, por esa razón se han realizado aproximaciones de tipo estadístico y probabilístco para poder explicar parte de la complejidad del autómata celular "*The Game of Life*".

El análisis se realiza en el modelo original "*The game of Life*" explicando el comportamiento de los estados en el espacio de evoluciones a través del tiempo, una vez que se identifica el comportamiento del modelo original se hace una extensión para el modelo en tres dimensiones. Un problema importante es saber si existe un autómata celular que sea el sucesor directo de *Life* en tres dimensiones, Carter Bays ha realizado extensas investigaciones para determinar que autómata celular en tres dimensiones es similar a *Life* proponiendo mas de una regla de evolución, sin embargo varios de sus resultados son cuantitativos basados en simulaciones, con el análisis basado en la teoría del campo promedio reducimos este número de reglas y se dan algunas condiciones probabilísticas necesarias para que un autómata celular en tres dimensiones tenga comportamientos similares a *Life*.

Finalmente se presenta el análisis de la *regla 110* estudiada por Mathew Cook, es un autómata celular que evoluciona en una dimensión, éste tiene muchas características similares a *Life* pero su principal diferencia es que no evoluciona en un fondo estable como lo hace *Life*. La *regla 110* es un modelo mas sencillo que *Life* pero puede soportar comportamientos tan complejos como *Life*, se realiza su análisis probabilístico con la teoría del campo promedio encontrando diferencias importantes con respecto a *Life*. Por otra parte se presenta un análisis determinístico con teoría de gráficas desarrollado por Harold V. McIntosh, lo que permite explicar con detalle los comportamientos de este autómata celular utilizando diagramas de de Bruijn extendidos. Una propuesta radica en poder emplear este análisis en *Life* (en autómatas celulares de dos dimensiones), además de mostrar la importancia de este autómata celular en particular con sus características propias y ver que la *regla 110* puede ser visto como un modelo original y plantear ahora si existen autómatas celulares en dos y tres dimensiones que sean similares a la *regla 110*.

Introducción

El origen de los autómatas celulares data de los año 50's con su precursor John von Neumann, donde plantea un modelo matemático que fuera capaz de soportar comportamientos complejos y pudiera reproducirse a sí mismo. Los autómatas celulares han tenido aplicaciones importantes como la modelación de la reacción química de Zhabotinsky, incendios forestales, codificación del DNA, encriptación de datos, computación en paralelo, simulación del proceso electoral, comportamiento colectivo de hormigas, modelación del flujo de automoviles, simulación del modelo depredador-presa, simulación de epidemias, simulación de modelos matemáticos, simulación de máquinas de Turing y modelación del sistema nervioso humano, entre otros.

El modelo original de von Neumann es un autómata celular que evoluciona en dos dimensiones, con veintinueve estados en su alfabeto y evoluciona con la vecindad de von Neumann, en [54] demostró la existencia de un constructor universal con su modelo. Una de las finalidades que planteó von Neumann para la teoría de autómatas celulares, es que dicho sistema pueda simular el sistema nevioso humano; además de poder obtener sistemas que a partir de componentes no confiables puedan producir sistemas confiables [51].

En la teoría de la computación la máquina de Turing [40] es el sistema de computación mas general conocido; los autómatas celulares tienen al menos el mismo poder computacional para poder realizar computación. En [10] se ha demostrado que el autómata celular *Life* puede hacer computación universal, en el estudio de los autómatas celulares ya se ha logrado simular una máquina de Turing en un autómata celular, pero lo interesante sería ver si una máquina de Turing puede simular un autómata celular.

A finales de los años 60's Martin Gardner publica en su columna mensual del *Scientific American* [18] el autómata celular en dos dimensiones conocido como "*The Game of Life*" nombrado así por su precursor John Horton Conway, este autómata tomó mucho interés ya que su sistema es muy sencillo de describir. A partir de entonces se han desarrollado diversos análisis para tratar de explicar de manera formal dichos comportamientos, sin embargo el estudio de los autómatas celulares en dos dimensiones no es fácil de generalizar ni mucho menos de clasificar. A pesar de este problema a mediados de los años 80's Carter Bays presenta resultados importantes [2] sobre sus investigaciones para tratar de encontrar una regla de evolución similar a *Life* en tres dimensiones, encontrando mas de una regla de evolución que presentan algunos comportamientos que existen en *Life*.

El problema de analizar reglas de evolución en tres dimensiones crece exponencialmente, pues el número de reglas de evolución es grandísimo. A medidados de los años 90's Mathew Cook presenta la *regla 110* [47] un autómata celular en una dimensión que tiene comportamientos similares a los de *Life*, pero su principal diferencia es que evoluciona en un fondo periódico en su espacio de evoluciones y *Life* no. Cook muestra algunos de sus resultados obtenidos de como la *regla 110* puede llegar a realizar computación universal, aunque el estudio que se realiza sobre esta regla de evolución es sobre su similitud que existe con *Life* y no demostrar si la *regla 110* puede o no puede hacer computación universal.

Las investigaciones mas importantes realizadas para tratar de caracterizar reglas de evolución en autómatas celulares de una dimensión en general, se han llevado a cabo de manera estadística aplicando la teoría de estructura local realizado por Howard Andrew Gutowitz [22], descartando la clasificación de Stephen Wolfram [57] que no es del todo rigurosa, dado que la clasificación de Wolfram está basada en la observación de las evoluciones a través del tiempo y no bajo alguna teoría que le de sustento a esa clasificación, además mostramos algunos contra ejemplos de dicha clasificación. Otro análisis importante fue clasificar comportamientos colectivos no triviales en autómatas celulares probabilísticos de altas dimensiones aplicando la teoría del campo promedio, lattices acopladas, técnicas de MonteCarlo y diagramas de bifurcaciones, realizado por Hugues Chaté y Paul Manneville en [16].

La teoría del campo promedio se aplica desde un enfoque directo como lo muestra Harold V. McIntosh en [44] y Chaté-Manneville en [16], para determinar el comportamiento de las reglas de evolución similares a *Life*, a través de su curva de probabilidad que representa la densidad de los estados en el espacio de evoluciones a través del tiempo. Establecemos una caracterización estadística con las reglas similares a *Life* en autómatas celulares de una y tres dimensiones, tomando como base el análisis que se hace con *Life*. En el caso de la *regla 110* realizamos su estudio estadístico y algunas proyecciones sencillas en dos dimensiones, ya que dicha regla evoluciona sobre un fondo periódico y *Life* no, de ahí surge la importancia de encontrar autómatas en dos dimensiones con características de la *regla 110* y *Life*, lo que nos dio como resultados hallar reglas de evolución que tienen características estadísticas y fenotípicas similares a *Life* y la *regla 110*. Logrando obtener una caracterización probabilística de las reglas en tres dimensiones que son similares a *Life* y estableciendo una relación fenotípica y probabilística con la *regla 110*, a pesar de contar con poca literatura que nos permita tener una teoría formal en autómatas celulares de dos y tres dimensiones, que en la actualidad no existe.

Entonces el problema es tratar de caracterizar estas reglas de evolución probabilísticamente hablando, que sean similares a *Life* y obtener una mejor aproximación con respecto al comportamiento de los estados en el espacio de evoluciones. Aplicamos la teoría del campo promedio para caracterizar las reglas de evolución en autómatas celulares de una, dos y tres dimensiones en general, de esta manera determinamos el comportamiento de las células a través del tiempo obteniendo su curva de probabilidad que muestra la densidad de los estados con respecto al espacio de evoluciones. Este análisis toma como punto de partida los trabajos realizados por Gutowitz en [22], [23], [24], [26] y [27], donde presenta de manera detallada y formal sus investigaciones en el estudio del comportamiento de los autómatas celulares en una dimensión a través de la teoría del campo promedio [1] y la teoría de estructura local [2], por otra parte se tomaron los estudios de Chaté-Manneville [16] en autómatas celulares probabilísticos de altas dimensiones a través de la teoría del campo promedio y técnicas de Monte Carlo.

Finalmente se describen los programas gráficos desarrollados en esta tesis *ACOSXL21*, *ACOSXSTVN*, *ACOSXSTM*, *ACOSXVN* y *ACOSXM*, para las plataformas *Mac OS X*, *Mac OS X Server*, *OpenStep* y *Windows*, estos programas sirvieron para poder reproducir cada una de las reglas estudiadas, estos programas fueron codificados con el lenguaje orientado a objetos *Objetive-C*, en la actualidad existe poco software que nos permita hacer estudios adecuados, ya que estos programas tienen varias limitaciones como la falta de interactividad con el usuario, la poca manipulación de parámetros, la plataforma en que son desarrollados y el escaso uso de análisis matricial, estadístico, probabilístico, de gráficas, entre otros. Un buen sistema desarrollado para el estudio de los autómatas celulares en una dimensión y que además cuenta con muchas

[1] Mean Field Theory.
[2] Local Structure Theory.

de las herramientas arriba mencionadas son el sistema NXLCAU [3] y LCAU [4] desarrollados por McIntosh, estos programas pueden ser adquiridos de manera libre en *http://delta.cs.cinvestav.mx/˜mcintosh*.

En la actualidad existe muy poca literatura que trata de explicar la complejidad del autómata celular "*The Game of Life*", así como la explicación de las estructuras que puede producir dicha regla de evolución, de aquí surge la necesidad de encontrar un análisis que nos lleve a tratar de entender un poco la complejidad de esta regla de evolución. A pesar de ser un modelo muy sencillo sus comportamientos son muy complejos, entonces el problema es que no hay una explicación de porque la regla de evolución *Life* tiene esos comportamientos y aunque se sabe que una formalización completa esta aún lejos de darse, en esta tesis se dá una aproximación probabilística de como interactuan los estados en el espacio de evoluciones.

En la tesis se presentan varias aportaciones con respecto a la literatura actual.

1. Primeramente se hace una extensión del análisis con la teoría del campo promedio calculando las densidades a través del tiempo, haciendo sustituciones del polinomio del campo promedio que genera la regla de evolución *Life*. Una vez que se identifican las características que sigue la curva de probabilidad de *Life*, se hace una primera comparación con otras reglas de evolución que presentan características similares a *Life*, encontrando diferencias importantes que indican porque estas reglas de evolución no pueden simular completamente los comportamientos de *Life*.

2. Se hace una clasificación de los autómatas celulares en tres dimensiones que son similares a *Life* con la teoría del campo promedio, ya que el análisis que existe actualmente en autómatas celulares que son similares a *Life* en el espacio tridimensional es un análisis cuantitativo basado en simulaciones de evoluciones. Aplicando la teoría del campo promedio se encontró que varias de las reglas de evolución propuestas por Bays que tienen comportamientos similares a *Life*, no conservan propiedades similares probabilísticamente y de esta manera se logró encontrar diferencias en estas reglas de evolución, este estudio comparativo no ha sido reportado por otros autores.

3. Se realiza el análisis probabilístico de la *regla 110* para determinar sus relaciones estadísticas que existen con *Life*, ya que la *regla 110* puede ser visto como un modelo original y no propiamente como una regla que tiene características similares a *Life*, además dada la importancia que esta regla de evolución ha tenido se describe un método para reproducir cada una de las estructuras que la *regla 110* produce en el espacio de evoluciones, este método se realiza utilizando diagramas de de Bruijn, logrando reproducir de manera exacta algunas de las estructuras que se pueden generar cuando se evoluciona el autómata celular, este método solo ha sido reportado en [46] por McIntosh.

4. Se realizaron 5 programas para modelar autómatas celulares de una y dos dimensiones, estos programas sirvieron para reproducir algunos datos obtenidos por otros autores. Los programas fueron codificados en objetos y se muestra la potencialidad que tienen algunas de las herramientas de la tecnología *WebObjects* para que estos 5 programas puedan correr en 4 sistemas operativos diferentes.

En el Capítulo 1 se define la notación básica de los autómatas celulares en una, dos y tres dimensiones, así como su estructura elemental. Se muestra la clasificación fenotípica de los autómatas celulares en una dimensión establecida por Wolfram y se describe la teoría del campo promedio, la útilidad que ofrece y su relación con la teoría de los autómatas celulares. En el Capítulo 2 se analiza el autómata celular en dos dimensiones *Life* describiendo su regla de evolución tal como la definió Conway, además se presenta una

[3] Para los sistemas NeXTSTEP y OPENSTEP.
[4] Para el sistema MS-Dos y Windows.

variante de *Life* conocido como *HighLife*, estas dos reglas se analizan con la teoría del campo promedio y se describen sus diferencias probabilísticas, ilustrando algunas de sus estructuras mas interesantes y su comportamiento de manera general en el espacio de evoluciones para ambas reglas.

En el Capítulo 3 se analizan los autómatas celulares en tres dimensiones propuestos por Bays con la teoría del campo promedio, diferenciando las reglas de evolución que presentan características similares a la regla *Life*, aprovechamos el estudio realizado por Bays que presenta varias reglas de evolución que pueden producir configuraciones *gliders* y *still life*, además ilustramos algunas de estas estructuras con las reglas mas representativas que presentó Bays, pues varios de sus resultados son de tipo fenotípico y varios de sus resultados formales no pueden ser proyectados a autómatas celulares de una y dos dimensiones, pero no dejan de ser importantes y por eso se discuten algunas de sus definiciones.

En el Capítulo 4 se analiza el autómata celular en una dimensión de orden $(2,1)$ *regla 110* con la teoría del campo promedio, mostrando la relación a nivel fenotípico que existe entre *Life* y la *regla 110* establecida por Cook. Ilustramos algunos de los *gliders* dados a conocer por Cook así como el fondo periódico en el que la regla evoluciona conocido como *ether* y que no tiene *Life*, además se describe el uso de los diagramas de de Bruijn extendidos para poder reproducir algunos *gliders* y el *ether*, finalmente establecemos sus características probabilísticas con *Life* y algunos de sus comportamientos complejos que lo relacionan con *Life*.

En el capítulo 5 se describen los resultados obtenidos así como las limitaciones que tiene este análisis, también se muestran los resultados experimentales obtenidos a nivel de simulación que son interesantes y los trabajos futuros que nos permitan tener un estudio mas formal de estos resultados. Finalmente en el apéndice A se describen las características a nivel usuario de los programas de entorno gráfico desarrollados para esta tesis, los requerimientos necesarios en cuanto a software y hardware, además mostramos las diferentes plataformas en que pueden ser usadas estas aplicaciones. También describimos las características a nivel desarrollador de la implementación en objetos con el lenguaje *Objective-C* y su transportabilidad en 4 plataformas haciendo notar la potencialidad de esta tecnología [5], al final se explica la construcción del programa ACOSXM.

[5] http://www.apple.com

Capítulo 1

Fundamentos

En este capítulo se dan los fundamentos básicos en la teoría de autómatas celulares. En la Sección 1.1 se define la estructura de los autómatas celulares en una dimensión. En la Sección 1.2 se define la estructura de los autómatas celulares en dos dimensiones haciendo notar las diferencias con los autómatas de una dimensión. En la Sección 1.3 se define la estructura de los autómatas celulares en tres dimensiones, su espacio de evoluciones y algunas limitantes que existen en estos tipos de autómatas celulares. En la Sección 1.4 se muestra la clasificación de los autómatas celulares establecida por Wolfram [57] y se hace notar que esta clasificación no es del todo general ilustrando algunos contra ejemplos, finalmente en la Sección 1.5 se define la teoría del campo promedio tal como la define Gutowitz en [24] y se explica su relación con la teoría de autómatas celulares.

1.1 Autómatas celulares en una dimensión

El estudio de los autómatas celulares en una dimensión ha tenido mucho interés a través de la historia, por esta razón se ha logrado obtener una amplia literatura en este tipo de autómatas celulares. Sea $\mathbb{Z}$ el conjunto de los enteros y $\mathbb{Z}^+$ el conjunto de los enteros positivos, entonces el espacio de evoluciones en una dimensión se representa como una sucesión de elementos que determinarán un arreglo lineal, x_i representa las posiciones de cada elemento dentro del arreglo por lo que $i \in \mathbb{Z}$. Cada una de las posiciones del arreglo es llamado *célula*, estas células pueden tomar elementos de un conjunto Σ y $\Sigma \in \mathbb{Z}^+$, este conjunto representa el número de estados que puede manejar el autómata celular en estudio por lo que $x_i \in \Sigma$, es decir, cada una de las células tendrá un elemento del conjunto Σ. Una sucesión de células es llamada una *configuración*, en general Wolfram representa a los autómatas celulares de una dimensión con dos parámetros (k, r), donde k representa el número de estados del conjunto Σ y r el número de vecinos con respecto a una célula central. Los *vecinos* son células que se encuentran ubicadas a la izquierda y a la derecha en igual número con respecto a la célula central, por lo tanto los vecinos mas la célula central forman una *vecindad* como se ilustra en la Figura 1.1.

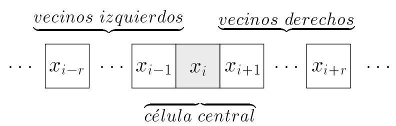


Figura 1.1: Vecindad de tamaño $2r + 1$

Los autómatas celulares en una dimensión se pueden representar como el sistema:

$$\{\Sigma,\, r,\, \varphi,\, c_i\}$$

donde Σ es el conjunto de estados, r el número de vecinos con respecto a una célula central, φ la función de trancisión y c_i la configuración inicial del sistema. El conjunto de estados Σ puede ser acotado determinando su cardinalidad $|\Sigma| = k$ por lo tanto $k \in \mathbb{Z}^+$ y el conjunto Σ es numerable como $\Sigma = \{0, 1, \ldots, k-1\}$; entonces se define un arreglo lineal asociando una posición $i-$ésima a una sucesión de x_i $\forall$ $i \in \mathbb{Z}$ y $x \in \Sigma$.

Una secuencia de longitud finita de símbolos en Σ es una *cadena* sobre Σ, el conjunto de todas las cadenas sobre Σ, es representado por el conjunto de todas las configuraciones en una dimensión $\Sigma^{\mathbb{Z}}$. Utilizando los dos parámetros (k, r) se puede determinar el número de células que conforman los vecinos y vecindades completas, $2r$ es el número de vecinos con respecto a una célula central, $2r + 1$ es el número de celulas que forman una vecindad, por lo tanto Σ^{2r+1} es el conjunto de todas las cadenas de longitud $2r+1$ que determina el número de vecindades para un r dado.

Una *configuración* es un asignamiento de estados Σ a cada uno de los elementos del arreglo lineal, una configuración finita de células es un arreglo lineal acotado en ambos extremos, es decir, en sus límites del extremo izquierdo y extremo derecho. Sea c una configuración no finita sobre Σ, entonces el conjunto de todas las configuraciones no finitas se denota como el conjunto $\mathcal{C}(\Sigma)$, por lo tanto $\mathcal{C}(\Sigma) \subseteq \Sigma^{\mathbb{Z}}$. Sea c_i una configuración finita y $\mathcal{C}_F(\Sigma)$ el conjunto de todas las configuraciones finitas, por lo tanto $c_i \in \mathcal{C}_F(\Sigma)$ y $\mathcal{C}_F(\Sigma) \subset \mathcal{C}(\Sigma)$, además esta configuración c_i evolucionará a tráves del tiempo t donde $t \in \mathbb{Z}$, finalmente una configuración finita c_i se determina por una secuencia de células de longitud l, tal que $c_i = x_0, x_1, \ldots, x_{l-1}$ y $l \in \mathbb{Z}^+$. Las configuraciones finitas tienen propiedades a la frontera, es decir, cuando se evalúan los extremos del arreglo lineal se concatena la célula inicial con la célula final y se forma un anillo, de esta manera se conservan las propiedades simétricas en el espacio de evoluciones del autómata celular como se ilustra en el Figura 1.2, si la configuración c_i es de longitud l entonces se concatenan las células $x_0 \diamond x_{l-1}$ donde '$\diamond$' es la operación concatenación.

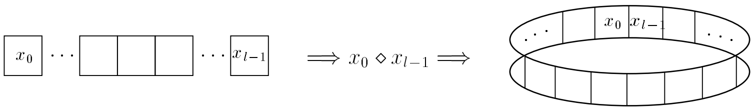


Figura 1.2: Propiedades a la frontera en una dimensión

La *función de transición* φ indica la transformación local que está determinada por las vecindades de longitud $2r + 1$ y la longitud de las vecindades deben de ser $2r + 1 \geq 2$, el número de vecindades está determinado por el conjunto Σ^{2r+1}, la función de transición es la parte importante en los autómatas celulares, una *regla de evolución* es la función de transición definida en cada una de las vecindades con respecto al valor que tenga k y r, cada una de estas vecindades serán transformadas en un elemento del conjunto Σ.

El número de células en una vecindad es igual a $2r + 1$, el número de vecindades para una regla en particular es igual a k^{2r+1} y el número de reglas de evolución que se pueden generar para un autómata celular de orden (k, r) es igual a $k^{k^{2r+1}}$, entonces la función de transición va a determinar la transformación local de cada una de las vecindades a un elemento de Σ como se representa a continuación:

$$x_i = \varphi(x_{i-r}, \ldots, x_i, \ldots, x_{i+r}) \tag{1.1.1}$$

para toda $x_i \in \Sigma$, por lo tanto la transformación local en general está definido como la función $\varphi : \Sigma^{2r+1} \to \Sigma$.

La función φ necesita de r vecinos a la izquierda y r vecinos a la derecha para realizar la transformación local, sin embargo cuando φ se encuentra en los extremos del arreglo lineal existe el problema de que faltarían células que completen las vecindades en el extremo izquierdo como en el extremo derecho, ésto se soluciona con la concatenación de la células $x_0 \diamond x_{l-1}$ como se ilustró en la Figura 1.2, es decir, utilizando las propiedades a la frontera. La transformación local es dada con cada una de las vecindades a un elemento del conjunto de estados, si en lugar de emplear vecindades se utilizan configuraciones finitas se obtiene una transformación global, es decir, la correspondencia que existe es entre configuraciones y no entre vecindades a un solo elemento. Sea Φ una transformación global, la transformación global es una correspondencia entre configuraciones c_i, entonces la función de transición para una transformación global se representa como:

$$x_0, x_1, \ldots, x_{l-1} = \Phi(x_0, x_1, \ldots, x_{l-1}) \tag{1.1.2}$$

donde $x_0, x_1, \ldots, x_{l-1} = c_i$ y $c_i \in \mathcal{C}_F(\Sigma)$, por lo tanto la transformación global está definido como la función $\Phi : \mathcal{C}_F(\Sigma) \to \mathcal{C}_F(\Sigma)$.

Se analiza un autómata celular de orden $(2, 1)$ para ejemplificar estos conceptos, en este caso $k = 2$ que indica el número de elementos en el conjunto Σ, estos dos elementos se puede representar como $\Sigma = \{0, 1\}$, el parámetro $r = 1$ indica que debe haber un vecino a la izquierda con respecto a una célula central y un vecino a la derecha con respecto a la misma célula central. El número de vecinos en total es igual a $2r = 2(1) = 2$, es decir, el vecino a la izquierda y el vecino a la derecha. La vecindad es igual a $2r+1 = 2(1)+1 = 3$ células, los dos vecinos mas la célula central determinan las tres células de la vecindad. El número de vecindades es igual a $k^{2r+1} = 2^{2(1)+1} = 2^3 = 8$, ésto quiere decir que se tienen ocho vecindades que representarán una regla en particular. El número de reglas que se pueden derivar con este orden es igual a $k^{k^{2r+1}} = 2^{2^{2(1)+1}} = 2^8 = 256$ reglas de evolución diferentes para este autómata celular.

La regla de evolución se puede representar de varias maneras, se emplea la notación binaria y decimal ya que en algunos casos tratar de representar una regla de evolución en notación binaria resulta muy difícil. La representación gráfica en el espacio de evoluciones en los autómatas celulares de una dimensión se obtiene apartir de una configuración inicial c_i, a esta configuración inicial c_i^t se le aplica la función de transición φ y de esta manera se obtienen las configuraciones $c_i^{t+1}, \ldots, c_i^{t+n}$ hasta un tiempo dado, donde t representa el tiempo y n el $n-$ésimo tiempo en que evolucionará el autómata celular como se ilustra en la Figura 1.3, por lo tanto t y $n \in \mathbb{Z}$.

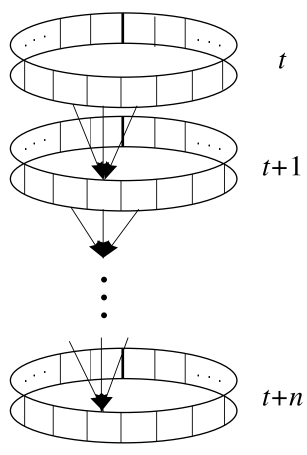


Figura 1.3: Espacio de evoluciones en una dimensión

El diagrama de evoluciones representa las configuraciones c_i a través del tiempo, los estados se representan con colores el estado '0' es blanco y el estado '1' es negro. En la Figura 1.4 se ilustra el espacio de evoluciones a través del tiempo con una configuración inicial aleatoria.

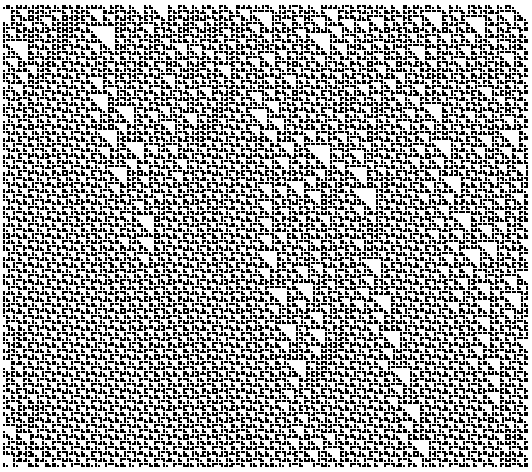


Figura 1.4: Diagrama de evoluciones en una dimensión regla 124

La función de transición φ para esta regla de evolución se representa como:

Regla 124	
$\varphi(0,0,0) = 0$	$\varphi(1,0,0) = 1$
$\varphi(0,0,1) = 0$	$\varphi(1,0,1) = 1$
$\varphi(0,1,0) = 1$	$\varphi(1,1,0) = 1$
$\varphi(1,1,0) = 1$	$\varphi(1,1,1) = 0$

Tabla 1.1: Regla de evolución 124

el número de vecindades es igual a ocho, éstas son el número de combinaciones posibles que existen en tres posiciones dando como resultado ocho combinaciones $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$. La regla de evolución es la cadena binaria que se deriva en cada una de las transformaciones, por ejemplo en la Tabla 1.1 se tiene la regla 124, esta regla de evolución se representa como la cadena 00111110 transformada en notación decimal $124 = (0 * 2^0) + (0 * 2^1) + (1 * 2^2) + (1 * 2^3) + (1 * 2^4) + (1 * 2^5) + (1 * 2^6) + (0 * 2^7)$.

En autómatas celulares de mayor orden la regla de evolución es muy difícil de representar, para solucionar este problema se emplean reglas totalísticas o semitotalísticas. Una regla totalística para un autómata celular de orden (k, r) se representa como:

$$\varphi(x_{i-r}, \ldots, x_i, \ldots, x_{i+r}) = \tau(x_{i-r} + \ldots + x_i + \ldots + x_{i+r}) \tag{1.1.3}$$

donde τ agrupa todas las vecindades que tengan un mismo valor entero de acuerdo a la suma de los elementos que forman una vecindad dada, sin importar el orden en que se encuentren. Las reglas semitotalísticas difieren únicamente en que la célula x_i no es tomada en cuenta en la suma de los elementos de la vecindad de τ, este tipo de reglas también son aplicables en autómatas celulares de dos y tres dimesiones.

En los autómatas celulares de una dimensión se han realizado estudios muy importantes y completos como el trabajo de Gustav A. Hendlund en [31], entre otros autores podemos mencionar a Erika Jen, Masakasu Nasu, David Hillman, Edward Fredkin, Tommaso Toffoli, Norman Margolus, Jarkko Kari, S. Amoroso, Y. N. Patt, Gutowitz, Wolfram, Andrew Wuensche, McIntosh y Karel Culik entre otros.

1.2 Autómatas celulares en dos dimensiones

Los autómatas celulares en dos dimensiones evolucionan en el plano cartesiano. Sea $\Sigma = \{0, 1\}$ el conjunto de estados, c_i la configuración inicial y su función de transición φ va a estar determinada por dos tipos de vecindades, la vecindad de von Neumann y la vecindad de Moore como se ilustran en la Figura 1.5 respectivamente.

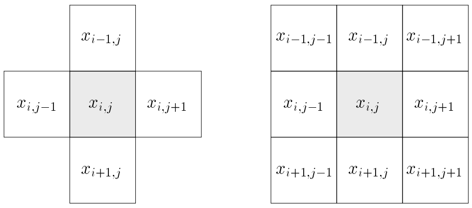


Figura 1.5: Vecindad de von Neumann y vecindad de Moore

En la teoría de autómatas celulares en dos dimensiones existe poca literatura que defina una generalización de dichos autómatas, ésto se debe porque las herramientas que se emplean en una dimensión no son prácticas para aplicarse en dos y tres dimensiones. Algunos estudios realizados sobre la teoría del comportamiento tanto local como global se han realizado de manera estadística en [3], [21], [24], [30], [39] y [48].

El espacio celular en dos dimensiones está definido por el producto cartesiano $\mathbb{Z} \otimes \mathbb{Z}$, entonces una célula $x_{i,j} \in \mathbb{Z} \otimes \mathbb{Z}$ es directamente conectada a una célula $\{(x_{i+k_1}, x_{j+k_2}) : Max\{|k_1|, |k_2|\} \leq 1\}$, es decir, la vecindad de Moore donde i, j y $k \in \mathbb{Z}$. La vecindad de Moore está formada por una célula central y ocho vecinos alrededor de ésta, si se utiliza la notación de una dimensión tendríamos que $k = 2$ y $r = 4$, entonces la vecindad tiene $2r+1 = 2(4)+1 = 9$ células en total, ésto origina $k^{2r+1} = 2^9 = 512$ vecindades y $k^{k^{2r+1}} = 2^{512}$ reglas de evolución. Nótese la dificultad de representar una regla de evolución en dos dimensiones, por eso es útil emplear reglas semitotalísticas para representar la función de transición con la vecindad de Moore.

Sea $\Sigma = \{0, 1\}$ el conjunto de estados, $\mathcal{V}$ es la vecindad isotrópica en realidad esta vecindad son los vecinos con respecto a una célula central por lo tanto $\mathcal{V} = 8$ y $\mathbf{x}_0$ la célula central en estudio donde $\mathbf{x}_0 = x_{i,j}$ y las células $\mathbf{x}_1, \ldots, \mathbf{x}_{\mathcal{V}} = x_{i-1,j-1}, \ldots, x_{i+1,j+1}$ son sus vecinos, para toda $\mathbf{x}_i \in \Sigma$. En la Ecuación 1.2.1 la función φ define la transformación local, las variables N_{min} y S_{min} indican el número mínimo de células ocupadas por el estado 1 en $\mathcal{V}$ y las variables N_{max} y S_{max} el número máximo de células ocupadas por el estado 1 en $\mathcal{V}$ en un tiempo t. Si $\mathbf{x}_0 = 0$ en el tiempo t, entonces $\mathbf{x}_0 = 1$ en el tiempo $t+1$ si $N_{min} \leq \sum_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq N_{max}$. Si $\mathbf{x}_0 = 1$ en el tiempo t, entonces $\mathbf{x}_0 = 1$ en el tiempo $t+1$ si $S_{min} \leq \sum_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq S_{max}$. Finalmente una regla semitotalística en dos dimensiones se representa como $R(S_{min}, S_{max}, N_{min}, N_{max})$[1], donde N y S deben tomar valores entre 1 y 8.

$$\varphi(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\mathcal{V}}) = \begin{cases} 1 \quad si & \begin{cases} \mathbf{x}_0 = 0 \quad y \quad N_{min} \leq \sum\limits_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq N_{max} \\ \\ \mathbf{x}_0 = 1 \quad y \quad S_{min} \leq \sum\limits_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq S_{max} \end{cases} \\ \\ 0 \quad \text{en otro caso} \end{cases} \tag{1.2.1}$$

[1] N significa nacimiento y S sobrevivencia de una célula $x_{i,j}$.

La vecindad de von Neumann se puede representar de manera análoga ajustando la Equación 1.2.1, esta vecindad suprime las células diagonales y conserva las células ortogonales con respecto a la vecindad de Moore, la función de transición φ tiene cuatro vecinos y una célula central por lo tanto $\mathcal{V} = 4$, si se emplea la notación de una dimensión $k = 2$ y $r = 2$, entonces se tienen $k^{2r+1} = 2^5 = 32$ vecindades y 2^{32} reglas de evolución. Sea $\Sigma = \{0, 1\}$ el conjunto de estados, $\mathbf{x}_0 = x_{i,j}$ la célula central y $\mathbf{x}_1 = x_{i-1,j}$, $\mathbf{x}_2 = x_{i,j-1}$, $\mathbf{x}_3 = x_{i,j+1}$, $\mathbf{x}_4 = x_{i+1,j}$ son los vecinos o la vecindad isotrópica para toda $\mathbf{x}_i \in \Sigma$. En la Ecuación 1.2.2 la función φ define la transformación local, las variables N_{min} y S_{min} indican el número mínimo de células ocupadas por el estado 1 en $\mathcal{V}$ y las variables N_{max} y S_{max} el número máximo de células ocupadas por el estado 1 en $\mathcal{V}$ en un tiempo t. Si $\mathbf{x}_0 = 0$ en el tiempo t entonces $\mathbf{x}_0 = 1$ en el tiempo $t + 1$ si $N_{min} \leq \sum_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq N_{max}$, pero si $\mathbf{x}_0 = 1$ en el tiempo t entonces $\mathbf{x}_0 = 1$ en el tiempo $t + 1$ si $S_{min} \leq \sum_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq S_{max}$, donde N y S deben tomar valores entre 1 y 4.

$$\varphi(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\mathcal{V}}) = \begin{cases} 1 \quad si & \begin{cases} \mathbf{x}_0 = 0 \quad y \quad N_{min} \leq \sum\limits_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq N_{max} \\ \\ \mathbf{x}_0 = 1 \quad y \quad S_{min} \leq \sum\limits_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq S_{max} \end{cases} \\ \\ 0 \quad \text{en otro caso} \end{cases} \tag{1.2.2}$$

En la Figura 1.6 se ilustra el espacio de evoluciones en dos dimensiones, es un arreglo bidimensional y las posiciones $x_{i,j}$ son ocupadas por elementos del conjunto $\Sigma = \{0, 1\}$, el estado 0 se representa con el color blanco y el estado 1 se representa con el color negro.

Figura 1.6: Espacio de evoluciones en dos dimensiones

En la Figura 1.7 se ilustran algunas de las estructuras mas interesantes de la regla *Life*, esta regla de evolución es una regla semitotalística y se representa como $R(2,3,3,3)$. Estas estructuras fueron construidas cuidadosamente, pero varias de ellas pueden ser generadas desde una configuración aleatoria a través del tiempo. Esta regla de evolución se explica y analiza con todo detalle en el Capítulo 2.

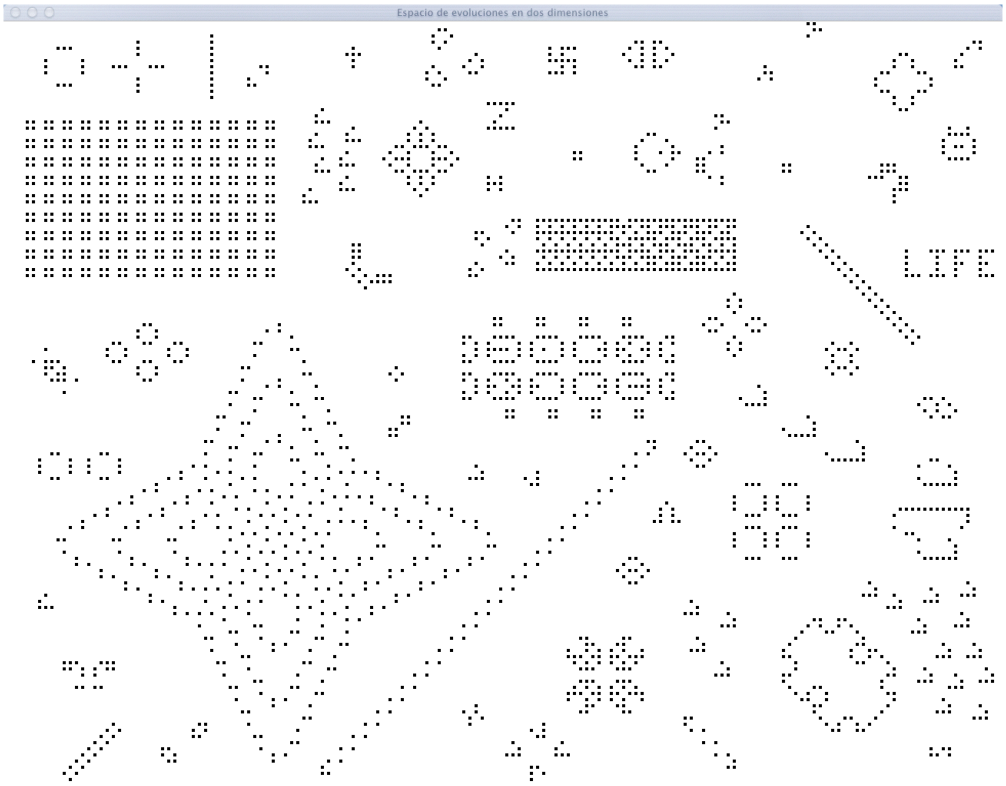


Figura 1.7: Configuraciones complejas en *Life*

En autómatas celulares de dos dimensiones no hay propiedades a la frontera, es decir, el arreglo bidimensional está definido por el producto cartesiano $\mathbb{Z} \otimes \mathbb{Z}$.

1.3 Autómatas celulares en tres dimensiones

Los autómatas celulares en tres dimensiones han sido ampliamente analizados por Bays en [2], [3], [4], [5], [6], [7] y [8]. Su estudio se enfoca principalmente en encontrar una regla de evolución en tres dimensiones que sea la sucesora de *Life* en el espacio tridimensional, muchos de sus resultados son de tipo cuantitativo, basados principalmente en la simulación de varias reglas de evolución en pequeños espacios tridimensionales para encontrar estructuras que sean similares a *Life* y de esta manera ha logrado obtener varias reglas de evolución que presentan características similares a *Life* en autómatas celulares de tres dimensiones.

Existe muy poca literatura que trata de generalizar una notación en autómatas celulares de tres dimensiones, la mayoría de los trabajos realizados en este tipo de autómatas han sido de tipo estadístico, tratando de encontrar comportamientos colectivos no triviales en el espacio de evoluciones, por otra parte se ha intentado encontrar aplicaciones en las áreas de la química y la arquitectura. Si bien los autómatas celulares en dos dimensiones son difíciles de representar, en tres dimensiones el problema crece exponencialmente.

Sea $\Sigma = \{0, 1\}$ el conjunto de estados, el espacio de evoluciones en tres dimensiones se determina por el producto $\mathbb{Z} \otimes \mathbb{Z} \otimes \mathbb{Z}$. Al igual que los autómatas celulares en dos dimensiones, en tres dimensiones también se utilizan reglas de evolución semitotalísticas y la función de transición φ solo utiliza la vecindad de Moore como se ilustra en la Figura 1.8, la vecindad de Moore en el espacio tridimensional es solo una extensión de dos dimensiones en tres dimensiones, donde $x_{i,j,k}$ es la célula central de la vecindad y $x_{i-1,j-1,k-1}, \ldots, x_{i+1,j+1,k+1}$ son los vecinos de la vecindad con respecto a la célula central, para toda $x_{i,j,k} \in \Sigma$. La vecindad de Moore en tres dimensiones tiene una célula central y 26 vecinos alrededor de ésta por lo que $\mathcal{V} = 26$, por lo tanto una vecindad en tres dimensiones está formada por 27 células.

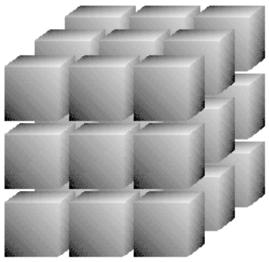


Figura 1.8: Vecindad de Moore en tres dimensiones

Para representar una regla de evolución φ se debe definir la transformación de cada vecindad que conforma una regla, los autómatas celulares en tres dimensiones tienen 2^{27} vecindades, lo que produce $2^{2^{27}}$ reglas de evolución. El problema de representar una regla de evolución crece exponencialmente conforme aumenta la dimensión del autómata celular, por esta razón se utilizan reglas semitotalísticas.

Sea $\Sigma = \{0, 1\}$ el conjunto de estados, $\mathcal{V} = 26$ el número de vecinos, entonces una célula $x_{i,j,k} \in \mathbb{Z} \otimes \mathbb{Z} \otimes \mathbb{Z}$ es directamente conectada a una célula $\{(x_{i+k_1}, x_{j+k_2}, x_{l+k_3}) : Max\{|k_1|, |k_2|, |k_3|\} \leq 1\}$, es decir, la vecindad de Moore en tres dimensiones, $\mathbf{x}_0 = x_{i,j,k}$ es la célula central y $\mathbf{x}_1, \ldots, \mathbf{x}_{\mathcal{V}} = x_{i-1,j-1,k-1}, \ldots, x_{i+1,j+1,k+1}$ son los vecinos alrededor de la célula central para toda $\mathbf{x}_i \in \Sigma$. En la Ecuación 1.3.1 la función φ define la transformación local, las variables N_{min} y S_{min} indican el número mínimo de células ocupadas por el estado 1 en $\mathcal{V}$ y las variables N_{max} y S_{max} el número máximo de células ocupadas por el estado 1 en $\mathcal{V}$ en un tiempo t. Si $\mathbf{x}_0 = 0$ en el tiempo t, entonces $\mathbf{x}_0 = 1$ en el tiempo $t + 1$ si $N_{min} \leq \sum_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq N_{max}$. Si $\mathbf{x}_0 = 1$ en el tiempo t, entonces $\mathbf{x}_0 = 1$ en el tiempo $t+1$ si $S_{min} \leq \sum_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq S_{max}$. Finalmente una regla semitotalística en tres dimensiones se representa como $R(S_{min}, S_{max}, N_{min}, N_{max})$, donde N y S deben tomar valores entre 1 y 26.

$$\varphi(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\mathcal{V}}) = \begin{cases} 1 \quad si & \begin{cases} \mathbf{x}_0 = 0 \quad y \quad N_{min} \leq \sum\limits_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq N_{max} \\ \\ \mathbf{x}_0 = 1 \quad y \quad S_{min} \leq \sum\limits_{i=1}^{\mathcal{V}} \mathbf{x}_i \leq S_{max} \end{cases} \\ \\ 0 \quad \text{en otro caso} \end{cases} \tag{1.3.1}$$

Los autómatas celulares en tres dimensiones son aún mas complicados para analizarlos, en la literatura de autómatas celulares en tres dimensiones se tienen análisis de tipo estadístico y probabilístico, tratando de explicar el comportamiento de reglas semitotalísticas, algunos trabajos importantes a este tipo de análisis se pueden ver en [3], [21] y [30].

El espacio de evoluciones que comúnmente manejan algunos programas en internet, es muy pequeño lo que impide poder visualizar comportamientos interesantes de estructuras periódicas fijas o con desplazamientos, en la Figura 1.9 se muestra un espacio de evoluciones en tres dimensiones en un arreglo de $80 \times 80 \times 80$ células, las células de color blanco representan el estado 0 y las células de color negro representan el estado 1, las ilustraciones que se muestran en autómatas celulares de tres dimensiones se obtuvieron con el programa *Tresvita 3.2* [2].

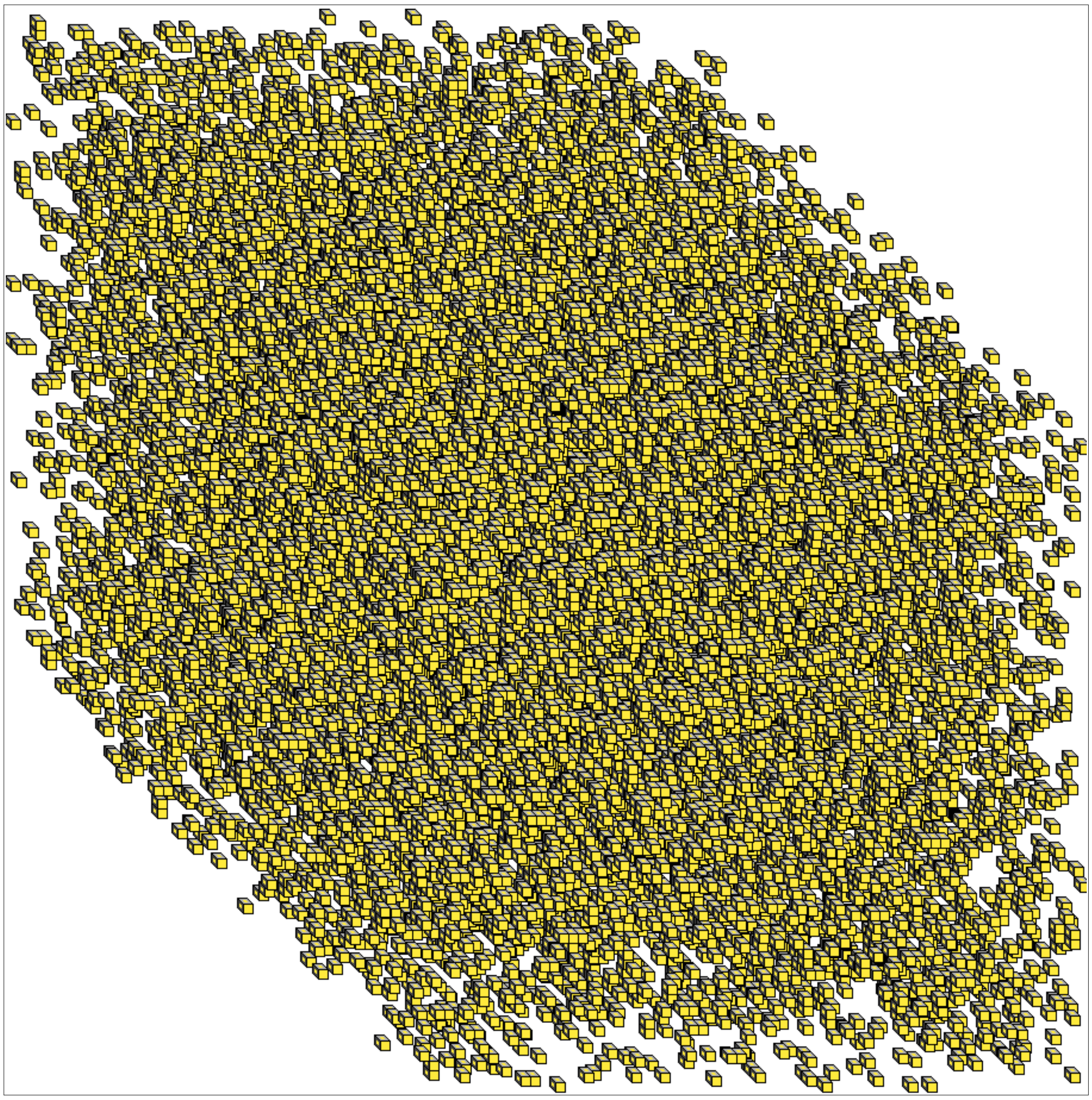


Figura 1.9: Diagrama de evoluciones en tres dimensiones

En el espacio de evoluciones tridimensional se pueden encontrar comportamientos complejos, caóticos o triviales. Bays presenta varias reglas con comportamientos complejos en [2]. En la Figura 1.10 se pueden ver algunas estructuras complejas construidas cuidadosamente y otras se obtuvieron de manera aleatoria

[2]http://www.kasprzyk.demon.co.uk/www/ALHome.html, programado por Alexander Mieczyslaw Kasprzyk, para la plataforma Macintosh con el sistema operativo MacOS 9, 1993.

aplicando la regla de evolución $R(4,5,5,5)$.

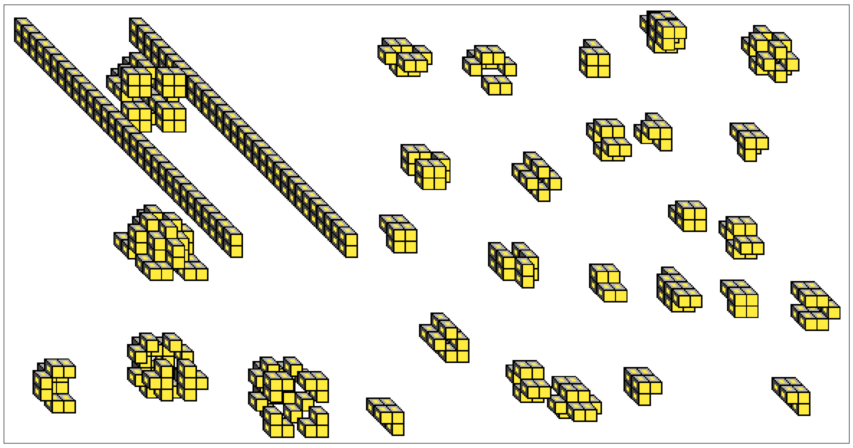


Figura 1.10: Evoluciones en tres dimensiones regla $R(4,5,5,5)$

1.4 Clases de Wolfram

La clasificación de Wolfram es de tipo fenotípico como lo discute Gutowitz en [26]. Se dice que es de tipo fenotípico porque su clasificación es realizada en base al comportamiento de las células a través del tiempo dentro del diagrama de evoluciones, es decir, en base a la observación.

- **Clase I**. Evolución a un estado constante.
- **Clase II**. Evolución a segmentos periódicos aislados.
- **Clase III**. Evolución que es siempre caótica.
- **Clase IV**. Evolución a segmentos caóticos aislados.

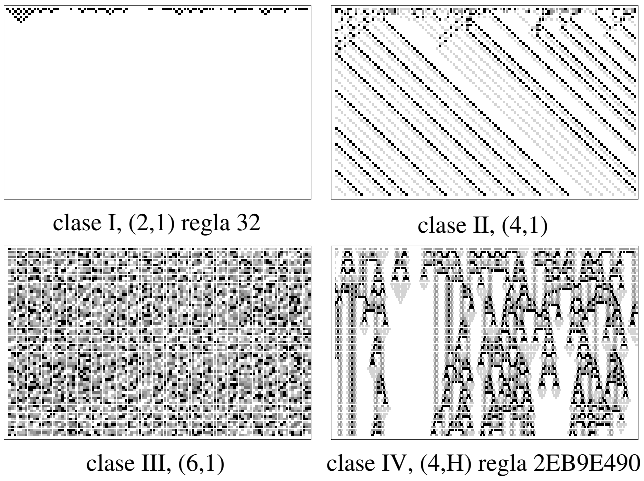


Figura 1.11: Clases de Wolfram

Los diagramas de evolución de la Figura 1.11 ilustran las clases de Wolfram, su clasificación la presentó en autómatas celulares de una dimensión, los autómatas celulares de la Figura 1.11 son autómatas celulares en una dimensión de diferentes ordenes, esta clasificación de tipo fenotípico es extendida a autómatas celulares de dos y tres dimensiones.

Esta clasificación no es general ya que pueden existir reglas de evolución que tienen comportamientos no triviales pero que se pueden describir perfectamente, por ejemplo un autómata celular reversible. Los *autómatas celulares reversibles* son autómatas que evolucionan con una regla de evolución y con esta regla se va construyendo su comportamiento en el espacio de evoluciones, pero este comportamiento puede ser reconstruido con una regla de evolución inversa, es decir, se puede reconstruir cada una de las configuraciones c_i hacia atrás en el tiempo. Por lo tanto su transformación local tiene la característica de tener una correspondencia biyectiva en cada una de sus vecindades, es decir, de manera local y consecuentemente de manera global, ésto implica que existe una función inversa φ^{-1} de la regla original φ que es única como se ilustra en la Figura 1.12.

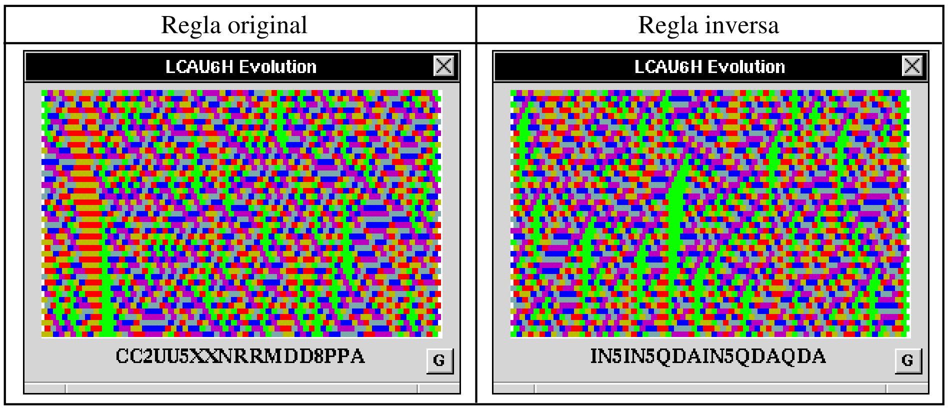
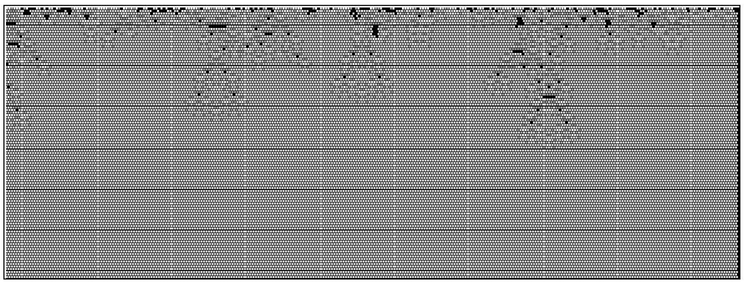


Figura 1.12: Autómata celular reversible en una dimensión

En la Figura 1.12 se ilustra un autómata celular reversible en una dimensión de orden $(k = 6, r = h)$ donde $h = 1/2$, es decir medio vecino a cada lado de la célula central, pero como eso no es posible se unen los dos medios vecinos en un solo vecino, por lo que se tiene una célula central y un vecino a la derecha que representa los dos medios vecinos. Los autómatas celulares con vecindades fraccionarias no son discutidos en esta tesis, tampoco los autómatas celulares reversibles, ni la representación hexadecimal de las reglas de evolución, únicamente se empleó como un ejemplo ilustrativo, para mas detalle de este tipo de autómatas celulares y representaciones puede consultarse el trabajo [33].

La complejidad en los autómatas celulares es muy difícil de describir, como se ilustró con el autómata celular reversible de la Figura 1.12, sin embargo varias contradicciones pueden ser encontradas en cada una de las clases que propone Wolfram.

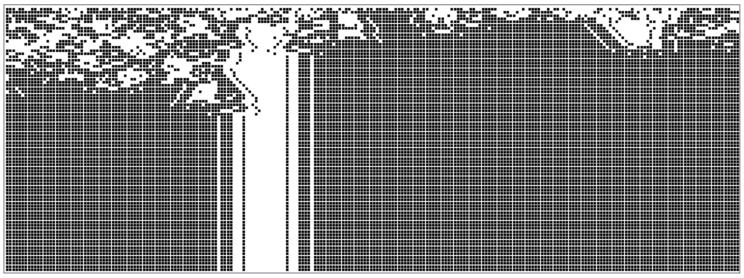
La clase I se caracteriza porque a partir de cualquier configuración aleatoria al momento de aplicar la regla de evolución, rápidamente el espacio de evoluciones es dominado por un solo estado del conjunto Σ. Entonces ¿la clase I puede tener estructuras caóticas aisladas en su espacio de evoluciones?, como se ilustra en la Figura 1.13. Nótese que estas estructuras caóticas aisladas pueden evolucionar por un tiempo mayor y despues entrar a un comportamiento uniforme o dominado por un solo estado, dando una falsa información si el autómata celular puede ser clasificado como clase IV en vez de ser clasificado como clase I, por lo tanto se puede ver que a nivel fenotípico la clasifación no es del todo confiable ni general.



(4,t) regla 71C71D74C71D74D11D74D14574D14516C71D74D11D74D14574D14516D14516591D74D14574D14516D14516594516596474D14516D14516594516596416596491

Figura 1.13: Autómata celular $(4, t)$ ¿clase I o clase IV?

La clase II se caracteriza porque a partir de cualquier configuración aleatoria se obtienen comportamientos periódicos desde las primeras generaciones y estos comportamientos periódicos se repiten hasta el infinito. Entonces ¿la clase II puede tener estructuras caóticas y despues de algunas generaciones tener un comportamiento periódico?, como se ilustra en la Figura 1.14.



(2,5) regla 3528

Figura 1.14: Autómata celular $(2, 5)$ ¿clase II o clase IV?

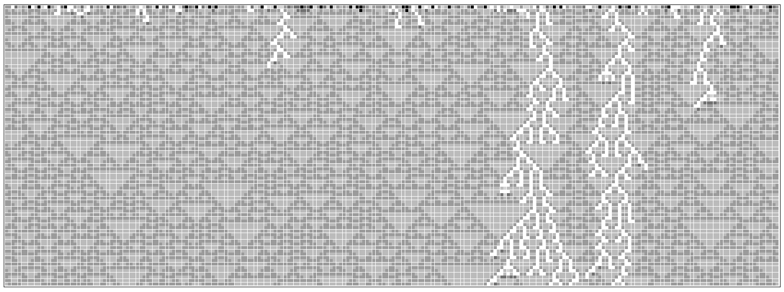
La Figura 1.14 muestra que este autómata celular tiene comportamientos caóticos al empezar a evolucionar la regla de evolución y despues de algunas generaciones entra en comportamientos periódicos.

(5,1) regla 683991868

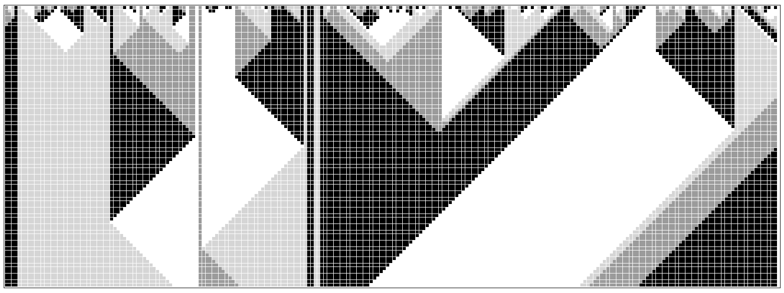
Figura 1.15: Autómata celular $(5, 1)$ ¿clase II o clase IV?

La evolución de la Figura 1.15 presenta comportamientos periódicos, también presenta estructuras caóticas aisladas, pero además nótese que estas estructuras caóticas aisladas pueden llegar a ser periódicas. En la clasificación de Wolfram este autómata celular sería del tipo clase II por tener comportamientos periódicos, pero es claro notar que algunas estructuras complejas pueden tener evoluciones muy largas y entonces esta regla de evolución ¿a que clase pertenece?.

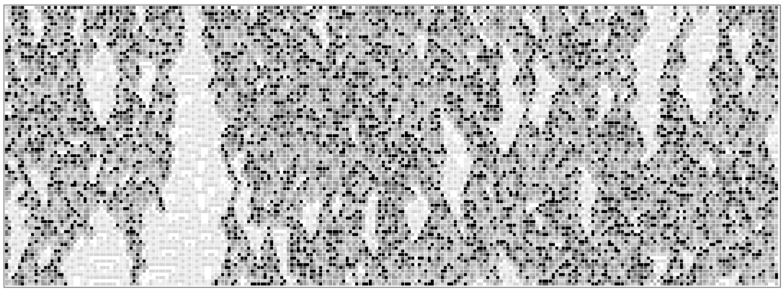
Algunas reglas de evolución con comportamientos mas extremos y mas complejos se muestran en la Figura 1.16; cabe mencionar que estos ejemplos fueron hallados aleatoriamente.



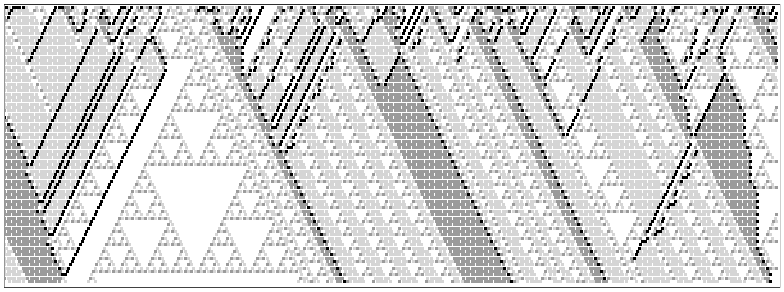
(6,1) regla 1994250871



(4,1) regla FCFF6504FCEAAA04FCEA655500EA6504



(8,1) regla mZTgdSJu9H2



(4,h) regla 64A3C5D8

Figura 1.16: Autómatas celulares de clases no definidas

La clasificación de Wolfram ha sido discutida y refutada por varios autores como en [3], [15], [23], [26], [39], [44] y [60], determinando que dicha clasificación no es general, inclusive se pueden encontrar diferencias a nivel fenotípico y ver que la clasificación de Wolfram no contiene todos los comportamientos posibles, porque existen autómatas celulares que no se encuentran clasificados en ninguna de las clases de Wolfram.

La clasificación en los autómatas celulares es mas compleja de lo que se puede suponer, como lo hace notar Culik en [15], la clasificación de los autómatas celulares es indecidible. Esta discrepancia en la clasificación de

Wolfram puede ser llevada a autómatas celulares de dos y tres dimensiones, aunque es mas complicado tratar de encontrar aleatoriamente reglas de evolución con diferencias de tipo fenotípico. Los análisis estadísticos y probabilísticos tratan de ayudar a explicar parte de la complejidad de dichos autómatas celulares, la teoría del campo promedio permite diferenciar el comportamiento de las reglas de evolución sin llegar a una clasificación. Es importante notar que es útil tener un análisis analítico que permita caracterizar de alguna manera las reglas de evolución en los autómatas celulares y apoyarnos en las simulaciones para tener un mejor entendimiento de sus comportamientos, en lugar de tener un análisis fenotítipo únicamente.

1.5 Teoría del campo promedio

La definición presentada en esta sección de la teoría del campo promedio se obtuvo del trabajo realizado por Gutowitz en [22], [23] y [25].

La *teoría del campo promedio* es un modelo simple que proporciona propiedades estadísticas de los autómatas celulares. La teoría del campo promedio se basa en que los elementos del conjunto de estados Σ son independientes entre si, es decir, no hay correlación entre cada uno de los elementos que se encuentran en el espacio de evoluciones, bajo esta condición es fácil estimar la probabilidad de los estados en una vecindad en términos de la probabilidad de un solo estado (el estado en que evoluciona la vecindad). La probabilidad de la vecindad es el producto de las probabilidades de los estados que forman la vecindad.

La probabilidad de que una célula tenga un estado en particular en el tiempo $t+1$, es la suma de las probabilidades de las vecindades que se transforman a este estado en el tiempo t. Se consideran únicamente las vecindades definidas para el estado de la célula en estudio, porque la regla de evolución es local. Sea φ la regla de evolución de un autómata celular de orden (k,r) con dos estados por célula, por lo tanto $\Sigma = \{0,1\}$.

La vecindad es de tamaño $2r+1$, k^{2r+1} es el número de vecindades, $\varphi(x_{i-r},\ldots,x_i,\ldots,x_{i+r})$ tomará el valor de 0 ó 1 dependiendo de la transformación de cada vecindad para una regla de evolución en particular, la vecindad $x_{i-r},\ldots,x_i,\ldots,x_{i+r}$ se representará como X. Sea p la probabilidad de tener el estado 1 en el tiempo t, $1-p$ es la probabilidad de tener el estado 0 en el tiempo t, v es el número de veces que aparece el estado 1 en la vecindad, $n-v$ es el número de veces que aparece el estado 0 en la vecindad, la sumatoria tomará cada una de las vecindades definidas por el conjunto Σ^{2r+1}, entonces la teoría del campo promedio calcula la densidad en el tiempo $t+1$ como:

$$p_{t+1} = \sum_{j=1}^{k^{2r+1}} \varphi_j(X) p_t^v (1-p_t)^{n-v} \tag{1.5.1}$$

nótese que k^{2r+1} es el número de vecindades y Σ^{2r+1} es el conjunto que define cada una de las vecindades, además la teoría del campo promedio puede ser empleada en autómatas celulares de diferentes dimensiones. Los puntos fijos que representa la teoría del campo promedio son una estimación de la densidad de los estados en un largo tiempo, esta densidad es independiente de la densidad inicial.

McIntosh hace notar la relación que existe entre la teoría del campo promedio y los polinomios de Bernstein en [44], la definición de los polinomios de Bernstein que se presenta se obtuvo de [37].

Sea una función $f(x)$ definida en el intervalo cerrado $[0,1]$ entonces la expresión:

$$B_n(x) = B_n^f(x) = \sum_{v=0}^{n} f\left(\frac{v}{n}\right) \begin{pmatrix} n \\ v \end{pmatrix} x^v (1-x)^{n-v} \tag{1.5.2}$$

es llamado el *Polinomio de Bernstein* de orden n de la función $f(x)$. $B_n(x)$ es un polinomio en x de grado $\leq n$ donde $v = 0, 1, \ldots, n-1$. El polinomio $B_n(x)$ sería introducido por S. Bernstein para dar una demostración de la aproximación del teorema de Weierstrass. Pero si $f(x)$ es continua en $[0, 1]$, entonces:

$$\lim_{x \to \infty} B_n(x) = f(x) \tag{1.5.3}$$

uniformemente en $[0, 1]$. Aquí hay muchas otras expresiones llamadas "integrales singulares" y la relación que tienen con los polinomios de Bernstein es la aproximación que realizan "generando" la función $f(x)$ además de reproducir algunas de sus propiedades. La integral singular mas conocida es la integral de Dirichlet's

$$s_n(x) = \frac{1}{\pi} \int_{-\pi}^{+\pi} f(t) \, \frac{sin(n+\frac{1}{2})(t-x)}{2sin\frac{1}{2}(t-x)} \, dt,$$

representado las sumas parciales $s_n(x)$ de las series de Fourier de la función $f(x)$ integrable en $[-\pi, +\pi]$. Otro ejemplo es la integral de Fejér que representa la media aritmética $\sigma_n = (s_0 + s_1 + \ldots + s_n)/(n+1)$ de $s_n(x)$. En general una integral singular puede ser escrita en la forma:

$$\Phi_n(x) = \int_a^b f(t) K_n(x,t) dt, \tag{1.5.4}$$

donde $K_n(x,t)$ es el "kernel" definido por $a \leq x \leq b$ y $a \leq t \leq b$, que tiene propiedades de las funciones $f(x)$ de una cierta clase, $\Phi_n(x)$ converge a $f(x)$ cuando $n \to \infty$.

El polinomio de Bernstein de la Ecuación 1.5.2 es una suma finita de un tipo correspondiente a la integral de la Ecuación 1.5.4. Ambas ecuaciones son casos especiales de la integral de Stieltjes en la variable t,

$$B_n(x) = \int_0^1 f(t) d_t K_n(x,t), \tag{1.5.5}$$

con el kernel

$$K_n(x,t) = \sum_{v \leq n,t} \begin{pmatrix} n \\ v \end{pmatrix} x^v (1-x)^{n-v}, \quad 0 < t \leq 1, \tag{1.5.6}$$

$$K_n(x,0) = 0$$

que es constante en algún intervalo $v/n \leq t < (v+1)/n, v = 0, 1, \ldots, n-1$ y tiene el salto

$$\begin{pmatrix} n \\ v \end{pmatrix} x^v (1-x)^{n-v}$$

que es el punto básico de la interpolación $t = v/n$. En este sentido la teoría de los polinomios de Bernstein, además de la teoría de las series de Fourier, son un capítulo de la teoría de integrales singulares.

Los polinomios de Bernstein están conectados con la teoría de la probabilidad, con problemas de momentos y con la teoría de sumas en series divergentes. Un problema complejo e interesante que no ha sido completamente resuelto, concierne a los polinomios de Bernstein en funciones analíticas.

La expresión:

$$p_v = p_{n,v}(x) = \begin{pmatrix} n \\ v \end{pmatrix} x^v (1-x)^{n-v} \tag{1.5.7}$$

contenida en la Ecuación 1.5.2 es la binomial o las probabilidades de Newton conocidas en la teoría de la probabilidad. Si $0 \leq x \leq 1$ es la probabilidad de un evento E, entonces $p_{n,v}(x)$ es la probabilidad de que E debe ocurrir exactemente v veces en n intentos independientes.

La relación que existe entre la teoría del campo promedio y los polinomios de Bernstein se establece entre las Ecuaciones 1.5.1 y 1.5.7. La Ecuación 1.5.1 genera el polinomio completo derivado de todas las evoluciones de cada una de las vecindades, es decir, analiza todos los términos que puede generar la regla de evolución, por otra parte el polinomio de Bernstein de la Ecuación 1.5.7 calcula todas las combinaciones que se pueden generar de un estado en una vecindad de tamaño $2r+1$, ésto da como resultado un término del polinomio de la Ecuación 1.5.1 en específico para un n y v dado, donde n tomará el valor $2r+1$ y v tomará el número de veces que un estado de Σ puede estar en la vecindad, es decir, v tomará valores entre $0 \leq v \leq 2r+1$.

Finalmente los polinomios de Bernstein ayudan a no calcular todos los términos posibles del polinomio, como lo hace la teoría del campo promedio, simplemente calculando las combinaciones para un número de veces en que estará un estado en la vecindad, excluyendo todos aquellos términos que no formarán parte del polinomio para una regla de evolución en específico, aunque se sigue empleando la sumatoria pero ahora de binomiales. Esta sumatoria a diferencia de la Ecuación 1.5.1, solo sumará los términos que son calculados por las binomiales.

Para ejemplificar estos conceptos se construye el polinomio de un autómata celular en una dimensión de orden $(k=2, r=1)$ *regla 110*. El conjunto de estados $\Sigma = \{0,1\}$, la vecindad es de tamaño $2r+1 = 2(1)+1 = 3$ y el número de vecindades es igual a $k^{2r+1} = 2^3 = 8$. Entonces la regla de evolución para la *regla 110* se representa como:

$$\begin{aligned}
\varphi(0,0,0) &= 0 \\
\varphi(0,0,1) &= 1 \\
\varphi(0,1,0) &= 1 \\
\varphi(0,1,1) &= 1 \\
\varphi(1,0,0) &= 0 \\
\varphi(1,0,1) &= 1 \\
\varphi(1,1,0) &= 1 \\
\varphi(1,1,1) &= 0
\end{aligned}$$

las vecindades son $\Sigma^{2r+1} = \{000, 001, 010, 011, 100, 101, 110, 111\}$, p es la probabilidad de obtener el estado 1 en la siguiente generación, el complemento de la probabilidad de p es q que es igual a $q = 1 - p$ y q es la probabilidad de obtener el estado 0 en la siguiente generación, por lo tanto $p + q = 1$ donde p y q tomarán valores del intervalo cerrado $[0,1]$.

Empleando la Ecuación 1.5.1, $\varphi(X)$ tomará el valor que produzcan las células de evolución para cada una de las vecindades de Σ^{2r+1}, p_t la probabilidad de obtener el estado 1 en la siguiente generación, $(1-p_t)$ la probabilidad de obtener el estado 0 en la siguiente generación, v es el número de veces que aparece el estado 1 en la vecindad y $n - v$ es el número de veces que aparece el estado 0 en la vecindad, de esta manera se tiene que:

$$\begin{aligned}
p_{t+1} &= (0)p_t^0(1-p_t)^3 + (1)p_t^1(1-p_t)^2 + (1)p_t^1(1-p_t)^2 \\
&\quad +(1)p_t^2(1-p_t)^1 + (0)p_t^1(1-p_t)^2 + (1)p_t^2(1-p_t)^1 \\
&\quad +(1)p_t^2(1-p_t)^1 + (0)p_t^3(1-p_t)^0
\end{aligned}$$

simplificando el polinomio se tiene que:

$$p_{t+1} = 2p_t(1-p_t)^2 + 3p_t^2(1-p_t)$$

poniendolo en términos de p y q finalemente se obtiene el polinomio de la *regla 110*:

$$p_{t+1} = 2p_t q_t^2 + 3p_t^2 q_t$$

como se había mencionado la sumatoria calcula todas las vecindades que deriva la regla de evolución, ahora se emplearán los polinomios de Bernstein de la Ecuación 1.5.7 para ver que resultado produce.

El grado del polinomio n es igual al tamaño de la vecindad, entonces $n = 2r+1 = 3$ y v tomará valores de $v = 0, 1, 2, 3$ que es el número de veces que puede aparecer el estado 1 en la vecindad, $x = p$ y $1 - x = q$.

Para $n = 3$ y $v = 0$ se tiene:

$$\begin{aligned} p_{0,3} &= \begin{pmatrix} 3 \\ 0 \end{pmatrix} x^0(1-x)^{3-0} = \left(\frac{3!}{0!(3-0)!}\right)(1)(1-x)^3 \\ &= \left(\frac{3!}{3!}\right)(1-x)^3 = (1-x)^3 \end{aligned}$$

en términos de p y q finalmente tenemos:

$$p_{0,3} = q^3.$$

Para $n = 3$ y $v = 1$ se tiene:

$$\begin{aligned} p_{1,3} &= \begin{pmatrix} 3 \\ 1 \end{pmatrix} x^1(1-x)^{3-1} = \left(\frac{3!}{1!(3-1)!}\right) x(1-x)^2 \\ &= \left(\frac{3!}{2!}\right) x(1-x)^2 = 3x(1-x)^2 \end{aligned}$$

en términos de p y q finalmente tenemos:

$$p_{1,3} = 3pq^2.$$

Para $n = 3$ y $v = 2$ se tiene:

$$\begin{aligned} p_{2,3} &= \begin{pmatrix} 3 \\ 2 \end{pmatrix} x^2(1-x)^{3-2} = \left(\frac{3!}{2!(3-2)!}\right) x^2(1-x)^1 \\ &= \left(\frac{3!}{2!}\right) x^2(1-x) = 3x^2(1-x) \end{aligned}$$

en términos de p y q finalmente tenemos:

$$p_{2,3} = 3p^2q.$$

Para $n = 3$ y $v = 3$ se tiene:

$$\begin{aligned} p_{3,3} &= \begin{pmatrix} 3 \\ 3 \end{pmatrix} x^3(1-x)^{3-3} = \left(\frac{3!}{3!(3-3)!}\right) x^3(1-x)^0 \\ &= \left(\frac{3!}{0!}\right) x^3 = x^3 \end{aligned}$$

en términos de p y q finalmente se tiene que:

$$p_{3,3} \quad = \quad p^3.$$

Nótese que los polinomios de Bernstein calculan todos los términos que puede generar el polinomio de la Ecuación 1.5.1, estos términos no toman en cuenta la regla de evolución, sin embargo empleando los dos conceptos se puede simplificar el calculo del polinomio de la teoría del campo promedio para reglas de evolución muy grandes.

La manera en que se simplifica el proceso es calculando los polinomios de Bernstein que determinan la probabilidad de obtener el estado 1 en la siguiente generación, es decir, tomando en cuenta aquellas vecindades que evolucionan al estado 1 y que la suma del estado 1 en cada una de las vecindades correspondan de igual manera, este concepto solo es utilizado para reglas de evolución totalísticas y semitotalísticas. Si las reglas de evolución no fueran de estos dos tipos, entonces se tiene que aplicar la Ecuación 1.5.1 y este método no puede ser aplicado.

1.6 Comentarios finales

Los autómatas celulares en una dimensión son los mas estudiados en la literatura de autómatas, su formalización a contribuido enormemente a encontrar aplicaciones en diferentes áreas científicas, son modelos sencillos y fáciles de implementar en una computadora, recomendamos las siguientes referencias que tratan sobre la Sección 1.1 [28], [31], [33], [41], [57], [58] y [59]. La formalización en autómatas celulares de dos dimensiones no es totalmente general, sin embargo se tratan de emplear nuevas herramientas que permitan establecer una representación mas rigurosa, los estudios realizados hasta ahora se han realizado de manera estadística, recomendamos las siguientes referencias que tratan sobre la Sección 1.2 [39], [48] y [53]. Los autómatas celulares en tres dimensiones son aún mas difíciles de analizar con herramientas estadísticas, la importancia de analizarlos es por su relación que existe con el mundo real que se encuentra en tres dimensiones, recomendamos las siguientes referencias que tratan sobre la Sección 1.3 [3], [21], [24] y [30]. Se puede ver que la clasificación de Wolfram no es general y es importante tratar de describir estos comportamientos, existen muchas diferencias tanto a nivel fenotípico como estadístico, recomendamos las siguientes referencias que tratan sobre la Sección 1.4 [3], [15], [21], [56] y [60]. La teoría del campo promedio encuentra una relación directa con autómatas celulares, en representar el comportamiento de los estados con respecto a su densidad a través del tiempo y poder obtener una descripción mas representativa de las reglas de evolución, recomendamos las siguientes referencias que tratan sobre la Sección 1.5 [22], [23], [24], [25], [26], [27], [37], [44] y [55].

Capítulo 2

Analizando el modelo original "The Game of Life" con la teoría del campo promedio

En este capítulo se presenta el autómata celular en dos dimensiones conocido como "*The Game of Life*", llamado así por su precursor Conway. En la Sección 2.1 se presenta la estructura de esta regla semitotalística como la describe Conway en [10]. En la Sección 2.2 se ilustran algunas de las estructuras mas interesantes conocidas en *Life*. En la Sección 2.3 se muestra una variante de *Life* conocido como *HighLife* que también es un autómata celular de regla semitotalística que evoluciona en dos dimensiones. En la Sección 2.4 se analizan estas reglas de evolución con la teoría del campo promedio determinando su caracterización a través de la curva de probabilidad en la identidad; en particular se hace énfasis en la regla de *Life*, para determinar el comportamiento de los estados a nivel global dentro del espacio de evoluciones al límite y de esta manera establecer una relación con autómatas celulares en tres y una dimensión que tengan características similares a *Life*.

2.1 Estructura de la regla que representa a $Life$

El autómata celular de Conway en dos dimensiones llamado "*The Game of Life*", es llamado así por su relación que tiene con la modelación de sistemas biológicos, tal como ecosistemas, incendios forestales, reacciones químicas, comportamientos colectivos de seres vivos, entre otros. Sea $\Sigma = \{0, 1\}$ el conjunto de estados, el estado 0 representa una célula muerta (color blanco) y el estado 1 representa una célula viva (color negro).

La función de trancisión φ emplea la vecindad de Moore Figura 1.5, la regla de evolución es semitotalística y es representada como $R(2, 3, 3, 3)$, la primera pareja de números 2,3 son un mínimo y un máximo respectivamente de células vivas necesarias para definir la sobrevivencia de una célula que está viva, la segunda pareja de números 3,3 también son un mínimo y un máximo respectivamente de células vivas necesarias para definir el nacimiento de una célula, entonces las variables de la Equación 1.2.1 toman los valores de $S_{min} = 2$, $S_{max} = 3$, $N_{min} = 3$ y $N_{max} = 3$, por eso es que S representa la sobrevivencia en el tiempo $t + 1$ de una célula que se encuentra viva en el tiempo t y N representa el nacimiento de una célula en el tiempo $t + 1$ de una célula que se encontraba muerta en el tiempo t.

Conway hace un amplio análisis para poder determinar una regla de evolución que tuviera dos características fundamentales:

1. Una configuración c_i no debe desaparacer rápidamente.

2. Una configuración c_i debe crecer ilimitadamente.

además se deben de definir tres condiciones, cuándo una célula debe de *nacer*, *sobrevivir* o *morir*. Estas condiciones son muy importantes para obtener las características mencionadas.

(a) **Nacimiento**. Una célula muerta en el tiempo t llega a vivir en el tiempo $t+1$ si tiene exactamente tres de sus vecinos vivos en el tiempo t.

(b) **Muerte por sobre-población**. Si una célula vive en el tiempo t y tiene cuatro o mas de sus ocho vecinos vivos en el tiempo t, esta célula debe morir en el tiempo $t+1$.

(c) **Muerte por aislamiento**. Si una célula vive en el tiempo t y tiene un vecino vivo o ninguno en el tiempo t, esta célula debe morir en el tiempo $t+1$.

(d) **Sobrevivencia**. Una célula que vive en el tiempo t deberá permanecer viva en el tiempo $t+1$, si y solo si, tiene dos o tres vecinos vivos en el tiempo t.

Por lo tanto la Equación 1.2.1 para la regla $R(2,3,3,3)$ en particular queda como:

$$\varphi(\mathbf{x}_0,\mathbf{x}_1,\ldots,\mathbf{x}_8) = \begin{cases} 1 \quad si & \begin{cases} \mathbf{x}_0 = 0 \quad y \quad 3 \leq \sum\limits_{i=1}^{8}\mathbf{x}_i \leq 3 \\ \\ \mathbf{x}_0 = 1 \quad y \quad 2 \leq \sum\limits_{i=1}^{8}\mathbf{x}_i \leq 3 \end{cases} \\ \\ 0 \quad \text{en otro caso} \end{cases} \tag{2.1.1}$$

se tiene que $3 \leq \sum_{i=1}^{8}\mathbf{x}_i \leq 3$ es el intervalo de células vivas entre N_{min} y N_{max} para un nacimiento y $2 \leq \sum_{i=1}^{8}\mathbf{x}_i \leq 3$ es el intervalo de células vivas entre S_{min} y S_{max} para la sobrevivencia. Las células se encuentran en nuestro espacio de evoluciones $\mathbb{Z}\otimes\mathbb{Z}$ donde las transformaciones para cada una de las células $x_{i,j}$ se efectúan de manera simultánea o en paralelo.

2.2 Comportamientos en $Life$

El interés que se originó en *Life* se debe principalmente a los comportamientos complejos que la regla puede producir, llegando a resultados muy importantes como fue el hecho de demostrar que la regla *Life* puede realizar computación universal [10].

En el espacio de evoluciones existen cuatro tipos de comportamientos:

1. Configuraciones que desaparecen.

2. Configuraciones estáticas.

3. Configuraciones periódicas.

4. Configuraciones periódicas con desplazamiento.

evolucionando sobre un fondo estable de puros 0's. En la literatura de *Life* se les han asignado nombres a cada una de estas estructuras como: *blinker, blanker, block, beehive, loaf, pond, tub, traffic lights, ships, baker, barber, train, beacon maker, bigship, hertz, Z, rocket, honey farm, snake, barge, boat, glider, flip flop, pants, glider gun, ticks, flashes, spaceships, boat maker, chesire cat, mega flip flop, pool table, pin wheel, catherine wheel, puffer, candelabra, mosaic kills virus, latin cross, toad, r pentomino, clock, swastika, etc.* [18] [19].

2.2.1 Configuraciones que desaparecen

Son estructuras de células vivas que desaparecen en pocas iteraciones, es decir, todas las células vivas van muriendo como se ilustra en la Figura 2.1.

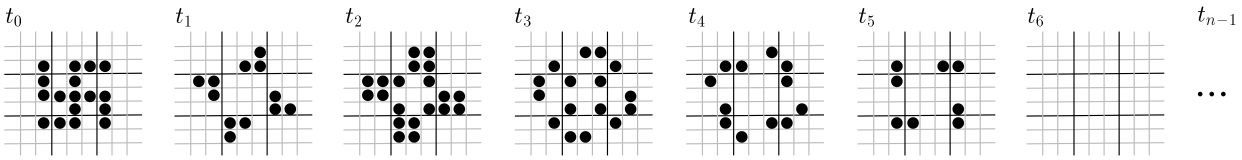


Figura 2.1: Configuración que desaparece, estructura *swastika*

La cruz *swastika* desaparece por completo a patir de la sexta generación, se puede observar la simetría que existe en el espacio de evoluciones a través del tiempo. En particular el autómata celular *Life* es uniforme, determinístico y simétrico.

2.2.2 Configuraciones fijas

Una configuración es estable a través del tiempo, si ésta no cambia su forma ni desaparece. A todas estas configuraciones se les conoce como *naturaleza muerta* o *still life*.

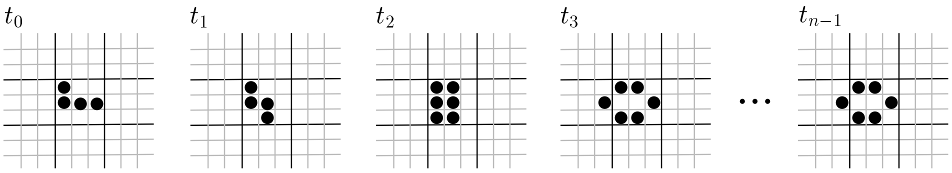


Figura 2.2: Configuración estática, estructura estable *beehive*

A partir de la tercera generación la estructura de nombre *beehive* compuesta de seis células, se mantiene estática a través del tiempo. *Life* tiene varias formas de estas estructuras, si se evoluciona una configuración aleatoria muchas de estas estructuras se producen casi inmediatamente.

2.2.3 Configuraciones periódicas

Son estructuras con un período dado y las configuraciones que producen estas estructuras son fijas. En la Figura 2.3 se puede ver como una población compuesta de tres células cambia de su posición horizontal a una posición vertical a través del tiempo.

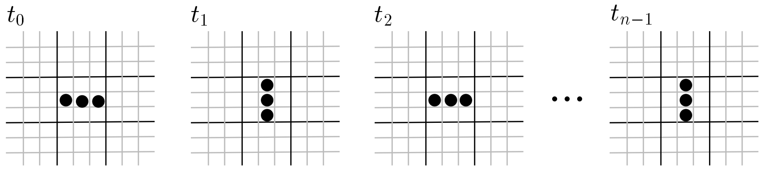


Figura 2.3: Configuración periódica, estructura periódica *blinker*

2.2.4 Configuraciones periódicas con desplazamiento

Este tipo de estructuras son muy interesantes e importantes en esta regla. Muchos de los descubrimientos mas relevantes están relacionados a este tipo de estructuras, determinando las trayectorias que pueden tener a través del espacio $\mathbb{Z} \otimes \mathbb{Z}$ y que comportamientos o estructuras tan complejas se pueden llegar a producir cuando estos *gliders* chocan unos con otros. Además estos choques se pueden realizar con cualquier otra estructura, varias de estas estructuras complejas que se han llegado a obtener se han logrado a través de la computadora paralela CAM (Cellular Automata Machine), desarrollada en el MIT por Tommaso Toffoli y Norman Margolus [53]. El *glider* mas relevante en la literatura de *Life* es el *glider* de cinco células Figura 2.4, además este *glider* tiene la característica de ser totalmente simétrico en un arreglo de 3×3, su periódo de desplazamiento es en cuatro iteraciones.

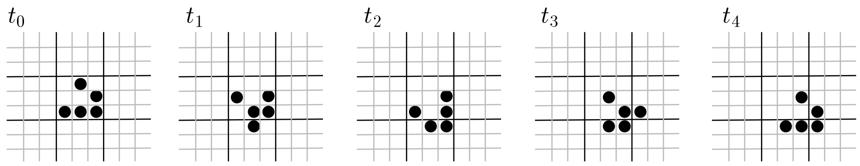


Figura 2.4: Configuración periódica con desplazamiento, *glider*

El *glider* de la Figura 2.4 tiene una población total de cinco células y esta población se conserva en cada paso. Es interesante notar que este *glider* mantiene el mismo número de células vivas en cada iteración y su desplazamiento es de forma diagonal con ocho trayectorias diferentes. Existen otros tipos de *gliders* que varian de forma, tamaño, período de desplazamiento y dirección en el espacio de evoluciones, por lo tanto comportamientos de este tipo fueron los que originaron un interés mas a fondo para poder explicar el origen y el alcance de estas estructuras.

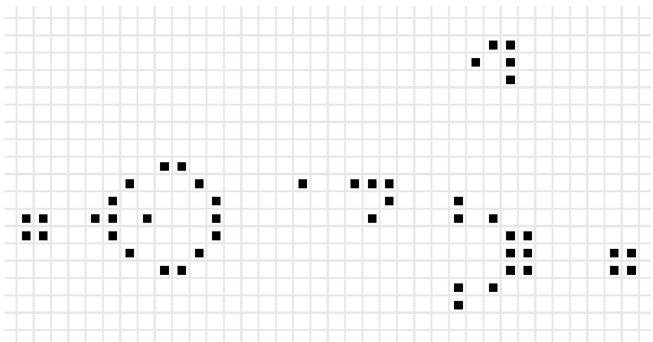


Figura 2.5: *Glider gun*

El *glider gun* de la Figura 2.5 es una estructura que produce *gliders* como el de la Figura 2.4, de manera continua cada treinta iteraciones. Pero además algo verdaderamente sorprendente es que treinta *gliders* como el de la Figura 2.4 pueden construir su propio *glider gun* [19], es decir, crece ilimitadamente y puede reproducirse así mismo.

2.3 Una variante de *Life* en dos dimensiones *HighLife*

El estudio de *Life* originó que otros investigadores buscaran reglas de evolución que tuvieran comportamientos tan complejos como dicha regla, fue así como se mostro una variante de la regla de Conway llamada *HighLife* por su precursor David I. Bell en [9]. Esta regla de evolución muestra comportamientos complejos similares a los de *Life* y se representa como $R(2,3,3,6)$, por ejemplo una configuración compuesta de 7 células vivas como la de la Figura 2.6:

Figura 2.6: Configuración incial para la regla *HighLife*

produce una estructura que es simétrica en cada una de sus iteraciones a través del tiempo, en la Figura 2.7 se puede ver su estructura despues de 195 evoluciones y con una población de 3820 células vivas.

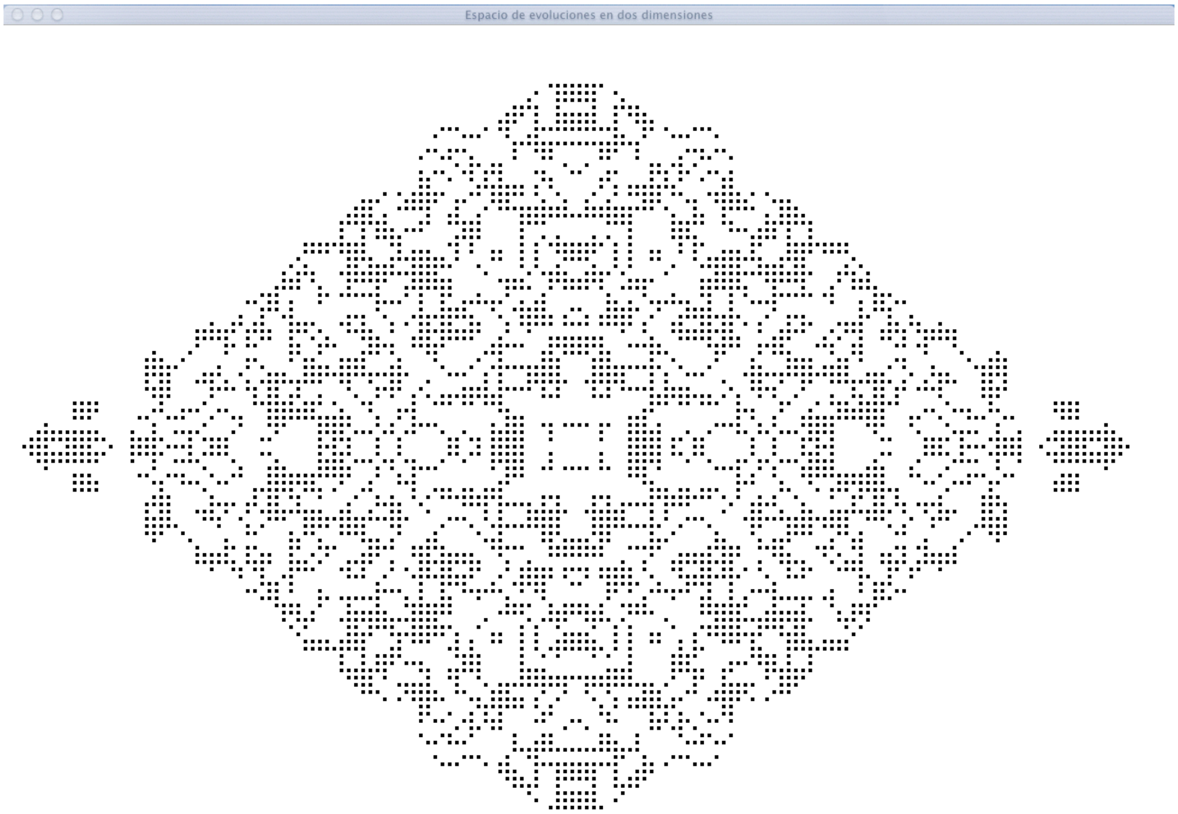


Figura 2.7: Comportamientos en *HighLife*

Este autómata celular tiene una diferencia importante con respecto a *Life*, casi desde cualquier configuración inicial, el autómata celular crece ilimitadamente, cubriendo todo el plano de manera rápida. Si el espacio de evoluciones empieza con una configuración aleatoria, despues de varias iteraciones su comportamiento no presenta un patrón característico, como por ejemplo las regiones aisladas que existen en *Life* que además son alcanzadas prácticamente desde cualquier configuración aleatoria. En la Figura 2.8 se muestran algunas configuraciones periódicas o estables en el espacio de evoluciones.

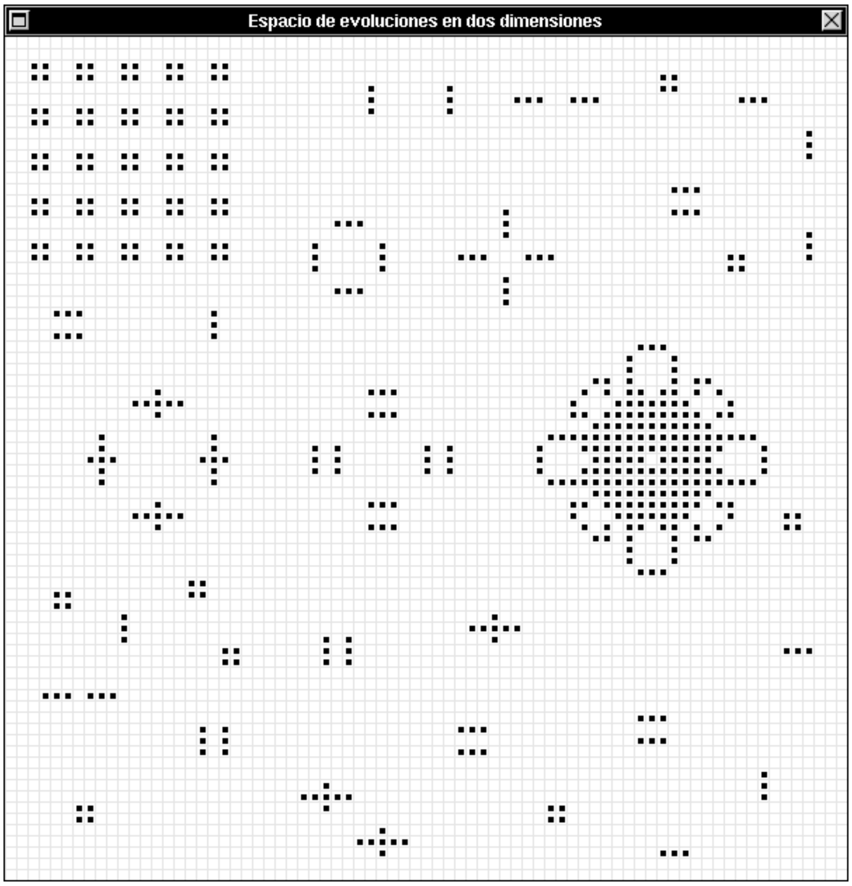


Figura 2.8: Naturaleza muerta y *blinkers* en *HighLife*

La función de la Ecuación 1.2.1 para la regla *HighLife* queda como:

$$\varphi(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_8) = \begin{cases} 1 \quad si & \begin{cases} \mathbf{x}_0 = 0 \quad y \quad 3 \leq \sum_{i=1}^{8} \mathbf{x}_i \leq 6 \\ \\ \mathbf{x}_0 = 1 \quad y \quad 2 \leq \sum_{i=1}^{8} \mathbf{x}_i \leq 3 \end{cases} \\ \\ 0 \quad \text{en otro caso} \end{cases} \tag{2.3.1}$$

donde la diferencia que existe con la regla de Conway es que $N_{max} = 6$, por esta razón es que *HighLife* crece rápidamente en pocas iteraciones, hasta cubrir el espacio bidimensional completamente, es decir, no significa que el espacio de evoluciones se llene de puros 1's, mas bien que el número de células vivas es mas que el de células muertas.

2.4 Aplicando la teoría del campo promedio

2.4.1 Analizando la regla $Life$ $R(2,3,3,3)$

La teoría del campo promedio ha sido empleada en [16], [22], [24], [30], [44] y [50], para tratar de explicar el comportamiento de los estados de las reglas de evolución, la mayoría de estos estudios describen el comportamiento de las células a través de su densidad en el espacio de evoluciones en el tiempo. El análisis propone un enfoque sencillo y práctico de como obtener dichas densidades e interpretar fácilmente los resultados, tal como lo plantean McIntosh en [44] y Chaté-Manneville en [16], graficando su curva de probabilidad en el tiempo inicial y calculando sus curvas de probabilidad a través del tiempo, sustituyendo iterativamente el polinomio de la Equación 1.5.1.

Se construye el polinomio de la teoría del campo promedio paso a paso para ver con claridad la relación que existe entre las Ecuaciones 1.5.1 y 1.5.7, y las variables definidas por S_{min}, S_{max}, N_{min} y N_{max}. La regla *Life* es la regla semiltotalística $R(2,3,3,3)$ entonces $S_{min}=2$, $S_{max}=3$, $N_{min}=3$ y $N_{max}=3$, el número de configuraciones que se pueden generar en la vecindad de Moore cuando $S_{min}=2$ es igual a 28, como se ilustra en la Figura 2.9.

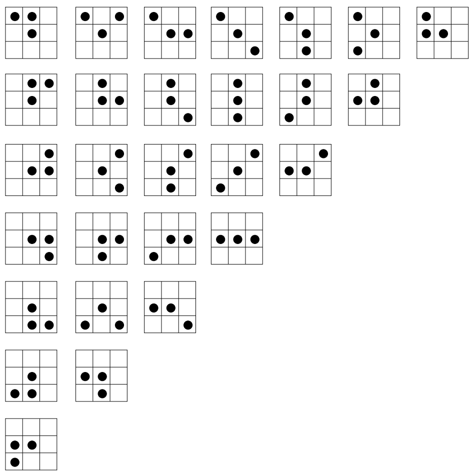


Figura 2.9: Configuraciones que determinan la sobrevivencia cuando $S_{min}=2$

La transformación local cuando $S_{min}=2$ determina 28 vecindades que se transforman a 1 en la siguiente generación, donde la célula central en este caso es igual a 1, la Figura 2.9 ilustra todas las combinaciones que se pueden generar en la vecindad de Moore, cuando la célula central está viva y se tienen dos vecinos vivos para que en la siguiente generación la célula central siga viva. Empleando notación de las Ecuaciones 1.5.1 y 1.5.7 se representa el polinomio del campo promedio de la siguiente manera:

$$p_{t+1}=\sum_{v=S_{min}}^{S_{max}}\left(\begin{array}{c}n-1\\v\end{array}\right)p_t^{v+1}(1-p_t)^{n-v-1}+\sum_{v=N_{min}}^{N_{max}}\left(\begin{array}{c}n-1\\v\end{array}\right)p_t^{v}(1-p_t)^{n-v} \qquad (2.4.1)$$

donde n representa el número de células que tiene una vecindad, $n-1$ son las combinaciones sin tomar en cuenta la célula central, v indica el número de veces que aparece el estado 1 en la vecindad de Moore, $n-v$ indica el número de veces que aparece el estado 0 en la vecindad de Moore, p_t es la probabilidad de tener 1's en la siguiente generación, $1-p_t$ es la probabilidad de tener 0's en la siguiente generación, la combinatoria

$\begin{pmatrix} n-1 \\ v \end{pmatrix}$ dará las constantes para cada uno de los términos del polinomio. La primera sumatoria calculará todas las combinaciones que se derivan del intervalo que definen S_{min} y S_{max} en el caso de la sobrevivencia. La segunda sumatoria calculará todas las combinaciones que se derivan del intervalo que definen N_{min} y N_{max} en el caso de los nacimientos. En el caso de la sobrevivencia al exponente v se le suma un elemento, porque al momento de definir la cantidad de 1's en la vecindad la célula central está viva. En el caso de los nacimientos v representa el número de veces que aparece el estado 1 en la vecindad y no se le suma ningún elemento porque la célula central está muerta.

La probabilidad de que se obtengan 1's es igual a p, para garantizar su complemento q es la probabilidad de que se obtengan 0's, por lo tanto $p + q = 1$ y $q = 1 - p$. Simplificando la Ecuación 2.4.1 empleando la variable q se tiene finalmente:

$$p_{t+1} = \sum_{v=S_{min}}^{S_{max}} \begin{pmatrix} n-1 \\ v \end{pmatrix} p_t^{v+1} q_t^{n-v-1} + \sum_{v=N_{min}}^{N_{max}} \begin{pmatrix} n-1 \\ v \end{pmatrix} p_t^{v} q_t^{n-v} \tag{2.4.2}$$

El exponente v para la variable p_t se calcula sumando el número de células ocupadas por el estado 1 dentro de la vecindad de Moore, el exponente $n-v$ para la variable q se calcula sumando el número de células ocupadas por el estado 0 dentro de la vecindad de Moore, entonces se tiene que $v = 3$ y $n - v = 9 - 3 = 6$, cuando $S_{min} = 2$. Nótese que las combinaciones se realizan en la vecindad isotrópica $\mathcal{V}$ (en los vecinos), donde la célula central $\mathbf{x}_0$ es ocupada por el estado 1, en realidad al momento de calcular las configuraciones no importa si $\mathbf{x}_0$ es ocupada por el estado 1 o el estado 0, sin embargo $\mathbf{x}_0$ es importante al momento de definir los exponentes del polinomio.

Ahora se realiza el cálculo cuando $S_{max} = 3$ como se ilustra en la Figura 2.10.

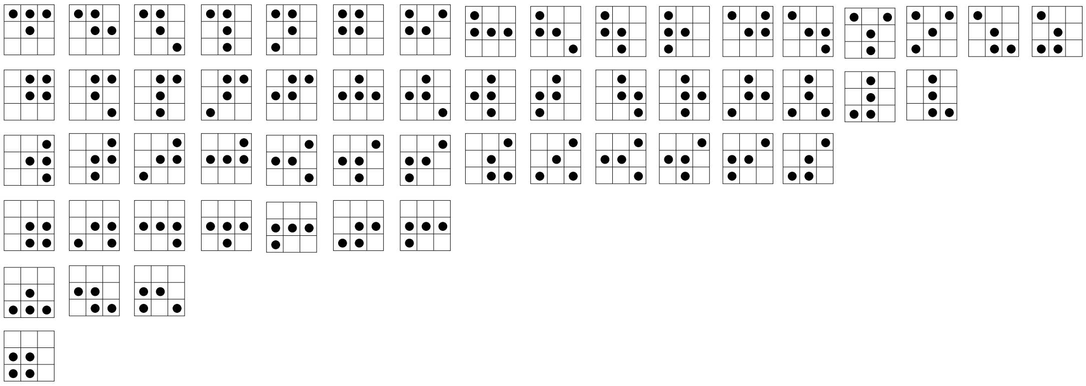


Figura 2.10: Configuraciones que determinan la sobrevivencia cuando $S_{max} = 3$

Cuando $S_{max} = 3$ se generan 56 vecindades que se transforman al estado 1 en la siguiente generación dentro de la vecindad de Moore, éstas son combinaciones sin repetición. Por lo tanto se calcula el número de combinaciones empleando el coeficiente binomial:

$$\begin{pmatrix} n-1 \\ v \end{pmatrix} = \frac{(n-1)!}{v!((n-1)-v)!} \tag{2.4.3}$$

donde $n = 2r + 1 = \mathcal{V}$ y v tomará valores de $v = \{0, 1, \ldots, n-1\}$. Por ejemplo para el caso de la Figura 2.9 se tiene que $v = S_{min} = 2$ y $n - 1 = \mathcal{V} = 8$, entonces se tiene que:

$$\begin{pmatrix} 8 \\ 2 \end{pmatrix} = \frac{(9-1)!}{2!((9-1)-2)!} = 28$$

donde este número representa las combinaciones sin repetición que se generan cuando $S_{min} = 2$ en la vecindad isotrópica $\mathcal{V} = 8$ o los vecinos de la vecindad de Moore. En el caso de la Figura 2.10 se tiene que $S_{max} = 3$, entonces se tiene que:

$$\begin{pmatrix} 8 \\ 3 \end{pmatrix} = \frac{(9-1)!}{3!((9-1)-3)!} = 56$$

donde este número representa las combinaciones sin repetición que se generan cuando $S_{max} = 3$ en la vecindad isotrópica $\mathcal{V} = 8$ o los vecinos de la vecindad de Moore. Nótese que para el caso cuando $N_{min} = 3$ y $N_{max} = 3$, se tienen 56 combinaciones igual que cuando $S_{max} = 3$, ésto se debe porque las combinaciones se realizan en las células de $\mathcal{V}$, es decir, los vecinos de la vecindad sin tomar en cuenta la célula central $\mathbf{x}_0$, por lo que las combinaciones de $S_{max} = 3$ son iguales a las combinaciones de $N_{min} = 3$ y $N_{max} = 3$, estos valores van a determinar las constantes del polinomio.

El grado del polinomio se determina por el número de células vivas v y por el número de células muertas $n-v$, que existen en los vecinos $\mathcal{V}$ de las Figuras 2.9 y 2.10, cuando $S_{min} = 2$ se tiene que $v = 3$ y $n-v = 6$, cuando $S_{max} = 3$ se tiene que $v = 4$ y $n-v = 5$, cuando $N_{min} = 3$ se tiene que $v = 3$ y $n-v = 6$ y cuando $N_{max} = 3$ se tiene que $v = 4$ y $n-v = 5$, finalmente el polinomio para la regla de *Life* es:

$$p_{t+1} = 28p_t^3q_t^6 + 56p_t^4q_t^5 + 56p_t^3q_t^6 \tag{2.4.4}$$

simplificando se tiene el polinomio de la teoría del campo promedio para *Life*:

$$p_{t+1} = 84p_t^3q_t^6 + 56p_t^4q_t^5 \tag{2.4.5}$$

la probabilidad que se genera cuando $S_{max} = 3$ es la misma probabilidad cuando $N_{min} = 3$, porque el número de 1's en la vecindad de Moore es igual en los dos casos, por eso se tienen tres términos en el polinomio en lugar de cuatro como pudiera pensarse.

La curva de probabilidad se obtiene poniendo el polinomio de la Ecuación 2.4.5 en términos de p:

$$p_{t+1} = 84p_t^3(1-p_t)^6 + 56p_t^4(1-p_t)^5 \tag{2.4.6}$$

donde p tomará valores de 0 a 1. La curva de probabilidad del campo promedio muestra directamente la información de como se comporta la densidad de los estados en el espacio de evoluciones. La densidad es interpretada a través de la existencia de puntos fijos en la curva de probabilidad que genera el polinomio. A continuación se da la definición de punto fijo.

Definición 2.4.1. Un punto x es periódico para una función f si $f^n(x) = x$ para algún $n \geq 1$ donde $n \in \mathbb{Z}^+$ y se dice que x tiene periódo n bajo f. Si $f(x) = x$, entonces x es llamado un *punto fijo* en f. $\square$

Los puntos fijos pueden ser estables o inestables. Sea f una función diferenciable, si el valor absoluto de la primera derivada $|f'(x)| < 1$, entonces x es un *punto fijo estable*. Si el valor absoluto de la primera derivada $|f'(x)| > 1$ entonces x es un *punto fijo inestable*. Si el valor de la primera derivada $f'(x) = 0$, entonces x es llamado un *punto crítico*, es decir, su tangente es horizontal al eje x en el plano cartesiano.

La gráfica de la Figura 2.11 muestra la curva de probabilidad que produce el polinomio de la regla de evolución *Life*, el eje horizontal q indicará la probabilidad de obtener 0's, el eje vertical p indica la probabilidad de obtener 1's, p y q podrán ser evaluados en el intervalo cerrado $[0, 1]$, por esta razón la diagonal mostrará los punto fijos de la curva al momento de intersectar algún punto de la curva con la diagonal, porque eso significa que $f(x) = x$ en ese punto.

El punto fijo estable que se encuentra en 0 sobre el eje x de la Figura 2.11 indica que en el espacio de evoluciones no hay células vivas, entonces la probabilidad de que se obtengan células vivas en la siguiente generación es 0, ésto es por la condición de que $S_{min} < 2$; por otra parte si el espacio de evoluciones se encuentra lleno de células vivas, entonces la probabilidad de que se obtengan células vivas en la siguiente generación es 0, esto se debe por la condición de que $S_{max} > 3$ y los valores de la curva son igual a 0 conforme nos acerquemos mas a 1 en el eje q.

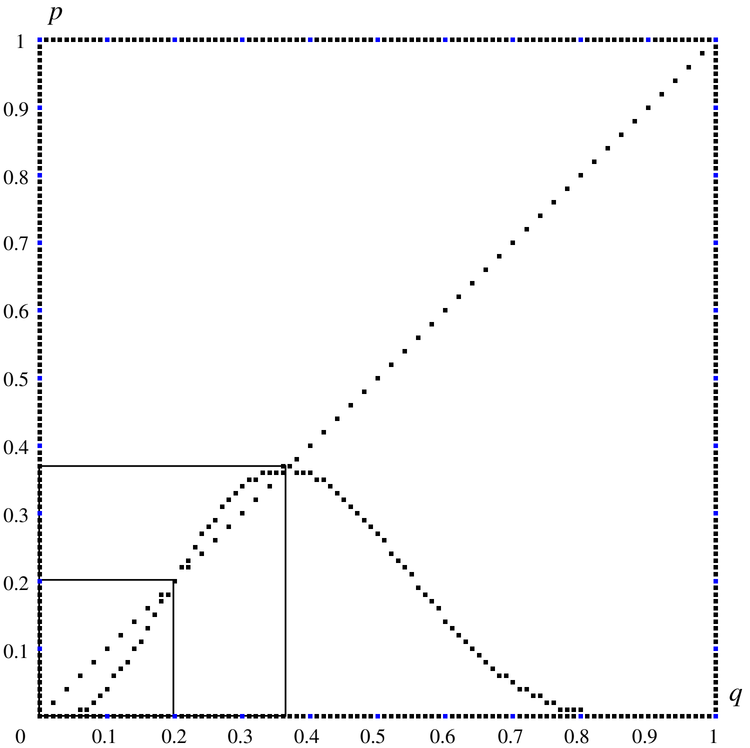


Figura 2.11: Curva de probabilidad de la regla *Life*

Nótese la existencia de otros dos puntos fijos, un punto fijo inestable aproximadamente en 0.2 que indica la existencia de comportamientos inpredecibles dentro de ciertas regiones locales, es decir, la densidad que tiene el autómata en ese momento puede mantenerse igual, crecer rápidamente o desaparecer por completo en unos cuantos pasos. El otro punto fijo es un punto crítico y se encuentra aproximadamente en 0.37 que indica la densidad promedio de células vivas que se pueden conservar en el espacio de evoluciones a través del tiempo. Para obtener el comportamiento en la siguiente generación se sustituye el polinomio p_t de la Ecuación 2.4.6, en él mismo para obtener la curva de probabilidad en p_{t+2} de la siguente manera:

$$\begin{aligned} p_{t+2} \quad = \quad & 84(84p_{t+1}^3(1-p_{t+1})^6 + 56p_{t+1}^4(1-p_{t+1})^5)^3(1-(84p_{t+1}^3(1-p_{t+1})^6 + 56p_{t+1}^4(1-p_{t+1})^5))^6 + \\ & 56(84p_{t+1}^3(1-p_{t+1})^6 + 56p_{t+1}^4(1-p_{t+1})^5)^4(1-(84p_{t+1}^3(1-p_{t+1})^6 + 56p_{t+1}^4(1-p_{t+1})^5))^5 \end{aligned}$$

este polinomio mostrará el comportamiento de la curva en la siguiente generación. Nótese que conforme se desee calcular mas generaciones el polinomio crece exponencialmente, los polinomios para la tercera y cuarta generación ocuparían tres páginas completas. En la Figura 2.12 se grafican las curvas de probabilidad para la segunda, tercera y cuarta generación de la regla de evolución *Life*.

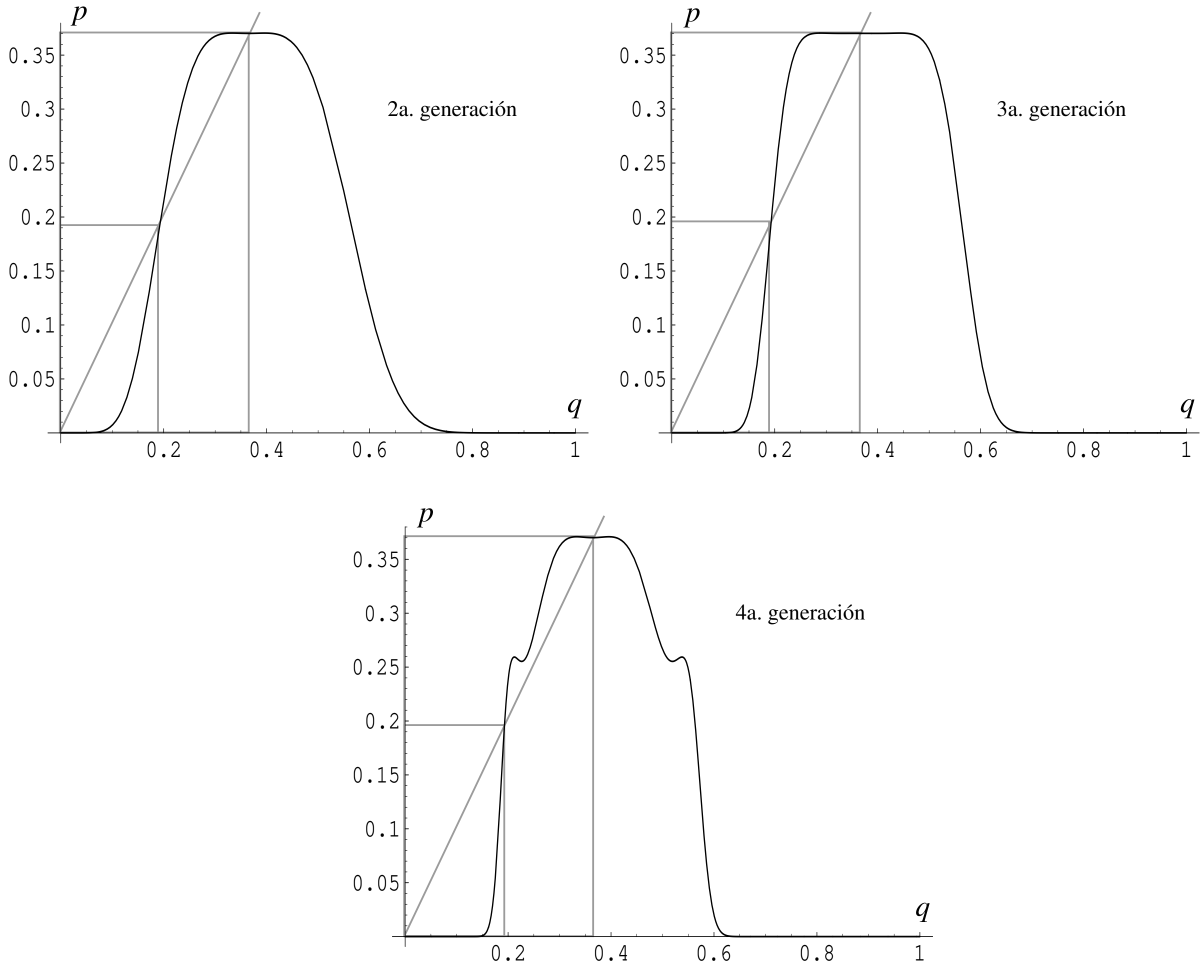


Figura 2.12: Curva de probabilidad de la regla *Life*, segunda, tercera y cuarta generación

El comportamiento de las células en cada generación delimita cada vez mas su densidad en el espacio de evoluciones, en promedio éste es el comportamiento que debe seguir cualquier configuración inicial aleatoria, se puede apreciar en la Figura 2.12 que el intervalo de la curva sobre el eje q se hace cada vez mas estrecho, ésto indica que la población de células vivas es cada vez menor, en efecto así ocurre en *Life*.

En la fase de experimentación las curvas de probabilidad se calcularon sustituyendo el polinomio p_0 en p_1, el polinomio p_0 en p_2 y así sucesivamente. Para tratar de acelerar el proceso en las curvas de probabilidad se sustituyo el polinomio p_0 en p_1, el polinomio p_1 en p_2 y así sucesivamente, pero los resultados al momento de generar la gráfica fueron idénticos, además de que para el segundo caso existe el problema de que el polinomio a calcular es mucho mas grande que para el primer caso.

La densidad promedio de 0.37 calculada para la regla de evolución *Life* coincide con otros resultados obtenidos en [50] y [24]. Todos los demás casos se estudian de la misma manera, para posteriormente establecer una caracterización a través de sus curvas de probabilidad.

2.4.2 Analizando la regla $HighLife$ $R(2, 3, 3, 6)$

La regla de evolución *HighLife* presenta comportamientos muy interesantes como se vio en la Sección 2.3, pero tiene la característica que desde cualquier configuración inicial aleatoria en su evolución no presenta comportamientos interesantes por si solo, como lo hace la regla *Life*.

El polinomio del campo promedio para la regla $R(2, 3, 3, 6)$ queda como:

$$p_{t+1} = 28p_t^3 q_t^6 + 56p_t^3 q_t^6 + 56p_t^4 q_t^5 + 28p_t^6 q_t^3 \tag{2.4.7}$$

simplicando se tiene:

$$p_{t+1} = 84p_t^3 q_t^6 + 56p_t^4 q_t^5 + 28p_t^6 q_t^3. \tag{2.4.8}$$

Se puede suponer que la curva de probabilidad de *HighLife* debe ser mas alta que la de *Life*, ya que el intervalo entre N_{min} y N_{max} originan muchos nacimientos en cada generación, se gráfica su curva del campo promedio en la Figura 2.13.

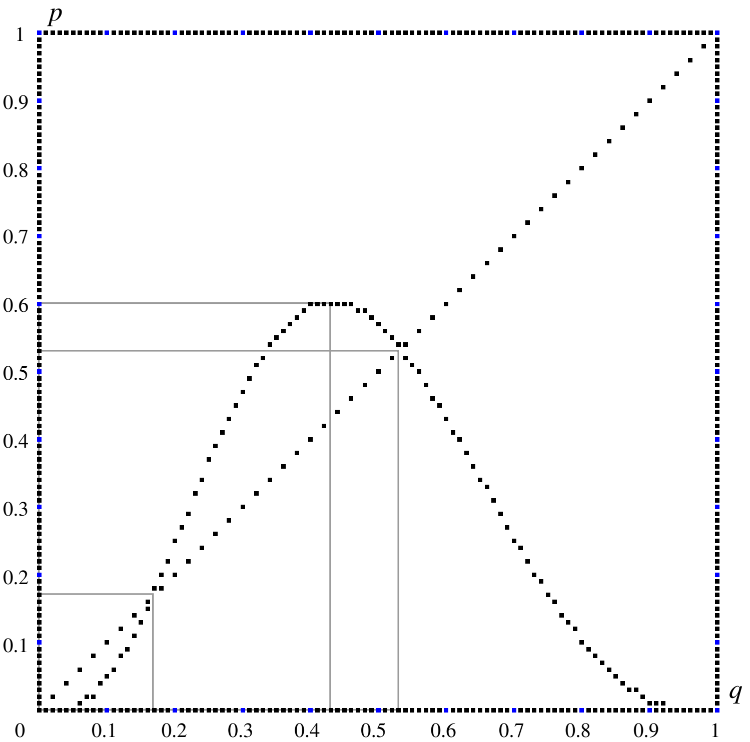


Figura 2.13: Curva de probabilidad de la regla *HighLife*

La Figura 2.13 muestra la curva de probabilidad para la regla *HighLife*, nótese la existencia de dos puntos fijos, un punto fijo inestable aproximadamente en 0.17 que indica la existencia de comportamientos inpredecibles donde la densidad de las células es difícil de definir, el otro punto fijo estable se encuentra aproximadamente en 0.52 que indica la existencia de densidades uniformes en el espacio de evoluciones, es decir, la densidad de células que existe en ese momento se conservará con muy poca alteración en la siguiente generación, ahora comparemos la curva de *HighLife* con la curva de *Life* en la Figura 2.14.

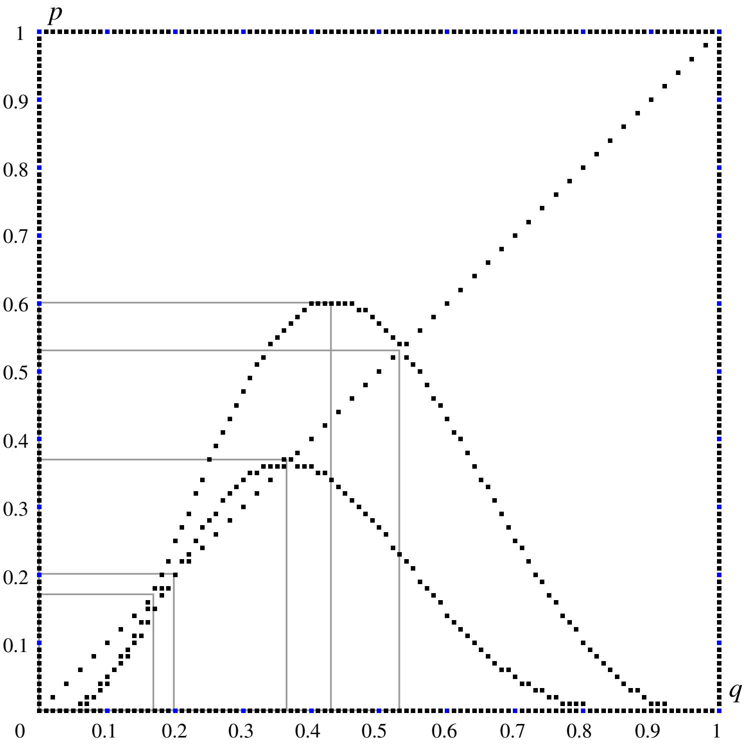


Figura 2.14: Comparando *HighLife* con *Life* en el campo promedio

En la Figura 2.14 se puede ver que la probabilidad promedio de obtener 1's en el espacio de evoluciones de *HighLife* es mas alta que la de *Life*, también una diferencia importante es la existencia de un punto fijo estable que no existe en *Life* y que si existe en *HighLife*, este punto fijo estable determina que las configuraciones aleatorias c_i con una densidad de células vivas en ese momento se mantendrá en las evoluciones siguientes y ésto provoca que se tenga un comportamiento uniforme a través del tiempo a partir del momento en que se alcanza una densidad que no variará en todo el tiempo en que sea evolucionado el autómata celular, es decir, no se tendrá la formación de regiones aisladas dentro del espacio de evoluciones por lo tanto *HighLife* no puede tener configuraciones complejas aleatoriamente, si puede crecer ilimitadamente, no puede desaparecer rápidamente y no puede crear *gliders* de manera natural y su densidad se mantiene uniforme a través del tiempo.

Esto es un problema importante en *HighLife* porque en el supuesto de que pudiera soportar comportamientos tan complejos como los que produce *Life*, entonces estas estructuras tendrían que construirse con mucho cuidado, sin embargo se han probado diversas densidades en varias configuraciones c_i aleatorias, comprobando que el rango de densidades es un poco mas grande que el de *Life*.

Por ejemplo si en la regla de *Life* se trabajan con configuraciones aleatorias de densidades mayores o iguales a 0.8, es muy difícil obtener 1's en la siguiente generación prácticamente todas las configuraciones c_i con esa densidad desaparecen en una evolución. Por otra parte si ambas reglas manejan densidades menores o iguales a 0.01 es poco probable que puedan llegarse a construir cualquier tipo de estructuras, aunque si se manejan densidades que se encuentren en el intervalo $0.03 < c_i < 0.1$ la regla de evolución *Life* puede llegar a generar varias de sus estructuras típicas y rápidamente establecer su densidad; por el contrario con la regla *HighLife* se obtienen estructuras que crecen rápidamente y en unas cuantas generaciones se cubre todo el plano sin mostrar algún patron o estructura interesante, además de que la variedad de estructuras fijas entre ambas reglas es muy notable.

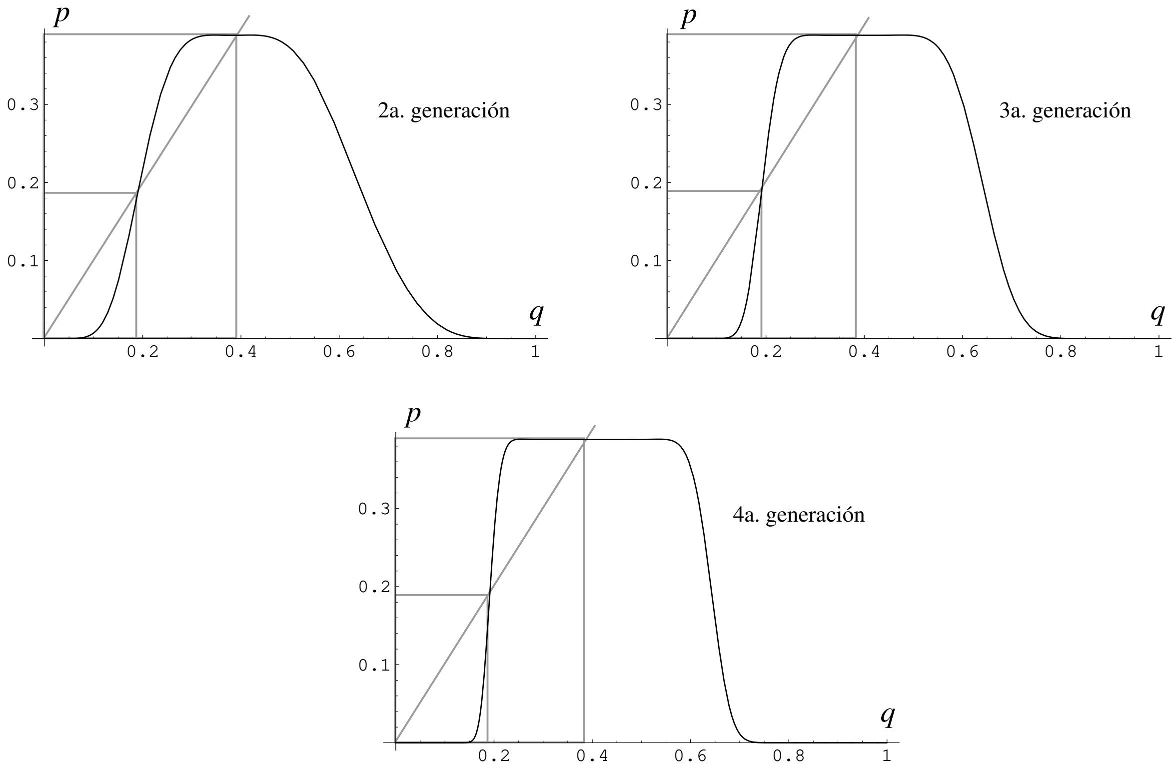


Figura 2.15: Curva de probabilidad de la regla *HighLife*, segunda, tercera y cuarta generación

En la Figura 2.15 se muestran las curvas de probabilidad de *HighLife* para el comportamiento de las estados en el espacio de evoluciones bidimensional en la segunda, tercera y cuarta generación. A diferencia de *Life* la regla *HighLife* amplia el rango de probabilidad de la densidad promedio, donde este rango indica que el número de células que se encuentran vivas se mantendrán vivas en la siguiente generación y en otro paso mas las células vivas serán mas, hasta cubrir la mayor parte de el espacio $\mathbb{Z} \otimes \mathbb{Z}$ pero sin cubrirlo por completo, a diferencia de *Life* la densidad promedio de células vivas se va acotando rápidamente lo que no sucede con *HighLife*.

Los autómatas celulares en dos dimensiones tienen una amplia variedad de reglas con comportamientos interesantes, por lo tanto *HighLife* no es la única variante de *Life*. Michael Magnier, Claude Lattaud y Jean-Claude Heudin en [38] presentan varias reglas de evolución que pueden soportar comportamientos complejos, caóticos, cíclicos o desaparecer rápidamente; algunas de ellas tienen sus propios *gliders* y estructuras periódicas como se ilustra en la Figura 2.16. El enfoque principal de este estudio es mostrar las clases de Wolfram que pueden existir en dos dimensiones, haciendo una comparación en cada una de las clases. Estos comportamientos pudieron ser reproducidos con los programas desarrollados en esta tesis.

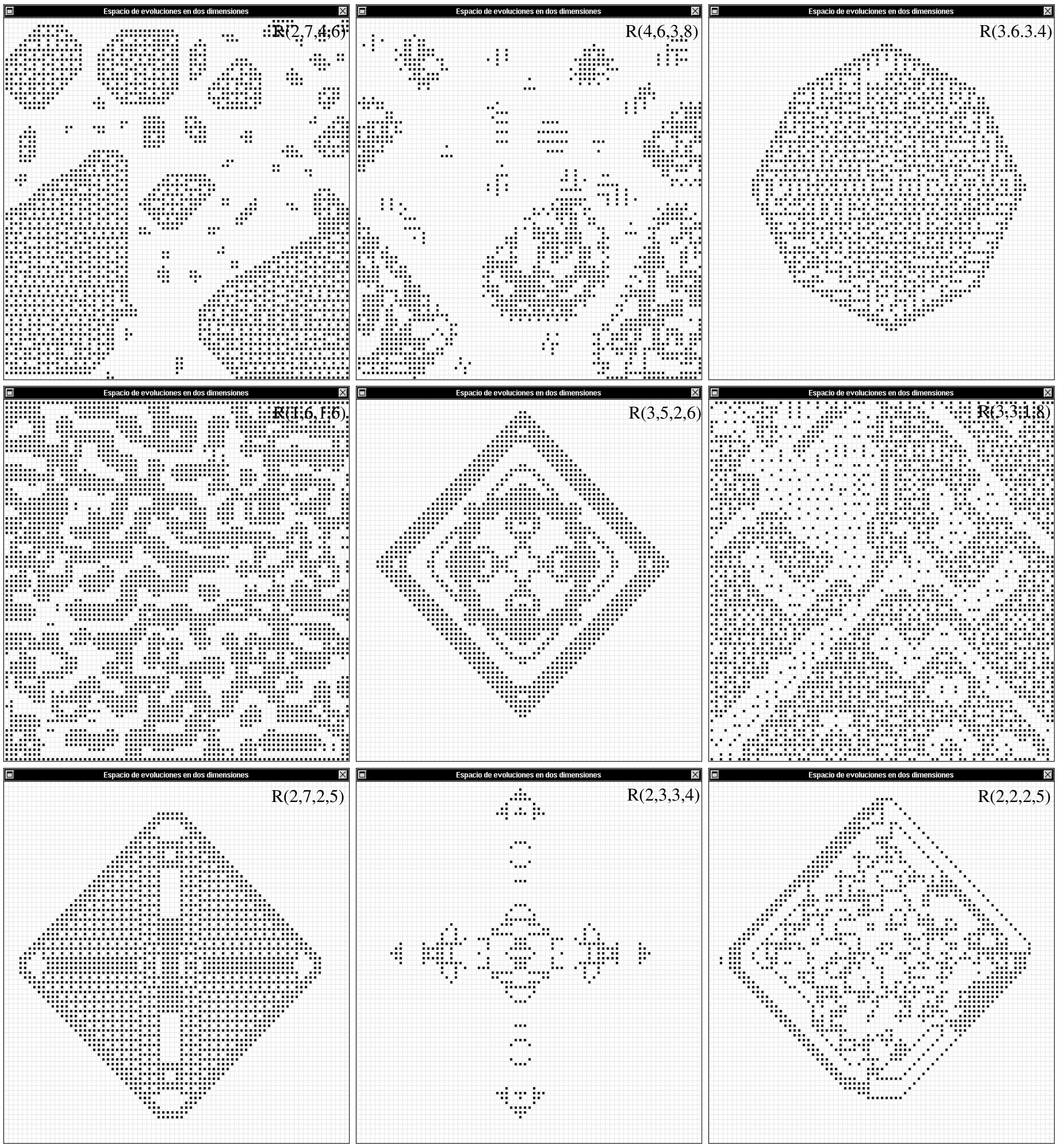


Figura 2.16: Reglas complejas en dos dimensiones

2.5 Comentarios finales

En este capítulo se describió con detalle la estructura de la regla *Life*, ilustrando algunos de sus comportamientos mas importantes y se definió de manera general las reglas semitotalísticas; recomendamos las siguientes referencias que tratan sobre la Sección 2.1 y la Sección 2.2 [10], [18], [19], [29], [42], [43] y [47]. Sin embargo de varias reglas de estudio en dos dimensiones que se han propuesto la mas destacada de ellas es la regla *HighLife*, la cual se definió de la misma manera como se definió *Life*, recomendamos la siguiente referencia que trata sobre la Sección 2.3 [9]. La propuesta fundamental del análisis probabilístico de la tesis con la teoría del campo promedio y que no ha sido reportada de esta manera, se aplica con detalle a *Life* especificando cada uno de los pasos a seguir y explicando con detalle cada uno de los resultados obtenidos, de igual manera se hizo el análisis con *HighLife* y se hace con todas las demás reglas de evolución para autómatas celulares de tres y una dimensión que se analizan, recomendamos las siguientes referencias que tratan sobre la Sección 2.4 [16], [24], [30], [32], [39], [44], [48] y [50].

Capítulo 3

Analizando autómatas celulares en tres dimensiones similares a "The Game of Life" con la teoría del campo promedio

En este capítulo se hace un análisis de autómatas celulares en tres dimensiones que tienen comportamientos similares a *Life*, aprovechando las investigaciones realizadas por Bays que presenta varias reglas de evolución que son meritórias a ser llamadas *Life*. En la Sección 3.1 se presentan algunas reglas de evolución propuestas por Bays con comportamientos similares a *Life*. En la Sección 3.2 se muestran estructuras interesantes de algunas reglas de evolución que justifican su parecido a *Life*, finalmente en la Sección 3.3 se analizan las reglas de evolución $R(5,7,6,6)$ y $R(4,5,5,5)$ con la teoría del campo promedio estableciendo su caracterización con respecto a *Life*.

3.1 Reglas de evolución parecidas a $Life$

Los estudios realizados por Bays [2], [3], [4], [5], [6], [7], [8] en autómatas celulares de tres dimensiones son importantes, ya que es el primero en establecer varias condiciones para determinar si una regla de evolución es meritória a ser llamada *Life en tres dimensiones*, sus estudios han mostrado que existen varias reglas de evolución en tres dimensiones que pueden tener comportamientos complejos, producir *gliders*, configuraciones fijas, periódicas y no desaparecer rápidamente.

El primer problema es que deben de existir reglas de evolución "similares" a *Life* en autómatas de una y dos dimensiones bajo las condiciones que establece Bays, pero varias de estas condiciones no pueden ser proyectadas a autómatas celulares de una y dos dimensiones. Se muestran algunas de las definiciones establecidas por Bays para deteminar si una regla de evolución puede simular a *Life* en tres dimensiones.

Definición 3.1.1. *Bays* [2]

Una regla de evolución semitotalística $R(S_{min}, S_{max}, N_{min}, N_{max})$ define un autómata celular del tipo "*The Game of Life*", si y solo si, se cumplen las siguientes condiciones :

1. Un *glider* debe existir de manera "natural" si se aplica la regla $R(S_{min}, S_{max}, N_{min}, N_{max})$ iterativamente a una configuración c_i aleatoria.

2. Todas las configuraciones c_i aleatorias que sean transformadas por la regla de evolución semitotalística $R(S_{min}, S_{max}, N_{min}, N_{max})$ deben tener un crecimiento limitado. $\square$

La primera condición implica la existencia de estructuras periódicas con desplazamientos en el espacio de evoluciones tridimensional $\mathbb{Z} \otimes \mathbb{Z} \otimes \mathbb{Z}$ sin necesidad de construirlos cuidadosamente, en *Life* comúnmente los *gliders* empiezan a generarse en el intervalo de generaciones $80 \leq t_i \leq 100$ para toda configuración c_i aleatoria de densidad promedio. La segunda condición implica que desde cualquier configuración inicial aleatoria c_i, la regla de evolución debe alcanzar una densidad promedio baja, lo que significa que no se cubra todo el espacio $\mathbb{Z} \otimes \mathbb{Z} \otimes \mathbb{Z}$ por un solo estado o que un solo estado domine el espacio de evoluciones.

Nótese que la Definición 3.1.1 puede ser aplicada a autómatas celulares de dos y una dimensión, por supuesto esta definición se basa en el modelo original de *Life*, lo que implica rápidamente que la regla *HighLife* no cumple ninguna de las dos condiciones, dado que desde cualquier configuración aleatoria c_i no se producen *gliders* de manera natural y no crece limitadamente.

Las demás definiciones mostradas a continuación están basadas para autómatas celulares en tres dimensiones y no pueden aplicarse a autómatas celulares de dos y una dimensión como se explica despues de la definición realizada por Bays.

Definición 3.1.2. *Bays* [2]

Una expanción de un objeto *Life* para construirlo en tres dimensiones, se obtiene haciendo copias de todas las células vivas $(x_i, y_i, 0)$ en el plano adyacente z, es decir, en la terna $(x_i, y_i, 1)$. $\square$

Un ejemplo bastante ilustrativo de la Definición 3.1.2 se puede realizar con el *glider* de cinco células que existe en *Life* Figura 2.4 y puede expanderse en tres dimensiones, haciendo copias de sus células vivas en el plano adyacente $z = 1$ y poder evolucionarlo en el espacio tridimensional $\mathbb{Z} \otimes \mathbb{Z} \otimes \mathbb{Z}$.

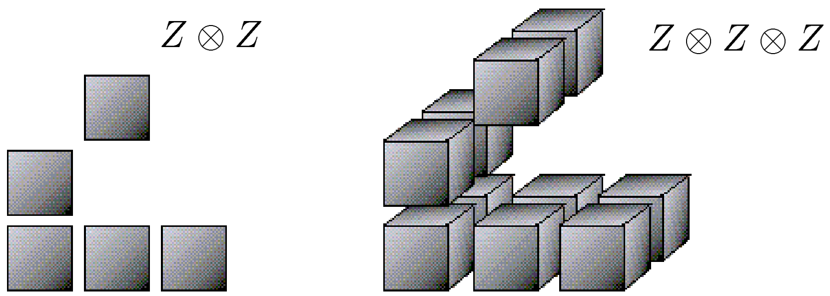


Figura 3.1: Expanciones de *Life* en tres dimensiones

Las copias se realizan únicamente en el eje $z = 1$, la regla de evolución en tres dimensiones que evoluciona el *glider* de la Figura 3.1 es la regla semitotalística $R(5, 7, 6, 6)$, por esa razón se pueden construir objetos análogos que existen en *Life* en la regla de evolución tridimensional $R(5, 7, 6, 6)$.

Definición 3.1.3. *Bays* [2]

Una proyección de un objeto *Life* en dos dimensiones debe existir en tres dimensiones, si y solo si, ambas de las siguientes condiciones se cumplen:

1. Todas las células vivas (x_i, y_i, z_i) deben de estar en dos planos adyacentes. Por ejemplo sean los planos $z = 0$ y $z = 1$.

2. La pareja de células $(x_i, y_i, 0)$ y $(x_i, y_i, 1)$ están ambas vivas o muertas. □

La Definición 3.1.3 es recíproca de la Definición 3.1.2, en la Definición 3.1.3 la primera condición dice que si se tiene un objeto *Life* en tres dimensiones, tal objeto no puede ser reproducido en el espacio tridimensional tal como se puede reproducir en dos dimensiones si sus copias se realizan en planos que no sean adyacentes, como se ilustra en la Figura 3.2. La segunda condición dice que las células que se encuentran en los planos adyacentes deberán estar vivas o muertas para garantizar que el objeto pueda reproducirse sin ningún problema.

Nótese que estos planos adyacentes pueden estar en cualquier dirección, porque las estructuras que existen en *Life* son simétricas en el espacio bidimensional $\mathbb{Z} \otimes \mathbb{Z}$ y esta característica debe conservarse con sus objetos análogos en tres dimensiones en el espacio tridimensional $\mathbb{Z} \otimes \mathbb{Z} \otimes \mathbb{Z}$.

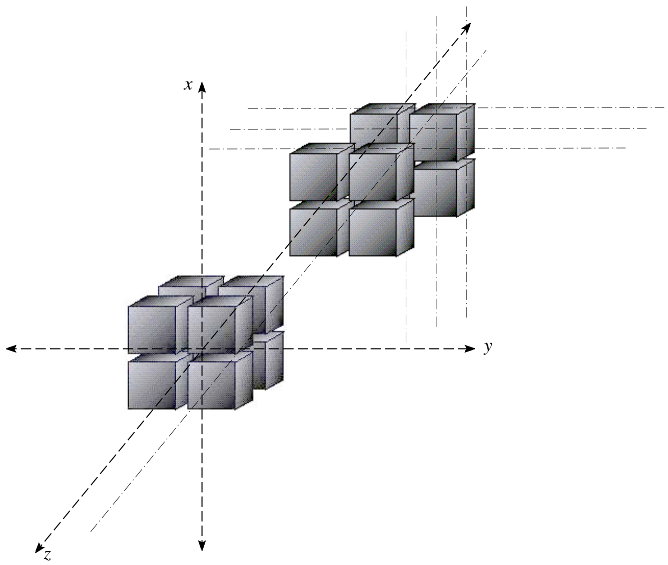


Figura 3.2: Planos adyacentes en tres dimensiones

Bays presenta diversas reglas de evolución que cumplen algunas condiciones necesarias para que sean posibles sucesoras de *Life en tres dimensiones*, como las reglas $R(5767)$, $R(5777)$, $R(5566)$, $R(5755)$, $R(4656)$, $R(4655)$, $R(6767)$, $R(4567)$, $R(6766)$, $R(5655)$, $R(5877)$, $R(4666)$, $R(4566)$, $R(3455)$, $R(3566)$, entre otras.

Revisando el comportamiento de cada una de estas reglas en tres dimensiones realizadas por Bays hemos considerado analizar la regla $R(5, 7, 6, 6)$, porque esta regla de evolución presenta muchas estructuras análogas a *Life* en el espacio tridimensional y la regla de evolución $R(4, 5, 5, 5)$ porque muestra comportamientos similares a *Life*, pero esta regla de evolución produce sus propios *gliders* y sus propias estructuras fijas que no son análogas a *Life* o en dos dimensiones por otras reglas de evolución.

3.2 Reglas interesantes en tres dimensiones

3.2.1 Regla de evolución $R(5,7,6,6)$

La regla de evolución $R(5,7,6,6)$ puede reproducir objetos *Life* en el espacio tridimensional, conservando su simetría a través del tiempo, Bays reportó esta regla en [2] mostrando algunas de las estructuras análogas de dos dimensiones a tres dimensiones, como se ilustró en la Figura 3.1 el *glider* de cinco células que existe en dos dimensiones puede ser reproducido en tres dimensiones por esta regla de evolución, pero en el Capítulo 2 se mostró que en *Life* existe una estructura que puede reproducir cada treinta pasos uno de estos *gliders* que se conoce como *glider gun* Figura 2.5, el cual no ha sido descubierto en tres dimensiones.

En la Figura 3.3 se ilustra la evolución de una configuración aleatoria mostrando algunas estructuras fijas, algunos *blinkers* y *gliders* que produce la regla naturalmente.

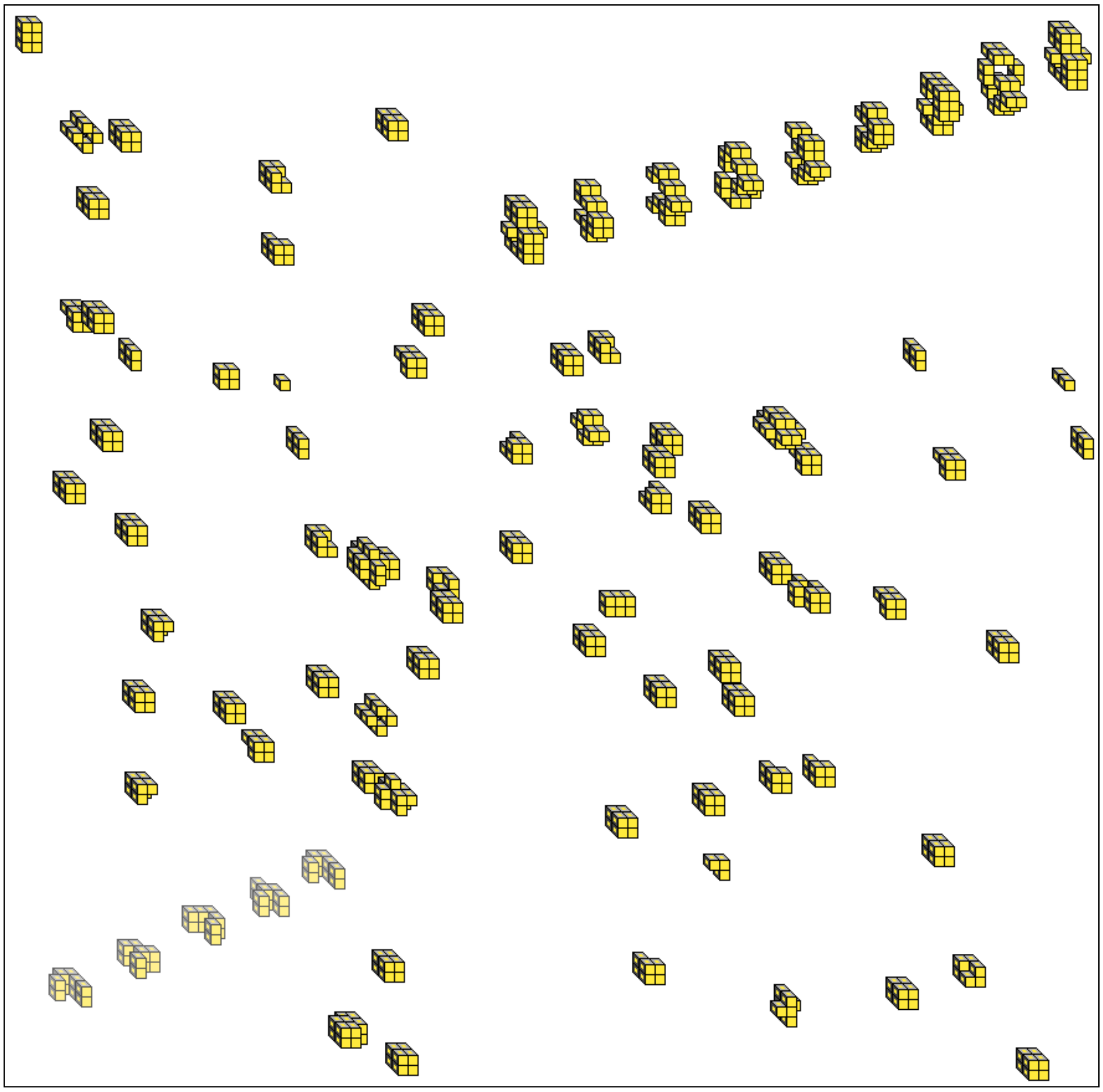


Figura 3.3: Comportamientos de la regla $R(5,7,6,6)$

La regla de evolución $R(5,7,6,6)$ se representa como:

$$\varphi(\mathbf{x}_0,\mathbf{x}_1,\ldots,\mathbf{x}_{26}) = \begin{cases} 1 \quad si & \begin{cases} \mathbf{x}_0 = 0 \quad y \quad 6 \leq \sum_{i=1}^{26} \mathbf{x}_i \leq 6 \\ \\ \mathbf{x}_0 = 1 \quad y \quad 5 \leq \sum_{i=1}^{26} \mathbf{x}_i \leq 7 \end{cases} \\ \\ 0 \quad \text{en otro caso} \end{cases} \tag{3.2.1}$$

3.2.2 Regla de evolución $R(4,5,5,5)$

La regla de evolución $R(4,5,5,5)$ a diferencia de la regla $R(5,7,6,6)$ produce varios objetos que no son análogos en dos dimensiones dentro de sus configuraciones, pero produce varios *gliders* propios y muchas estructuras periódicas y fijas tal como lo hace *Life*, Bays reporta en detalle todas sus características en [2].

En la Figura 3.4 se ilustran algunos *gliders*, *blinkers* y estructuras fijas o mejor conocidas como *still life* de la regla $R(4,5,5,5)$, aunque con el estudio que se realizó con la teoría del campo promedio se verá que esta regla de evolución tiene diferencias probabilísticas importantes con respecto a *Life*.

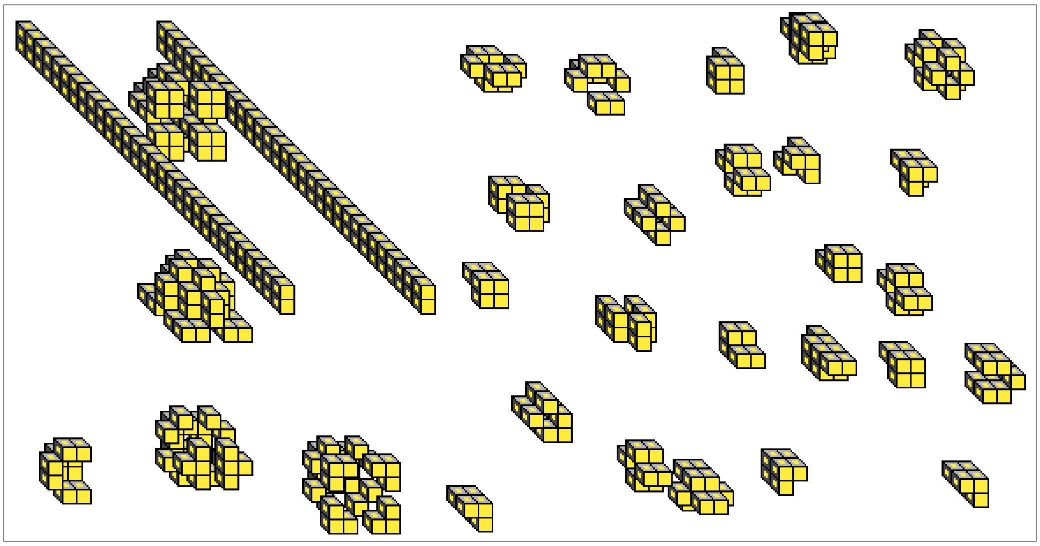


Figura 3.4: Comportamientos de la regla $R(4,5,5,5)$

Esta regla de evolución $R(4,5,5,5)$ se representa como:

$$\varphi(\mathbf{x}_0,\mathbf{x}_1,\ldots,\mathbf{x}_{26}) = \begin{cases} 1 \quad si & \begin{cases} \mathbf{x}_0 = 0 \quad y \quad 5 \leq \sum_{i=1}^{26} \mathbf{x}_i \leq 5 \\ \\ \mathbf{x}_0 = 1 \quad y \quad 4 \leq \sum_{i=1}^{26} \mathbf{x}_i \leq 5 \end{cases} \\ \\ 0 \quad \text{en otro caso} \end{cases} \tag{3.2.2}$$

Esta regla de evolución es importante ya que podría realizar fenómenos tan complejos como lo hace *Life*, lo que implica que si en tres dimensiones existe mas de una regla de evolución que puedan construir estructuras tan complejas como lo hace *Life*, entonces deben de existir otras reglas de evolución en autómatas celulares de dos y una dimensión que realizen lo que hacen esas reglas de evolución tridimensionales. Se puede ver que la regla de evolución $R(4,5,5,5)$ podría ser la sucesora directa de *Life* en tres dimensiones, ya que sería la traslación inmediata agregando el eje z^- y z^+ al plano cartesiano definido por x y y, sumandole esos cuadrantes a la regla *Life*, obteniendo la regla $R(2+2, 3+2, 3+2, 3+2) = R(4,5,5,5)$.

3.3 Aplicando la teoría del campo promedio

3.3.1 Analizando la regla $R(5,7,6,6)$

La regla de evolución $R(5,7,6,6)$ es muy interesante ya que puede reproducir muchas estructuras de *Life* que son análogas en tres dimensiones, entre estas estructuras se encuentran algunas conocidas como el *block*, *blinkers*, algunos *still life*, el *glider* de cinco células, entre otros. El grado del polinomio crece rápidamente porque se tiene una célula central y 26 vecinos, entonces si se utilizan reglas de evolución completas tendríamos $2^{2^{27}}$ reglas de evolución y aunque se emplean reglas semitotalísticas se tienen 390,625 posibles reglas de evolución a estudiar, por lo que el número de reglas de evolución a analizar es muy amplio.

El polinomio del campo promedio se calcula de la misma manera como se mostró en la Sección 2.4.1, tomando encuenta que la vecindad de Moore se representa en tres dimensiones y no en el plano. Empleando la Ecuación 2.4.2 el polinomio tiene tres términos originados por $S_{min} = 5$, $S_{max} = 7$, $N_{min} = 6$ y $N_{max} = 6$:

$$p_{t+1} = 65780 p_t^6 q_t^{21} + 657800 p_t^8 q_t^{19} + 230230 p_t^6 q_t^{21} \tag{3.3.1}$$

simplificando se tiene que:

$$p_{t+1} = 296010 p_t^6 q_t^{21} + 657800 p_t^8 q_t^{19}. \tag{3.3.2}$$

El polinomio de la regla $R(5,7,6,6)$ tiene tres términos, cuando $S_{min} = 5$ la variable $v = 6$ porque tiene cinco 1's en sus vecinos mas la célula central que seguirá viva en la siguiente generación en la vecindad de Moore en tres dimensiones y $n - v = 27 - 6 = 21$ donde 21 representa la cantidad de 0's en la vecindad de Moore en tres dimensiones en ese momento. El segundo término representa las combinaciones sin repetición de $S_{max} = 7$, donde se tienen 7 vecinos vivos y 26 posibles células a ocupar, la variable $v = 8$ porque la célula central está ocupada por el estado 1 en esa configuración y $n - v = 27 - 8 = 19$. El tercer término está determinado por $N_{min} = 6$ donde $v = 6$ y $n - v = 27 - 6 = 21$, en este caso la probabilidad que origina N_{min} es igual a la probabilidad que origina N_{max}, por esta razón solo se tiene un término y no dos como pudiera pensarse. En la Figura 3.5 se gráfica el polinomio de la Ecuación 3.3.2.

El procedimiento para gráficar el polinomio es el mismo que se explicó en la Sección 2.4.1, la curva de probabilidad de la Figura 3.5 presenta un rango de células vivas mas estrecho que el de *Life* en el eje q, ésto se debe porque el número de células vivas es menor al número de células muertas en una vecindad mas grande que la que existe en dos dimensiones, nótese que la curva es tangencial a la diagonal. McIntosh en [44] menciona los diferentes tipos de comportamientos que pueden tener las curvas de probabilidad, diciendo que la clase IV podrían ser todas aquellas reglas de evolución que tengan curvas de probabilidad que sean tangenciales a la diagonal.

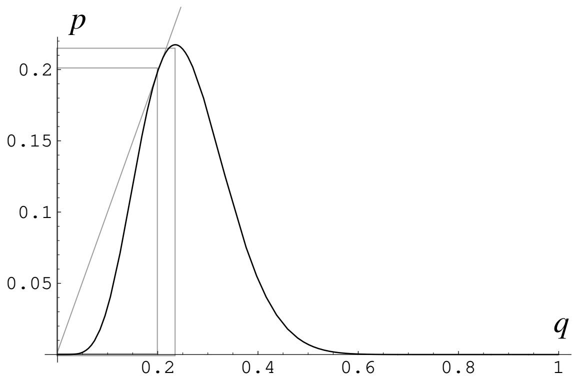


Figura 3.5: Curva de probabilidad de la regla $R(5, 7, 6, 6)$

En la Figura 3.6 se muestra el comportamiento de los estados en la segunda, tercera y cuarta generación respectivamente, al igual que *Life* las curvas de probabilidad van acotando la curva sobre el eje q rápidamente en cada generación, esto es fácil de comprobar ya que si ponemos una configuración aleatoria c_i, la cantidad de 1's decrece muy rápido. Una característica importante es la existencia de un punto fijo inestable aproximandamente en 0.2 que garantiza un comportamiento de densidades no predecible en el espacio, donde pueden formarse configuraciones crecientes, locales, que desaparezcan o que se mantengan igual.

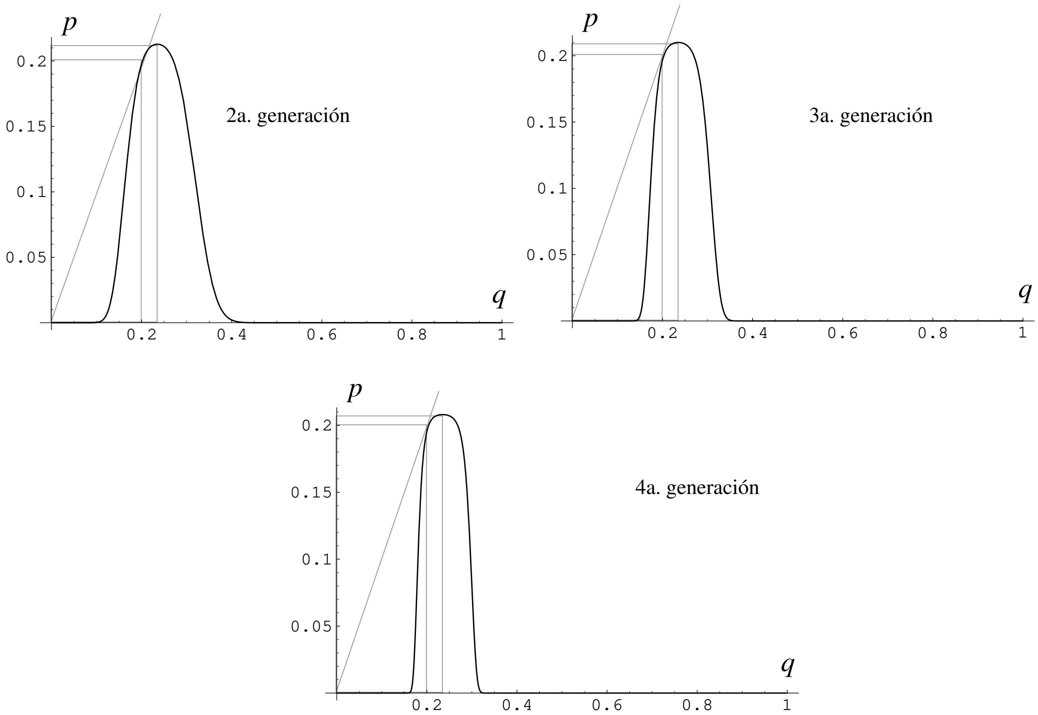


Figura 3.6: Curvas de probabilidad de la regla $R(5, 7, 6, 6)$ segunda, tercera y cuarta generación

El comportamiento de la curva es uniforme con respecto a su punto máximo y no presenta cambios en este punto crítico a través del tiempo, la similitud establecida con *Life* es que tiene un punto fijo inestable recordemos que *Life* no tiene puntos fijos estables, la probabilidad de tener 1's se va acotando en cada generación al igual que *Life* y el comportamiento de la curva se conserva a través del tiempo al igual que *Life*.

3.3.2 Analizando la regla $R(4, 5, 5, 5)$

La regla de evolución $R(4, 5, 5, 5)$ estudiada por Bays en [2], muestra estructuras muy interesantes que son independientes de las que produce la regla $R(5, 6, 7, 7)$, sus *gliders* y configuraciones periódicas fijas son de construcción diferente y se producen de manera natural, además de que no son análogas en dos dimensiones. Esto originó un interés especial que nos lleva a realizar el estudio probabilístico para saber como se comporta la regla de evolución y que relación existe con *Life*. Probando con varias configuraciones aleatorias se puede ver que el comportamiento de las células es similar al de la regla $R(5, 6, 7, 7)$, se estaciona rápidamente pero no desaparece por completo porque produce muchas estructuras periódicas fijas y *still life*.

La regla de evolución $R(4, 5, 5, 5)$ en primera instancia puede ser vista como la sucesora lógica de *Life*, si se suma la ordenada z a la regla *Life* se tiene que $R(2+2, 3+2, 3+2, 3+2)$ produce la regla de evolución en tres dimensiones $R(4, 5, 5, 5)$, pues el eje z está en el espacio $\mathbb{Z} \otimes \mathbb{Z}$ lo que implica la existencia de planos: z^+ y z^-. Los valores de las variables $S_{min} = 4$, $S_{max} = 5$, $N_{min} = 5$ y $N_{max} = 5$, determinan los coeficientes y los exponentes del polinomio del campo promedio que queda como:

$$p_{t+1} = 14950 p_t^5 q_t^{22} + 65780 p_t^6 q_t^{21} + 65780 p_t^5 q_t^{22} \tag{3.3.3}$$

simplicando se tiene:

$$p_{t+1} = 80730 p_t^5 q_t^{22} + 65780 p_t^6 q_t^{21}. \tag{3.3.4}$$

Nótese que los términos del polinomio de la Ecuación 3.3.4 son análogos a los de la Ecuación 2.4.5 que representa a *Life*. La gráfica de probabilidad del polinomio de la Ecuación 3.3.4 queda de la siguiente manera:

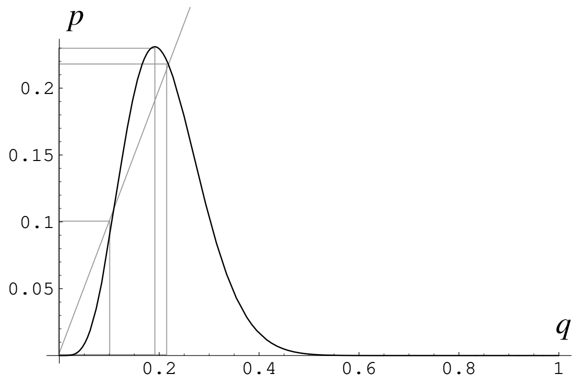


Figura 3.7: Curva de probabilidad de la regla $R(4, 5, 5, 5)$

la gráfica de la Figura 3.7 muestra un acotamiento de la probabilidad promedio necesaria para la existencia del estado 1, existe un punto fijo inestable aproximadamente en 0.1 y un punto fijo estable en aproximadamente 0.22, la primera diferencia con respecto a la regla *Life* es la existencia de un punto fijo estable que no existe en *Life*.

La curva de probabilidad de la Figura 3.7 muestra que la probabilidad de tener 1's en el espacio $\mathbb{Z} \otimes \mathbb{Z} \otimes \mathbb{Z}$ en la siguiente generación no es tan alto como la que representa *Life*, ésto se debe porque la ordenada z

en el espacio tridimensional implica mas vecinos en la vecindad y menos células vivas que determinen mas población en la misma vecindad, consecuentemente la densidad poblacional es mas baja que la de *Life*, tal como se mostró para la regla $R(5,7,6,6)$.

Se evoluciono esta regla de evolución con varias configuraciones aleatorias de diferente orden, observadando que no desaparece por completo rápidamente y tampoco llena el espacio de puros 1's, mostrando la existencia de algunos *blinkers*, *still life* y *gliders* que son diferentes de la regla $R(5,7,6,6)$.

La curva de probabilidad de la Figura 3.7 en sus extremos inferiores sobre el eje q muestra el mismo comportamiento que la regla $R(5,7,6,6)$ y $R(2,3,3,3)$, se puede notar que la densidad poblacional promedio en tres dimensiones es menor que la que existe en dos dimensiones.

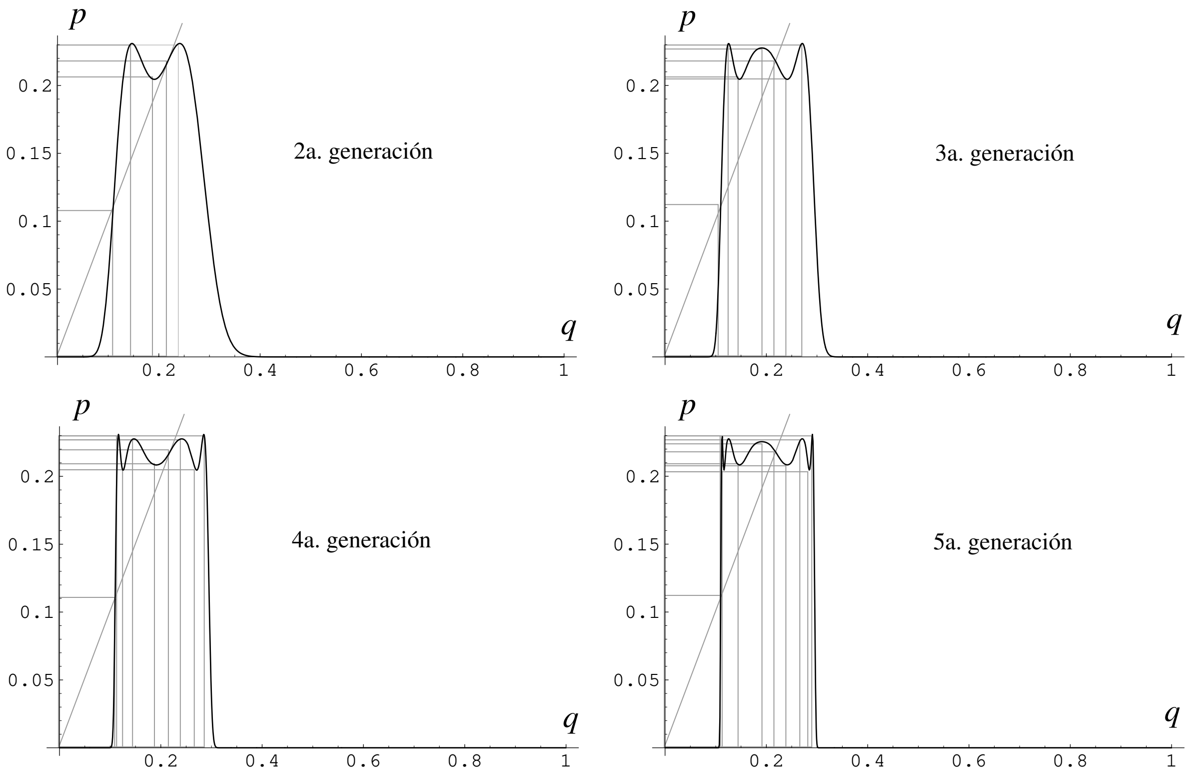


Figura 3.8: Curvas de probabilidad de la regla $R(4,5,5,5)$ segunda, tercera, cuarta y quinta generación

La Figura 3.8 muestra las curvas de probabilidad de los polinomios del campo promedio para la segunda, tercera, cuarta y quinta generación respectivamente.

Nótese que existen puntos fijos estables tangenciales a la diagonal y puntos fijos inestables alternandose de generación en generación pares e impares a partir de la segunda generación. En primera instancia se tienen varios puntos máximos y mínimos por generación lo que implica inestabilidad en la densidad promedio de células vivas que se encuentran en el espacio de evoluciones, también nótese que la cantidad de estos puntos máximos y mínimos va aumentando conforme se calculan mas generaciones. Es muy interesante ver que esta regla de evolución aunque no tiene una densidad promedio en su curva de probabilidad y tiene un punto fijo inestable, presenta características muy similares a *Life* a nivel de simulación, Bays reporta esta regla en [2] ilustrando algunos *gliders* que ha descubierto y configuraciones muy interesantes que la regla produce en el espacio de evoluciones, aunque no reporta la existencia de un *glider gun* o un *mega flip-flop*.

Las curvas de probabilidad de la Figura 3.8 al igual que la regla de evolución $R(5,7,6,6)$ y la regla *Life* van acontando la densidad promedio en el eje q en cada generación, pero la diferencia principal que tiene con respecto a *Life* es la existencia de varios puntos máximos y mínimos que se encuentran en el intervalo de la densidad promedio de la curva, por lo tanto esta regla de evolución aunque puede soportar comportamientos complejos no muestra una densidad promedio al límite como la regla $R(5,7,6,6)$ y tampoco como la regla *Life*.

3.4 Comentarios finales

El estudio de los autómatas celulares en tres dimensiones es mas complicado dado el número de células que pueden trabajar en la vecindad dentro del espacio tridimensional y por la cantidad de reglas de evolución que se pueden generar, primero existe un problema de representación ya que existen pocos programas que pueden hacerlo y con sus respectivas limitaciones, segundo el grado de los polinomios de Bernstein es muy alto. Sin embargo se puede apreciar que el análisis con la teoría del campo promedio es útil, ya que permite agrupar las reglas de evolución de acuerdo al comportamiento de los estados a través del tiempo, se seleccionaron estas dos reglas de evolución aprovechando los estudios realizados por Bays en autómatas celulares que evolucionan en tres dimensiones, se puede ver que particularizar una regla de evolución en tres dimensiones similar a *Life* es mas complejo que un estudio analítico y esto puede ser visto por la regla $R(4,5,5,5)$ que estadísticamente hablando no tiene un comportamiento como *Life*, aunque la regla no está exenta de tener su propio *glider gun* y realizar una cierta computación, recomendamos las siguientes referencias que tratan sobre la Sección 3.1 y 3.2 [2], [4], [5], [6], [7], [8] y [17], para la Sección 3.3 recomendamos [3], [21], [24], [30] y [32].

Capítulo 4

Analizando el autómata celular en una dimensión "regla 110" con diagramas de de Bruijn y la teoría del campo promedio

En este capítulo se presenta el autómata celular en una dimensión de orden (2,1) *regla 110*, esta regla de evolución muestra algunas características similares a *Life* presentadas por Cook en [14]. En la Sección 4.1 se define su estructura elemental y algunas herramientas empleadas para su estudio. En la Sección 4.2 se muestran los tipos de comportamientos tal como su fondo periódico y sus *gliders*. En la Sección 4.3 se muestra su análisis con la teoría del campo promedio.

4.1 Estructura de la *regla* 110

La *regla 110* es una autómata celular que evoluciona en una dimensión de orden ($k = 2, r = 1$) y presenta algunas características similares a *Life*, dentro de las estructuras que genera esta regla de evolución puede producir configuraciones fijas, configuraciones periódicas fijas y configuraciones periódicas con desplazamientos, pero a diferencia de *Life* todas estas estructuras evolucionan sobre un fondo periódico llamado "*ether*" por Cook, además de que puede soportar comportamientos complejos y reproducirse así mismo.

Cook establece una comparación de *Life* con la *regla 110* en [14], es decir, ambas reglas pueden soportar estructuras persistentes que pueden estar fijas o desplazarse a través del espacio de evoluciones, además de que estas estructuras pueden interactuar entre ellas.

La función de transición φ para la *regla 110* se representa como:

Regla 110	
$\varphi(0,0,0) \to 0$	$\varphi(1,0,0) \to 0$
$\varphi(0,0,1) \to 1$	$\varphi(1,0,1) \to 1$
$\varphi(0,1,0) \to 1$	$\varphi(1,1,0) \to 1$
$\varphi(0,1,1) \to 1$	$\varphi(1,1,1) \to 0$

Tabla 4.1: Regla de evolución 110

Life y la *regla 110* pueden empezar en una configuración inicial aleatoria y despues de un tiempo establecerse en el espacio de evoluciones. Cook muestra tres condiciones importantes que relacionan la *regla 110* con la regla de *Life*.

1. Una célula muerta x_i en el tiempo t debe nacer en el tiempo $t+1$, si su vecino derecho x_{i+1} está vivo.
2. Una célula viva x_i en el tiempo t debe morir en el tiempo $t+1$, si ambos de sus vecinos x_{i-1} y x_{i+1} están vivos.
3. En otro caso la célula x_i en el tiempo t seguirá igual en el tiempo $t+1$.

La *regla 110* tiene configuraciones que existen únicamente como configuraciones iniciales, pero estas configuraciones nunca se podrán generar a través de la evolución desde cualquier otra configuración, es decir, no tiene ancestros. Este tipo de configuraciones se les llaman pertenecientes al "*Jardín del Edén*", por ejemplo la configuración 01010 es una configuración que nunca se podrá generar en el diagrama de evoluciones, la única manera de que exista en el diagrama de evoluciones es en la configuración inicial; por su parte *Life* también tiene configuraciones pertenecientes al "*Jardín del Edén*" [10].

El estudio sobre la *regla 110* ha causado un gran interés ya que Cook ha mencionado en [14] que dicha regla puede hacer computación universal, aunque no es claro como se puede llegar a comprobar. Nuestro interés no radica en determinar si dicha regla puede hacer o no puede hacer computación universal, mas bien en establecer las características que relaciona la *regla 110* con la regla de *Life*. Tomando en cuenta que la principal diferencia consiste en que la *regla 110* evoluciona en un fondo periódico y *Life* no, otro punto importante es que si la *regla 110* presenta comportamientos similares a *Life* en una dimensión con sus propias características, entonces implica que dichos comportamientos deben existir en autómatas celulares de dos y tres dimensiones, comportandose como *Life* pero con un fondo periódico. Por otra parte es importante resaltar que si el sistema de Cook puede simular todo lo que hace *Life*, se comprobaría que el sistema que planteo von Neumann originalmente en [54], es llevado a un sistema mas sencillo que el modelo de Conway y este modelo es la *regla 110*.

4.2 Diagramas de de Bruijn en la *regla* 110

Las características en las estructuras de *Life* son sencillas y muy complejas a la vez, éstas son alcanzadas desde casi cualquier configuración aleatoria. Ahora bien *Life* es una regla semitotalística y la *regla 110* no es una regla totalística ni semitotalística, lo que implica que sus posibles proyecciones en autómatas celulares de dos y tres dimensiones tengan que realizarse definiendo cada una de las vecindades que se derivan con la vecindad de von Neumann y la vecindad de Moore. Cook presenta ocho tipos de *gliders* y un *glider gun* en [14], no describe sus métodos o las herramientas que emplea para construir dichos *gliders*, sin embargo logramos reproducir estos *gliders* empleando teoría de gráficas con los diagramas de de Bruijn [45], tal como lo describe McIntosh en [46].

Se emplean diagramas de de Bruijn [45] para determinar la tranformación local en cada una de las vecindades de la regla de evolución, McIntosh reporta estas relaciones en [46] describiendo la importancia de emplear estas herramientas que proporcionan la información rápidamente, estos diagramas se ajustan muy bien a las reglas de los autómatas celulares en una dimensión aunque originalmente fueron empleados para el estudio de la teoría de registro de corrimientos. La teoría de registro de corrimientos es una disciplina basada en el tratamiento de secuencias traslapando.

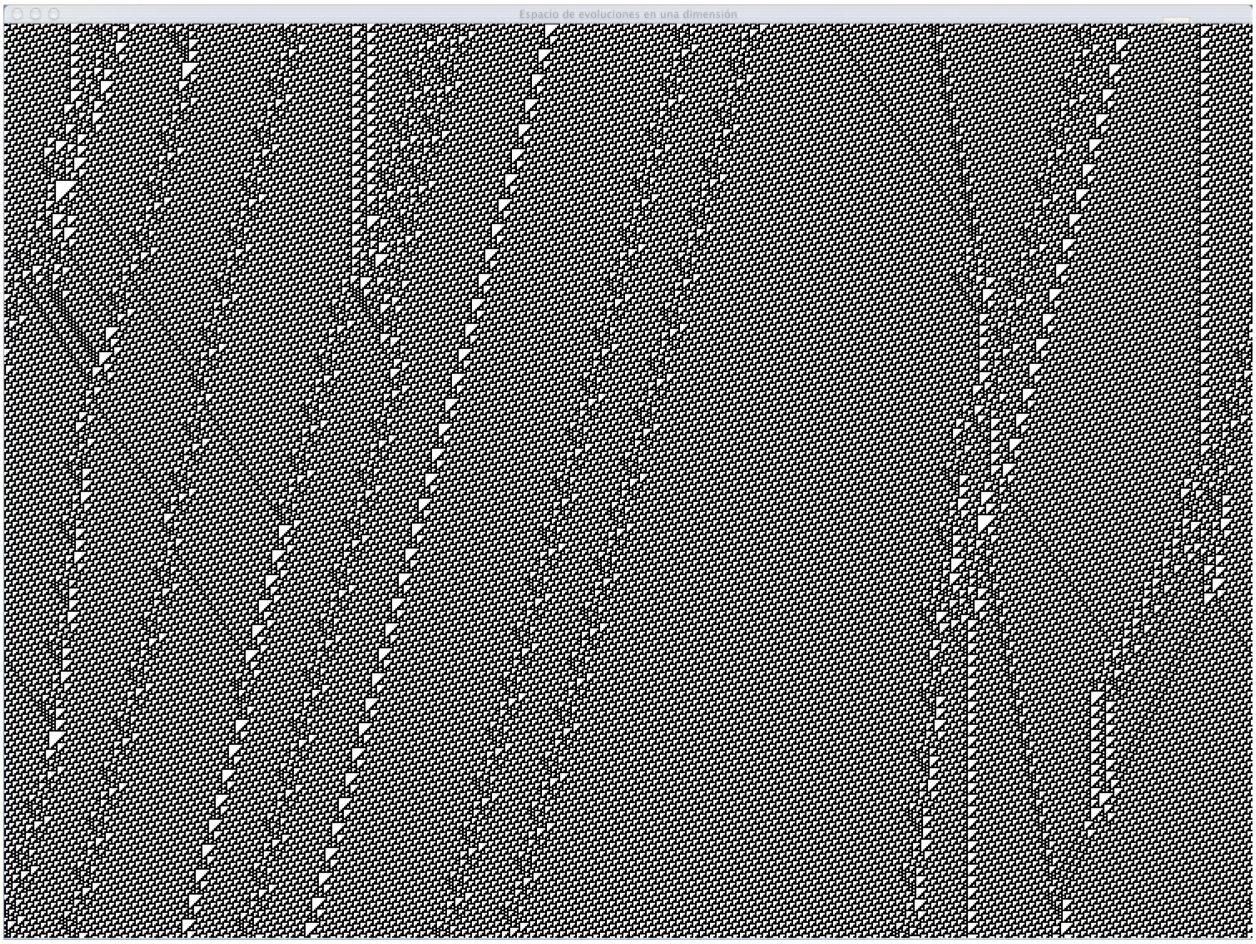


Figura 4.1: Evolución típica de la *regla 110*

En el espacio de evoluciones de la Figura 4.1 se puede ver la existencia de estructuras periódicas que se desplazan a través de un fondo periódico llamado *ether* Figura 4.4, donde estas estructuras periódicas son los *gliders* que se desplazan sin sufrir cambios en sus estructuras por el *ether* y a su vez el *ether* tampoco se destruye por los desplazamientos de los *gliders*. La complejidad de la *regla 110* se produce cuando los *gliders* coalisionan entre ellos, originando otros *gliders*. Estas coalisiones son muy variadas pues la estructura que se genera depende de que *gliders* son los que coalisionan, que desplazamiento tienen los *gliders*, a que altura coalisionan, entre otras cosas.

Los vértices del *diagrama de de Bruijn* son secuencias de símbolos de algún alfabeto, estos símbolos pueden ser secuencias de vértices de una gráfica en específico. Las aristas del diagrama describen como tales secuencias pueden traslapar, por lo tanto diferentes grados de traslape producen diagramas diferentes, el traslape se produce entre un símbolo inicial, los símbolos que traslapan y un símbolo terminal.

Por ejemplo la secuencia binaria 0011 traslapa con la secuencia 0110 tomando el 0 de la primera secuencia como el símbolo inicial, 011 como los elementos que traslapan entre la primera y segunda secuencia, el útimo 0 de la segunda secuencia como el símbolo terminal, las aristas pueden ser etiquetadas de acuerdo a la nueva secuencia que se produce que es 0**011**0 y producir una variedad de caminos.

Cuando los símbolos son enteros consecutivos pueden ser tratados como elementos de un anillo o quiza un campo finito.

La matriz de incidencia del diagrama de de Bruijn es:

$$M_{i,j} = \begin{cases} 1 \quad si \qquad\qquad j \quad = \begin{cases} ki \\ ki+1 \\ \vdots \\ ki+k-1 \end{cases} (mod \quad k^{2r}) \\ \\ 0 \quad \text{en otro caso} \end{cases} \tag{4.2.1}$$

donde k y r deteminan el orden del autómata celular. Sea un autómata celular de orden $(k = 2, r = 1)$. La matriz de incidencia para el diagrama de de Bruijn de la Figura 4.2 queda como:

$$M_{i,j} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Esta matriz se obtuvo empleando la Ecuación 4.2.1. Si $i = 0$ entonces $j = ki = 2*0 = 0$ y $j = (2*0)+1 = 1$ por lo tanto $M_{0,0} = 1$ y $M_{0,1} = 1$. Si $i = 1$ entonces $j = ki = 2*1 = 2$ y $j = (2*1)+1 = 3$ por lo tanto $M_{1,2} = 1$ y $M_{1,3} = 1$. Si $i = 2$ entonces $j = ki = 2*2 = 4$ y $j = (2*2)+1 = 5$, 4 módulo 4 es 0 y 5 en módulo 4 es 1, por lo tanto $M_{2,0} = 1$ y $M_{2,1} = 1$. Si $i = 3$ entonces $j = ki = 2*3 = 6$ y $j = (2*3)+1 = 7$, 6 módulo 4 es 2 y 7 en módulo 4 es 3, por lo tanto $M_{3,2} = 1$ y $M_{3,3} = 1$, en todos los demas casos $M_{i,j} = 0$. Finalmente se ve que la Ecuación 4.2.1 representa una gráfica de de Bruijn para cualquier autómata celular de orden (k, r).

En la Figura 4.2 se muestra el diagrama de de Bruijn genérico para los autómatas celulares de orden $(k = 2, r = 1)$.

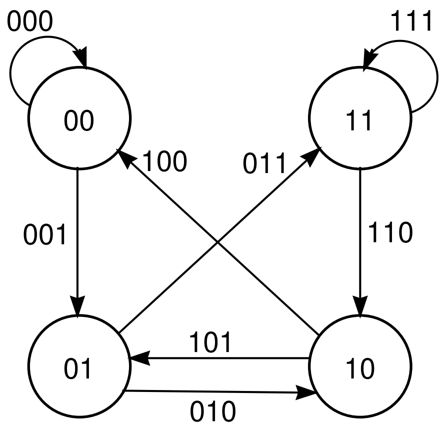


Figura 4.2: Diagrama de de Bruijn genérico $(2, 1)$

El modulo $k^{2r} = 2^2 = 4$ es el número de vértices en el diagrama de de Bruijn, j tomará valores de $k*i = 2i$ hasta $(k*i)+k-1 = (2*i)+2-1 = 2i-1$. Los vértices tienen fracciones de vecindad originada por las secuencias 00, 01, 10 y 11. El traslape define cada una de las aristas como:

Formando vecindades	
(0,**0**) $\diamond$ (**0**,0)	0**0**0
(0,**0**) $\diamond$ (**0**,1)	0**0**1
(0,**1**) $\diamond$ (**1**,0)	0**1**0
(0,**1**) $\diamond$ (**1**,1)	0**1**1
(1,**0**) $\diamond$ (**0**,0)	1**0**0
(1,**0**) $\diamond$ (**0**,1)	1**0**1
(1,**1**) $\diamond$ (**1**,0)	1**1**0
(1,**1**) $\diamond$ (**1**,1)	1**1**1

Tabla 4.2: Traslapes en el diagrama de de Bruijn genérico (2,1)

En la Tabla 4.2 se muestran los traslapes que se derivan de los elementos de cada vértice, estos traslapes son las aristas del diagrama de de Bruijn genérico para los autómatas celulares de orden $(2,1)$ de la Figura 4.2.

El diagrama de de Bruijn genérico para un autómata $(2,1)$ tiene cuatro vértices $\{0,1,2,3\}$ que corresponden a cuatro vecindades parciales de dos células $\{00,01,10,11\}$, con ocho aristas representando sus vecindades de tamaño $2r+1$ Tabla 4.1, entonces el número de vértices en el diagrama de de Bruijn está determinado por k^{2r} y el número de aristas es igual a k^{2r+1}.

El diagrama de de Bruijn para la *regla 110* se deriva del diagrama de de Bruijn genérico de la Figura 4.2, el diagrama de de Bruijn genérico puede ser diferenciado con respecto a una regla de evolución en particular. En la Figura 4.3 se puede ver como cada arista representa el estado al que va a evolucionar la vecindad, el estado 1 está de color negro y el estado 0 está de color gris, éste es el diagrama de de Bruin para la *regla 110*.

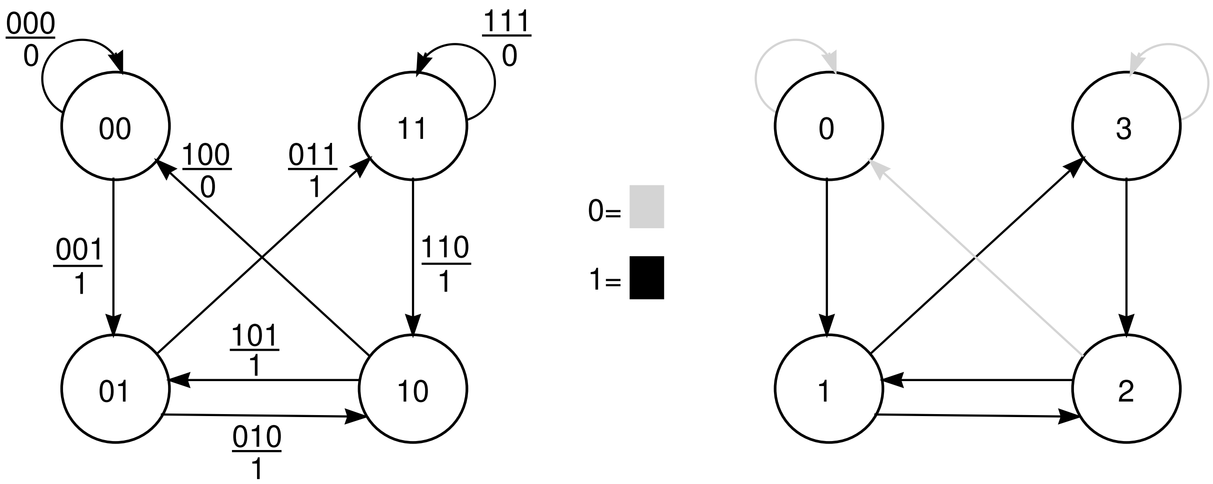


Figura 4.3: Diagrama de de Bruijn para la *regla 110*

Los diagramas de de Bruijn se emplean para describir como se pueden reproducir de manera mas sencilla y práctica los resultados presentados por Cook en [14], cabe señalar que estas técnicas no han sido reportadas por otros autores, McIntosh describe su método en [46].

El diagrama de de Bruijn puede ser extendido para formar vecindades mas grandes y determinar el desplazamiento de las células en t generaciones, ésto origina diagramas de de Bruijn mas grandes y complicados de calcular, por ejemplo calculemos las vecindades de tamaño cinco.

Vecindades de tamaño cinco aplicando *regla 110*			
traslape	vecindad generación 0	generación 1	generación 2
$(0,0,0,0) \diamond (0,0,0,0)$	00000	000	0
$(0,0,0,0) \diamond (0,0,0,1)$	00001	001	1
$(0,0,0,1) \diamond (0,0,1,0)$	00010	011	1
$(0,0,0,1) \diamond (0,0,1,1)$	00011	011	1
$(0,0,1,0) \diamond (0,1,0,0)$	00100	110	1
$(0,0,1,0) \diamond (0,1,0,1)$	00101	111	0
$(0,0,1,1) \diamond (0,1,1,0)$	00110	111	0
$(0,0,1,1) \diamond (0,1,1,1)$	00111	110	1
$(0,1,0,0) \diamond (1,0,0,0)$	01000	100	0
$(0,1,0,0) \diamond (1,0,0,1)$	01001	101	1
$(0,1,0,1) \diamond (1,0,1,0)$	01010	111	0
$(0,1,0,1) \diamond (1,0,1,1)$	01011	111	0
$(0,1,1,0) \diamond (1,1,0,0)$	01100	110	1
$(0,1,1,0) \diamond (1,1,0,1)$	01101	111	0
$(0,1,1,1) \diamond (1,1,1,0)$	01110	101	1
$(0,1,1,1) \diamond (1,1,1,1)$	01111	100	0
$(1,0,0,0) \diamond (0,0,0,0)$	10000	000	0
$(1,0,0,0) \diamond (0,0,0,1)$	10001	001	1
$(1,0,0,1) \diamond (0,0,1,0)$	10010	011	1
$(1,0,0,1) \diamond (0,0,1,1)$	10011	011	1
$(1,0,1,0) \diamond (0,1,0,0)$	10100	110	1
$(1,0,1,0) \diamond (0,1,0,1)$	10101	111	0
$(1,0,1,1) \diamond (0,1,1,0)$	10110	111	0
$(1,0,1,1) \diamond (0,1,1,1)$	10111	110	1
$(1,1,0,0) \diamond (1,0,0,0)$	11000	100	0
$(1,1,0,0) \diamond (1,0,0,1)$	11001	101	1
$(1,1,0,1) \diamond (1,0,1,0)$	11010	111	0
$(1,1,0,1) \diamond (1,0,1,1)$	11011	111	0
$(1,1,1,0) \diamond (1,1,0,0)$	11100	010	1
$(1,1,1,0) \diamond (1,1,0,1)$	11101	011	1
$(1,1,1,1) \diamond (1,1,1,0)$	11110	001	1
$(1,1,1,1) \diamond (1,1,1,1)$	11111	000	0

Tabla 4.3: Extensión de vecindades en el diagrama de de Bruijn con la *regla 110*

En la Tabla 4.3 se necesita que las vecindades parciales bajo la operación traslape '$\diamond$' formen una vecindad de tamaño par, despues se aplica la *regla 110* de la Tabla 4.1 sin tomar en cuenta la propiedad a la frontera, ésto produce una vecindad de tamaño $2r + 1$ en la generación 1, se aplica nuevamente la regla de volución en esta vecindad y se obtiene el estado al que va a evolucionar la vecindad de tamaño cinco.

Por ejemplo la secuencia 0110 traslapa con la secuencia 1101 para formar la vecindad 01101, al aplicar la *regla 110* se tiene que 01101 genera la secuencia 111 que es una vecindad mas pequeña, si se vuelve aplicar la regla de evolución a la secuencia 111 se tiene el estado 0, entonces la vecindad 01101 evoluciona a 0 en dos generaciones.

Las fracciones de vecindad compuestas por 4 células derivan 16 vértices para el diagrama de de Bruijn de este orden y 32 aristas que son las vecindades, despues se calcula la célula de evolución que le corresponde a cada vecindad de ese tamaño. Este método es el que se emplea para generar diagramas de de Bruijn extendidos, que van a determinar como se puede construir una estructura que la *regla 110* produsca en particular.

McIntosh hace notar que el problema de la *regla 110* es un problema de cubrir el espacio con triángulos, aunque Cook no aplica este enfoque ya que su análisis lo presenta con configuraciones muy grandes, con la unión de varias configuraciones c_i de diferentes tamaños en diferentes ciclos de longitud variada; si uno desea hacer a mano tales procesos es una labor muy difícil de finalizar.

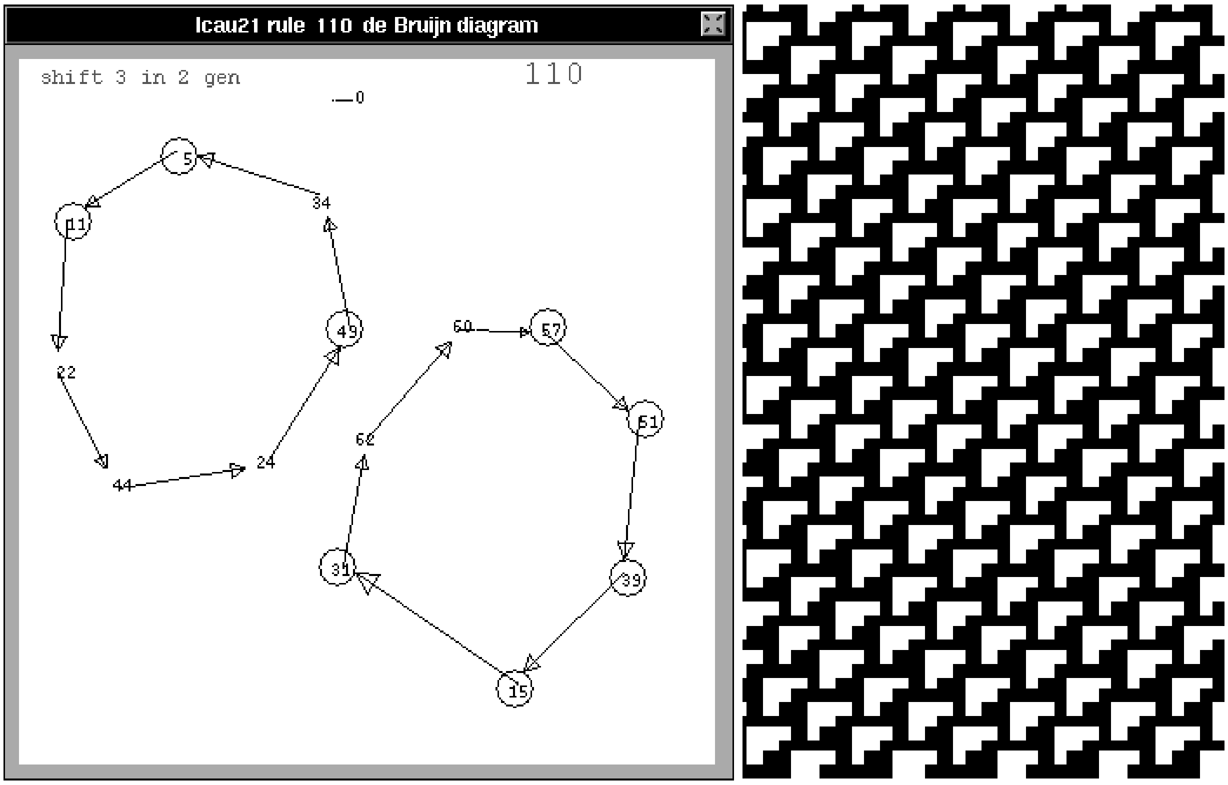


Figura 4.4: *Ether* en la *regla 110*

En la Figura 4.4 se determina el fondo no periódico con configuraciones de tamaño seis, por ejemplo el vértice 5 (000101) traslapa con el vértice 11 (001011) formando la vecindad 11 (0001011), que evoluciona al estado 1 aplicandole la *regla 110* como se describe en la Tabla 4.3, nótese que hay $k^{2(3)+1} = 128$ vecindades de tamaño seis. El corrimiento de los estados de estas estructuras es de un desplazamiento de dos células a la derecha en tres generaciones.

La manera en que se van a identificar las estructuras es siguiendo las secuencias de los estados que tienen las aristas, porque las aristas van a estar representadas por un color, este color es el estado al que evoluciona una vecindad dada, entonces se forma una secuencia de estados simplemente tomando el estado que tiene cada arista hasta cerrar un ciclo y ver que la secuencia resultante produce una estructura en particular.

En la Tabla 4.4 se toma la secuencia de estados 1000101 que produce el ciclo de la izquierda en la Figura 4.4, esta secuencia de estados debe producir el *ether* si se llena la configuración inicial con esta secuencia de estados, entonces en el espacio de evoluciones se tendrá puro *ether*. Por otra parte el diagrama de de Bruijn indica que esta secuencia o cualquier permutación de ella, debe desplazarse 3 células a la derecha en 2 generaciones.

En la Tabla 4.4 la configuración 1000101 muestra como todas sus células se mueven al mismo tiempo tres posiciones a la derecha en dos generaciones marcadas con †, la configuración 1001111 también sufre

Configuración ether	
generación	binario
0	1000101 †
1	1001111
2	1011000 †
3	1111001
4	0001011 †
5	0011111
6	0110001 †
7	1110011
8	0010110 †
9	0111110
10	1100010 †
11	1100111
12	0101100 †
13	1111100
14	1000101 †
15	1001111

Tabla 4.4: Secuencia 1000101 que produce puro *ether*

este cambio de desplazar todas sus células tres posiciones a la derecha en dos generaciones. Finalmente permutaciones de la cadena 1000101 puede reproducir la estructura *ether*.

4.2.1 *Gliders* en la *regla* 110

Cook muestra ocho tipos de *gliders* y un *glider gun*, estos resultados fueron descritos en [14] pero dado algunos problemas legales, no sean reportado con detalle como se obtuvieron estos resultados. La característica importante en esta regla de evolución es que puede llegar hacer computación, Cook ha analizado a fondo esta regla y menciona que dicha regla puede realizar *computación universal*.

La finalidad de estudiar esta regla de evolución no es determinar si la *regla 110* puede o no puede hacer computación universal, mas bien establecer sus relaciones con *Life* aunque se sabe que *Life* puede realizar computación como se demostro en [10].

El uso de los diagramas de de Bruijn extendidos no han sido reportados para la reproducción de *gliders* o cualquier otras estructuras que existan en el espacio de evoluciones de la *regla 110*, Cook presentó varias configuraciones para que fueran analizadas por otros investigadores en el grupo de discución *LifeCA*, la condición inicial: 00010011011111{15922-14}110{396-402}1{342-346}{653-1108}{343-996}[1] establecida por Cook es difícil de analizar. Por ejemplo la parte 110{396 − 402} indica que la configuración 110 debe repetirse en la configuración inicial 396 veces, a su vez la configuración resultante debe copiarse 402 veces, ésta es la configuración inicial a estudiar.

Se puede apreciar que es un trabajo muy laborioso y que requiere una buena cantidad de tiempo para su estudio. Empleando los diagramas de de Brujin se reproducen algunos de los *gliders* mostrados y clasificados por Cook en [14], los diagramas de de Bruijn extendidos que se muestran a continuación omitien todos los detalles mostrados en la Sección 4.2.

[1]e-mail recivido por los miembros de LifeCA@onelist.com.

Glider A

En la Figura 4.5 se puede ver que una estructura de este tipo puede ser calculada con el diagrama de de Bruijn en 3 generaciones y con un desplazamiento de 4 células a la izquierda, tomando la secuencia de uno de los ciclos sabemos que el *glider* A puede ser reproducido por la secuencia de estados 000111, si se toma el ciclo definido por las vecindades parciales $28 \to 56 \to 113 \to 227 \to 199 \to 142 \to 28$, las células de evolución que representarán sus vecindades definen la secuencia 000111.

Nótese que cada uno de estos ciclos de longitud seis definen un *glider* A, aunque las secuencias sean diferentes, por ejemplo tenemos que las secuencias de estados 111110 y 001101 también definen la construcción de *gliders* A en el espacio de evoluciones.

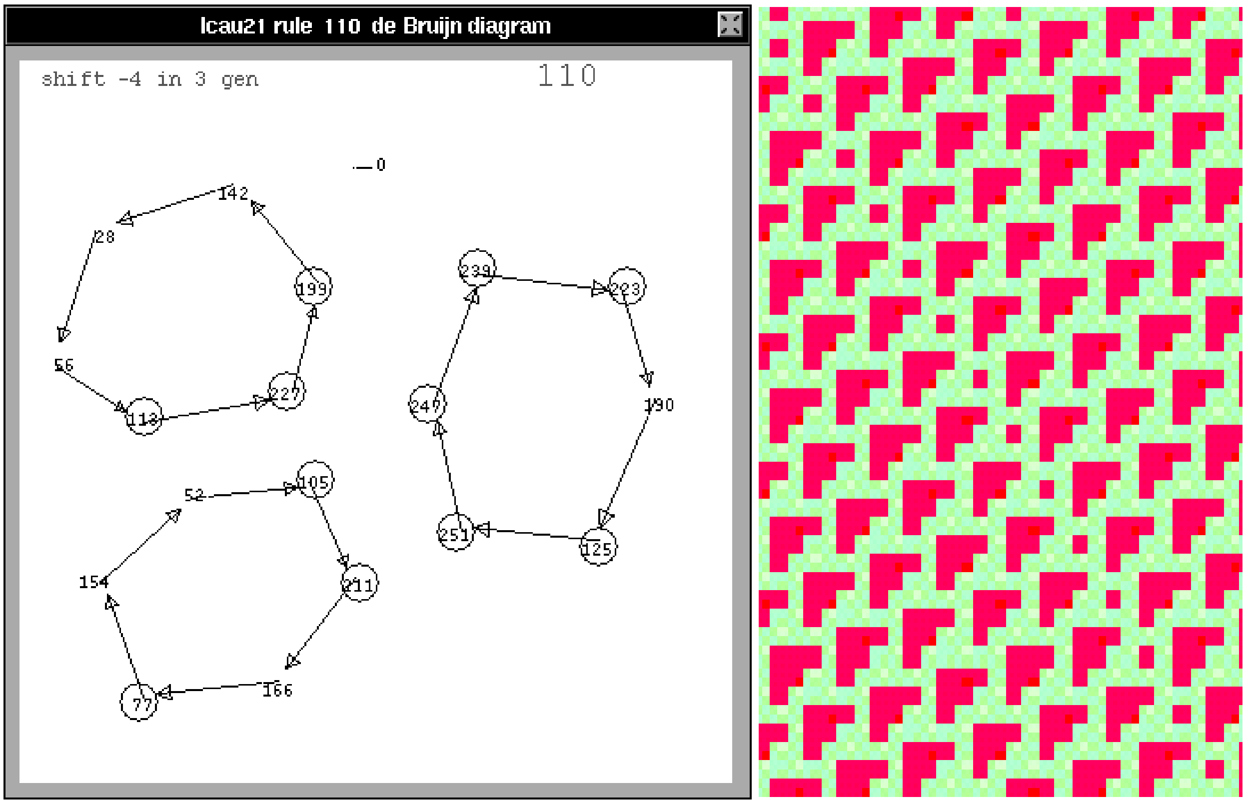


Figura 4.5: *Glider* A

Es importante resaltar el hecho que la construcción de un *glider* sigue una permutación de estados con respecto a una secuencia dada, aquí se puede ver un problema menor muy interesante en los diagramas de de Bruijn extendidos.

Tratar de establecer una clasificación de acuerdo al tipo de secuencias que pueden generar un *glider* en particular, dado que existen varias combinaciones de estados que pueden producir una estructura en particular y sabiendo que hay varias configuraciones que producen una estructura dada, ver cual sería la diferencia entre utilizar una u otra configuración para obtener un resultado en el espacio de evoluciones con otras estructuras.

Glider B

El *glider* B de la Figura 4.6 se desplaza de manera inversa al *glider* A y se pueden reproducir varios de ellos con el diagrama de de Bruijn en 5 generaciones con un desplazamiento de 10 células a la derecha.

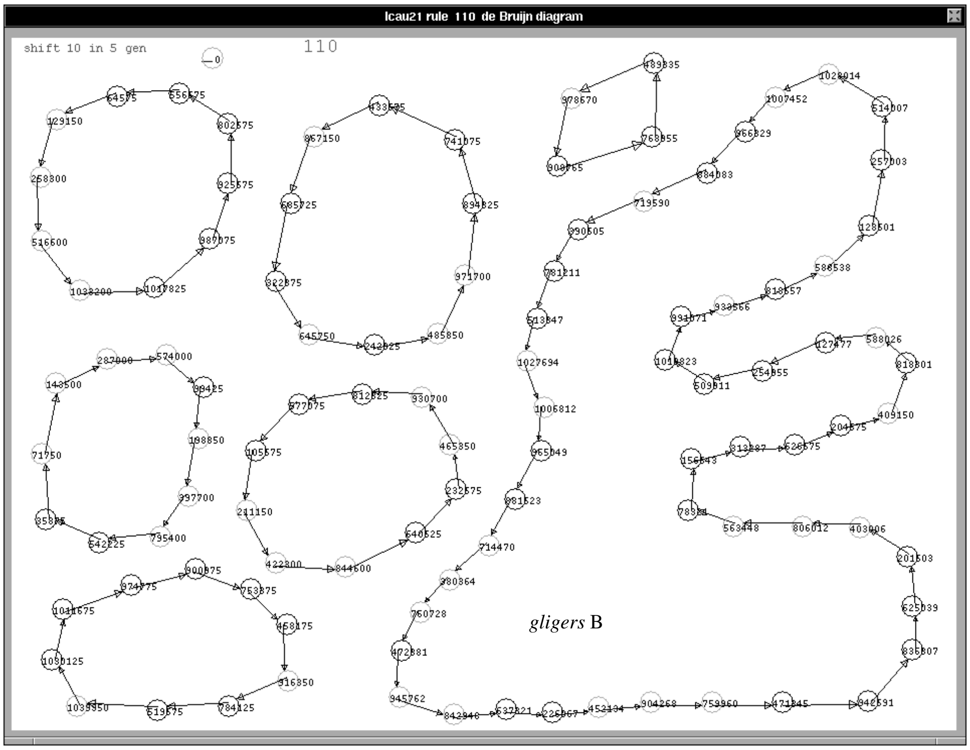


Figura 4.6: Diagrama de de Bruijn para obtener *gliders* B

El ciclo que se genera en este diagrama de de Bruijn es extenso, lo que implica un proceso mas grande que realizar para obtener dicho diagrama, tal ciclo se ilustra con la etiqueta de "*gliders* B" en la Figura 4.6. La Figura 4.7 ilustra varios *gliders* B.

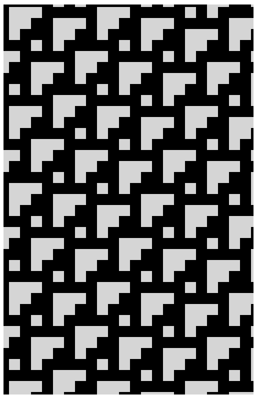


Figura 4.7: *Gliders* B

En la Figura 4.8 se ilustran algunos *gliders* dados a conocer por Cook en [14].

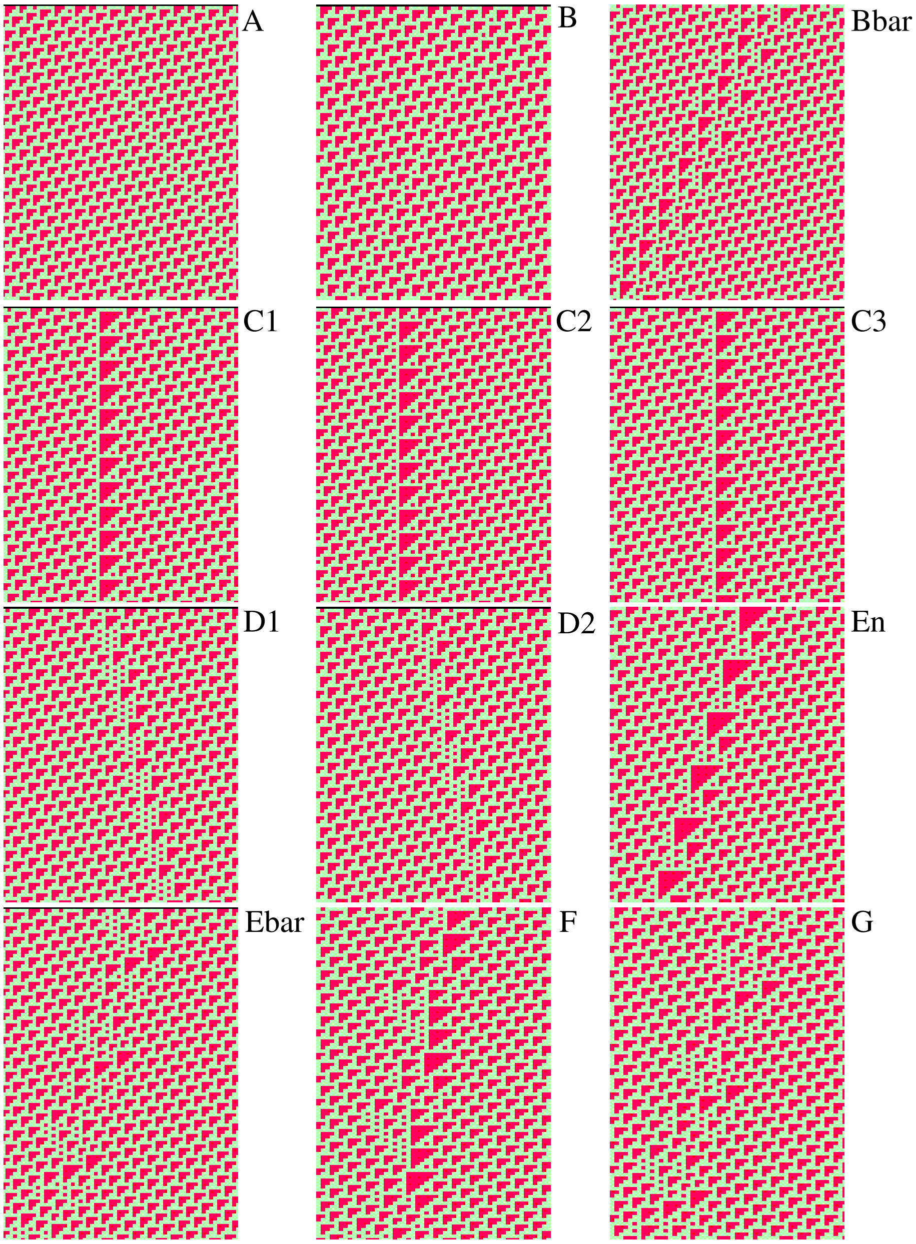


Figura 4.8: Algunos *gliders* de la *regla 110*

Los gliders de la Figura 4.8 tienen un tamaño fijo y un periódo de desplazamiento a través del tiempo, por eso son estructuras periódicas con desplazamientos. Nótese que las estructuras son de diferentes tamaños, algunas son grandes como el *glider gun* que está constituido de 20 células con un desplazamiento en 77 generaciones, ésto implicaría que el diagrama de de Bruijn asociado al *glider gun* debe ser enorme.

Otra característica muy importate son los choques que se pueden producir con estos *gliders* y originar comportamientos complejos, este hecho provocó que se llegara a la conjetura de que la *regla 110* puede realizar computación universal siguiendo la lógica que se empleó con la regla de *Life*, analizando sus choques entre *gliders* como lo demostró Conway en [10] donde *Life* puede hacer computación universal con estos choques. *Life* y la *regla 110* tienen la característica de que si se quiere llegar a reproducir por ejemplo un contador binario a través del espacio de evoluciones, no es común obtenerlo con configuraciones aleatorias, es decir, hay que construirlo con cuidado en la configuración inicial para contruir dicho contador binario. En [34] se muestra la construcción de un contador binario con la *regla 110* emplando tres *gliders*, el *glider* E_n se incrementará en uno E_{n+1} si choca con un *glider* B y se decrementará en uno E_{n-1} si choca con un *glider* A.

Algunos *gliders* mas grandes son difíciles de obtener con los diagramas de de Bruijn, los diagramas aunque pueden ser calculados implican una labor de ordenamiento con paciencia, como se muestra en la Figura 4.9.

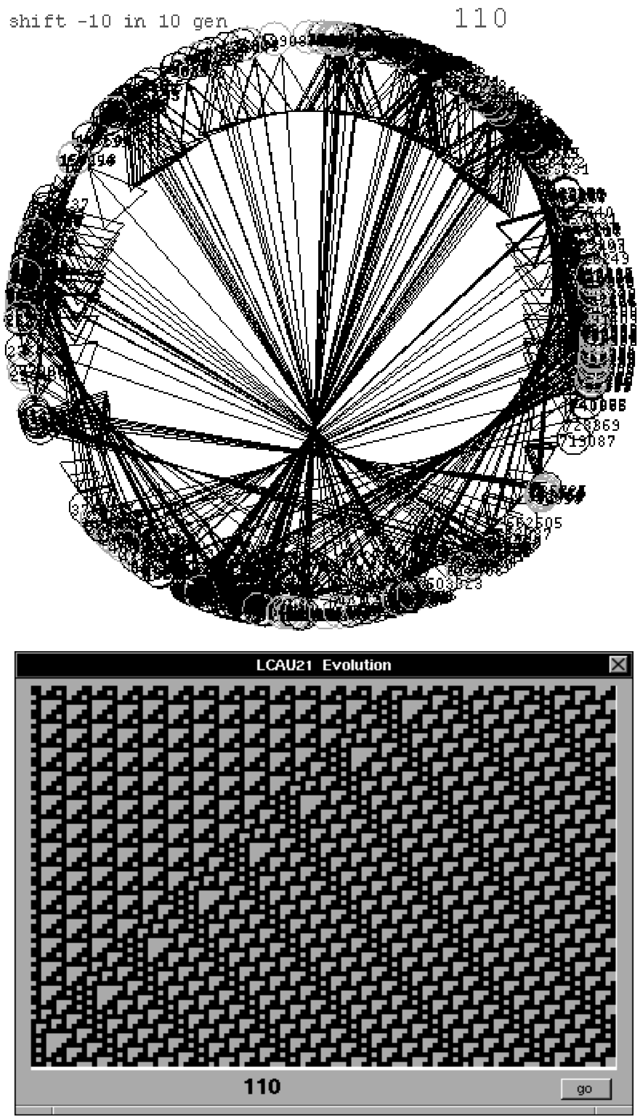


Figura 4.9: Diagrama de de Bruijn extendido en 10 generaciones

En la Figura 4.9 se muestra el diagrama de de Bruijn en 10 generaciones con un desplazamiento de 10 células a la izquierda, se selecciono el nodo 211150 del diagrama de de Bruijn y se puede ver que en el espacio de evoluciones se generan varios *gliders* con el *ether*. Aunque también puede llenarse el fondo con un solo *glider* y meter una línea de puro *ether*, es decir, el *ether* sería como un *glider* y el *glider* como el *ether*.

4.3 Aplicando la teoría del campo promedio

4.3.1 Analizando la *regla* 110

El polinomio del campo promedio para la *regla 110* queda como:

$$p_{t+1} = 2p_t q_t^2 + 3p_t^2 q_t. \tag{4.3.1}$$

La *regla 110* definida en la Tabla 4.1, se puede ver que existen dos vecindades con dos células en el estado 0 y una célula con el estado 1, por lo tanto $v = 1$ y $n - v = 2$. Existen tres vecindades con dos células en el estado 1 y una célula con el estado 0, por lo tanto $v = 2$ y $n - v = 1$.

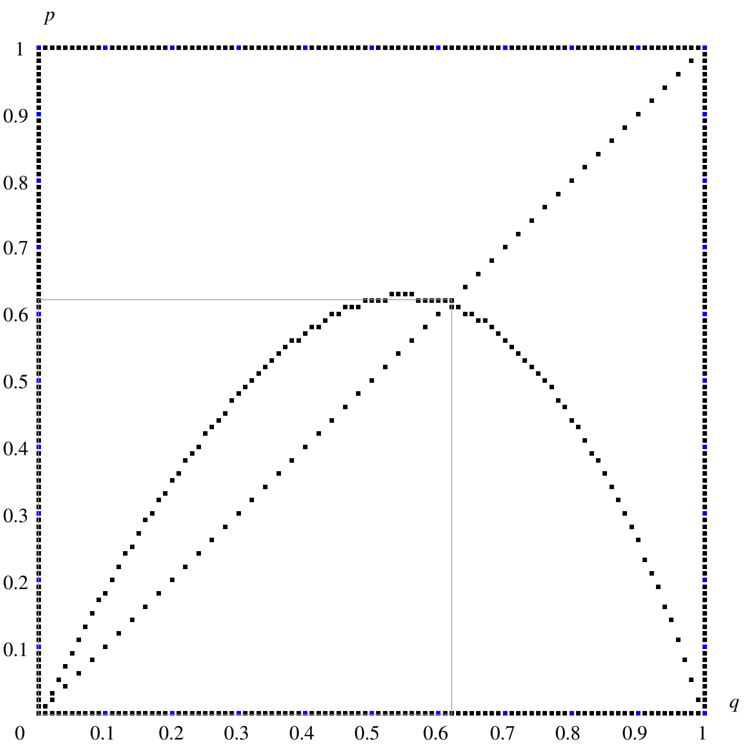


Figura 4.10: Curva de probabilidad de la *regla 110*

En la Figura 4.10 se muestra la curva de probabilidad de la *regla 110*, observando que la curva tiene una alta probabilidad de generar 1's en su espacio de evoluciones. Nótese la existencia de un punto fijo estable en aproximadamente 0.61, ésto indica que se tendrán comportamientos de estados que mantienen una misma densidad en las siguientes generaciones, es decir, si se tiene un fondo periódico de puro *ether* y un *glider* evolucionando a través de ese fondo periódico, la cantidad de 0's y 1's es la misma conforme el *glider* se vaya desplazando.

Es importante notar que la curva de probabilidad de la *regla 110* tiene su punto máximo en aproximadamente 0.56, lo que implica que la probabilidad de que exista el estado 1 es mas alta que la probabilidad de que exista el estado 0. Por otra parte veamos que en los extremos de la curva sobre el eje q, la curva muestra un levantamiento muy rápido y es porque la cantidad de 1's que se generan en un paso es muy rápido, por ejemplo si en nuestra configuración inicial c_i únicamente ponemos una célula con el estado 1 y todas las demás células en el estado 0, veremos que cuando evolucionemos esa configuración tendremos que en cada generación crece en un factor doble, de acuerdo a cada generación c_i será al doble en c_{i+1} y c_{i+1} será el doble en c_{i+2} y así sucesivamente, hasta que el triángulo que se está formando encuentre el límite de la longitud del anillo y se equilibre la existencia de los estados. Además este fenómeno es idéntico cuando una célula en la configuración inicial tiene el estado 0 y todas las demás células tienen el estado 1, por eso es que la curva de probabilidad tiene esos crecimientos en ambos extremos.

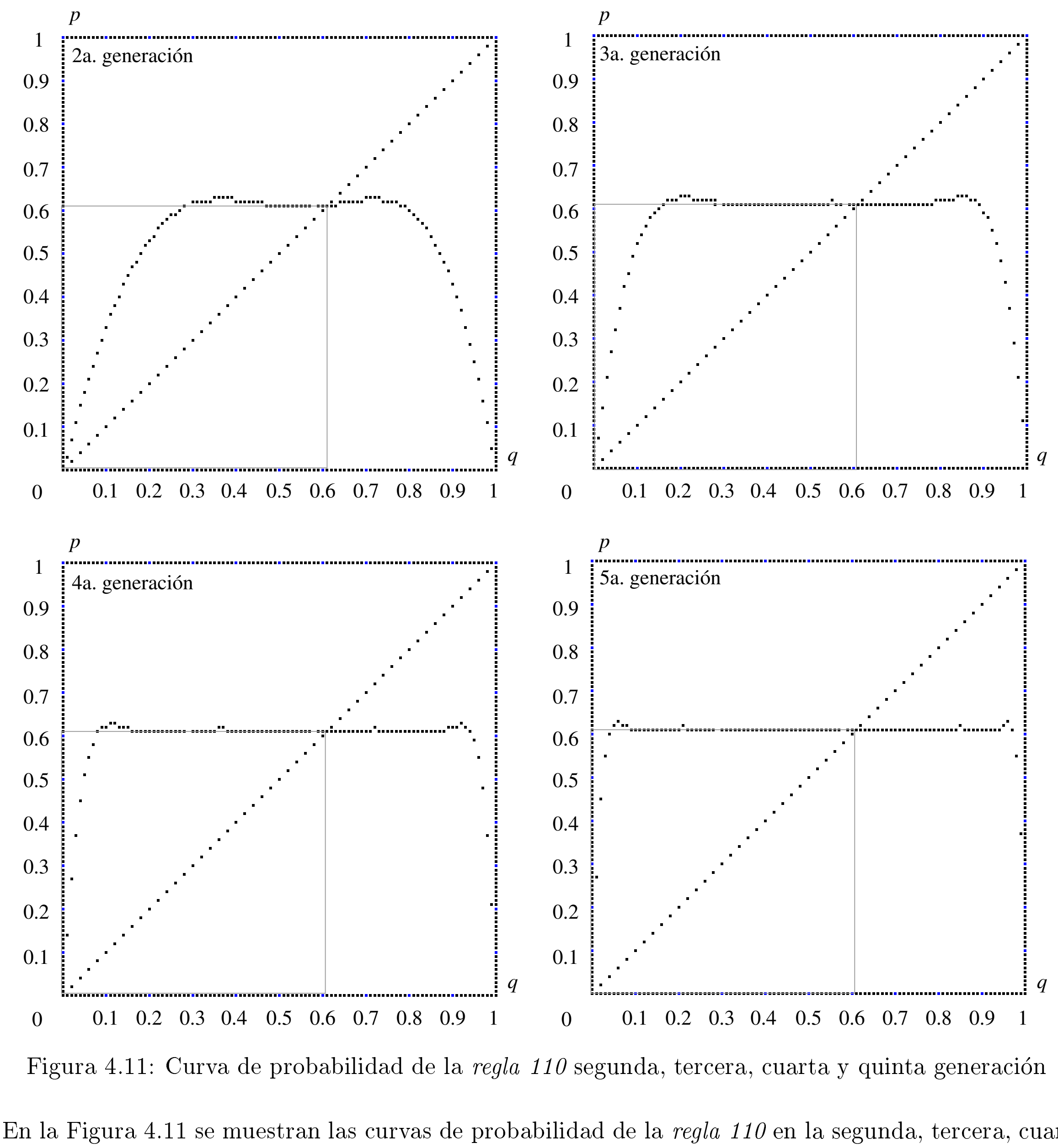


Figura 4.11: Curva de probabilidad de la *regla 110* segunda, tercera, cuarta y quinta generación

En la Figura 4.11 se muestran las curvas de probabilidad de la *regla 110* en la segunda, tercera, cuarta y quinta generación. Se puede ver que la probabilidad de tener 1's en cada generación se sigue manteniendo alta y ésto se puede explicar desde la regla de evolución, donde se tienen 5 vecindades que se transforman al estado 1 y 3 vecindades que se transforman al estado 0. El punto fijo estable deja de existir ya que la derivada en el punto de intersección vale 0, además se mantiene en cada generación ampliando cada vez mas el rango de probabilidad en obtener 1's en la siguiente generación en el eje x. Finalmente se puede ver que sigue una distribución uniforme en su curva de probabilidad, también mantiene el punto fijo inalterable en cada generación al igual que *Life*, pero su diferencia está en el rango de la probabilidad que tienen los estados en particular del estado 1, ésto se debe por el *ether* que existe en la *regla 110* y que no existe en *Life*, pero se puede ver que la *regla 110* tiene algunas características probabilísticas similares a *Life*. Sin lugar a duda a pesar de tener algunas similitudes con *Life*, la *regla 110* puede ser visto como un modelo original.

4.4 Comentarios finales

La *regla 110* de ser comprobado plenamente su funcionamiento, se puede llegar a decir que es un modelo mas sencillo que puede reproducir comportamientos como los que realizan los modelos de Conway y von Neumann, un autómata celular que trabaja en una dimensión, únicamente con dos estados y con una vecindad de tres células, además de la útilidad que se demostró al emplear diagramas de de Bruijn para reproducir parte de estas estructuras, recomendamos las siguientes referencias que tratan sobre la Sección 4.1 y la Sección 4.2 [14], [34], [35], [45], [46] y [58]. El estudio analítico también resulto útil ya que permite establecer una relación probabilística con la regla *Life* a pesar de su diferencia notable, de que *Life* no evoluciona en un fondo periódico como lo hace la *regla 110*, sin embargo no es un factor que impida que la *regla 110* pueda simular a *Life*, recomendamos las siguientes referencias que tratan sobre la Sección 4.1 y la Sección 4.2 [22], [23], [46], [45] y [46].

Capítulo 5

Conclusiones

En este capítulo se dan las conclusiones finales de la investigación realizada en esta tesis. En la Sección 5.1 se describe la relevancia de los resultados obtenidos así como las limitaciones que tiene este análisis. En la Sección 5.2 se describen algunos resultados experimentales interesantes que se obtuvieron a través del proceso de investigación en autómatas celulares de dos y una dimensión. En la Sección 5.3 se discuten los trabajos futuros en base a los resultados obtenidos, para esquematizar de alguna manera las rutas posibles a seguir en busca de una buena formalización teórica de los comportamientos de *Life* en general.

5.1 Resultados obtenidos

El autómata celular "*The game of Life*" es un modelo que se caracteriza por su sencillez, sin embargo a pesar de ser un modelo muy sencillo ha demostrado tener una complejidad impresionante. La teoría del campo promedio permite establecer características probabilísticas de la regla de evolución a nivel global y es aquí donde se tiene la primera limitante de este estudio, porque no describe el comportamiento a nivel local, que es precisamente donde se encuentra toda la complejidad de la regla.

La regla de evolución *Life* a pesar de ser una regla semitotalística merece mas estudio en cada una de las vecindades; como se menciono el comportamiento a nivel global se tiene en el conjunto de configuraciones finitas $\mathcal{C}_F(\Sigma)$ y la teoría del campo promedio permite determinar como se comportarán los estados del conjunto de estados por medio de su densidad a través del tiempo. La densidad que muestran cada uno de los estados permite deducir el comportamiento de los estados de una manera global y general, es decir, se sabe que son condiciones necesarias pero no suficientes para poder obtener evoluciones de ese tipo a nivel global, sin embargo estas condiciones hay que formalizarlas de manera mas rigurosa a nivel local en los conjuntos de configuraciones finitas $\mathcal{C}_F(\Sigma)$ e infinitas $\mathcal{C}(\Sigma)$.

El análisis con la teoría del campo promedio es sencillo y fácil de interpretar, por esta razón se han planteado cuestiones como la de establecer una clasificación de los autómatas celulares a través de esta teoría, es decir, de una manera analítica y no fenotípica. Aunque establecer una clasificación en los autómatas celulares está aún lejos de darse, es importante tratar de acercar sus características probabilísticas y estadísticas, para poder seguir buscando herramientas mas útiles que ayuden a explicar mejor estos fenómenos.

Es claro que el entendimiento de *Life* no es sencillo y el estudio en tres dimensiones implica crecer el problema de manera exponencial, sin embargo se aprovecha el trabajo realizado por Bays y aplicando la teoría del campo promedio se puede ver que las reglas de evolución tienen diferencias probabilísticas que ayudan a identificar mejor a cada regla de evolución y no solo confiar en los hechos fenotípicos. La importancia de estudiar estos comportamientos en tres dimensiones es por el simple hecho de que el mundo real está en tres dimensiones, pero es claro que su formalización es mas complicada.

El análisis realizado con la teoría del campo promedio se realiza de manera iterativa, es decir, se estudia la densidad de los estados de generación en generación, sin embargo este análisis puede ser mas refinado haciendo el análisis de manera compuesta; de manera compuesta no se obtiene la densidad por generaciones si no por configuraciones, ésto nos llevaría a deducir mas rápido el comportamiento global de los estados. Gutowitz utiliza la teoría de estructura local en autómatas celulares de una dimensión que proporciona resultados mas exactos, sin embargo aplicar la teoría de estructura local en mas dimensiones es mas complicado.

Por otra parte la aparición de la *regla 110* vino a dar un giro importante en los comportamientos que tiene y que además son similares a *Life*, la importancia radica en que es un modelo mas sencillo que *Life*, lo que permite emplear herramientas que se utilizan en una dimensión y de esta manera entender mejor sus comportamientos para obtener una mejor explicación de ellos. Como se explico en el Capítulo 4 los diagramas de de Bruijn determinan de manera precisa y directa la reproducción de algunos *gliders* que produce la *regla 110* y en general de las estructuras que puede producir la regla, de aquí surje el planteamiento de extender estos diagramas de de Bruijn a dos dimensiones y ver si se pueden reproducir las estructuras de *Life* con secuencias de estados a través de una gráfica. Posteriormente ésto permitiría explicar estructuras tan complejas como el *glider gun* y justificar los resultados aplicando teoría de matrices y teoría de gráficas.

Finalmente se puede decir que el estudio de la teoría del campo promedio en autómatas celulares similares a *Life*, ha dado resultados importantes en cuanto a características probabilísticas y que además permitió ver de manera mas clara parte de la complejidad de *Life*. La teoría del campo promedio es fácil de interpretar y aunque tiene limitaciones es útil como una aproximación mas clara y directa en la densidad que siguen los estados del autómata al límite en el espacio de evoluciones de manera global, ya que la teoría del campo promedio puede ser aplicada en autómatas celulares que manejen mas estados, mas vecinos y mas dimensiones.

5.2 Resultados experimentales

En el proceso de investigación se hallaron reglas de evolución con comportamientos interesantes, aunque algunos de estos resultados no se relacionan con *Life* propiamente. Una regla que presenta comportamientos interesantes es la regla semitotalística $R(2,4,3,3)$ que evoluciona con la vecindad de Moore, desde cualquier configuración aleatoria el espacio de evoluciones muestra la construcción de laberintos, estos laberintos se forman de manera natural y se van definiendo poco a poco, la construcción definitiva se alcanza en promedio entre 100 y 200 generaciones.

La construcción de estos laberintos es a través de estructuras fijas, aunque en algunas regiones se identifican estructuras periódicas muy parecidas a los *blinkers*, una evolución de este tipo se ilustra en la Figura 5.1, se empleó una cadena de 7 células en forma de renglón y el comportamiento final se alcanzo en 277 generaciones con una población de 2864 células vivas.

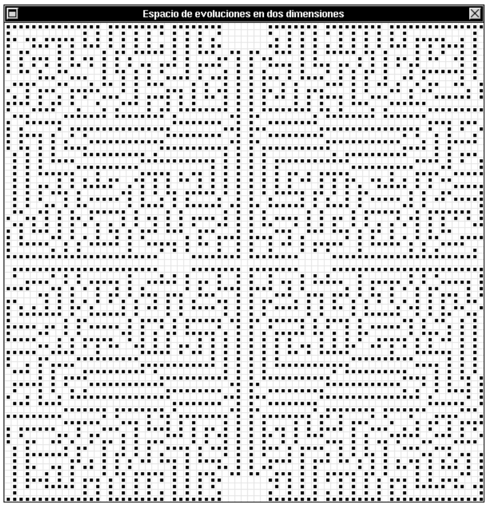


Figura 5.1: Regla semitotalística $R(2, 4, 3, 3)$ empleando vecindad de Moore

Otro tipo de evolución interesante es aquella que va construyendo un fondo fijo poco a poco, aunque se debe mencionar que esta regla no puede ser reproducida si se forma un toro con las células superiores e inferiores, se fijan los límites superiores e inferiores. La regla de evolución se obtuvo usando las 512 vecindades que tiene la vecindad de Moore como se ilustra en la Figura 5.2.

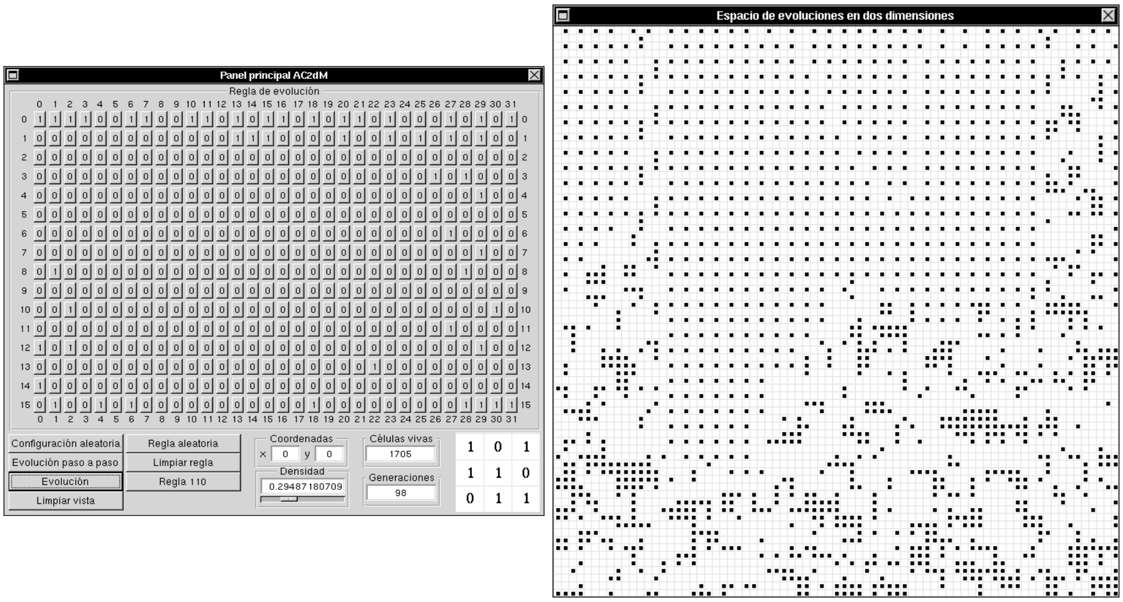


Figura 5.2: Comportamientos complejos empleando vecindad de Moore completa

Recordemos que la cantidad de reglas de evolución que se pueden derivar con las 512 vecindades son 2^{512} que es una cantidad grandísima. Por lo que el estudio en este tipo de reglas es muy amplio y se necesitaría demasiado tiempo para poder analizar cada una de estas reglas de evolución.

En referente a la *regla 110* se lograrón obtener proyecciones muy simples de esta regla en autómatas de dos dimensiones y una dimensión. Por ejemplo con el autómata celular de una dimensión de orden $(2, 2)$ regla 3CFC3CFC, se tienen evoluciones idénticas a los de la *regla 110*, para poder explicar este hecho veamos su regla de evolución en la Figura 5.3.

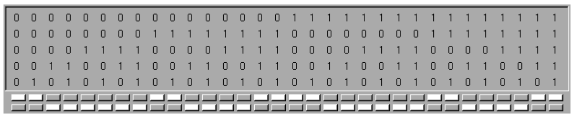


Figura 5.3: Autómata celular (2,2) regla 3CFC3CFC

La explicación consiste en que de las cinco células que conforman cada vecindad en esta regla, únicamente se toman las tres células centrales y aquellas que coincidan con las de la *regla 110* evolucionarán de igual manera, por esta razón es que se obtinen evoluciones idénticas. Empleando este enfoque con autómatas celulares en dos dimensiones que utilizan la vecindad de von Neumann, se puede hacer lo mismo proyectando únicamente las células horizontales o verticales que sean igual que las vecindades de la *regla 110* y definiendo una evolución del tipo de la *regla 110* como configuración inicial, con esto veremos como planos de configuraciones en dos dimensiones se corren de abajo a arriba o de izquierda a derecha en cada generación manteniendo estructuras del tipo de la *regla 110*.

Es importante ver que la *regla 110* puede tener proyecciones en dos dimensiones donde exista un fondo periódico y llenando el espacio con triángulos, por otra parte encontrar proyecciones en tres dimensiones tratando de encontrar estructuras como tetraedros desplazandose en el espacio tridimensional que simulen comportamientos de la *regla 110*.

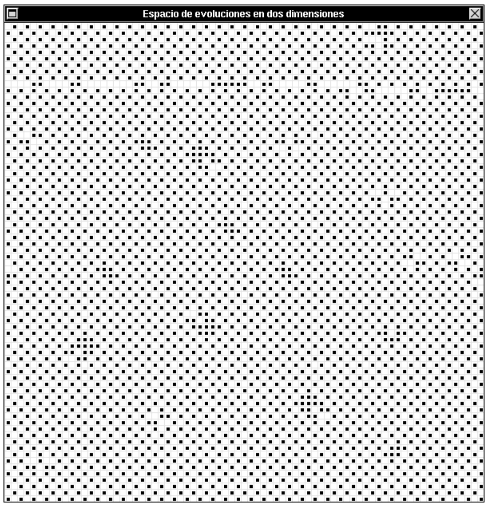


Figura 5.4: Regla semitotalística $R(2, 2, 2, 4)$ empleando vecindad de von Neumann

Analizando autómatas celulares en dos dimensiones se encontró la regla semitotalística $R(2, 2, 2, 4)$ que emplea la vecindad de von Neumann y evoluciona en dos dimensiones, esta regla de evolución tiene la característica de evolucionar a través de un fondo periódico como la *regla 110* y poder coexistir con estructuras periódicas como *blinkers*, cubriendo todo el espacio de evoluciones como se ilustra en la Figura 5.4. Estos comportamientos pueden obtenerse con configuraciones iniciales aleatorias con densidades promedio.

5.3 Trabajos futuros

El estudio de los autómatas celulares con comportamientos similares a *Life* resulta difícil de generalizar, sin embargo con el estudio probabilístico a través de la teoría del campo promedio se puede ver con mas claridad la siguiente etapa de estos estudios.

1. Realizar un análisis con la teoría del campo promedio de manera compuesta para reafirmar los resultados obtenidos en esta tesis.

2. Realizar un programa que pueda calcular raíces de polinomios con coeficientes muy altos y obtener resultados exactos de los polinomios del campo promedio.

3. Extender el estudio de los diagramas de de Bruijn en autómatas celulares de dos dimensiones para poder reproducir *gliders* o *still lifes* de manera precisa.

4. Determinar reglas de evolución en dos y tres dimensiones que realmente reproduscan problemas como es el tratar de llenar el espacio de evoluciones con triángulos y obtener comportamientos como la *regla 110*.

5. Realizar un programa completo para modelar autómatas celulares en tres dimensiones adecuadamente.

Apéndice A

Programas gráficos desarrollados en "Objetive-C"

En este apéndice se muestran los 5 programas desarrollados para esta tesis de entorno gráfico en autómatas celulares de una y dos dimensiones, estos programas estan códificados con el lenguaje orientado a objetos *Objetive-C*, implementados inicialmente en el sistema operativo OpenStep y transportados a Mac OS X "Acua", Mac OS X Server "Rhapsody" y Windows NT, haciendo notar la potencialidad de transportabilidad en cada una de las plataformas mencionadas.

Estos programas son: ACOSXL21 autómata celular en una dimensión de orden $k = 2$ y $r = 1$, ACOSXSTV autómata celular en dos dimensiones que trabaja con la vecindad de von Neumann y reglas de evolución semitotalísticas donde $\Sigma = \{0, 1\}$, ACOSXSTM autómata celular en dos dimensiones que trabaja con la vecindad de Moore y reglas de evolución semitotalísticas donde $\Sigma = \{0, 1\}$, ACOSXV autómata celular en dos dimensiones que trabaja con la vecindad de von Neumann y reglas de evolución completas donde $\Sigma = \{0, 1\}$ y ACOSXM autómata celular en dos dimensiones que trabaja con la vecindad de Moore y reglas de evolución completa donde $\Sigma = \{0, 1\}$.

El software que se usó y que es aplicable en cualquiera de las cuatros plataformas mencionadas es el *WebObjects 4.0.1 Developer*, en este apéndice solo se describe la construcción del programa ACOSXM con el sistema OpenStep en la Sección A.6, ya que los otros programas siguen un esquema similar. Finalmente se muestran pantallas de ejecución de cada una de las aplicaciones en cada una de las plataformas.

Los programas códificados en Windows NT tienen detalles que deben tomarse en cuenta, estos detalles no consisten precisamente de la aplicación en si. El sistema operativo MS-Dos no tiene un buen manejo de procesos y tampoco un buen manejo de la memoria RAM, esto implica que las aplicaciones como el ACOSXV y el ACOSXM consuman muchos recursos de la máquina, en realidad los programas pueden correr en Windows 95/98/2000 pero estos sistemas operativos son menos estables que el Windows NT, para correr adecuadamente se recomiendan 128MB de memoria RAM, pentium II a 233 Mhz y Windows NT como mínimo.

A.1 ACOSXL21

Autómata celular en una dimensión de orden $k = 2$ y $r = 1$, se puede variar el tamaño de las células, establecer configuraciones aleatorias de diferentes densidades, continuar la evolución si se desea y limpiar el espacio de evoluciones.

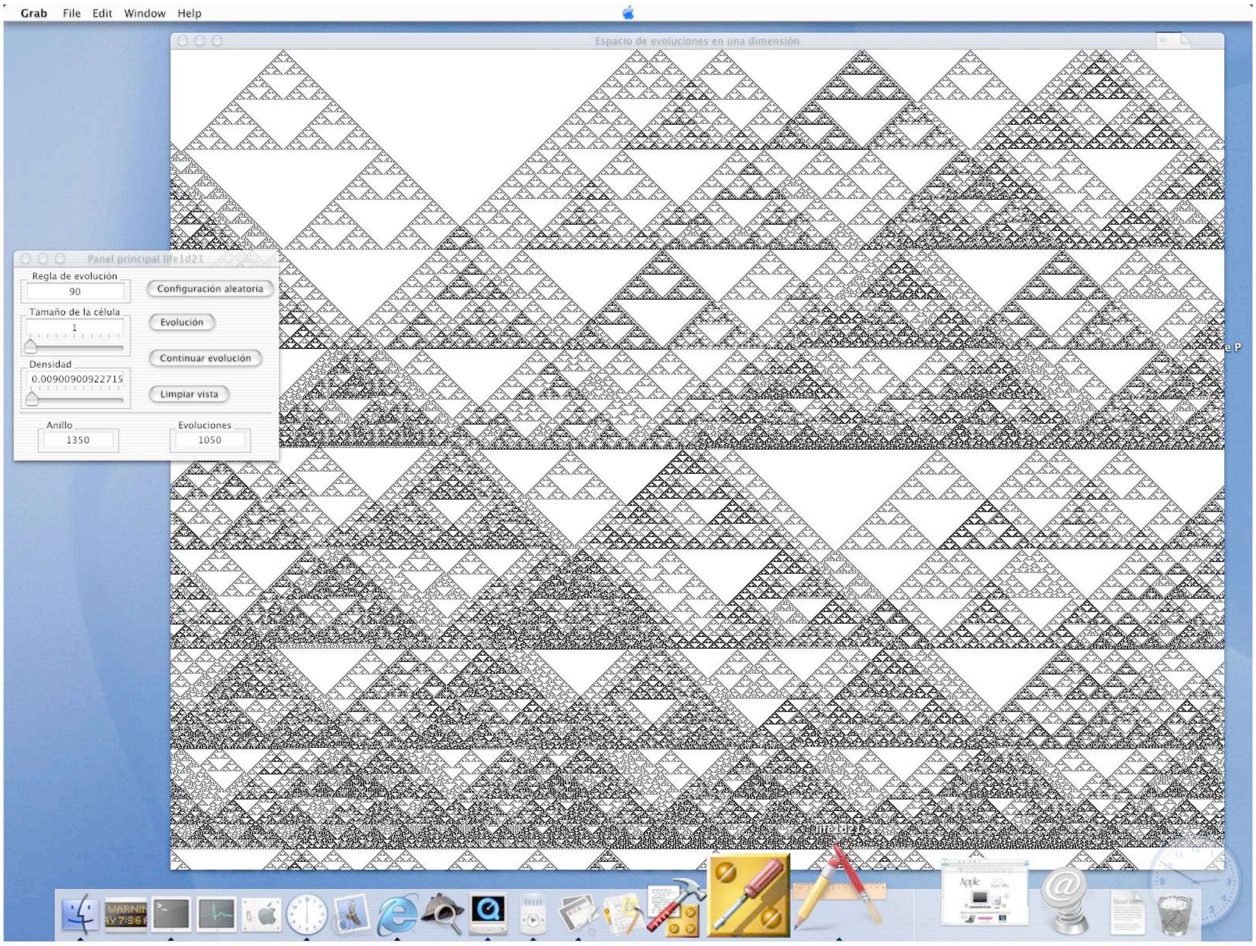


Figura A.1: ACOSXL21 para Mac OS X

En la Figura A.1 se muestra la aplicación ACOSXL21 en la plataforma Mac OS X y puede correr en las máquinas G3, G4 y G4 Cube de Macintosh.

A.2 ACOSXSTV

Autómata celular en dos dimensiones que trabaja con la vecindad de von Neumann y emplea reglas semi-totalísticas, se puede variar la regla de evolución en enteros de 1 a 4, establecer configuraciones aleatorias de diferentes densidades, introducir con el ratón células dentro del espacio de evoluciones, evolucionar paso a paso, evolucionar de manera contínua y detenerla en el momento que se desee pulsando el mismo botón, limpiar el espacio de evoluciones y se puede introducir una evolución del tipo de la *regla 110*.

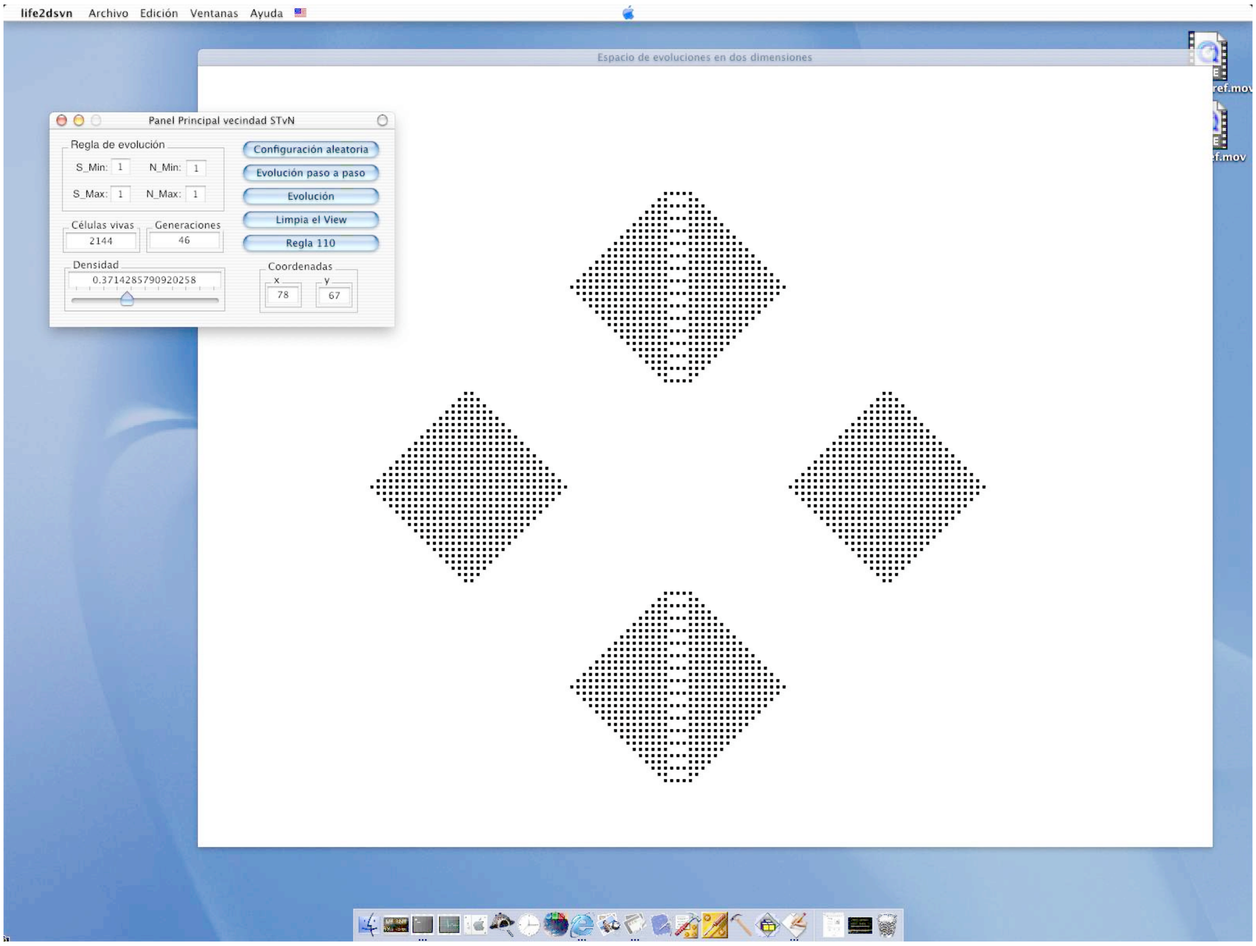


Figura A.2: ACOSXSTV para Mac OS X

En la Figura A.2 se muestra la aplicación ACOSXL21 en la plataforma Mac OS X y puede correr en las máquinas G3, G4 y G4 Cube de Macintosh.

A.3 ACOSXSTM

Autómata celular en dos dimensiones que trabaja con la vecindad de Moore y emplea reglas semitotalísticas, se puede variar la regla de evolución en enteros de 1 a 8, establecer configuraciones aleatorias de diferentes densidades, introducir con el ratón células dentro del espacio de evoluciones, evolucionar paso a paso y evolucionar de manera contínua, detenerla en el momento que se desee pulsando el mismo botón y limpiar el espacio de evoluciones.

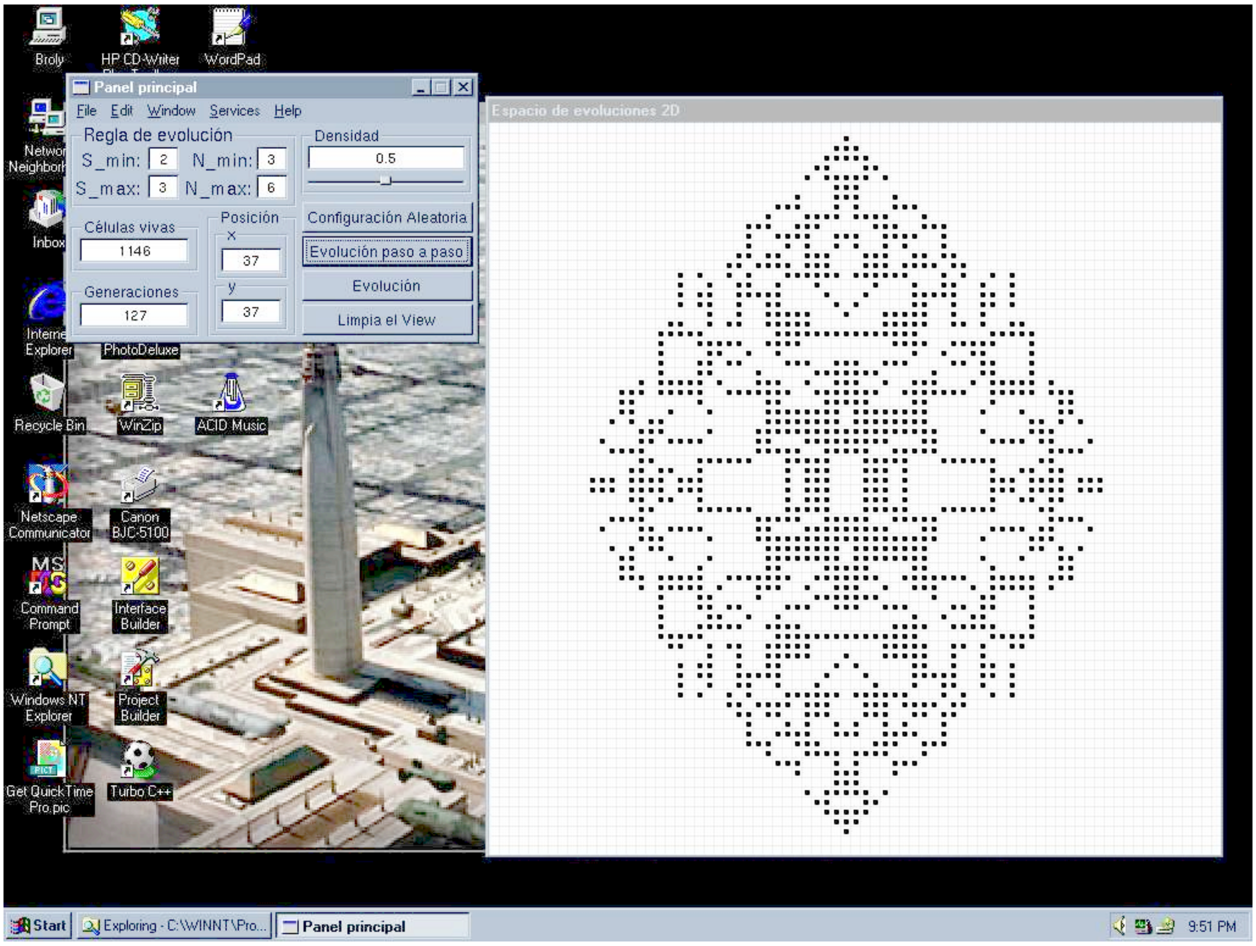


Figura A.3: ACOSXSTM para Windows NT

En la Figura A.3 se muestra la aplicación ACOSXSTM en la plataforma Windows NT y puede correr en las máquinas con Windows NT Workstation y Windows NT Server.

A.4 ACOSXV

Autómata celular en dos dimensiones que trabaja con la vecindad de von Neumann completa, es decir, se pueden manejar las 32 vecindades que se derivan de esta vecindad, la regla de evolución cambia su estado únicamente dando un clik a los botones, se pueden introducir reglas aleatorias, limpiar el arreglo de los botones, establecer configuraciones aleatorias de diferentes densidades, introducir con el ratón células dentro del espacio de evoluciones, evolucionar de manera contínua y detenerla en el momento que se desee pulsando el mismo botón, evolucionar paso a paso, introducir una evolución del tipo de la *regla 110* y limpiar el espacio de evoluciones.

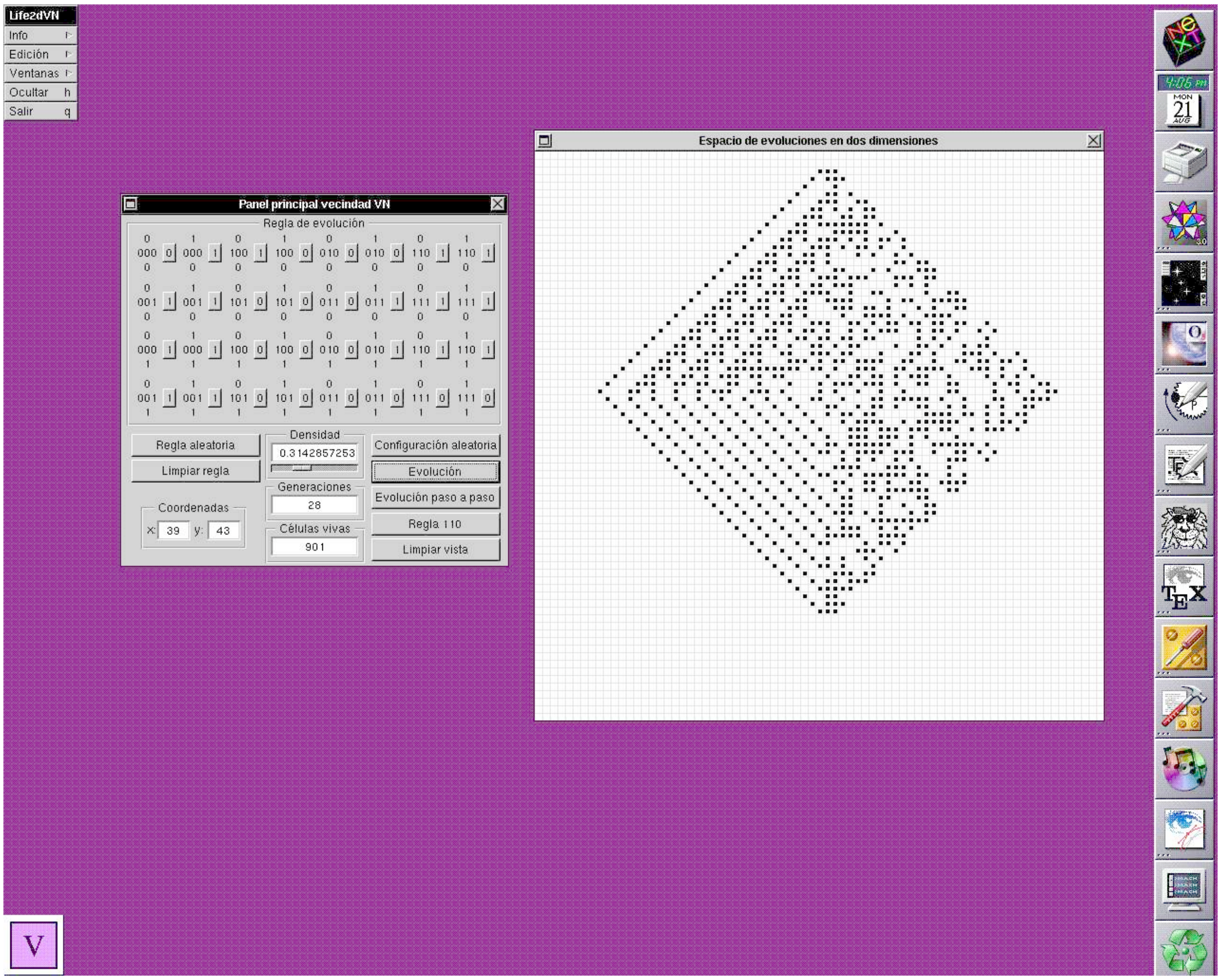


Figura A.4: ACOSXV para OpenStep

En la Figura A.4 se muestra la aplicación ACOSXV en la plataforma OpenStep y puede correr en las máquinas Pc's con el sistema operativo OpenStep, Next Station y Next Cube de Next Computer.

A.5 ACOSXM

Autómata celular en dos dimensiones que trabaja con la vecindad de Moore completa, es decir, se pueden manejar las 512 vecindades que se derivan de esta vecindad, la regla de evolución cambia su estado únicamente dando un clik a los botones y se muestra la vecindad que es activada, se pueden introducir reglas aleatorias, limpiar el arreglo de los botones, establecer configuraciones aleatorias de diferentes densidades, introducir con el ratón células dentro del espacio de evoluciones, evolucionar de manera contínua y detenerla en el momento que se desee pulsando el mismo botón, evolucionar paso a paso, introducir una evolución del tipo de la *regla 110* y limpiar el espacio de evoluciones.

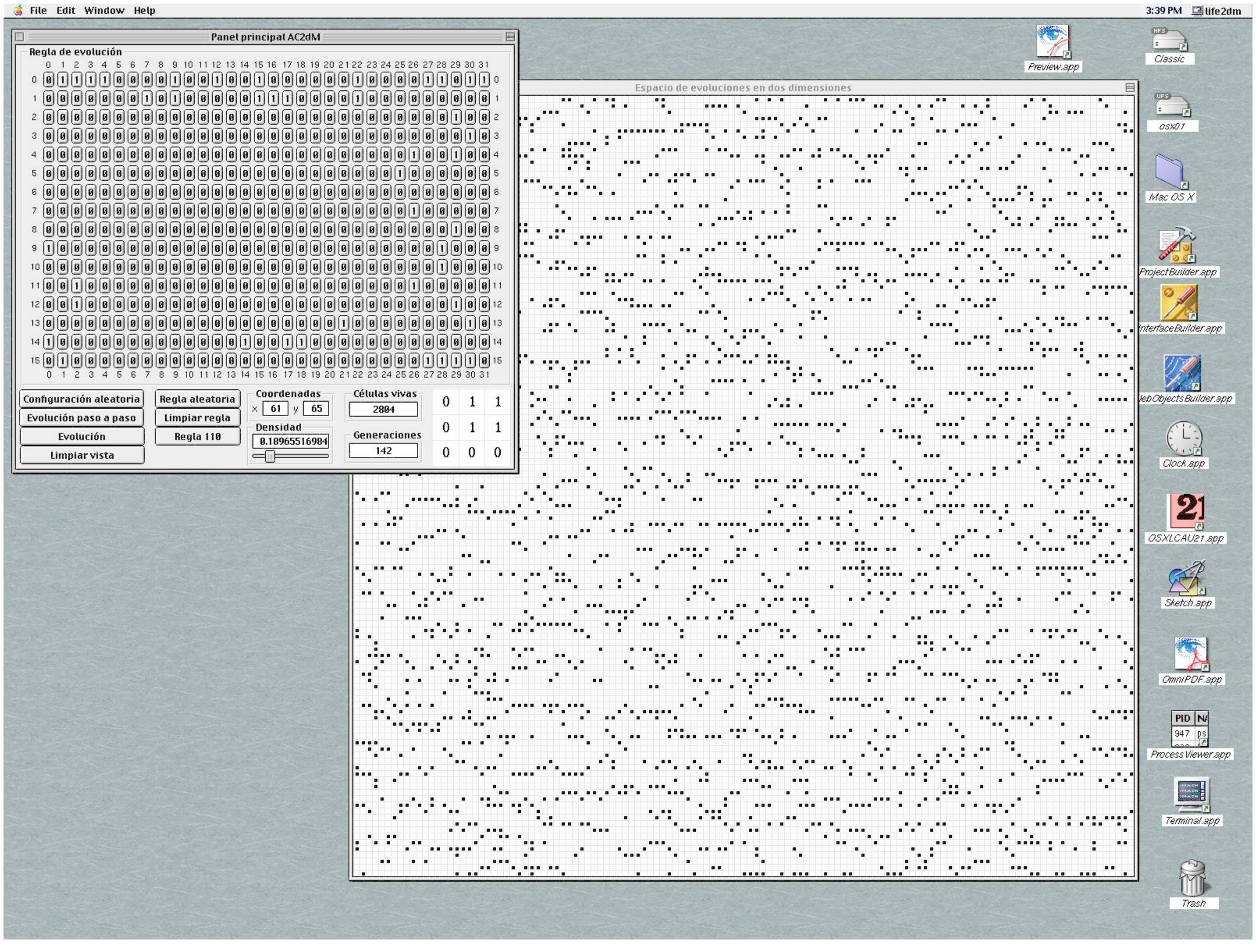


Figura A.5: ACOSXM para Mac OS X Server

En la Figura A.5 se muestra la aplicación ACOSXM en la plataforma Mac OS X Server y puede correr en las máquinas G3, G4 y G4 Cube de Macintosh.

A.6 Construcción del programa ACOSXM

Para poder desarrollar una aplicación con WebObjects se emplean dos herramientas importantes, la primera de ellas es el *Project Builder* Figura A.6 que permite manejar todos los recursos que usaremos a través de la construcción del proyecto.

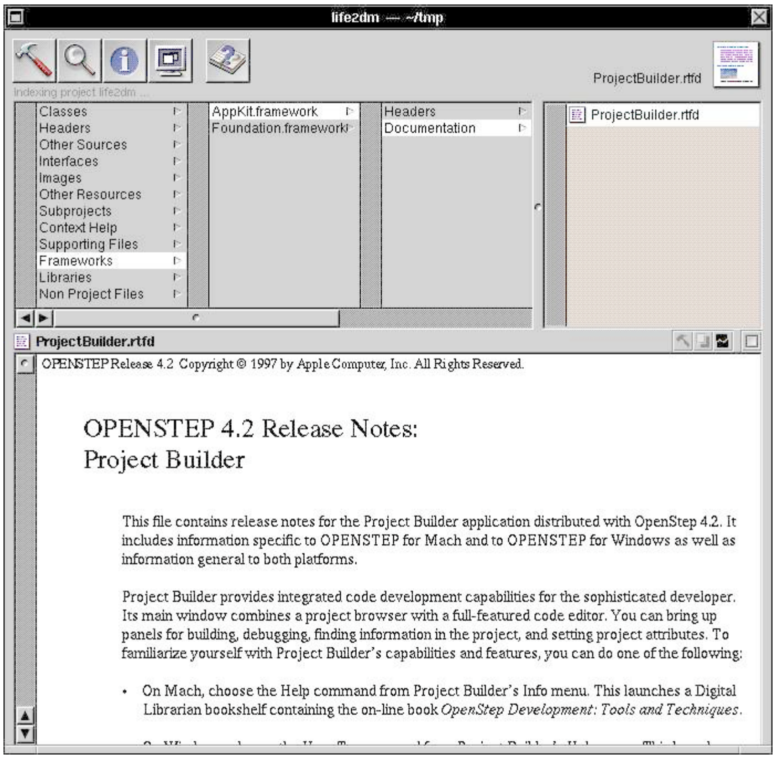


Figura A.6: Project Builder

La segunda herramienta importante es el *Interface Builder* Figura A.7 que permite crear nuestra interface gráfica de una manera muy sencilla.

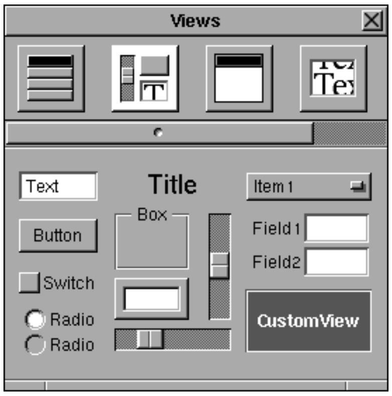


Figura A.7: Interface Builder

Primero abriremos el Project Builder y crearemos un proyecto nuevo de tipo aplicación como se ilustra en la Figura A.8.

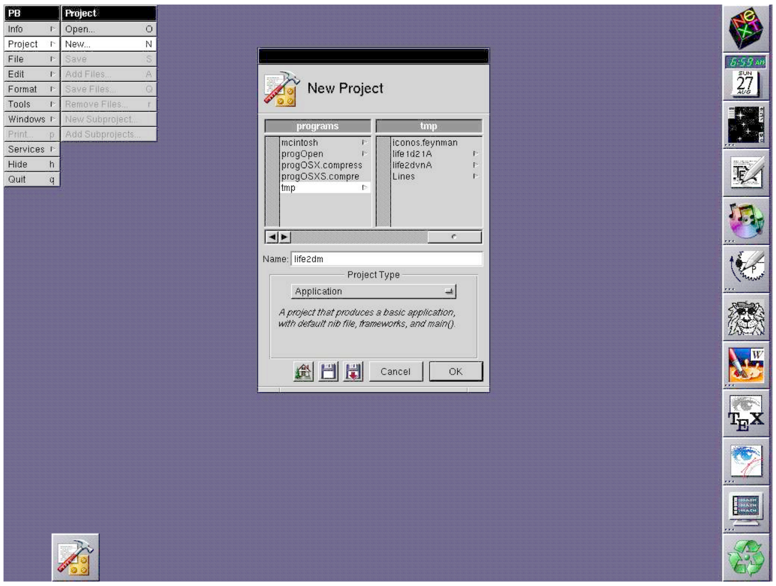


Figura A.8: Proyecto de tipo aplicación

El Project Builder creará un folder con el nombre que se le dio a la aplicación que contendrá todos los archivos necesarios que necesatará el proyecto, en el primer browser de izquierda a derecha nos muestra la clasificación de recursos que podemos usar, para empezar a construir la interface gráfica daremos un click en Interfaces y doble click en el archivo life2dm.nib, esto provocará que se abra automáticamente el Interface Builder como se muestra en la Figura A.9.

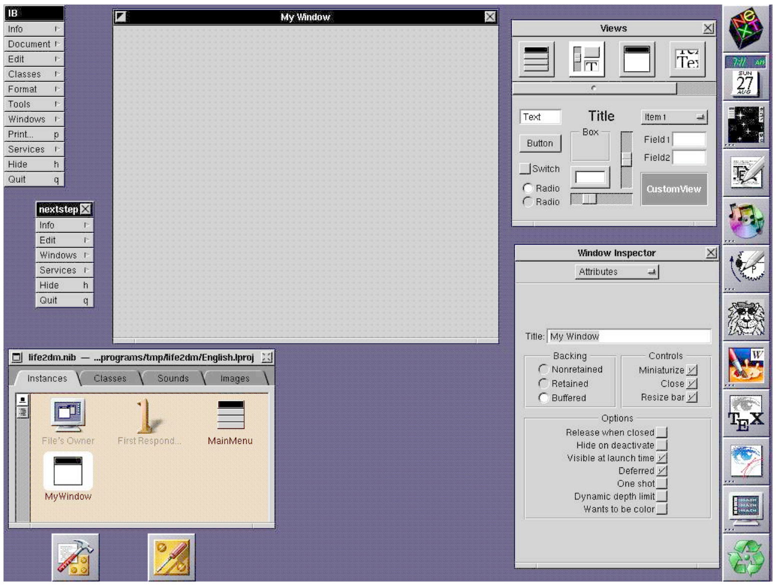


Figura A.9: Interface Builder inicial

Emplearemos dos ventanas de tipo NSWindow y arrastraremos los objetos de las paletas del Interface Builder que necesitemos hacia nuestras ventanas, para obtener dos ventanas como el de la Figura A.10 y sus atributos serán definidos a través del Inspector, que se encuentra en Tools → Inspector.

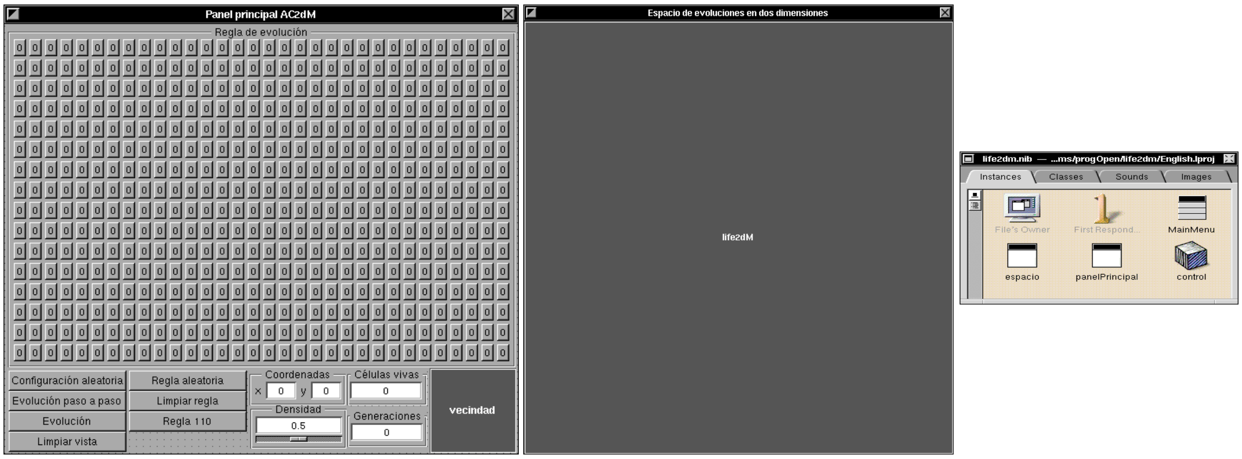


Figura A.10: Ventanas Panel principal y Espacio de evoluciones en dos dimensiones

La ventana de Panel principal tendrá 512 botones de tipo NSButton que cambiarán de la etiqueta 0 a 1 y viceversa, esta matriz de botones se hace seleccionando un botón con la etiqueta 0 y con la tecla Alt le damos un clik a la esquina inferior derecha y sin soltar el ratón lo arrastramos hasta crear una matriz de 16×32 botones, despues emplearemos 6 botones que controlarán las acciones de dibujar una configuración aleatoria (Configuración aleatoria), evolucionar paso a paso (Evolución paso a paso), evolucionar de manera continua (Evolución), limpiar el espacio de evoluciones (Limpiar vista), poner reglas aleatorias (Regla aleatoria), limpiar los botones de la regla (Limpiar regla) y finalmente dibujar una configuración como la *regla 110* en el espacio de evoluciones (Regla 110). El botón con la etiqueta Evolución deberá tener una etiqueta alterna que se llamará Parar evolución y sera de tipo Toggle con el Inspector. Un objeto view para dibujar la vecindad que representará cada botón, un campo de texto (Células vivas) que contará el número de células vivas en cada generación, un campo de texto (Generaciones) de tipo NSTextField que contará el número de generaciones que se vayan produciendo, dos campos de texto (x, y) que nos indicará la posición cuando una célula sea introducida o borrada en el espacio de evoluciones y un campo de texto (Densidad) que nos indicará la densidad de células en el espacio de evoluciones en una configuración aleatoria, a su vez este campo de texto será manipulado por un objeto de tipo NSSlider. Despues creamos cuatro tipos de subclases en la carpeta Classes de nuestra ventana life2dm.nib, una subclase de tipo NSObject llamada *control*, dos subclases de tipo NSView llamadas *vecindad* y *life2dM* y otra subclase de tipo NSMatrix llamada *matriz*.

Una vez creadas las clases se crearán sus archivos de código situandonos en el menú Classes → Create Files..., creará los archivos control.h, vecindad.h, matriz.h y life2dm.h en la carpeta Headers del Project Builder, también se crearán los archivos control.m, vecindad.m, matriz.m y life2dm.m en la carpeta Classes del Project Builder. En los archivos Headers se definirán los objetos y las acciones a las que respoderán los objetos que estan en la interface gráfica. La subclase *matriz* será conectada a la matriz de botones, *vecindad* al view que se encuentra en el Panel principal, *life2dm* será conectada al view que representará el espacio de evoluciones y la subclase control controlará la comunicación con todas las demás subclases como se ilustra en la Figura A.11

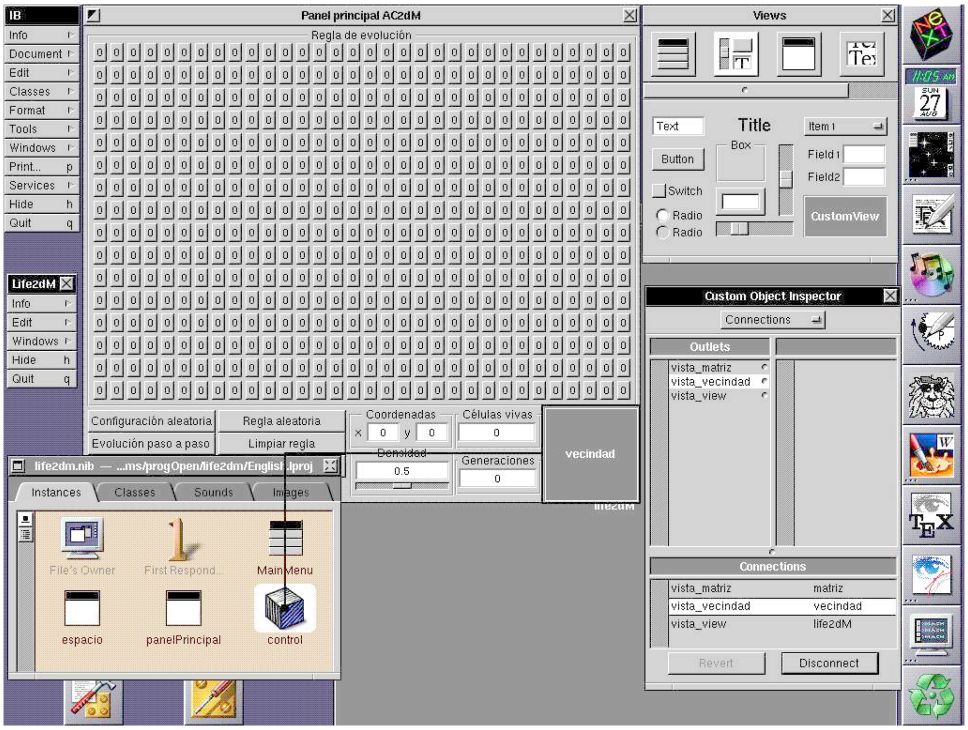


Figura A.11: Conexión de objetos

La subclase control será instanciada en Classes → Instantiate, los botones mandarán mensajes a la instancia control y estos mandarán mensajes a la subclase que pueda responder. Los objetos tienen asignados métodos a los cuales pueden responder, las clases pueden tener varios métodos que responderán a un mensaje dado, la conexión entre objetos y métodos se hará una vez que se haya realizado la programación correspondiente en cada una de las clases. Finalmente para construir la aplicación en el Project Builder iremos a Tools → ProjectBuild → Build & Run y se creará el archivo life2dm.app que es el archivo ejecutable de la aplicación, en los sistemas Acua, Rhapsody y OpenStep, en el caso de Windows NT el archivo se llama life2dm.exe. La transportación únicamente consiste en redefinir el archivo life2dm.nib en el Inspector del Project Builder que está en Tools → Inspector... y compilar en la nueva plataforma, aunque en Windows NT se debe reconstruir en life2dm-windows.nib de lo contrario fallará la construcción del archivo life2dm.exe, en el sistema Acua es recomendable revisar si la Interface no ha sido alterada ya que Mac OS X cuenta con mas objetos y la jerarquía de clases es mas extensa, además a diferencia de las otras plataformas Acua ya no emplea funciones PostScript para dibujar, mas bien objetos que responden a métodos de dibujo. Acontinuación mostraremos la codificación de cada una de las clases.

```
/***************** control.h *******************/

#import <AppKit/AppKit.h>
#import "matriz.h"
#import "life2dM.h"
#import "vecindad.h"

@interface control : NSObject
{
  /*Conecta a la clase NSMatrix matriz*/
```

```
  id vista_matriz;
  /*Conecta a la clase life2dM*/
  id vista_view;
  /*Conecta a la clase vecindad*/
  id vista_vecindad;
}

/*Manda el tag y la posicion de la celula*/
-(void)mandarPosicionTag:(id)sender;
/*Boton paso a paso*/
-(void)evolPaso:(id)sender;
/*Boton evolucion*/
-(void)evol:(id)sender;
/*Limpiar regla*/
-(void)limpiarRegla:(id)sender;
/*Regla aleatoria*/
-(void)reglaAleatoria:(id)sender;

@end
```

/**************** control.m ********************/**

```
#import "control.h"
int matriz_regla1[16][32];

@implementation control

/*Manda el tag y la posicion de la celula*/
-(void)mandarPosicionTag:(id)sender
{
  int renglon;
  int columna;
  int identificador;
  int estado;
  /*Captura el tag de la celula*/
  identificador=[[sender selectedCell] tag];
  /*Captura el renglon y la columna de la matriz*/
  renglon=[sender selectedRow];
  columna=[sender selectedColumn];
  /*Captura el estado*/
  estado=[[[vista_matriz cellAtRow:renglon column:columna] title] intValue];
  /*Le manda la posicion, el estado y el identificadora a la clase matriz*/
  [vista_matriz capturarPosicion:renglon:columna:identificador:estado];
  /*Le manda la vecindad a a la clase vecindad*/
  [vista_vecindad dibujarVecindad:identificador];
}

/*Boton paso a paso*/
-(void)evolPaso:(id)sender
{
  int i,j;
  /*Llamar a pasoaPaso de la clase life2dM*/
  for(i=0;i<16;i++)
        for(j=0; j<32; j++)
              matriz_regla1[i][j]=[vista_matriz traerVecindad:i:j];
  /*Traer la regla de control*/
  for(i=0;i<16;i++)
        for(j=0;j<32;j++)
              [vista_view establecerRegla:i:j:matriz_regla1[i][j]];
  /*Ejecuta el metodo paso a paso de la clase life2dM*/
  [vista_view pasoaPaso:sender];
}

/*Boton evolucion*/
-(void)evol:(id)sender
{
  int i,j;
  /*Llamar a pasoaPaso de la clase life2dM*/
  for(i=0;i<16;i++)
```

```
        for(j=0;j<32;j++)
                matriz_regla1[i][j]=[vista_matriz traerVecindad:i:j];
    /*Traer la regla de control*/
    for(i=0;i<16;i++)
        for(j=0;j<32;j++)
                [vista_view establecerRegla:i:j:matriz_regla1[i][j]];
    /*Ejecuta el metodo paso a paso de la clase life2dM*/
    [vista_view toggleRun:sender];
}

/*Limpiar regla*/
-(void)limpiarRegla:(id)sender
{
    int i,j;
    for(i=0;i<16;i++)
        for(j=0;j<32;j++)
                matriz_regla1[i][j]=0;
    [vista_matriz limpiarArreglo:self];
}

/*Regla aleatoria*/
-(void)reglaAleatoria:(id)sender
{
    int i,j;
    float y,flag=0.4;
    time_t segundos;
    time(&segundos);
    srand((unsigned)(segundos % 65536));
    for(i=0;i<16;i++)
    {
        for(j=0;j<32;j++)
        {
                y = rand()/pow(2,31);
                if(y>=flag)
                        matriz_regla1[i][j]=0;
                else
                        matriz_regla1[i][j]=1;
                [vista_matriz arregloAleatoria:i:j:matriz_regla1[i][j]];
        }
    }
}

@end

/****************** life2dm.h *******************/

#import <AppKit/AppKit.h>
#import "control.h"

@interface life2dM : NSView
{
    /*Cuenta el numero de celulas vivas por cada generacion*/
    id cellvivas;
    /*Cuenta el numero de iteraciones calculadas*/
    id generaciones;
    /*Determina la cantidad de celulas vivas al inicio*/
    id densidad;
    /*Captura coordenada directa del view*/
    id posicionx;
    id posiciony;
    NSTimer *linesTimer;
    BOOL running;
}

/*Inicializa la grafica*/
- initWithFrame:(NSRect)rects;
/*Dibuja la grafica*/
-(void)drawRect:(NSRect)rects;
/*Dibuja una celula*/
```

```
-(void)dibujaPunto:(int)x:(int)y:(float)rojo:(float)verde:(float)azul;
/*Dibuja una linea*/
-(void)dibujarLinea:(int)x1:(int)y1:(int)x2:(int)y2:(float)rojo:(float)verde:(float)azul;
/*Dibuja la malla*/
-(void)dibujaMalla:(id)sender;
/*Configuracion aleatoria*/
-(void)configuracionAleatoria:(id)sender;
/*Configuracion aleatoria*/
-(void)dibujaConfiguracion:(id)sender;
/*Traer regla de control*/
-(void)establecerRegla:(int)renglon:(int)columna:(int)estado;
/*Evolucion paso a paso*/
-(void)pasoaPaso:(id)sender;
/*Utiliza regla 110 para llenar el plano inicial*/
-(void)regla110:(id)sender;
/*Limpia la vista*/
-(void)limpiarVista:(id)sender;
/*Limpia todo*/
-(void)limpiarTodo:(id)sender;
/*Activa el mouse con mouse down*/
-(void)mouseDown:(NSEvent *)event;
- (void)dealloc;
- (void)toggleRun:sender;
- (void)animate:(NSTimer *)timer;

@end

/****************** life2dm.m ********************/

#import "life2dM.h"
#define LINES 600
#define RING 600
#define FACTOR 8
#define NUM 50
int longx,longy;
int arreglo[LINES/FACTOR][RING/FACTOR],inicial[LINES/FACTOR][RING/FACTOR];
int auxiliar[RING/FACTOR],matriz_aux[16][32];
int iteraciones;
int neighborhood[8];
int FLAG;
int ring,lines,lineas;

/*Convierte decimal en binario*/
int rule_binary(int ruleaux)
{
  int modulo;
  int divisor;
  int i=0;
  do
  {
        modulo = fmod(ruleaux,2);
        divisor = ruleaux/2;
        neighborhood[i] = modulo;
        ruleaux = divisor;
        i++;
  }while(i<8);
  return 0;
}

/*Reviza frontera superior*/
int check_sup(int k)
{
  int help;
  help=k;
  if(k<0)
        help=ring-1;
  if(k>ring-1)
        help=k-ring;
  return(help);
```

```
}

/*Reviza frontera inferior*/
int check_inf(int l)
{
  if(l>ring-1)
          l -= ring;
  return(l);
}

/*Reviza frontera izquierda*/
int check_izq(int k)
{
  int help;
  help=k;
  if(k<0)
          help=ring-1;
  if(k>ring-1)
          help=k-ring;
  return(help);
}

/*Reviza frontera derecha*/
int check_der(int l)
{
  if(l>ring-1)
          l -= ring;
  return(l);
}

@implementation life2dM

/*Inicializa la grafica*/
- initWithFrame:(NSRect)rects
{
  int i,j;
  [super initWithFrame:rects];
  [self allocateGState];
  FLAG=FACTOR;
  ring=RING/FACTOR;
  lines=LINES/FACTOR;
  iteraciones=0;
  /*Limpiar arreglos*/
  for(i=0;i<ring;i++)
  {
          auxiliar[i]=0;
          for(j=0;j<lines;j++)
          {
                  inicial[i][j]=0;
                  arreglo[i][j]=0;
          }
  }
  [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(appWillHide:)
  name:NSApplicationWillHideNotification object:[NSApplication sharedApplication]];
  [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(windowWillMiniaturize:)
  name:NSWindowWillMiniaturizeNotification object:nil];
  running = NO;
  return self;
}

/*Dibuja la grafica*/
-(void)drawRect:(NSRect)rects
{
  /*Limites del cuadro*/
  NSRect limites=[self bounds];
  /*Largo y ancho del cuadro y el incremento en radianes*/
  longx=limites.size.width;
  longy=limites.size.height;
  [[NSColor whiteColor] set];
```

```
  NSRectFill([self bounds]);
  /*Dibuja la malla*/
  [self dibujaMalla:self];
}

- (void)dealloc
{
  if (running)
  {
        [linesTimer invalidate];
        [linesTimer release];
  }
  [[NSNotificationCenter defaultCenter] removeObserver:self];
  [super dealloc];
}

- (void)toggleRun:(id)sender
{
  if (running)
  {
        [linesTimer invalidate];
        [linesTimer release];
        running = NO;
  }
  else
  {
        linesTimer = [[NSTimer scheduledTimerWithTimeInterval:0.0 target:self selector:
        @selector(animate:) userInfo:nil repeats:YES] retain];
        running = YES;
  }
}

- (void)animate:(NSTimer *)timer
{
  int i,j;
  int identificador,cociente,modulo;
  for(i=0;i<lines;i++)
  {
        for(j=0;j<ring;j++)
        {
              identificador=(pow(2,0)*inicial[i-1][j-1]) + (pow(2,1)*inicial[i-1][j]) +
              (pow(2,2)*inicial[i-1][j+1]) + (pow(2,3)*inicial[i][j-1]) + (pow(2,4)*inicial[i][j]) +
              (pow(2,5)*inicial[i][j+1]) + (pow(2,6)*inicial[i+1][j-1]) + (pow(2,7)*inicial[i+1][j]) +
              (pow(2,8)*inicial[i+1][j+1]);
              cociente=identificador/32;
              modulo=fmod(identificador,32);
              arreglo[i][j]=matriz_aux[cociente][modulo];
        }
  }
  /*Actualiza la siguiente configuracion*/
  for(i=0;i<lines;i++)
        for(j=0;j<ring;j++)
              inicial[i][j]=arreglo[i][j];
  for(i=0;i<lines;i++)
        for(j=0;j<ring;j++)
              arreglo[i][j]=0;
  /*Dibuja la nueva configuracion*/
  [self dibujaConfiguracion:self];
  iteraciones++;
}

- (void)windowWillMiniaturize:(NSNotification *)notification
{
  if (running && ([notification object] == [self window]))
        [self toggleRun:nil];
}

- (void)appWillHide:(NSNotification *)notification
{
```

```
  if (running)
          [self toggleRun:nil];
}

/*Dibuja una linea*/
-(void)dibujarLinea:(int)x1:(int)y1:(int)x2:(int)y2:(float)rojo:(float)verde:(float)azul
{
  PSsetrgbcolor(rojo,verde,azul);
  PSmoveto(x1,y1);
  PSlineto(x2,y2);
}

/*Dibuja una celula*/
-(void)dibujaPunto:(int)x:(int)y:(float)rojo:(float)verde:(float)azul
{
  /*Activa la accion del cuadro */
  [self lockFocus];
  PSnewpath();
  /*Dibujar Punto*/
  PSsetrgbcolor(rojo,verde,azul);
  PSarc(x,y,2,0,360);
  PSfill();
  /*Desactiva el cuadro*/
  PSstroke();
  [self unlockFocus];
}

/*Dibuja la malla*/
-(void)dibujaMalla:(id)sender
{
  int i;
  PSnewpath();
  /*Lineas verticales*/
  for(i=0;i<RING;i+=FACTOR)
          [self dibujarLinea:i:0:i:longy:0.80:0.80:0.80];
  /*Lineas horizontales*/
  for(i=0;i<LINES;i+=FACTOR)
          [self dibujarLinea:0:i:longx:i:0.80:0.80:0.80];
  PSstroke();
}

/*Configuracion aleatoria*/
-(void)configuracionAleatoria:(id)sender
{
  int i,j;
  float y,flag;
  time_t segundos;
  iteraciones=0;
  /*Calcular valores aleatorios*/
  time(&segundos);
  srand((unsigned)(segundos % 65536));
  flag=[densidad floatValue];
  for(i=0;i<lines;i++)
  {
          for(j=0;j<ring;j++)
          {
                  y = rand()/pow(2,31);
                  if(y>=flag)
                        inicial[i][j] = 0;
                  else
                        inicial[i][j] = 1;
          }
  }
  /*Dibuja la nueva configuracion*/
  [self dibujaConfiguracion:self];
}


/*Dibuja configuracion*/
```

```
-(void)dibujaConfiguracion:(id)sender
{
  int i,j,aux_x=0,aux_y=0,contador=0;
  /*Activa la accion del cuadro */
  [self lockFocus];
  /*Limpia el view*/
  [self limpiarVista:self];
  /*Dibujar la configuracion inicial*/
  for(i=0,aux_y=3;i<lines;i++,aux_y+=FACTOR)
  {
          for(j=0,aux_x=5;j<ring;j++,aux_x+=FACTOR)
          {
                   if(inicial[i][j]==1)
                   {
                          [self dibujaPunto:aux_x:aux_y:0:0:0];
                          contador++;
                   }
          }
  }
  /*Pinta el momento de generar el punto*/
  [[self window] flushWindow];
  [self unlockFocus];
  /*Dice cuantas celulas estan vivas*/
  [cellvivas setIntValue:contador];
  [generaciones setIntValue:iteraciones];
}

/*Utiliza regla 110 para llenar el plano inicial*/
-(void)regla110:(id)sender
{
  int i,j,tmp;
  float y,flag;
  time_t segundos;
  /*Limpiar arreglos*/
  for(i=0;i<lines;i++)
  {
          for(j=0;j<ring;j++)
          {
                   inicial[i][j]=0;
                   auxiliar[j]=0;
          }
  }
  /*Calcular valores aleatorios*/
  time(&segundos);
  srand((unsigned)(segundos % 65536));
  flag=[densidad floatValue];
  for(j=0;j<ring;j++)
  {
          y = rand()/pow(2,31);
          if(y>=flag)
                   inicial[0][j] = 0;
          else
                   inicial[0][j] = 1;
  }
  /*Configuracion inicial lineal*/
  for(j=0;j<ring;j++)
          auxiliar[j]=inicial[0][j];
  /*Transforma la regla en binario*/
  rule_binary(110);
  /*Calcula la evolucion*/
  for(i=1;i<lines;i++)
  {
          /*Calcula la siguiente evolucion*/
          for(j=0;j<ring;j++)
          {
                   tmp = auxiliar[check_inf(j-1)]*4 + auxiliar[j]*2 + auxiliar[check_sup(j+1)];
                   inicial[i][j]=neighborhood[tmp];
          }
          for(j=0;j<ring;j++)
```

```
                auxiliar[j]=inicial[i][j];
  }
  /*Dibuja la nueva configuracion*/
  [self dibujaConfiguracion:self];
}

/*Traer regla de control*/
-(void)establecerRegla:(int)renglon:(int)columna:(int)estado
{
  /*Traer regla de control*/
  matriz_aux[renglon][columna]=estado;
}

/*Evolucion paso a paso*/
-(void)pasoaPaso:(id)sender
{
  int i,j;
  int identificador,cociente,modulo;
  /*Aplicar funcion de transicion*/
  for(i=0;i<lines;i++)
  {
        for(j=0;j<ring;j++)
        {
                identificador=(pow(2,0)*inicial[i-1][j-1]) + (pow(2,1)*inicial[i-1][j]) +
                (pow(2,2)*inicial[i-1][j+1]) + (pow(2,3)*inicial[i][j-1]) + (pow(2,4)*inicial[i][j]) +
                (pow(2,5)*inicial[i][j+1]) + (pow(2,6)*inicial[i+1][j-1]) + (pow(2,7)*inicial[i+1][j]) +
                (pow(2,8)*inicial[i+1][j+1]);
                cociente=identificador/32;
                modulo=fmod(identificador,32);
                arreglo[i][j]=matriz_aux[cociente][modulo];
        }
  }
  /*Actualiza la siguiente configuracion*/
  for(i=0;i<lines;i++)
        for(j=0;j<ring;j++)
                inicial[i][j]=arreglo[i][j];
  for(i=0;i<lines;i++)
        for(j=0;j<ring;j++)
                arreglo[i][j]=0;
  /*Dibuja la nueva configuracion*/
  [self dibujaConfiguracion:self];
  iteraciones++;
}

/*Limpia el view*/
-(void)limpiarVista:(id)sender
{
  [[NSColor whiteColor] set];
  NSRectFill([self bounds]);
  [self dibujaMalla:self];
}

/*Limpia todo*/
-(void)limpiarTodo:(id)sender
{
  int i,j;
  [self display];
  /*Limpiar arreglos*/
  for(i=0;i<lines;i++)
  {
        auxiliar[i]=0;
        for(j=0;j<ring;j++)
        {
                inicial[i][j]=0;
                arreglo[i][j]=0;
        }
  }
  iteraciones=0;
  [cellvivas setIntValue:0];
```

```
  [generaciones setIntValue:iteraciones];
  /*Dibuja la nueva configuracion*/
  [self dibujaConfiguracion:self];
}

/*Activa el mouse con mouse down*/
-(void)mouseDown:(NSEvent *)event
{
  int cuenta_multiplo=0,auxiliar1=0;
  int m_der=0,m_izq=0,m_inf=0,m_sup=0,pos_x=0,pos_y=0,coor_x=0,coor_y=0;
  [posicionx setIntValue:[event locationInWindow].x];
  [posiciony setIntValue:[event locationInWindow].y];
  /*Restringuiendo valores para X*/
  if([posicionx intValue]<=0)
        [posicionx setIntValue:5];
  if([posicionx intValue]>(longx-7))
        [posicionx setIntValue:(longx-7)];
  /*Restringuiendo valores para Y*/
  if([posiciony intValue]<=0)
        [posiciony setIntValue:5];
  if([posiciony intValue]>(longy-7))
        [posiciony setIntValue:(longy-7)];
  /*Determina el lugar adecuado de la celula para el eje X*/
  cuenta_multiplo=[posicionx intValue]/FACTOR;
  coor_x=cuenta_multiplo;
  auxiliar1=cuenta_multiplo+1;
  m_der=cuenta_multiplo*FACTOR;
  m_izq=auxiliar1*FACTOR;
  if([posicionx intValue]>=m_der && [posicionx intValue]<=m_izq)
        pos_x=m_der+5;
  /*Limpia variables de conteo*/
  cuenta_multiplo=0;
  auxiliar1=0;
  /*Determina el lugar adecuado de la celula para el eje Y*/
  cuenta_multiplo=[posiciony intValue]/FACTOR;
  coor_y=cuenta_multiplo;
  auxiliar1=cuenta_multiplo+1;
  m_inf=cuenta_multiplo*FACTOR;
  m_sup=auxiliar1*FACTOR;
  if([posiciony intValue]>=m_inf && [posiciony intValue]<=m_sup)
        pos_y=m_inf+3;
  /*Visualiza las coordenadas*/
  [posicionx setIntValue:coor_x];
  [posiciony setIntValue:coor_y];
  /*Almacena arreglo de celulas*/
  if(inicial[coor_y][coor_x]==0)
  {
        inicial[coor_y][coor_x]=1;
        /*Activa la accion del cuadro */
        [self lockFocus];
        PSnewpath();
        /*Dibujar punto*/
        PSsetrgbcolor(0,0,0);
        PSarc(pos_x,pos_y,2,0,360);
        PSfill();
        /*Desactiva el cuadro*/
        PSstroke();
        /*Pinta el momento de generar el punto*/
        [[self window] flushWindow];
        [self unlockFocus];
  }
  else
  {
        inicial[coor_y][coor_x]=0;
        /*Activa la accion del cuadro */
        [self lockFocus];
        PSnewpath();
        /*Dibujar punto*/
        PSsetrgbcolor(1,1,1);
```

```
        PSarc(pos_x,pos_y,2,0,360);
        PSfill();
        /*Desactiva el cuadro*/
        PSstroke();
        /*Pinta el momento de generar el punto*/
        [[self window] flushWindow];
        [self unlockFocus];
  }
}

@end

/****************** matriz.h ********************/

#import <AppKit/AppKit.h>

@interface matriz : NSMatrix
{
}

/*Inicializa la matriz de botones*/
- initWithFrame:(NSRect)rects;
/*Captura la posicion de la celula y el tag*/
-(void)capturarPosicion:(int)renglon:(int)columna:(int)identificador:(int)estado;
/*Determina la vecindad en base el tag de la celula que va a evolucionar*/
-(int)traerVecindad:(int)renglon:(int)columna;
/*Limpia el arreglo*/
-(void)limpiarArreglo:(id)sender;
/*Arreglo aleatorio*/
-(void)arregloAleatoria:(int)renglon:(int)columna:(int)estado;

@end

/****************** matriz.m ********************/

#import "matriz.h"
/*Matriz que guarda la regla de evolucion*/
int matriz_regla[16][32];

@implementation matriz

/*Inicializa la matriz de botones*/
- initWithFrame:(NSRect)rects
{
  int i,j;
  [super initWithFrame:rects];
  /*Inicializa la matriz_regla en ceros*/
  for(i=0;i<16;i++)
        for(j=0;j<32;j++)
              matriz_regla[i][j]=0;
  return self;
}

/*Captura la posicion de la celula y el tag*/
-(void)capturarPosicion:(int)renglon:(int)columna:(int)identificador:(int)estado
{
  /*Determina el estado en la posicion capturada*/
  if(estado==0)
  {
        [[self cellAtRow:renglon column:columna] setTitle: [NSString stringWithFormat:@"%d",1]];
        /*Almacenar valores en la matriz original*/
        matriz_regla[renglon][columna]=1;
  }
  if(estado==1)
  {
        [[self cellAtRow:renglon column:columna] setTitle: [NSString stringWithFormat:@"%d",0]];
        /*Almacenar valores en la matriz original*/
        matriz_regla[renglon][columna]=0;
  }
```

```
}

/*Determina la vecindad en base el tag de la celula que va a evolucionar*/
-(int)traerVecindad:(int)renglon:(int)columna
{
/*Le mando el arreglo matriz_regla a control*/
return matriz_regla[renglon][columna];
}

/*Limpia el arreglo*/
-(void)limpiarArreglo:(id)sender
{
  int i,j;
  for(i=0;i<16;i++)
  {
        for(j=0;j<32;j++)
        {
              [[self cellAtRow:i column:j] setTitle: [NSString stringWithFormat:@"%d",0]];
              matriz_regla[i][j]=0;
        }
  }
}

/*Arreglo aleatorio*/
-(void)arregloAleatoria:(int)renglon:(int)columna:(int)estado
{
  matriz_regla[renglon][columna]=estado;
  if(matriz_regla[renglon][columna]==0)
        [[self cellAtRow:renglon column:columna] setTitle: [NSString stringWithFormat:@"%d",0]];
  else
        [[self cellAtRow:renglon column:columna] setTitle: [NSString stringWithFormat:@"%d",1]];
}

@end
```

/**************** vecindad.h ********************/**

```
#import<AppKit/AppKit.h>

@interface vecindad : NSView
{
}

/*Inicializa la grafica*/
- initWithFrame:(NSRect)rects;
/*Dibuja la grafica*/
-(void)drawRect:(NSRect)rects;
/*Dibuja una linea*/
-(void)dibujarLinea:(int)x1:(int)y1:(int)x2:(int)y2:(float)rojo:(float)verde:(float)azul;
/*Dibuja la malla*/
-(void)dibujaMalla:(id)sender;
/*Dibuja la vecindad*/
-(void)dibujarVecindad:(int)vecino;

@end
```

/**************** vecindad.m ********************/**

```
#import "vecindad.h"
int vecin[9];
int largox,largoy;

/*Convierte decimal en binario para la vecindad de Moore*/
int vecindad_binary(int vecino)
{
  int modulo;
  int divisor;
  int i=0;
  do
```

```
  {
        modulo = fmod(vecino,2);
        divisor = vecino/2;
        vecin[i] = modulo;
        vecino = divisor;
        i++;
  }while(i<9);
  return 0;
}

@implementation vecindad

/*Inicializa la grafica*/
- initWithFrame:(NSRect)rects
{
  [super initWithFrame:rects];
  [self allocateGState];
  return self;
}

/*Dibuja la grafica*/
-(void)drawRect:(NSRect)rects
{
  /*Limites del cuadro*/
  NSRect limites=[self bounds];
  largox=limites.size.width;
  largoy=limites.size.height;
  [[NSColor whiteColor] set];
  NSRectFill([self bounds]);
  [self dibujaMalla:self];
}

/*Dibuja una linea*/
-(void)dibujarLinea:(int)x1:(int)y1:(int)x2:(int)y2:(float)rojo:(float)verde:(float)azul
{
  PSsetrgbcolor(rojo,verde,azul);
  PSmoveto(x1,y1);
  PSlineto(x2,y2);
}

/*Dibuja la malla*/
-(void)dibujaMalla:(id)sender
{
  int i,j,flag;
  flag=largox/3;
  PSnewpath();
  /*Lineas verticales*/
  for(i=0,j=flag;i<3;i++,j=j+flag)
        [self dibujarLinea:j:0:j:largoy:0.80:0.80:0.80];
  /*Lineas horizontales*/
  for(i=0,j=flag;i<3;i++,j=j+flag)
        [self dibujarLinea:0:j:largox:j:0.80:0.80:0.80];
  PSstroke();
}

/*Dibuja la vecindad*/
-(void)dibujarVecindad:(int)vecino
{
  char s[5];
  /*Transformar de decimal a binario*/
  vecindad_binary(vecino);
  /*Limpiar view*/
  [self display];
  /*Poner texto*/
  [self lockFocus];
  PSsetrgbcolor(0,0,0);
  PSnewpath();
  PSselectfont("Times-Roman",18);
  sprintf(s,"%d",vecin[0]); PSmoveto(13,76); PSshow(s);
```

```
  sprintf(s,"%d",vecin[1]); PSmoveto(46,76); PSshow(s);
  sprintf(s,"%d",vecin[2]); PSmoveto(79,76); PSshow(s);
  sprintf(s,"%d",vecin[3]); PSmoveto(13,43); PSshow(s);
  sprintf(s,"%d",vecin[4]); PSmoveto(46,43); PSshow(s);
  sprintf(s,"%d",vecin[5]); PSmoveto(79,43); PSshow(s);
  sprintf(s,"%d",vecin[6]); PSmoveto(13,11); PSshow(s);
  sprintf(s,"%d",vecin[7]); PSmoveto(46,11); PSshow(s);
  sprintf(s,"%d",vecin[8]); PSmoveto(79,11); PSshow(s);
  [self unlockFocus];
}

@end
```

Lista de Figuras

Lista de Tablas

Bibliografía

[1] Michel Barsley, *Fractals everywhere*, Academic Press, 1988.

[2] Carter Bays, *Candidates for the Game of Life in Three Dimensions*, Complex Systems **1**, pp. 373-400, 1987.

[3] Carter Bays, *Classification of Semitotalistic Cellular Automata in Three Dimension*, Complex Systems **2**, pp. 235-254, 1988.

[4] Carter Bays, *A Note on the Discovery of a New Game of Three-dimensional Life*, Complex Systems **2**, pp. 255-258, 1988.

[5] Carter Bays, *The Discovery of a New Glider for the Game of Three-Dimensional Life*, Complex Systems **4**, pp. 599-602, 1990.

[6] Carter Bays, *New Game of Three-Dimensional Life*, Complex Systems **5**, pp. 15-18, 1991.

[7] Carter Bays, *A New Candidate Rule for the Game of Three-Dimensional Life*, Complex Systems **6**, pp. 433-441, 1992.

[8] Carter Bays, *Further Notes on the Game of Three-Dimensional Life*, Complex Systems **8**, pp. 67-73, 1994.

[9] David I. Bell, *HighLife - An Interesting Variant of Life*, Article for review received by life@cs.arizona.edu, April 17, 1994.

[10] Elwyn Berlekamp, John Conway and Richard Gut, *Winning Ways for your Mathematical Plays*, Academic Press, vol 2, chapter 25, 1982.

[11] Hendrik J. Blok and Birger Bergersen, *Synchronous versus asynchronous updating in the "game of Life"*, Physical Review E **59**, num. 2, pp. 3876-3879, 1999.

[12] David J. Buckingham and Paul B. Callahan, *Tight bounds on periodic cell configurations in Life*, April 15, 1997.

[13] E. F. Cood, *Cellular Automata*, Academic Press, 1968.

[14] Matthew Cook, *Introduction to the activity of rule 110*, (copyrigth 1994-1998 Matthew Cook) http://w3.datanet.hu/˜cook/Workshop/CellAut/Elementary/Rule110/110pics.html, January 1999.

[15] Karel Culik II and Sheng Yu, *Undecidability of CA Classification Schemes*, Complex Systems **2**, pp. 177-190, 1988.

[16] Hugues Chaté and Paul Manneville, *Collective Behaviors in Spatially Extended Systems with Local Interactions and Synchronous Updating*, Progress in Theoretical Physics, vol. **87**, pp. 1-60, 1992.

[17] Lee Earl Meeker, *Four-Dimensional Cellular Automata and The Game of Life*, Thesis Master of Science, University of South Carolina, 1998.

[18] Martin Gardner, *Mathematical Games - The fantastic combinations of John H. Conway's new solitaire game Life*, Scientific American **223**, pp 120-123, 1970.

[19] Martin Gardner, *Wheels, Life and other mathematical amusements*, W. H. Freeman and Company Press, New York, 1983.

[20] J. B. C. Garcia, M. A. F. Gomes, T. I. Jyh, T. I. Ren and T. R. M. Sales, *Nonlinear dynamics of the cellular-automaton "game of Life"*, Physical Review E **48**, pp. 3345-3351, no. 5, November 1993.

[21] R. W. Gerling, *Classification of three-dimensional cellular automata*, Physica A **162**, pp. 187-195, 1990.

[22] Howard Andrew Gutowitz, *Local structure theory for cellular automata*, Thesis Ph.D., University of Rockefeller, 1987.

[23] Howard A. Gutowitz, Jonathan D. Victor and Bruce W. Knight, *Local structure theory for cellular automata*, Physica D **28**, pp. 18-48, 1987.

[24] Howard A. Gutowitz and Jonathan D. Victor, *Local structure theory in more that one dimension*, Complex Systems **1**, pp. 57-68, 1987.

[25] Howard A. Gutowitz and Jonathan D. Victor, *Local structure theory: calculation on hexagonal arrays, and iteraction of rule and interaction of rule and lattice*, Journal of Statistical Physical **54**, pp. 495-514, 1989.

[26] Howard A. Gutowitz, *Mean Field vs. Wolfram Classification of Cellular Automata*, http://www.santafe.edu/~hag/mfw/mfw.html, November 1989.

[27] Howard A. Gutowitz and Chris Langton, *Mean Field Theory of The Edge of Chaos*, Proccedings of ECAL, Lecture Notes in Computer Science, **929** Springer 1995.

[28] Howard Gutowitz and C. Domian, *The Topological Skeleton of Cellular Automaton Dynamics*, Physica D (submitted), 1995.

[29] J. Hardouin Duparc, *Generalization of "Life"*, Dynamical Systems and Cellular Automata, Academic Press, London, 1985.

[30] Jan Hemmingsson, *A totalistic three-dimensional cellular automaton with quasiperiodic behavior*, Physica A **183**, pp 255-261, 1985.

[31] Gustav A. Hedlund, *Endomorphisms and automorphisms of the shift dynamical systems*, Mathematical Systems Theory **3**, pp 320-375, 1969.

[32] Francisco Jiménez Morales, *Evolving three-dimensional cellular automata to perform a quasiperiod-3 collective behavior task*, Physical Review E **60**, pp 4934-4940, 1999.

[33] Genaro Juárez Martínez, *Grados de Reversibilidad en Autómatas Celulares Lineales*, Tesis de Licenciatura, Universidad Nacional Autónoma de México, Campus ENEP Acatlán, 1998.

[34] Genaro Juárez Martínez y Juan Carlos Seck Tuoh Mora, *Analizando gliders en la regla 110*, Reporte de Investigación, Centro de Investigaciones y Estudios Avanzados del I.P.N., sin públicar, 1999.

[35] Kristian Lindgren and Mats G. Nordahl, *Universal Computing in Simple One-Dimensional Cellular Automata*, Complex Systems **4**, pp. 299-318, 1995.

[36] Douglas Lind and Brian Marcus, *An Introduction to Symbolic Dynamics and Coding*, Cambridge University Press, 1995.

[37] G. G. Lorentz, *Bernstein Polynomials*, University of Toronto Press, Toronto, 1953.

[38] Michael Magnier, Claude Lattaud and Jean-Claude Heudin, *Complexity Classes in the Tow-dimensional Life Cellular Automata Subspace*, Complex Systems **11**, no. 6, pp. 419-436, 1997.

[39] S. S. Manna and D. Stauffer, *Systematics of transitions of square-lattice cellular automata*, Physica A **162**, pp. 176-186, 1990.

[40] Marvin L. Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall, 1967.

[41] Harold V. McIntosh, *Linear Cellular Automata*, Universidad Autónoma de Puebla, Instituto de Ciencias, Departamento de Aplicación de Microcomputadoras, 1987.

[42] Harold V. McIntosh, *Life's Still Lifes*, Universidad Autónoma de Puebla, Instituto de Ciencias, Departamento de Aplicación de Microcomputadoras, 1988.

[43] Harold V. McIntosh, *A Zoo of Life Forms*, Universidad Autónoma de Puebla, Instituto de Ciencias, Departamento de Aplicación de Microcomputadoras, 1988.

[44] Harold V. McIntosh, *Wolfram's Class IV and a Good Life*, Physica D **45**, pp. 105-121, 1990.

[45] Harold V. McIntosh, *Linear cellular automata via de Bruijn diagrams*, Universidad Autónoma de Puebla, Instituto de Ciencias, Departamento de Aplicación de Microcomputadoras, 1991.

[46] Harold V. McIntosh, *Rule 110 as it relates to the presence of gliders*, http://delta.cs.cinvestav.mx/~mcintosh, February 2000.

[47] Melanie Mitchell, *Computation in Cellular Automata: A Selected Review*, Santa Fe Intitute, NM 87501 U.S.A., September 1996.

[48] Norman H. Packard and Stephen Wolfram, *Two-dimensional cellular automata*, Journal of Statistical Physics **38**, pp. 901-946, 1985.

[49] Clark Robinson, *Dynamical Systems: stability, symbolic dynamics and chaos*, CRC Press, 1994.

[50] L. S. Schulman and P. E. Seiden, *Statistical mechanics of a dynamical system based on Conway's game of Life*, Journal of Statistical Physics **19**, pp. 293-314, 1978.

[51] Claude E. Shannon, *Von Neumann's Contributions to Automata Theory*, Automata Studies, pp. 123-129, 1958.

[52] Harold S. Stone, *Discrete Mathematical Structures and their Applications*, Computer Science Series, Stanford University, 1973.

[53] Tommaso Toffoli and Norman Margolus, *Cellular Automata Machines*, The MIT Press, Cambridge, Massachusetts, 1987.

[54] John von Neumann, *Theory of Self-reproducing Automata* (edited and completed by A. W. Burks), University of Illinois Press, 1966.

[55] Stephen Wolfram, *Statistical mechanics of cellular automata*, Reviews of Modern Physics **55**, pp. 601-644, 1983.

[56] Stephen Wolfram, *Universality and complexity in cellular automata*, Physica D **10**, pp. 1-35, 1984.

[57] Stephen Wolfram, *Theory and Aplications of Cellular Automata*, World Scientific Press, Singapore, 1986.

[58] Stephen Wolfram, *Cellular Automata and Complexity: collected papers*, Addison-Wesley Publishing Company, 1994.

[59] Andrew Wuensche and Mike Lesser, *The Global Dynamics of Cellular Automata*, Santa Fe Institute in the Sciences of Complexity, 1992.

[60] Andrew Wuensche, *Classifying Cellular Automata*, Complexity **4**, pp. 47-66, 1999.