CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL IPN
INGENIERÍA ELÉCTRICA
SECCIÓN COMPUTACIÓN

PROCEDIMIENTO PARA PRODUCIR COMPORTAMIENTOS COMPLEJOS EN REGLA 110

Tesis que presenta:

Genaro Juárez Martínez

PARA OBTENER EL GRADO DE
DOCTOR EN CIENCIAS
EN LA ESPECIALIDAD DE
INGENIERÍA ELÉCTRICA

Asesores:

Harold V. McIntosh

Sergio V. Chapa Vergara

MÉXICO DF OCTUBRE, 2006

A mis padres:

Samuel Juárez y Maria Luisa Martínez Yonca

y hermanos:

Erik Juárez Martínez, Fabian Juárez Martínez y

Georgina Verónica Juárez Martínez

A la niña más linda y tierna:

Adriana de la Paz Sánchez Moreno

Contenido

Agradecimientos

Principalmente al invaluable apoyo y cariño de mis padres Samuel Juárez y Maria Luisa Martínez Yonca a quienes quiero mucho. A mis hermanos Erik, Fabian y Georgina que siempre me han apoyado en todo momento y también quiero mucho. A la niña que quiero con todo mi corazón Adriana de la Paz.

Un agradecimiento muy especial a los profesores y amigos Harold V. McIntosh, Juan Carlos Seck Tuoh Mora, Andrew Adamatzky, Matthew Cook, Andrew Wuensche, David Hillman, Paul Chapman, Fred Lunnon y Cristopher Moore, por sus asesorías, contribuciones y discusiones en Regla 110. Al Departamento de Aplicación de Microcomputadoras de la Universidad Autónoma de Puebla, que en todo momento siempre me han apoyado y ayudado mucho. Al apoyo de CONACyT siempre constante y muy responsable en todo el tiempo que estuve becado por la institución (número de registro 139509). Al apoyo por parte del British Council Mexico a través de una excelente persona y amiga Lucia Pérez Moreno. Al invaluable apoyo de The Royal Society of England. Al apoyo de la University of the West of England y al Engineering & Physical Sciences Research Council. Al International Centre of Unconventional Computing Group y a la Universitët Osnabrück Dustchland.

Otro agradecimiento muy especial a todos los investigadores que han estado interesados en esta investigación y contribuyeron de diferentes maneras con documentos, mensajes, correciones, críticas y opiniones muy importantes en varios temas de este documento. A los excelentes investigadores y amigos: Kenneth Steiglitz, George Maydwell, Yurii Rogozhin, Manfred Kudlek, Stephen Wolfram, Markus Redeker, Nino Boccara, Thomas Worsch, Germán González, Rafael Baquero, Sergio Chapa, Tim Tyler, Edwin Clark, Matthew Frank, Mirko Rahn, Melanie Mitchell, David Drysdale, Henry Cohn, Steven Krantz, Todd Rowland, Carlos Gershenson, Martin Schneider y Shmuel Penchas.

A amigas y amigos muy apreciados: Miriam Josseline y Karen Rangel, Lizeth Contreras, Clarissa Corral, Jessica, Joana, Julio, Maria Elena, Alejandra Sánchez y al grupo iGEM México. Al comité revisor y a la Sección de Computación del CINVESTAV.

México DF
Septiembre, 2006

Genaro Juárez Martínez
http://uncomp.uwe.ac.uk/genaro/

Resumen

El objetivo de esta investigación en el autómata celular Regla 110, es desarrollar un procedimiento para construir condiciones iniciales que permitan producir y controlar comportamientos complejos a través de choques entre estructuras (conocidos como gliders, partículas, auto-localidades móviles o patrones en la literatura de autómata celular), donde los gliders representan el conjunto de elementos (objetos de su universo) que existen en el espacio de evoluciones de la Regla 110.

Las construcciones de condiciones iniciales se realizan a través de una representación que llamamos fases, donde las fases son un conjunto finito de expresiones regulares. Utilizamos diagramas de de Bruijn para determinar las expresiones regulares en la Regla 110 y también utilizamos un análisis por mosaicos que describe la construcción general del espacio de evoluciones. La combinación de ambos análisis: los diagramas de de Bruijn y mosaicos, dan origen a nuestro procedimiento por fases.

La investigación en la Regla 110 inicia representando el espacio de evoluciones a través de mosaicos que nos permite determinar propiedades de todos los gliders de la Regla 110 como son: distancia, período, desplazamiento, velocidad, puntos de contacto y forma interna, para toda estructura que exista en su espacio de evoluciones.

En el estudio de autómata celular con comportamiento complejo como son: el Juego de la Vida, Regla 54, High Life, Life-3d, regla Beehive, la misma Regla 110; no se tiene reportado ningún procedimiento para codificar condiciones iniciales en un autómata celular con comportamiento complejo. De esta manera, nuestro procedimiento para codificar y construir condiciones iniciales es nuevo no sólo en la Regla 110 sino también en una dimensión.

Finalmente, implementamos el procedimiento en un programa de computación que permite construir condiciones iniciales codificadas en fases para obtener tanto evoluciones simples como complejas en la Regla 110. Entre estas evoluciones podemos mencionar: la producción de gliders, la producción de grandes triángulos, la reproducción del sistema tag cíclico y la construcción de operaciones lógicas, todos ellos basados en choques. Cada una de las evoluciones mencionadas son resueltos y presentamos su respectiva codificación para

su reproducción en nuestra notación de fases.

Abstract

The goal of this investigation in the cellular automaton Rule 110 is to develop a procedure for constructing initial conditions able to produce complex behaviors through collisions between structures (known like gliders, particles, mobile self-localization or patterns in the literature of cellular automata), representing the set of elements in the evolution space of Rule 110.

The construction of initial conditions is realized using a particular representation called *phases*, composed by a finite set of regular expressions. We use de Bruijn diagrams to determine the regular expressions in Rule 110 and also an analysis by tiles describing the general construction of the evolution space. The combination with both tools: de Bruijn diagrams and tiles, forms our procedure based on phases.

The study begins representing the evolution space through tiles allowing to determine general properties for every glider in Rule 110, for instance: distances, periods, shifts, speeds, contact points and internal structure.

In cellular automata as: the Game of the Life, Rule 54, High Life, Life-3d, Beehive rule, the same Rule 110, it has not been reported some procedure to codify initial conditions yielding complex behaviors. This way, the relevance of the paper consists on presenting a novel procedure not only in Rule 110 but in the one-dimensional case, to codify and construct interesting initial conditions.

Finally, we implement the procedure in a computer system codifying initial conditions in phases to simulate both simple and complex problems in Rule 110. Some of the problems treated in the paper are: the production of gliders, the production of great triangles, the reproduction of the cyclic tag system and the construction of logic operations, all them collision-based structures. Each one of the mentioned problems is solved and its respective codification for reproduction is exemplified with our phases notation.

Prefacio

El estudio del autómata celular (AC) Regla 110 ofrece un interesante caso de estudio en teoría de AC con comportamiento complejo en una dimensión (1D). Nuestro interés por estudiar la Regla 110 surge dada la amplia variedad de gliders que existen en su espacio de evoluciones y el número de choques que se pueden producir, originando grandes cantidades de información que surgen en su espacio.

El resultado principal de nuestra investigación en la Regla 110 es el desarrollo de un procedimiento para construir condiciones iniciales, codificadas a través de un conjunto finito de expresiones regulares que son obtenidas a través de las fases para todos los gliders, consecuentemente obtenemos un lenguaje regular basado en gliders para la Regla 110. Además, el procedimiento es implementado en un programa de computación donde el procedimiento es aplicado para codificar condiciones iniciales desde una ventana de fases (nuestra máquina de estados finitos). Es importante recalcar que el procedimiento y el programa son nuevos en el estudio de la Regla 110.

La construcción de condiciones iniciales es a través de una codificación expresada en fases, donde las fases representan una secuencia del conjunto finito de expresiones regulares que determinan a cada glider. Las secuencias representan cada uno de los gliders específicamente y no son considerados extensiones o paquetes de ellos, porque el conjunto es no-finito.

El procedimiento es implementado en un programa de computación y lo llamamos "OSXLCAU21." El código y aplicación del programa son de dominio público disponible para los sistemas operativos OPENSTEP, Mac OS X y Windows.[1]

Una vez determinado el lenguaje regular basado en gliders lo aplicamos para construir condiciones iniciales y resolvemos algunos problemas complicados en la Regla 110 para ilustrar su utilidad, como son: la producción de gliders, la producción de grandes triángulos, la reproducción del sistema tag cíclico y la construcción de operaciones lógicas, todos ellos basados en choques entre dos o más gliders.

Finalmente la tesis está organizada de la siguiente manera:

- Capítulo 1. Se dan los antecedentes históricos de la teoría de AC y la Regla 110.

[1]Disponible desde `http://uncomp.uwe.ac.uk/genaro/OSXCASystems.html`

En el caso de la Regla 110 se discuten los resultados obtenidos por otros autores y señalamos nuestras aportaciones. Al final de este capítulo se discuten problemas fundamentales en teoría de AC: complejidad, auto-reproducción y constructor universal. Los conceptos originalmente planteados por John von Neumann, por sí mismos son temas muy extensos y difíciles de analizar. Sin embargo, una introducción muy general de ellos nos permite tener un panorama respecto de la dirección en la que el estudio de la Regla 110 debe dirigirse.

- Capítulo 2. Se presentan todos los gliders hasta ahora conocidos en la Regla 110 y proponemos una clasificación más general. Se discuten las características que existen en el espacio de evoluciones de la Regla 110 a través de mosaicos y los diagramas de de Bruijn, donde éstos originan la representación por fases para cada uno de los gliders. Se explica detalladamente cómo son inferidas las fases, como son obtenidas las expresiones regulares y cómo son clasificadas para cada uno de los gliders. Al final del capítulo se presenta el procedimiento para construir condiciones iniciales utilizando las fases, ilustrando algunas construcciones para ejemplificar su representación y significado.

- Capítulo 3. Se aplica el procedimiento en cuatro problemas para ilustrar su utilidad y funcionamiento, construyendo condiciones iniciales en la Regla 110. El primer problema es de auto-organización a través de los elementos que pueden ser construidos en la Regla 110, éstos son la producción de todos los gliders. El segundo problema es la construcción de grandes triángulos. El tercer problema es la reproducción de los componentes del sistema tag cíclico para simular su correcto funcionamiento en el espacio de evoluciones con la Regla 110. El cuarto y último problema es la implementación de operaciones lógicas en la Regla 110.

- Capítulo 4. Se dan las conclusiones, limitaciones y alcances del procedimiento para el estudio de la Regla 110 y el trabajo que se pretende realizar después.

- Apéndice A. Se describe el funcionamiento del programa de computación OSXL-CAU21 mostrando cada una de sus características de manera general, resaltando lo más importante.

- Apéndice B. Se presenta el conjunto finito de expresiones regulares para todos los gliders (sin extensiones) hasta ahora conocidos de la Regla 110.

Capítulo 1

Preliminares

En este capítulo se presentan los antecedentes históricos en teoría de autómata celular (AC) y de la Regla 110. En el caso de la Regla 110, se comparan las contribuciones de otros autores y nuestras aportaciones. Se presenta la notación básica en teoría de AC en una dimensión (1D) y los conceptos que dan origen a nuestro interés por analizar la Regla 110. Los conceptos de complejidad y problemas fundamentales en teoría de AC son discutidos en un contexto general, con la intención de ofrecer un panorama general del interés por estudiar la Regla 110 y problemas aún más complicados por resolver.

1.1 Marco histórico

La teoría de AC puede dividirse en tres etapas cada una con contribuciones importantes: la primera con su precursor John von Neumann, la segunda con John Horton Conway y la tercera con Stephen Wolfram.

La primera etapa comienza a mitad de los años 40's con su precursor von Neumann, quien desarrolla un sistema con esencialmente dos características: soportar comportamiento complejo y capacidad de auto-reproducción. En ese tiempo von Neumann no contaba con la tecnología necesaria para llevar a cabo una implementación de su modelo, pero su amigo Stanislaw M. Ullam le propuso llevar su idea a un modelo matemático manejando células en un espacio de dos dimensiones y es así como surge la "Teoría de Autómata Celular" [46]. El modelo de von Neumann es un AC que evoluciona en dos dimensiones con veintinueve elementos en su conjunto de estados y la función de transición es definida por la vecindad de von Neumann (vecinos ortogonales).

La segunda etapa comienza a finales de los años 60's con Conway presentando un AC en dos dimensiones capaz de soportar comportamiento complejo, mejor conocido como "El

Juego de la Vida" [13]. Aquí, la diferencia más importante con respecto al AC de von Neumann es que el conjunto de estados es reducido a dos elementos y la función de transición evoluciona con la vecindad de Moore [37]. Otro resultado importante en el área se da en 1982, cuando Conway y otros [6] demuestran que dicho AC es universal, construyendo una máquina de registros a través de compuertas lógicas.

La tercera etapa inicia a mitad de los 80's con Wolfram, quien realiza el primer estudio de AC en 1D y establece una notación para representar AC en 1D. Un resultado importante es la clasificación de AC a través de su comportamiento en el espacio de evoluciones, donde dicha clasificación es proyectada a AC de mayor dimensión [51]. Las clases son: Clase I, que representa comportamiento estable, Clase II, que representa comportamiento periódico, Clase III, que representa comportamiento caótico y Clase IV, que representa comportamiento complejo. A partir de dicha clasificación Wolfram conjetura que todo AC Clase IV es capaz de soportar computación universal [49].

Finalmente, en enero de 1998 Matthew Cook resuelve la conjetura de Wolfram al demostrar que la Regla 110 es universal, como se puede ver en [10, 53, 21].

1.2 Notación básica en autómata celular

Un autómata celular (AC) es un sistema dinámico discreto[1] que evoluciona en un arreglo regular [48]. Nuestro interés particular es en el caso 1D.

El espacio de evoluciones en 1D es una sucesión de elementos que determinan un arreglo lineal, cada x_i representa un estado en la i-ésima posición dentro del arreglo. Cada uno de los elementos del arreglo es llamado *célula*, estas células toman elementos del conjunto finito de estados Σ (nuestro alfabeto), este conjunto representa el número de estados que puede manejar el autómata celular en estudio por lo que $x_i \in \Sigma$. De esta manera, una sucesión de células es llamada una *configuración*.

Wolfram representa a los AC de 1D con dos parámetros (k, r) [51], donde k representa el número de estados del conjunto Σ, es decir, su cardinalidad $|\Sigma| = k$ 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 x_i, por lo tanto, los vecinos más la célula central forman una *vecindad*: $x_{i-r}, \ldots, x_i, \ldots, x_{i+r}$.

Con los dos parámetros (k, r) se puede determinar el número de células que conforman

[1] Un *sistema dinámico discreto* consiste de un conjunto no vacio X y una transformación $f : X \to X$. Para un $n \in \mathbb{Z}^+$ la n-ésima iteración de f es la n-ésima composición $f^n = f \circ \cdots \circ f$; definimos f^0 como la identidad. Si f es invertible, entonces, $f^{-n} : f^{-1} \circ \cdots \circ f^{-1}$ (n veces). De esta manera, $f^{n+m} = f^n \circ f^m$, donde las iteraciones forman un grupo si f es invertible y un semigrupo en otro caso [7].

los vecinos y vecindades, $2r$ es el número de vecinos con respecto a una célula central y $2r+1$ es el número de células que forman una vecindad.

Una configuración finita de células es un arreglo lineal acotado en ambos extremos. 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 conserva la simetría en el espacio de evoluciones del AC. 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.

La *función local* φ es una función total que indica la transformación específica de cada vecindad de longitud $2r+1$ a un elemento de Σ. El número de vecindades diferentes está determinado por el conjunto Σ^{2r+1}. La función de transición es la parte importante en los AC. Una *regla de evolución* es la función de transición aplicada en cada una de las vecindades con respecto al valor que tenga k y r, cada vecindad tendrá una correspondencia con un elemento del conjunto de estados 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 determina la transformación local de cada una de las vecindades a un elemento de Σ.

Por lo tanto, la transformación local está definida como la función:

$$\varphi : \Sigma^{2r+1} \to \Sigma. \tag{1.2.1}$$

La transformación local da origen a una transformación global, donde la correspondencia es entre configuraciones. Sea Φ una transformación global y se representa como:

$$\Phi_{\varphi} : \Sigma^{\mathbb{Z}} \to \Sigma^{\mathbb{Z}}. \tag{1.2.2}$$

Un estado x_i de un AC es llamado un *estado estable*, si cumple la siguiente condición:

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

Una configuración con todas las células en un estado estable x_i es llamado una *configuración homogénea* de x_i.

Un estado estable x_i en un AC es identificado y llamado el estado quieto del AC. La configuración homogénea de x_i es llamada la configuración quieta.

Sea c_0 y c_1 dos configuraciones de un AC. Si $\Phi^i_\varphi(c_0) = c_1$ para algún $i \geq 0$, entonces decimos que c_0 evoluciona a c_1. Para una configuración c_0 la secuencia infinita:

$$c_0, \Phi_\varphi(c_0) = c_1, \Phi^2_\varphi(c_1) = c_2, \ldots, \Phi^i_\varphi(c_n) = c_{n+1}, \ldots$$

es llamada la *secuencia de evolución* de c_0 (también conocido como el *espacio de evoluciones*). Si la secuencia llega a ser periódica entonces decimos que c tiene una evolución periódica.

Dados los conceptos anteriores podemos formalmente representar un autómata celular a través de una 4-tupla $\{\Sigma, r, \varphi, c_0\}$, donde Σ es el conjunto finito de estados, r es el radio de vecindad, φ la función local y c_0 el estado inicial del sistema. Evolucionando en una d-dimensión sobre un espacio regular.

En [49] Wolfram presenta su clasificación en AC con cuatro clases. Usualmente, se convierte la función local codificada en binario a notación decimal para representar cada regla de evolución. A continuación presentamos las clases de Wolfram en 1D definidas por Karel Culik II en [11].

Definición 1.2.1. Un AC es de clase I si existe un estado estable x_i tal que todas las configuraciones finitas evolucionan a la *configuración homogénea*.

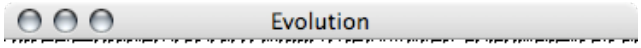

Figura 1.1: AC clase I – regla de evolución 8.

Definición 1.2.2. Un AC es de clase II si existe un estado estable x_i y cualquier configuración finita tiene una evolución que llega a ser periódica.

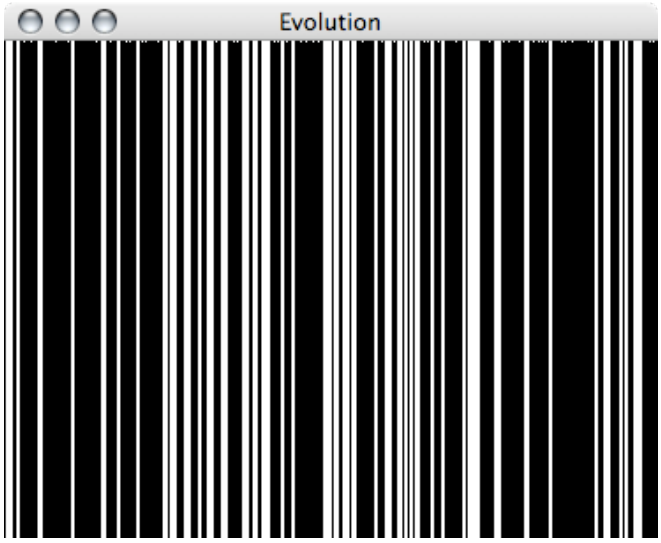

Figura 1.2: AC clase II – regla de evolución 236.

Definición 1.2.3. Un AC es de clase III si existe un estado estable x_i y para alguna pareja de configuraciones finitas c_1 y c_2, es decidible si c_1 evoluciona a c_2.

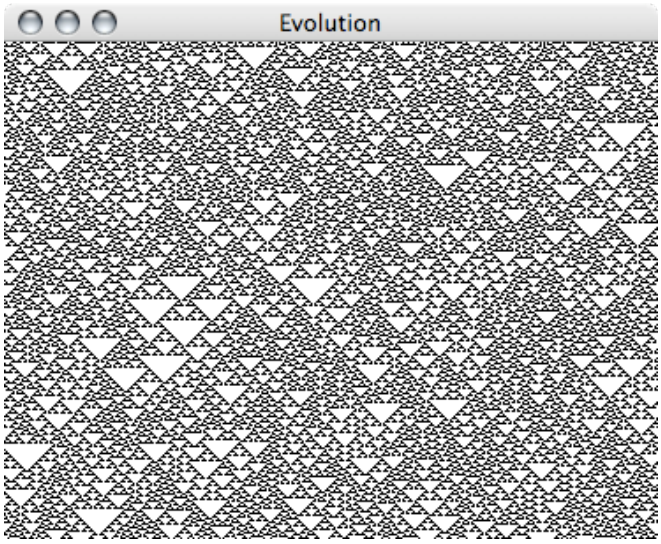

Figura 1.3: AC clase III – regla de evolución 22.

Definición 1.2.4. Un AC es de clase IV si incluye todo AC, es decir, todas las demás clases.

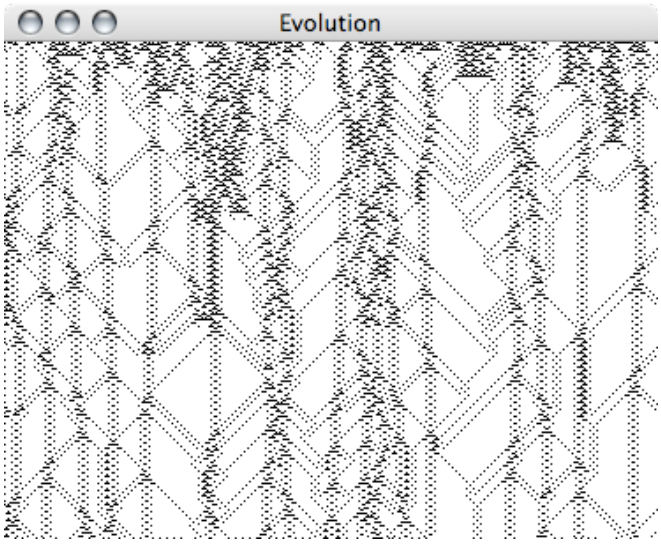

Figura 1.4: AC clase IV – regla de evolución 54.

Tratando de representar con ejemplos las definiciones anteriores recurrimos a reglas de evolución de una dimensión de orden (2,1).

El primer caso dice que cualquier configuración finita debe evolucionar al estado estable. Podemos ver que en la Figura 1.1 para la regla de evolución 8 se cumple que para cualquier segmento de la configuración inicial, éste es evolucionando al estado estable.

El segundo caso dice que cualquier configuración finita debe llegar a ser periódica. Podemos ver que en la Figura 1.2 para la regla de evolución 236 se cumple que para cualquier segmento de la configuración inicial, éste es evolucionando en bloques periódicos. En estas dos clases estudiadas el tiempo de evolución debe ser lineal.

El tercer caso dice que dado una pareja de configuraciones es decidible si la primera configuración puede evolucionar a la segunda configuración. Podemos ver que en la Figura 1.3 para la regla de evolución 22 se cumple que estas configuraciones son determinadas.

El cuarto caso dice que todo AC es clase IV si incluye todo AC. Podemos ver que en la Figura 1.4 para la regla de evolución 54 el espacio de evoluciones contiene elementos de todas las clases.

En particular cada clase es contenida en la siguiente. De esta manera:

$$\text{Clase I} \subset \text{Clase II} \subset \text{Clase III} \subset \text{Clase IV}.$$

Una nota importante dice que para distinguir un AC Clase III de un AC Clase IV, es demostrar que dicho AC sea computacionalmente universal, entonces, es Clase IV. Un resultado más importante es que es indecidible determinar que clase es cada AC [11].

1.3 Antecedentes en la Regla 110

El estudio sobre la Regla 110 inicia con las investigaciones de Wolfram en AC de 1D [51]. La Regla 110 es un AC que evoluciona en 1D de orden $(2, 1)$, es decir, dos estados en su conjunto de estados y un vecino en cada extremo con respecto a una célula central. La Regla 110 pertenece a la Clase IV de acuerdo a la clasificación de Wolfram [51], porque tiene comportamientos complejos en su espacio de evoluciones, es decir, regiones con comportamiento uniforme, periódico y caótico.

El segundo estudio dedicado a la Regla 110 es el de Wentian Li y Mats G. Nordahl en 1992 [29], donde se desarrolla un estudio probabilístico para explicar estabilidad global y se ilustran algunos de los comportamientos más frecuentes en el espacio de evoluciones de la Regla 110.

En enero de 1998 Cook [10] demuestra que la Regla 110 es universal. Al simular un sistema tag cíclico basado en choques entre bloques de gliders [53]. En enero de 1999 Cook determina una clasificación para todos los gliders encontrados hasta ahora en el espacio de evoluciones de la Regla 110 [9, 10, 23, 53, 22].

Finalmente, Harold V. McIntosh en 1999 desarrolla una investigación fundamentada en que la Regla 110 es un problema de cubrir el espacio de evoluciones con triángulos de diferentes tamaños [32]. Ilustrando que todos los gliders y en general todo el espacio de evoluciones de la Regla 110 es caracterizado a través de estos mosaicos. A continuación discutimos de manera general cada resultado obtenido hasta ahora en el estudio de la Regla 110.

Resultados de Li y Nordahl

Li y Nordahl aplican la teoría del campo promedio para obtener una descripción estadística de las evoluciones en la Regla 110 [29]. El estudio es realizado en el espacio de evoluciones de manera global y no local, determinando aproximadamente el comportamiento del espacio de evoluciones estadísticamente.

Resultados de Wolfram y Lind

Wolfram se interesa particularmente en los comportamientos de la Regla 110 al momento de clasificar todas las reglas de evolución elementales [51] (de orden (2,1)) y en 1985 establece la conjetura de que la Regla 110 puede llegar a hacer computación universal, dada la amplia variedad de gliders y choques que existen en su espacio de evoluciones.

Doug Lind establece la primera clasificación de gliders en la Regla 110. Su clasificación puede ser consultada en [51] (Apéndice - Tabla 15, pág. 577).[2] Los gliders clasificados por Lind son los que aparecen con más frecuencia desde condiciones iniciales aleatorias en el espacio de evoluciones de la Regla 110, identificando el fondo periódico y caracterizando 12 gliders a través de secuencias.

Resultados de Cook

Cook demuestra que la Regla 110 es universal en [10, 53]. La demostración se realiza implementando el funcionamiento de un sistema tag cíclico a través de choques entre gliders, donde diferentes bloques de gliders representan los operadores y datos.

En 1999 Cook determina una clasificación para todos los gliders conocidos hasta ahora en la Regla 110 como se puede ver en [9, 10, 53, 22]. En la clasificación, Cook encuentra nuevas y complicadas extensiones de gliders (dejando abierta la posibilidad de encontrar más gliders), discutiendo similitudes entre la Regla 110 y el Juego de la Vida de Conway.

En diciembre del 2002, Cook encuentra un camino para construir grandes triángulos en la Regla 110. La construcción es realizada a través de secuencias que determinan la condición inicial para producir el triángulo. De esta manera, Cook disminuye el intervalo establecido por McIntosh en la búsqueda de grandes triángulos.

Resultados de McIntosh

McIntosh desarrolla una investigación en la Regla 110 planteando el problema de cubrir el espacio de evoluciones con triángulos de diferentes tamaños [32].

El análisis realizado por McIntosh cuenta con varias herramientas para su estudio: diagramas de de Bruijn que calculan expresiones regulares a través de secuencias periódicas, diagrama de subconjuntos (conjunto potencia) que calculan configuraciones que no pueden ser producidos por el AC, diagrama de parejas (producto cartesiano) que calculan secuencias

[2] El apéndice es disponible desde: `http://www.stephenwolfram.com/publications/articles/ca/86-caappendix/16/text.html`

que pueden ser construidos por más de un camino, es decir, que tienen múltiples ancestros, diagramas de ciclos que calculan secuencias periódicas a través de ciclos de cierta longitud, análisis matricial y otras herramientas como la teoría del campo promedio y probabilidad por bloques proyectadas como curvas en el plano cartesiano, contornos o superficies.

El problema de cubrir el espacio de evoluciones con triángulos no es mencionado por Cook o Wolfram. Este enfoque permite describir detalladamente los gliders, los diferentes tipos de fondos periódicos y todo el espacio de evoluciones [15].

Un problema como consecuencia de enumerar todos los mosaicos que la Regla 110 puede construir, es conocer, ¿cuál es el triángulo más grande que puede existir? y si éste puede ser producto de alguna composición o existe otro camino para construirlo. McIntosh en [33] establece una relación entre ellos produciendo diferentes mosaicos a través de choques y determinando un límite para el máximo tamaño de un mosaico que la Regla 110 puede construir, establecido por Juan C. S. T. Mora en Agosto del 2001 [33].

Nuestros resultados

Iniciamos nuestra investigación en la Regla 110 con los resultados de Cook y McIntosh: la clasificación de gliders y la descripción del espacio de evoluciones por triángulos y diagramas de de Bruijn respectivamente.

El primer problema fue encontrar una forma de controlar y codificar el espacio de evoluciones de la Regla 110. La manera como se realiza es por medio de las propiedades de los mosaicos que representan el fondo periódico y éstas son reflejadas en todas las estructuras que existen en la Regla 110. Nuestro interés por encontrar una manera de codificar el espacio de evoluciones es para construir condiciones iniciales y entonces producir cualquier comportamiento en el espacio de la Regla 110, donde las producciones se basan en choques entre gliders.

Nuestra contribución en el estudio de la Regla 110 es el lenguaje regular basado en gliders establecido en el procedimiento para codificar condiciones iniciales a través de secuencias periódicas que llamamos FASES f_i_1, donde las fases determinan una medida horizontal aplicada al espacio de evoluciones y determinan el conjunto finito de expresiones regulares para todas las estructuras periódicas de la Regla 110. Posteriormente a cada expresión regular le asignamos una representación por fases y de esta manera proponemos una codificación para construir condiciones iniciales que operan bajo las reglas de las expresiones regulares en el espacio de evoluciones de la Regla 110. Por lo tanto, dos resultados importantes son obtenidos: un lenguaje regular y la máquina de estados finitos (para los gliders) en la Regla 110 (debemos mencionar que éstos no han sido determinados en algún otro AC con comportamiento

complejo).

Una vez obtenido el procedimiento, se procedió a implementarlo en un sistema de computación para el estudio de choques entre gliders de la Regla 110 llamado OSXLCAU21. Nuestro sistema se ha desarrollado únicamente para la construcción de condiciones iniciales, buscando ofrecer una herramienta fácil de utilizar para construir componentes que después pueden ser llevados a enormes simulaciones en el espacio de evoluciones de la Regla 110.

Finalmente, aplicamos el procedimiento a cuatro problemas importantes en el estudio de la Regla 110. El primer problema resuelto consiste en construir cada uno de los elementos (gliders) que existen en el espacio de evoluciones de la Regla 110 y que era desconocido hasta ahora, principalmente en los gliders más complicados [22, 23] (propiedad de cerradura). El segundo problema parcialmente resuelto consiste en la construcción de grandes triángulos a través de choques entre gliders. Nuestra contribución es en cuatro mosaicos, que son hasta ahora, los más grandes producidos por choques [33, 24]. El tercer problema resuelto consiste en la reconstrucción de los componentes del sistema tag cíclico. Con ésto se ofrece otra codificación de dicha máquina a través de fases y se verificó su funcionamiento en el espacio de evoluciones de la Regla 110 [21]. El cuarto problema resuelto consiste en la implementación de operaciones lógicas a través de choques entre grupos de gliders [24].

1.4 Conceptos iniciales

A continuación discutimos brevemente las principales ideas que von Neumann conceptualizó para la teoría de AC y conceptos previos que nos serán útiles en las secciones posteriores.

La finalidad de este marco introductorio es contar con un panorama general de nuestro interés por estudiar la Regla 110.

1.4.1 Mosaicos en la Regla 110

Estudiamos el concepto de mosaico porque es una de las principales características que determina el espacio de evoluciones de la Regla 110 y es dominado por triángulos de diferentes tamaños. Ellos serán de gran utilidad porque podremos indentificar y clasificar propiedades importantes.

Un *plano de mosaicos* $\mathcal{T}$ es una familia contable de conjuntos cerrados[3] $\mathcal{T} = \{T_0, T_1, \ldots\}$

[3]Un conjunto es *cerrado bajo una operación* si al aplicar la operación a los miembros del conjunto produce otro miembro del mismo conjunto. Por ejemplo el conjunto de los enteros positivos $\mathbb{Z}^+$ es cerrado bajo la operación suma pero no para la operación sustración: $1 - 2 \ni \mathbb{Z}^+$.

que cubren el plano sin intervalos o intersecciones [15]. Explícitamente, la unión de los conjuntos:

$$\mathcal{T} = \bigcup_{i=0}^{n} T_i \tag{1.4.1}$$

(conocidos como mosaicos de $\mathcal{T}$) están en todo el plano. Por "el plano" nos referimos al producto cartesiano $\mathbb{Z} \times \mathbb{Z}$.

La Regla 110 cubre el espacio de evoluciones a través de diferentes conjuntos de triángulos T_n, donde n es un entero mayor igual que cero y representa el tamaño del triángulo contando las células en estado cero en alguno de sus catetos. Los mosaicos se dividen en dos conjuntos: α y β $\forall$ $n \geq 2$ [32] (cada conjunto α o β determina su propia familia contable de mosaicos donde $|\{T_n^{\alpha}\}| = |\{T_n^{\beta}\}|$, como se muestra en la figura 1.5). Por ejemplo, diferentes mosaicos α y β son detallados en la construcción del glider H y glider gun (figura 2.3).

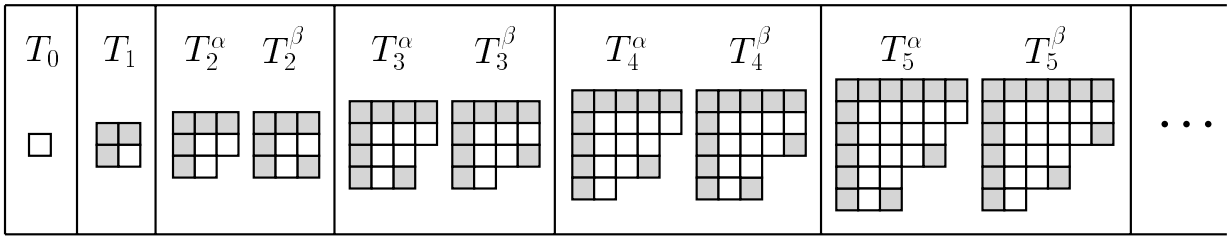

Figura 1.5: Dos conjuntos de mosaicos en la Regla 110: α y β.

Podemos definir un mosaico T_0 y representarlo con el estado 0. Por ejemplo, cuando la configuración inicial es cubierta por las expresiones: 0*, 1* y (10)*, el espacio de evoluciones es establecido por una evolución homogénea $x_i = 0$ (o el mosaico T_0). Sin embargo, el comportamiento no es el mismo para los mosaicos $T_1, T_2^{\alpha}, T_2^{\beta}, T_3^{\alpha}, T_3^{\beta}, \ldots, T_n^{\alpha}, T_n^{\beta}, \ldots$. De esta manera, el espacio de evoluciones puede ser cubierto totalmente por un mosaico T_n cuando toma valores entre $0 \leq n \leq 4$. Entonces, para $n \geq 5$ el espacio de evoluciones es cubierto por al menos dos diferentes mosaicos T_n.

Sean T_i y $T_j \in \mathcal{T}$ con $i \neq j$, entonces ambos mosaicos no pueden operar en el plano bajo la función de la Regla 110 si cubren el espacio parcialmente (existen huecos) o intersectan en sus células.

Una pregunta que surge es conocer el mosaico más grande que la Regla 110 puede construir en su espacio de evoluciones. En la actualidad, el límite es establecido con el mosaico T_{45} [33]; por lo tanto, la Regla 110 no puede construir un mosaico mayor. En la actualidad existe un camino para producir mosaicos T_n donde $0 \leq n \leq 33$. Los mosaicos T_{43}, T_{44} y T_{45} fueron calculados a través de una búsqueda especializada determinando los ancestros para

cada mosaico [33]. Finalmente, otro problema abierto es determinar una construcción para los mosaicos que se encuentran en el intervalo $34 \leq n \leq 42$.

De esta manera, la familia de mosaicos $\{T_n^{\alpha}\}$ y $\{T_n^{\beta}\}$ permiten describir detalladamente el espacio de evoluciones de la Regla 110. El segundo punto importante es que los mosaicos establecen propiedades a través de márgenes periódicos en las estructuras periódicas (gliders y éter). Su interpretación es muy importante para derivar las fases, inclusive, en estructuras no periódicas.

1.4.2 Lenguajes y autómatas

Un *alfabeto* es un conjunto finito de símbolos. Por ejemplo, $\Sigma = \{0, 1\}$ es nuestro alfabeto. Un *lenguaje* (*formal*) es un conjunto de cadenas de símbolos de algún alfabeto [18].

La *longitud* de una cadena w denotada $|w|$, es el número de símbolos en la cadena (su cardinalidad). La cadena vacia ϵ es la cadena que no tiene ningún símbolo, de esta manera, $|\epsilon| = 0$.

El conjunto vacio ϕ y el conjunto con la cadena vacia $\{\epsilon\}$ son lenguajes. Sin embargo, ellos son distintos; el último tiene un miembro y el primero no. El conjunto de *palíndromes* (cadenas que se leen igual hacia adelante y hacia atrás) sobre el alfabeto $\{0, 1\}$ es un lenguaje infinito [18]. Otro lenguaje es el conjunto de todas las cadenas sobre un alfabeto específico Σ, por lo tanto, representamos este lenguaje como Σ^*. Por ejemplo, si $\Sigma = \{a\}$ entonces $\Sigma^* = \{\epsilon, a, aa, aaa, \ldots\}$.

En el estudio de la teoría de lenguajes formales surgen varios problemas interesantes. Uno de ellos es determinar que tipo de lenguaje se está derivando y como podemos garantizar que es ése lenguaje y no otro. Esta jerarquía es conocida y establecida como la clasificación de Chomsky [18].

Nuestro interés radica en el estudio de lenguajes determinados por conjuntos regulares. Porque el conjunto de expresiones que determinamos por cada glider de la Regla 110, puede ser asociado como una expresión regular en particular. De esta manera, también es necesario tener en cuenta algunos conceptos conocidos en el estudio de máquinas de estados finitos para complementar nuestro estudio.

El autómata finito es un modelo matemático con un sistema de entradas y salidas discretas. El sistema puede estar en algún estado del conjunto finito de estados. El estado del sistema contiene la información de las entradas recibidas que son necesarias para determinar el comportamiento del sistema con respecto a las entradas subsecuentes.

Un autómata finito consiste de un conjunto finito de estados y un conjunto de transiciones de estado a estado que ocurre sobre los símbolos seleccionados desde algún alfabeto. Para

cada símbolo existe una transición a cada estado (puede regresar al mismo estado). Un estado, usualmente representado como q_0 es el estado inicial en donde el autómata inicia. Algunos estados son designados como estados finales o estados de aceptación.

Una gráfica dirigida llamado un *diagrama de transición* es asociado con un autómata finito como sigue. Los vértices de la gráfica corresponden a los estados del autómata finito. Si existe una transición del estado q al estado p para una entrada a entonces existe una arista etiquetada con a desde el estado q al estado p en el diagrama de trancisiones. El autómata finito acepta una cadena w si la secuencia de transiciones corresponde a los símbolos de w conducido del estado inicial a un estado de aceptación (o final).

Definición 1.4.1. Un *autómata finito* (AF) es una 5-tupla $< Q, \Sigma, \delta, q_0, F >$ donde Q es un conjunto finito de estados, Σ es un alfabeto finito de símbolos, $q_0 \in Q$ es el estado inicial, $F \subseteq Q$ es el conjunto de estados terminales y δ es la función de transición transformando $Q \times \Sigma \rightarrow Q$. De manera que, $\delta(q, a)$ determina el nuevo estado para cada estado q y símbolo a evaluado en la función.

Para describir el comportamiento del AF sobre un cadena debemos extender la función de transición δ para aplicarla de un estado a una cadena, en lugar, de un estado a un símbolo. Definimos una función $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$. La idea es que $\hat{\delta}(q, w)$ debe ser el estado del AF después de leer la cadena w habiendo iniciado del estado q. De esta manera, $\hat{\delta}(q, w)$ es el único estado p tal que existe un camino en el diagrama de transiciones del AF desde q a p etiquetado con la palabra w. Formalmente definimos:

(a) $\hat{\delta}(q, \epsilon) = \{q\}$, y

(b) para todas las cadenas w y un símbolo a: $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$.

Es conveniente utilizar δ en vez de $\hat{\delta}$ dado que los argümentos están definidos para ambas funciones.

Una cadena w es aceptada por un AF $M =< Q, \Sigma, \delta, q_0, F >$ si $\delta(q_0, w) = p$ para algún $p \in F$. Un *lenguaje aceptado* por M, representado como $L(M)$, es el conjunto $\{w | \delta(q_0, w) \in F\}$. El tipo de lenguajes aceptados por un AF es de nuestro interés porque éllos complementan nuestro análisis establecido con las expresiones regulares.

Históricamente ésta importante relación es establecida por S. C. Kleene. Demostrando que la teoría de expresiones regulares puede ser expresada con AF y la teoría de AF puede ser expresada con expresiones regulares [35].

Entonces podemos ver que un lenguaje es un *conjunto regular* si el conjunto es aceptado por algún AF. Los lenguajes aceptados por un AF son descritos por expresiones conocidas como *expresiones regulares*. Particularmente, los lenguajes aceptados por autómatas finitos son precisamente la clase de lenguajes descritos por expresiones regulares.

Sea Σ un conjunto finito de símbolos y L, L_1 y L_2 conjuntos de cadenas de Σ^*. La concatenación de L_1 y L_2 representado como L_1L_2 es el conjunto $\{xy|x \in L_1 \wedge y \in L_2\}$. Esto es, las cadenas en L_1L_2 son formadas seleccionando una cadena L_1 seguida por una cadena L_2 en todas las posibles combinaciones. Se define $L^0 = \{\epsilon\}$ y $L^i = LL^{i-1}$ para $i \geq 1$. La *cerradura de Kleene* (o cerradura) de L representado como L^* es el conjunto:

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

y la cerradura positiva de L representado como L^+ es el conjunto:

$$L^+ = \bigcup_{i=1}^{\infty} L^i.$$

Definición 1.4.2. Las *expresiones regulares* sobre un alfabeto y los conjuntos que ellos representan, son definidos recursivamente como:

1. ϕ es una expresión regular y representa el conjunto vacio.

2. ϵ es una expresión regular y representa el conjunto $\{\epsilon\}$.

3. Para cada $a \in \Sigma$, a es una expresión regular y representa el conjunto $\{a\}$.

4. Si r y s son expresiones regulares representando los lenguajes R y S respectivamente, entonces $(r + s)$, (rs), y (r^*) son expresiones regulares que representan los conjuntos $R \cup S$, RS y R^* respectivamente.

Cuando es necesario distingir entre una expresión regular r y el lenguaje determinado por r utilizamos $L(r)$.

La teoría de lenguajes formales nos da un esquema para estudiar conjuntos de cadenas desde un alfabeto finito. Los lenguajes pueden ser vistos como entradas de algunas clases de máquinas o como el producto final de un sistema de sustituciones.

El modelo básico de estas máquinas para estos lenguajes y necesaria para toda computación, es la máquina de Turing. De esta manera, las máquinas que reconocen cada una de las familias de los lenguajes son descritas como una máquina de Turing con restricciones.

lenguaje	estructura
recursivamente enumerables	máquina de Turing
sensitivos de contexto	autómata de límite lineal
libres de contexto	autómata de pila
regular	autómata finito

Tabla 1.1: Clases de lenguajes.

La importancia de asociar una máquina o sistema para determinar cada tipo de lenguaje nos lleva a una clasificación de lenguajes (tabla 1.1 [17]).

Nuestro interés se encuentra en los lenguajes que son determinados por conjuntos regulares. Porque aunque podemos derivar cada una de las palabras reconocidas a través de un AF o un diagrama de de Bruijn, solo necesitamos identificar aquellas cadenas que representan una estructura en la Regla 110 (gliders), para manipular el espacio de evoluciones con cada una de las entidades de las partículas.

Los conjuntos regulares pueden ser reconocidos por máquinas con una cantidad de memoria finita (máquina de estados finitos). Otro camino para representar cadenas en un lenguaje regular es por expresiones regulares. Por ejemplo, un lenguaje formado por todas las cadenas en el alfabeto $\Sigma = \{0, 1\}$ que no contienen dos unos consecutivos es regular y puede ser construido por la siguiente expresión regular: $(0 + 1)(00 + 01)^*$.[4]

El lenguaje regular basado en gliders L_{R110} propuesto y determinado para la Regla 110 es el resultado importante en nuestra investigación y éste no ha sido reportado por ningún otro autor hasta el momento. Además, la aplicación del subconjuto regular nos permite solucionar problemas que estaban sin resolver, a través de la construcción de condiciones iniciales codificadas por las fases. Finalmente, el lenguaje encontrado ofrece una herramienta poderosa para buscar una solución a problemas complicados que permanecen sin resolver en la Regla 110 (ver la subsección 1.4.5 de Regla 110 de este capítulo).[5]

1.4.3 Complejidad

La teoría de la complejidad clasifica problemas basandose en el grado de dificultad que hay para resolverlos. Un problema es clase P (tiempo polinomial) si el número de pasos necesarios para resolverlo es limitado por alguna potencia del tamaño del problema. Un problema es

[4]Para ver ejemplos completos y propiedades de los lenguajes formales, gramáticas, máquina de estados finitos y máquinas de Turing, consultar [4, 18, 35, 43, 45].

[5]Debemos señalar que el lenguaje regular L_{R110} no implica que la evolución de la Regla 110 sea regular en su conjunto límite [17], porque el lenguaje regular solo es conservado en la construcción de las condiciones iniciales.

clase NP si permite una solución en tiempo no polinomial. Particularmente, la clase de problemas P es un subconjunto de la clase de problemas NP.

La complejidad de un conjunto de instrucciones como el mínimo procedimiento efectivo requerido para reproducirlo es conocido como la *complejidad algorítmica* (también conocida como la complejidad de Kolmogorov).

1.4.4 Problemas fundamentales

Von Neumann determina dos características esenciales que deben soportar los AC: comportamiento complejo y auto-reproducción. Esto lleva a von Neumann a plantear dos preguntas fundamentales en la teoría de AC: ¿cómo puedo construir componentes confiables a partir de organismos no-confiables? y ¿qué tipo de organización lógica es suficiente para que un AC sea capaz de auto-reproducirse?. Ambas preguntas no son fáciles de responder y representan áreas muy extensas y difíciles de analizar. De esta manera, debemos tratar de sintetizar estos conceptos de la manera más simple posible.

Von Neumann pensó en construir una máquina capaz de solucionar problemas complejos y consideraba que una máquina con tal complejidad debía contener mecanismos de auto-control y auto-reparación. Un problema era establecer diferencias entre procesos y datos, considerando que éstos están en igualdad en el espacio celular, entonces, ideó la forma de construir una máquina capaz de auto-reproducirse.

El primer AC con capacidades de soportar auto-reproducción sería propuesto por von Neumann [46], evolucionando en un espacio Euclidiano. La estructura para llevar a cabo la auto-reproducción sería realizada en miles de células, donde cada una de estas células podía tener uno de veintinueve estados posibles. La regla de evolución es evaluada con la vecindad de von Neumann (vecinos ortogonales). Por lo tanto, tenemos un problema combinatorio y la regla de von Neumann crece exponencialmente, por esa razón en la actualidad se tiene implementada parcialmente como puede verse en [38].

Con su regla de evolución, von Neumann tuvo éxito en encontrar estructuras de células orientándose en ellas mismas para generar nuevas estructuras o idénticas y aunque este resultado es una forma de vida muy primitiva, es muy interesante porque se espera que una máquina puede construir únicamente un objeto de menor complejidad que él mismo. Un AC con capacidades de auto-reproducción tiene la capacidad de construir una máquina capaz de crear nuevas máquinas de complejidad y capacidades idénticas o superiores a sí mismo.

La regla de von Neumann además tiene la capacidad de implementar una máquina universal de Turing. Esto significa que existe una configuración inicial del AC que produce la solución de algún procedimiento efectivo. La propiedad es de interés teórico y no tanto de

interés práctico. Por lo tanto, la propiedad de computación universal significa que alguna lógica (compuertas lógicas) puede ser simulada por una regla de un AC.

Demostrar la existencia de una máquina con capacidades de auto-reproducción, en particular indica la posibilidad de que máquinas arbitrariamente complejas son capaces de soportar auto-reproducción y ésto es lo que distinge el resultado de von Neumann del que obtiene Christopher G. Langton en [26]. Aunque la operación de la máquina de Langton puede ser descompuesta en dos actividades: copiar y decodificar, no incorpora algo como un AC constructivo general. De esta manera, la capacidad de decodificación es menor; además, existe una sola descripción que puede ser procesada de manera efectiva.

Paralelamente, von Neumann plantea la idea de un constructor universal, en un sentido o significado que no era propiamente de computación. Para ello requería de un decodificador y una descripción para obtener el fenómeno de descendencia. Además, requería de una copia de la descripción, agregándolo como parte de la descendencia.

El concepto de constructor universal [46] es introducido por von Neumann, como un paso preliminar para dar entrada al problema de obtener crecimiento espontáneo y abierto en la complejidad del AC. El concepto lo formuló como una analogía al resultado de Turing de su máquina universal.

El constructor universal concurre en una pequeña parte del patrón para establecer la auto-reproducción. La unidad supervisora y las partes relacionadas con manipulación de memoria es mayor. Muchos de los patrones son dados por las unidades que codifican y decodifican la información transmitida o la recepción de otras partes del patrón. Von Neumann estimó el tamaño total de sus patrones para la auto-reproducción en 200,000 células [46] (este orden puede variar dependiendo el diseño).

Las investigaciones realizadas por von Neumann en la teoría de AC complejo se describen como el problema de obtener máquinas auto-reproductivas no triviales, donde lo no trivial es interpretado como un requisito que la máquina debe tener en un ambiente de una computación universal.

El resultado importante es que una máquina de Turing particular puede ser configurada de manera que pueda simular las operaciones (computaciones) de otra máquina y de esta manera realizar la misma computación de alguna máquina dada. Turing llama a esta máquina *universal* [35] y von Neumann se refiere a esta propiedad como "universalidad lógica."

El concepto de máquina universal no debe ser generalizado (no a algún AC en particular) a algún sistema a través de un sistema dado que soporte una noción de "autómata realizando computaciones."

El concepto esencial en la máquina universal de Turing es que sus operaciones son programadas por una lista de instrucciones. Se soporta un pequeño conjunto básico de instrucciones las cuales sirven para describir completamente el comportamiento computacional de una máquina de Turing arbitraria en términos de una secuencia finita de tales instrucciones. De esta manera, una máquina universal de Turing se construye para simular las computaciones de una máquina de Turing arbitraria, simplemente suministrando una descripción (codificación) apropiada de la máquina. De tal forma, no encontramos cosas tales como *constructores Turing* o *reproductores Turing*.

La idea básica de von Neumann es generalizar el análisis de Turing considerando máquinas abstractas que puedan operar o manipular cosas que pueden llegar a ser construidas por sí mismas. Von Neumann consideraba un número de sistemas distintos que no son equivalentes en algún sentido general y que no están completamente formalizados. Sin embargo, una constante a través de todo su trabajo fue el introducir alguna analogía al concepto de máquina lógica universal, pero relativo a alguna noción de "construcción" en vez de "computación." El nuevo concepto de von Neumann se refiere a un tipo particular de máquina que debe ser llamado *constructor universal*.

De esta manera, una máquina universal puede simular la computación de alguna máquina y un constructor universal debe ser capaz de construir alguna máquina. Consecuentemente, se debe notar que aquí no existe algún constructor Turing universal. En este nivel, la analogía entre estos dos desarrollos es separada y la palabra universal tiene sus respectivos dominios para la máquina y el constructor.

Después de los resultados obtenidos por von Neumann, varios investigadores se dieron a la tarea de reducir la complejidad del AC con dichas capacidades, como lo realizó Edward F. Codd en 1968 [8]. Codd encontró un AC con las mismas capacidades al propuesto por von Neumann pero manejando únicamente ocho elementos en su alfabeto.

La siguiente reducción se logra con el AC de Conway conocido como "El Juego de la Vida" a principios de 1970 [6]. Conway logra encontrar una regla de evolución que puede soportar comportamiento complejo y computación universal, al implementar una máquina de registros a través de compuertas lógicas [6]. La reducción se logra en un AC de dos dimensiones pero con únicamente dos elementos en su alfabeto y evolucionando con la vecindad de Moore (vecinos ortogonales y diagonales).

Con el avance de las computadoras, el Juego de la Vida se volvió muy popular, lo que despertó el interés de varios investigadores en el estudio de AC y direccionando varias líneas en simulación de procesos complejos, tal como: reacciones químicas, formación de patrones, vida artificial, crecimiento de epidemias, comportamiento colectivo no-trivial, fenómenos reversibles, computación no-convencional y otros más [1, 2, 53, 14].

Varios problemas fueron surgiendo a partir de ese momento. Uno de ellos era demostrar la existencia de un patrón que tenga un crecimiento constante y que nunca se detenga en el espacio de evoluciones, lo que se conoce como crecimiento ilimitado [40]. Muchos intentos fueron realizados para encontrar tal patrón. Uno de ellos fue el objeto *arcon*[6] formado de siete elementos, determinando un crecimiento constante en un largo tiempo, pero se pudo comprobrar que este crecimiento no era para siempre porque llega a ser estable en 5,206 generaciones (sin embargo no deja ser interesante su evolución). Las técnicas para encontrar tal patrón fueron varias. Finalmente, un grupo de estudiantes del MIT encabezado por R. William Gosper, Robert April, Michael Beeler, Richard Howell, Rich Schroeppel y Michael Specier resolvieron este problema al encontrar un generador de estructuras periódicas, conocido en la literatura de AC como "glider gun" [40].

1.4.5 Regla 110

Dados los antecedentes anteriores nuestro interés por estudiar la Regla 110 es encontrar los elementos necesarios donde un AC más simple puede tener las mismas capacidades al AC de von Neumann. El AC de von Neumann evoluciona en dos dimensiones con veintinueve estados y cinco células en la función de transición, por tanto, tiene 29^5 (20,511,149) vecindades para construir una regla de evolución. El AC de Conway evoluciona en dos dimensiones con dos estados y ocho células en la función de transición, por tanto, tiene 2^9 (512) vecindades para construir una regla de evolución. La Regla 110 evoluciona en una dimensión con dos estados y tres células en la función de transición, por tanto, tiene 2^3 (8) vecindades para construir una regla de evolución en ese orden. La complejidad en la función cambia representativamente, eso nos haría pensar que debemos tener menor complejidad en el espacio de evoluciones. Sin embargo, la complejidad exponencial espacial es conservada en cada sistema.

De esta manera, la Regla 110 se convierte en un candidato adecuado para soportar comportamiento complejo en su espacio de evoluciones, originado por la amplia variedad de gliders que existen en su espacio de evoluciones. Por lo tanto, los problemas establecidos y encontrados en las reglas de von Neumann y Conway deben ser proyectados a la Regla 110, donde la solución de esos problemas deben ser resueltos gradualmente. Por ejemplo, ahora es conocido que la Regla 110 es universal [10] con un sistema equivalente Turing, pero no es demostrado si los elementos pueden determinar una máquina de Turing como tal, un resultado todavía mayor sería demostrar que la Regla 110 es intrínsicamente universal.

[6]Las palabras involucradas y un amplio número de ejemplos en estructuras del Juego de la Vida pueden ser consultados desde `http://pentadecathlon.com/index.shtml`

Finalmente, problemas aún más complicados es determinar si la Regla 110 puede soportar un constructor universal o auto-reproducción o ambos.

1.5 Conclusiones del Capítulo 1

Los antecedentes en la teoría de AC ayudan a comprender los antecedentes en la Regla 110 y nuestro interés por analizarla. Finalmente, el estudio de la Regla 110 no es únicamente motivado por su computabilidad, sino que también se busca encontrar cada uno de los elementos que son expresados en los AC de von Neumann y Conway.

Capítulo 2

Determinando fases en la Regla 110

A continuación se presenta la estructura y notación de la Regla 110. Los gliders y su clasificación son discutidos, presentando sus propiedades básicas. Posteriormente, se explica y desarrolla detalladamente la determinación de las fases en la Regla 110: en los gliders y estructuras no periódicas. Se finaliza con el procedimiento para controlar choques codificando condiciones iniciales, determinando el subconjunto finito de expresiones regulares que originan un lenguaje regular en Regla 110 y la máquina de estados finitos para todos los gliders.

2.1 Estructura de la Regla 110

La Regla 110 pertenece a la Clase IV de acuerdo a la clasificación de Wolfram [51], porque tiene regiones uniformes, regiones periódicas y regiones caóticas, todas ellas en el mismo espacio de evoluciones como podemos ver en la figura 2.1.

En la figura. 2.1 se puede apreciar el fondo periódico determinado por un mosaico de tamaño tres, donde los gliders pueden desplazarse en el tiempo sin ser alterados o destruidos. Cook llama a este fondo periódico "éter" [10]. En Regla 110 no tenemos un estado estable s, en su lugar tenemos una secuencia periódica que es representado como una palabra en nuestro lenguaje regular (ver Apéndice B).

La función de transición local que determina el comportamiento de la Regla 110 es:

$$\begin{array}{ll} \varphi(0,0,0) \to 0 & \varphi(1,0,0) \to 0 \\ \varphi(0,0,1) \to 1 & \varphi(1,0,1) \to 1 \\ \varphi(0,1,0) \to 1 & \varphi(1,1,0) \to 1 \\ \varphi(0,1,1) \to 1 & \varphi(1,1,1) \to 0 \end{array}$$

Tabla 2.1: Regla de evolución 110.

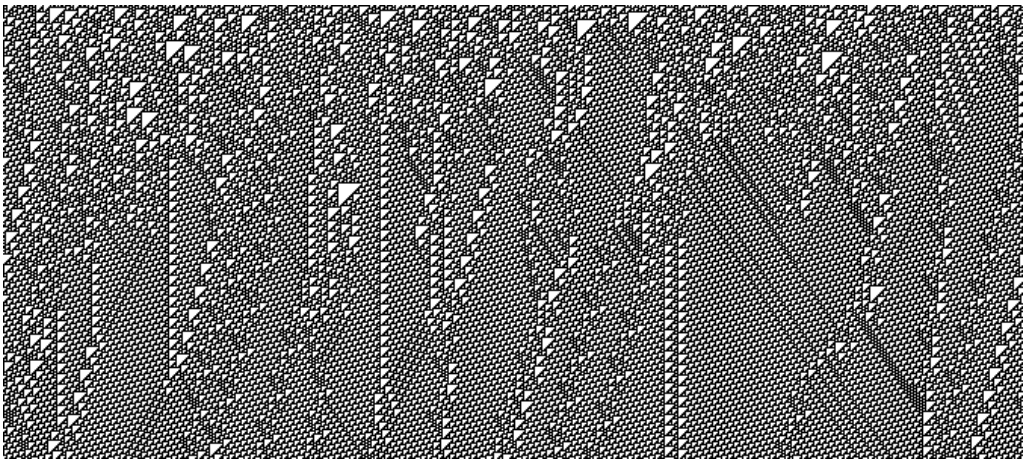

Figura 2.1: Evolución aleatoria en la Regla 110.

La regla de evolución es expresada en notación binaria 01110110 (su representación en decimal es el número 110). La manera en como es evaluado el espacio de evoluciones es sobre un arreglo lineal tomando una célula central y evaluamos el valor de la vecindad para determinar el nuevo elemento central en la siguiente generación: $\varphi(x_{i-1}^t, x_i^t, x_{i+1}^t) \rightarrow x_i^{t+1}$.

La evaluación se realiza en cada x_i del arreglo de manera simultánea, es decir, en paralelo para generar el siguiente arreglo, determinando el espacio de evoluciones.

2.1.1 Gliders en la Regla 110

En esta sección presentamos todos los gliders hasta ahora conocidos en la Regla 110. Además utilizamos la clasificación propuesta por Cook [10] de aquí en adelante (ilustrada en la figura 2.2).

Los gliders son presentados en dos formas: en su estructura simple y en extensiones o paquetes de ellos (como se ilustra en la figura 2.2). Los gliders con superíndice $n \in \mathbb{Z}^+$ indica que esos gliders pueden extenderse arbitrariamente. Las extensiones hacia la izquierda son con los gliders $\bar{B}$ y $\hat{B}$, y las extensiones hacia la derecha son con los gliders E y G. Al final de la lista se muestra un glider gun extendido, donde la extensión es originada por un glider $\bar{E}$. Esta extensión fue descubierta al momento de buscar un glider gun, producida por alguna composición de gliders y aunque no es clasificado por Cook en [10], también él tenía conocimiento de dicha estructura. En la figura 2.2 podemos ver ejemplos de gliders extendidos y paquetes con su respectiva notación.

En el espacio de evoluciones de la Regla 110 podemos ver tres trayectorias en los gliders. Desplazamiento de izquierda a derecha con los gliders A, D_1 y D_2. Desplazamiento de derecha

a izquierda con los gliders B, $\bar{B}$, $\hat{B}$, E, $\bar{E}$, F, G, H y glider gun. La última trayectoria es con gliders que no tienen desplazamiento C_1, C_2 y C_3. Los gliders tienen un período que es determinado por el número de generaciones necesarias para repetir su estructura en el espacio de evoluciones y el desplazamiento es el cambio de células x_{i+d} o x_{i-d}, donde $d \in \mathbb{Z}$ indica el número de células que se está desplazando el glider en cada período.

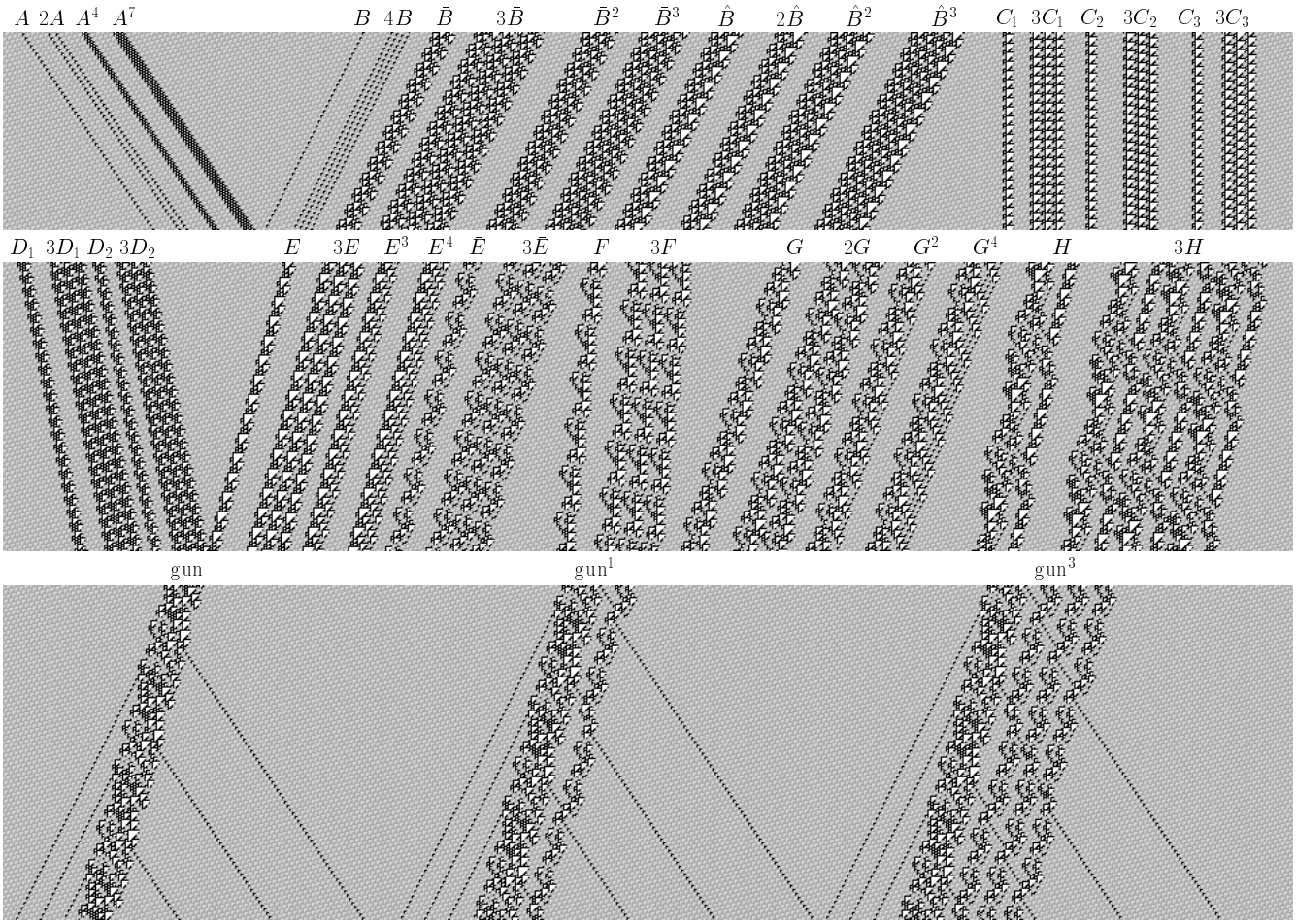

Figura 2.2: Clasificación de gliders en la Regla 110.

Es importante señalar que el subconjunto de expresiones regulares Ψ_{R110} que determinamos para todos los gliders en la Regla 110 no incluye extensiones o paquetes de éllos, es únicamente el glider en individual. Porque al tratar de enumerar todas esas extensiones o paquetes el subconjunto base de expresiones crece en diferentes módulos. Por lo tanto, el número de secuencias en el subconjunto es la unión de los períodos de cada glider:

$$\Psi_{R110} = \bigcup_{i=1}^{p} w_{i,g} \; \forall \; (w_i \in \Sigma^* \wedge g \in \mathcal{G}) \tag{2.1.1}$$

donde $\mathcal{G}$ es el conjunto de todos los gliders en Regla 110 y $p \geq 3$ el período. De esta manera, podemos hablar de un lenguaje regular L_{R110} que es construido a partir de las expresiones de Ψ_{R110}. Debemos notar que este lenguaje no es todo el lenguaje de la Regla 110, éste es unicamente el que es definido por las expresiones que representan los gliders, entonces tenemos que:

$$L_{R110} = \{w | w \in \Psi_{R110} \text{ y puede operar bajo las reglas básicas: } \cdot, +, *\}. \tag{2.1.2}$$

El lenguaje L_{R110} es determinado en base a las expresiones Ψ_{R110} que determinan cada glider y éste no ha sido publicado o presentado por otros autores. La determinación es establecida por los diagramas de de Bruijn y la discretización de los mosaicos donde ambos dan origen a nuestro análisis que hemos llamado "fases." Donde las fases indican con presición la posición y momento exacto donde cada glider (o partícula) es posicionada en la condición inicial.

Al aplicar este subconjunto de expresiones regulares operandolo con las operaciones básicas construimos condiciones iniciales que nos permiten obtener evoluciones importantes. El principal intéres que tenemos es para controlar, producir y manejar choques entre gliders. Convirtiendo este subconjunto en una herramienta muy poderosa para codificar condiciones iniciales en la Regla 110, además este subconjunto es implementado en un programa de computación. Aplicaciones inmediatas con resultados importantes en el estudio de la Regla 110 son realizadas en cientos, miles, millones y miles de millones de células, como veremos en el capítulo 3.

Ahora describimos las propiedades que cada glider tiene en diferentes aspectos, tales como: el nombre, márgenes periódicos, velocidad, ancho de la estructura y cubrimiento de glider en el espacio de evoluciones (ver tabla 2.2).

A continuación explicamos la tabla 2.2. La columna *estructura* indica el nombre del glider o estructura periódica. Las siguientes cuatro columnas etiquetadas *márgenes*, indican el número de márgenes periódicos que existen en cada glider. Los márgenes son divididos en márgenes con valor par representado por '*ems*' y márgenes con valor impar '*oms*' que a su vez son divididos en dos grupos: izquierdo y derecho, porque los gliders tienen márgenes pares e impares en sus fronteras izquierdas o derechas (o superior e inferior). Particularmente, las propiedades de los márgenes son explicadas detalladamente en la sección 2.2.4, donde se discute su origen, interpretación y representación para cada glider.

La columna $v_g \ \forall \ g \in \mathcal{G}$ indica la velocidad que tiene cada glider en el espacio de evoluciones, donde la velocidad es calculada dividiendo el desplazamiento d que tiene el glider entre su período p. Los tres tipos de trayectorias son identificadas en esta columna. Velocidad

positiva indica desplazamiento hacia la derecha. Velocidad negativa indica desplazamiento hacia la izquierda y velocidad cero indica que el glider no tiene desplazamiento.

La columna *ancho* indica el número mínimo y máximo de células necesarias que determina una cadena periódica en el arreglo lineal y representa un glider u otra estructura periódica. Por ejemplo, en el glider C_1 tenemos dos valores, esto quiere decir que con nueve o veintitrés células podemos definir dicho glider en la condición inicial.

estructura	márgenes izquierdo - derecho				v_g	ancho	cubrimiento
	ems	*oms*	*ems*	*oms*			
e_r	.	1	.	1	$2/3 \approx 0.666666$	14	T
e_l	1	.	1	.	-1/2 = -0.5	14	T
A	.	1	.	1	$2/3 \approx 0.666666$	6	T
B	1	.	1	.	-2/4 = -0.5	8	P
$\bar{B}^n$	3	.	3	.	-6/12 = -0.5	22	T
$\hat{B}^n$	3	.	3	.	-6/12 = -0.5	39	T
C_1	1	1	1	1	0/7 = 0	9-23	P
C_2	1	1	1	1	0/7 = 0	17	P
C_3	1	1	1	1	0/7 = 0	11	P
D_1	1	2	1	2	2/10 = 0.2	11-25	P
D_2	1	2	1	2	2/10 = 0.2	19	P
E^n	3	1	3	1	$-4/15 \approx -0.266666$	19	P
$\bar{E}$	6	2	6	2	$-8/30 \approx -0.266666$	21	P
F	6	4	6	4	$-4/36 \approx -0.111111$	15-29	P
G^n	9	2	9	2	$-14/42 \approx -0.333333$	24-38	P
H	17	8	17	8	$-18/92 \approx -0.195652$	39-53	P
glider gun	15	5	15	5	$-20/77 \approx -0.259740$	27-55	P

Tabla 2.2: Propiedades en los gliders de la Regla 110.

La última columna *cubrimiento* indica que gliders pueden cubrir completamente el espacio de evoluciones de la Regla 110. El cubrimiento puede ser total 'T' o parcial 'P,' donde cubrimiento total implica que el glider no necesita de mosaicos adicionales para cubrir por completo el espacio de evoluciones. Un cubrimiento parcial implica que al menos es necesaria la intervención de otro mosaico para que el glider y el nuevo mosaico puedan cubrir completamente el espacio de evoluciones. Esta representación es orientada al problema establecido por McIntosh de cubrir el espacio con diferentes mosaicos y encontrar qué conjuntos de gliders cumplen con esa condición, sólo que en este caso no estamos con uno o dos mosaicos en particular; en su lugar se aplican conjuntos de varios mosaicos de diferentes tamaños.

Con lo anterior podemos comentar otra dirección donde es posible buscar posibles contrucciones complejas pero no a través de condiciones iniciales, si no, a través de mosaicos. A partir de un conjunto inicial X podemos construir una familia de diferentes conjuntos X_i tal que, $\{\ldots \supset X_{i+1} \supset X_i \supset \ldots \supset X\}$ y cada conjunto debe producir un patrón diferente, entones ahora realizaremos operaciones con los mosaicos en el plano cartesiano como si fuera un rompecabezas pero sin violar las conecciones válidas determinadas por la Regla 110. Por lo tanto, en el sentido de Hao Wang podemos encontrar una composición de diferentes conjuntos X_i para implementar una secuencia de mosaicos operando una función lógica y por otro camino hablar de computación universal basada en dichas construcciones [15].

2.1.2 Clases de gliders en la Regla 110

Como hemos venido comentado los gliders y el espacio de evoluciones pueden ser descritos a través de diferentes mosaicos. Al momento de realizar la representación de gliders a través de los triángulos encontramos que los gliders pueden ser divididos en dos clases: *naturales* '$\mathcal{G}_N$' y *compuestos* '$\mathcal{G}_C$,' tal que, $\mathcal{G}_N \subset \mathcal{G}_C$. Los gliders $\mathcal{G}_N$ son aquellos que están formados sin alguna composición y los gliders $\mathcal{G}_C$ están formados por la composición de al menos un par de gliders $\mathcal{G}_N$. Esta relación se ilustra para cada glider de la Regla 110 en la tabla 2.3 ('dc' representa una descomposición caótica).

gliders $\mathcal{G}_N$	gliders $\mathcal{G}_C$
A	$E = A \to B$, $A^2 \to$ dc
B	$F = A \to B$, $B \to$ dc, $A^2 \to C_2$
$\bar{B}$	$G = A^2 \to$ dc
$\hat{B}$	$H = B \to$ dc, $A \to$ dc, $A \to E$, $D_1 \to$ dc
Cs	gun $= A^2 \to$ dc
Ds	
E	

Tabla 2.3: Clases de gliders en la Regla 110.

En los gliders de mayor período como son el glider H y glider gun, los choques internos que son producidos ayudan a mantener la forma de dicha estructura de construcción complicada, como podemos ver en la figura 2.3.

En el caso del glider H podemos ver claramente cómo la participación de un glider E (parte izquierda) ayuda a mantener la estructura de un glider más grande, porque el glider E choca periódicamente con la parte izquierda. La estabilidad entre ambas estructuras da origen a un glider más complicado: el glider H. Además, podemos ver la existencia

de un glider D_1 en la parte izquierda del glider H que sólo evoluciona en un período e inmediatamente es afectado por la parte interna después de haber chocado con el glider E.

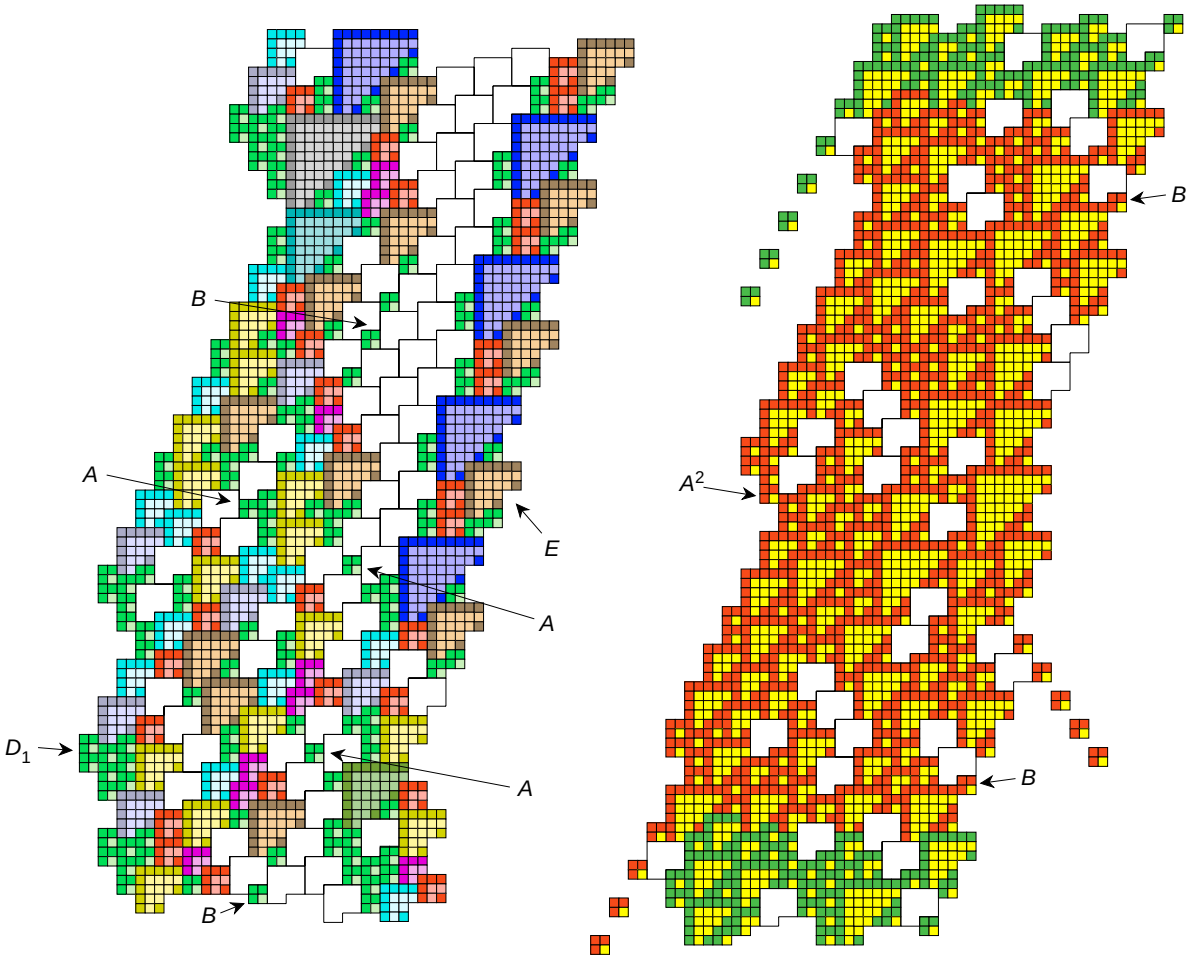

Figura 2.3: Dos tipos de gliders $\mathcal{G}_C$.

En el caso del glider gun también se pueden identificar varios fragmentos de otros gliders como son los gliders D_i, E, $\bar{E}$, F y G (figura 2.3). En este momento, debemos hacer notar que todos los gliders $\mathcal{G}_C$ tienen una característica en común: tienen el margen superior del glider $\bar{E}$, la construcción puede ser identificada en los márgenes superiores de los gliders F, G, H y glider gun.

En la estructura del glider gun es difícil identificar choques entre gliders naturales porque el interior del glider está formado en su mayor parte por regiones no periódicas. En el caso de los gliders $\bar{E}$, F y G, los choques internos son más visibles.

Finalmente, la clasificación de gliders propuesta permite pensar en la construcción de gliders $\mathcal{G}_C$ más grandes y complicados en su construcción o meta-gliders [24]. Los cuales

deben ser construidos entre varios gliders y cuya forma interna debe ser controlada por choques que ayuden a mantener la estructura. Ejemplos de gliders compuestos en otros AC pueden ser consultados en [54].

2.2 Determinando un lenguaje regular en la Regla 110

En esta sección explicamos cómo son determinadas y representadas las fases en el espacio de evoluciones de la Regla 110. El análisis inicia con la descripción del espacio de evoluciones por mosaicos [15] y aplicamos diagramas de de Bruijn [31, 47] para determinar el subconjunto finito de expresiones regulares basado en gliders. Ambos enfoques dan origen a la interpretación de "fases" en la Regla 110 y una vez determinadas las fases se explica el procedimiento para controlar choques específicos entre gliders codificando condiciones iniciales a través del subconjunto base de expresiones regulares Ψ_{R110} para determinar nuestro lenguaje regular L_{R110} basado en gliders.

2.2.1 Diagramas de de Bruijn

Los diagramas de de Bruijn [31, 47] se ajustan muy bien a las reglas de AC en una dimensión, aunque originalmente fueron empleados para el estudio de la teoría de registros de corrimientos (dicha teoría se basa en el tratamiento de secuencias donde algunos de sus elementos de la cadena intersectan con algún otra). A continuación explicamos que es un diagrama de de Bruijn. Ilustrando sus construcciones al momento de determinar las cadenas w para un par de gliders $\mathcal{G}$ en la Regla 110.

Definición 2.2.1. Para un autómata celular A en una dimensión de orden (k, r), el diagrama de de Bruijn es definido como una gráfica dirigida etiquetada con k^{2r} vértices y k^{2r+1} aristas. Los vértices son etiquetados con los elementos del alfabeto de longitud $2r$. Una arista es dirigida desde el vértice i al vértice j, si y sólo si, los r símbolos que conforman los vértices son los mismos y de esta manera poder formar una vecindad $2r+1$ que es representada como $i \diamond j$. Cuando éste es el caso, la arista conectando el vértice i al vértice j es etiquetada con el valor $\varphi(i \diamond j)$ (el valor de la vecindad definida en la función local) [47].

La matriz de conectividad M que determina el diagrama de de Bruijn se representa como:

$$M_{i,j} = \begin{cases} 1 & \text{si } j = ki, ki+1, \ldots, ki+k-1 \pmod{k^{2r}} \\ 0 & \text{en otro caso} \end{cases} \tag{2.2.1}$$

Ahora presentamos la matriz de conectividad y el diagrama de de Bruijn general para un AC 1D (2,1) (figura 2.4). Posteriormente discutimos su construcción.

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

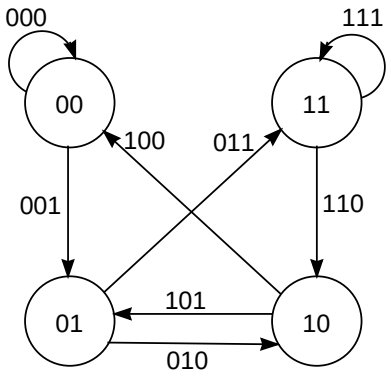

Figura 2.4: Diagrama de de Bruijn general para AC de orden (2,1).

La matriz de conectividad se obtuvo empleando la Ec. 2.2.1 y es calculada de la siguiente manera. 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 demás casos $M_{i,j} = 0$. Finalmente, tenemos que la ecuación 2.2.1 representa una gráfica dirigida de de Bruijn para cualquier AC de orden (k, r).

El módulo $k^{2r} = 2^2 = 4$ indica el número de vértices en el diagrama de de Bruijn y j debe tomar valores de $k * i = 2i$ hasta $(k * i) + k - 1 = (2 * i) + 2 - 1 = 2i - 1$. Los vértices tienen fracciones de vecindad originadas por las secuencias 00, 01, 10 y 11. Por lo tanto, las intersecciones determinan cada una de las conexiones.

En la tabla 2.4 se muestran las intersecciones que se derivan de los elementos de cada vértice. Las intersecciones son las aristas del diagrama de de Bruijn para un AC de orden

$(2, 1)$ como podemos ver en la figura 2.4.

El diagrama de de Bruijn para un AC de orden $(2, 1)$ 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 $2r + 1$.

$(0,\mathbf{0}) \diamond (\mathbf{0},0)$	$0\mathbf{0}0$
$(0,\mathbf{0}) \diamond (\mathbf{0},1)$	$0\mathbf{0}1$
$(0,\mathbf{1}) \diamond (\mathbf{1},0)$	$0\mathbf{1}0$
$(0,\mathbf{1}) \diamond (\mathbf{1},1)$	$0\mathbf{1}1$
$(1,\mathbf{0}) \diamond (\mathbf{0},0)$	$1\mathbf{0}0$
$(1,\mathbf{0}) \diamond (\mathbf{0},1)$	$1\mathbf{0}1$
$(1,\mathbf{1}) \diamond (\mathbf{1},0)$	$1\mathbf{1}0$
$(1,\mathbf{1}) \diamond (\mathbf{1},1)$	$1\mathbf{1}1$

Tabla 2.4: Intersecciones determinan aristas en el diagrama de de Bruijn.

A través del diagrama de de Bruijn las gráficas pueden representar cadenas, configuraciones o clases de configuraciones de un AC en su espacio de evoluciones. Las aristas del diagrama están asociadas con las vecindades de un AC usando los mismos símbolos que asocian los vértices en un paso de evolución.

Los vértices del diagrama de de Bruijn son secuencias de símbolos del conjunto de estados y los símbolos son secuencias de vértices de una gráfica en específico. Las aristas del diagrama de de Bruijn describen cómo tales secuencias se pueden intersectar. Consecuentemente, diferentes grados de intersección producen diferentes diagramas de de Bruijn [31].

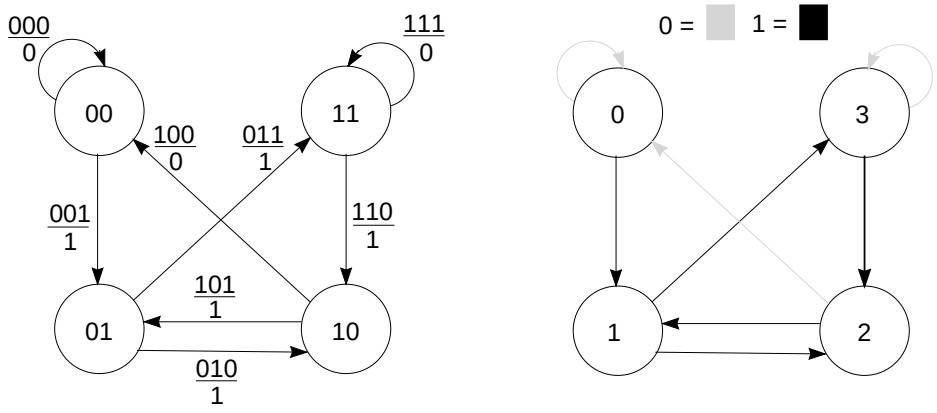

Figura 2.5: Diagrama de de Bruijn para la Regla 110.

El diagrama de de Bruijn para la Regla 110 se deriva del diagrama de de Bruijn genérico (figura 2.4) y es calculado en la figura 2.5. El color de la arista representa el estado al que evoluciona cada vecindad, como se ilustra en el segundo diagrama de la misma figura. La

asignación de colores es muy útil porque permite extraer directamente la cadena asociada dentro del diagrama, incluso cuando el diagrama tiene cientos de vértices y aristas.

Ahora debemos discutir otra variante donde el diagrama de de Bruijn puede ser extendido para determinar secuencias más grandes a través de su período y el desplazamiento de las células en el espacio de evoluciones de la Regla 110. Un problema es que para calcular diagramas de de Bruijn extendidos, éstos crecen exponencialmente en el orden de k^{2r^n} $\forall$ $n \in \mathbb{Z}^+$.

Por ejemplo, calculemos las secuencias de tamaño cinco y su respectivo diagrama de de Bruijn.

configuraciones de tamaño cinco			
intersección	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 2.5: Extensión de vecindades en el diagrama de de Bruijn.

En la tabla 2.5 se calculan todas las intersecciones entre las vecindades parciales de cuatro elementos (primera columna), generando todas las secuencias de longitud cinco (segunda columna). Las vecindades parciales bajo la operación intersección '$\diamond$' forman una vecindad. Después se aplica la regla de evolución 110 sin tomar en cuenta la propiedad a la frontera. Esto produce una secuencia de tamaño tres en la generación 1 (tercera columna). Se aplica nuevamente la regla de evolución en esta secuencia y se obtiene el estado al que va a evolucionar la vecindad (cuarta columna) de tamaño cinco.

Por ejemplo, la secuencia 0110 intersecta con la secuencia 1101 (renglón catorce) para formar la vecindad 01101. Después aplicamos la función de transición y se obtiene la secuencia 111 que es una vecindad más pequeña. Aplicamos nuevamente la función de transición a la nueva secuencia y obtenemos el estado 0, por lo tanto, la vecindad 01101 evoluciona a 0 en dos generaciones. El significado importante es que los valores de las aristas determinan una secuencia periódica que se desplaza en el tiempo en dos generaciones. El desplazamiento lo podemos determinar evolucionando la secuencia periódica y contando el número de posiciones que se está desplazando cada célula en el espacio de evoluciones.

Tenemos que las fracciones de vecindad compuestas por 4 células producen 16 vértices para el diagrama de de Bruijn extendido de este orden y 32 aristas que son todas las vecindades. Después se calcula la célula de evolución que le corresponde a cada vecindad de ese tamaño y cuando se llega a la célula de evolución se determina el número de generaciones necesarias donde las secuencias periódicas que están en ese diagrama de de Bruijn extendido se desplazan en el espacio de evoluciones. Finalmente, ésta es la manera en que se generan e interpretan los diagramas de de Bruijn extendidos.

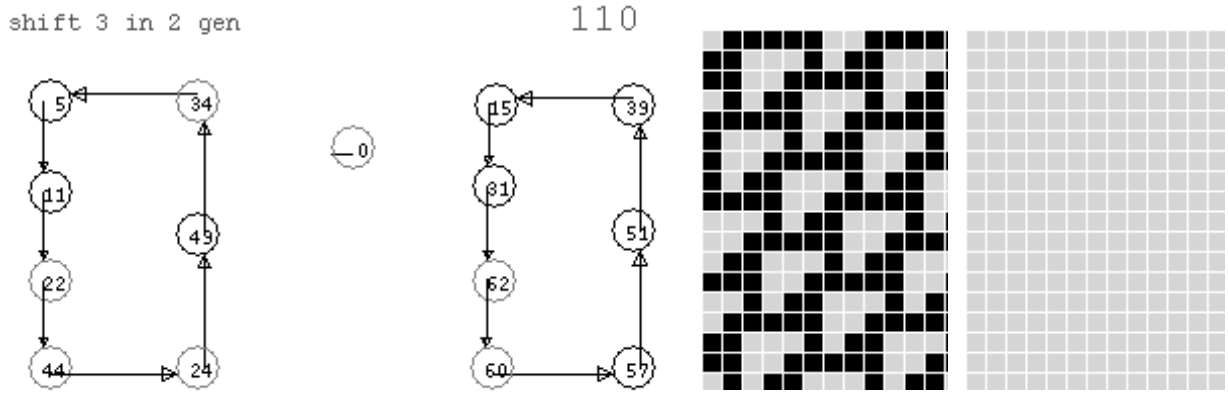

Figura 2.6: Diagrama de de Bruijn extendido determinando mosaicos: T_0 y T_3^{α}.

En este caso el diagrama de de Bruijn extendido calculado en la tabla 2.5 es ilustrado en la figura 2.6. Las gráficas de la izquierda muestran los ciclos que existen en ese diagrama

de de Bruijn (a la derecha están sus respectivas evoluciones). Eso significa que no todos los vértices ofrecen información relevante. En nuestro estudio sólo nos interesan los vértices que forman ciclos, porque los ciclos determinan las secuencias periódicas siguiendo una ruta en particular dentro del diagrama: entonces tenemos tres ciclos. La primera evolución ilustra el comportamiento de las cadenas $(1100010)^*$ ó $(1100111)^*$ determinadas por los ciclos del diagrama de de Bruijn de longitud 7, donde el estado es representado por el color del vértice (por ejemplo, el vértice 5 (000101) intersecta con el vértice 11 (001011) formando la vecindad 11 (0001011), que evoluciona al estado 1 como se describe en la tabla 2.5). En este caso ambos ciclos producen el mosaico T_3^α con diferentes cadenas. Además, las cadenas se desplazan tres elementos a la derecha cada dos generaciones.

El comportamiento para el tercer ciclo del diagrama de de Bruijn representado por el vértice 0 produce todas las secuencias 0^+. La segunda evolución de la figura 2.6 ilustra el comportamiento de esta secuencia que es dominada por el mosaico T_0 (evolución homogénea).

Los diagramas de de Bruijn extendidos[1] son diagramas que calculan todos las secuencias periódicas de una regla de evolución a través de los ciclos que determina el diagrama. Los diagramas calculan el desplazamiento de una secuencia periódica para un determinado número de generaciones. De esta manera, podemos calcular diagramas de de Bruijn que determinen todas las secuencias periódicas para cada glider en la Regla 110.

Para ilustrar cómo son determinadas las secuencias en un glider, ahora calculamos el diagrama de de Bruijn que determina el glider A en la Regla 110, discutiendo cómo son extraidas las secuencias periódicas del diagrama de de Bruijn que representan dicho glider y a su vez forman parte del conjunto de expresiones regulares.

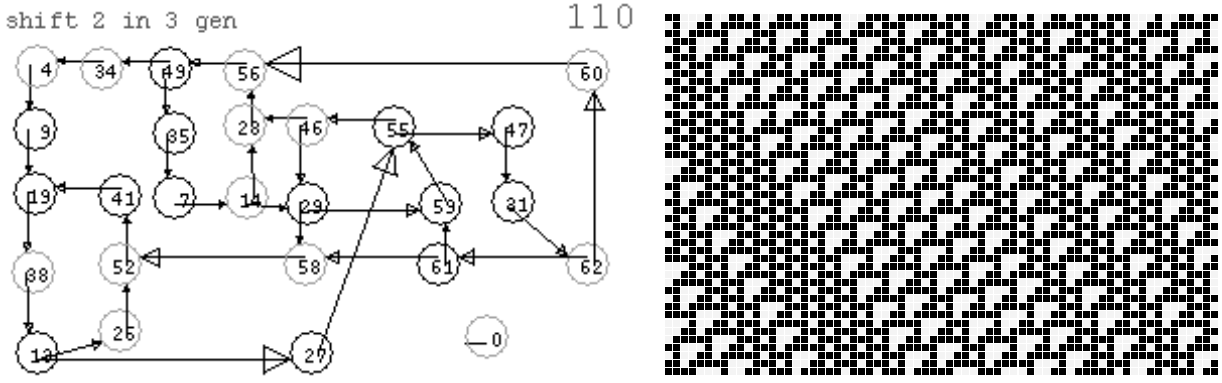

Figura 2.7: Diagrama de de Bruijn calculando el glider A y el éter.

[1]Los diagramas de de Bruijn son calculados con el sistema NXLCAU21 desarrollado por McIntosh en el sistema operativo NextStep (OpenStep y LCAU21 para MsDos). La aplicación y el código fuente son disponibles desde: `http://delta.cs.cinvestav.mx/~mcintosh/oldweb/software.html`

Conocemos que el glider A se desplaza dos células a la derecha en tres generaciones (tabla 2.2). Calculamos el diagrama de de Bruijn extendido (2-shift,3-gen) ilustrado en la figura 2.7. Los ciclos del diagrama calculan las secuencias periódicas para representar el glider A. Ahora bien, las secuencias están en el diagrama de de Bruijn pero no están clasificadas u ordenadas. Por lo tanto, tenemos que clasificarlas y ordenarlas.

En la figura tenemos dos ciclos: un ciclo formado por el vértice 0 y un ciclo grande de 26 vértices que a su vez es formado por 9 ciclos internos. La evolución de la derecha ilustra la asignación de diferentes secuencias periódicas produciendo el glider A en diferentes cantidades.

Siguiendo las rutas a través de las aristas obtenemos las secuencias o expresiones regulares que determinan una fase del glider A. Por ejemplo, tenemos que los ciclos formados por:

I. La expresión (1110)* vértices 29, 59, 55, 46 determina A^n gliders.

II. La expresión (111110)* vértices 61, 59, 55, 47, 31, 62 determina nA gliders con un mosaico T_3 entre cada glider A.

III. La expresión (11111000100110)* vértices 13, 27, 55, 47, 31, 62, 60, 56, 49, 34, 4, 9, 19, 38 determina el éter en una fase (en la siguiente subsección veremos que corresponde a la fase $e(\mathrm{f}_1\_1)$).

El ciclo de período uno representado por el vértice 0 produce una evolución homogénea s con el estado 0. La evolución de la derecha (figura 2.7) muestra diferentes grupos de gliders A. La condición inicial es construida siguiendo alguno de los siete ciclos posibles del diagrama de de Bruijn o varios de ellos a la vez. Pasar de un ciclo a otro determina el número de gliders A o el número de mosaicos T_3^{β}.

Los diagramas de de Bruijn determinan las transiciones calculando el valor de la arista al que evoluciona esa configuración. En la figura 2.7 cuando obtenemos las secuencias para representar el glider A y el éter podemos ver que el diagrama de de Bruijn es equivalente a un AF. Porque ellos determinan las cadenas que representan cada glider o estructura periódica de la Regla 110. Por lo tanto podemos clasificar y ordenar cada una de las cadenas para representarlas en una expresión regular.

Por ejemplo, si construimos el AF que representa unicamente el éter para una fase en particular entonces tenemos el diagrama de transiciones para esta máquina ilustrado en la figura 2.8. Este AF puede evaluar palabras de Σ^* pero puede distinguir (contar) todas las palabras que correspondan al éter – $e(\mathrm{f}_1\_1)$ – desde una cadena de entrada dada.

Construir un diagrama de transiciones en el AF de menor complejidad representa construir el diagrama más simple. En la figura 2.9 construimos el AF que representa el éter

$e(\mathrm{f_1\_1})$). El estado terminal $q_0 \in F$ determina que cadena es la que es aceptada unicamente en esta máquina de estados finitos. Ahora calculamos y analizemos el AF que representa el glider A ilustrado en la figura 2.10, dicho diagrama puede ser encontrado en el diagrama de de Bruijn de la figura 2.7.

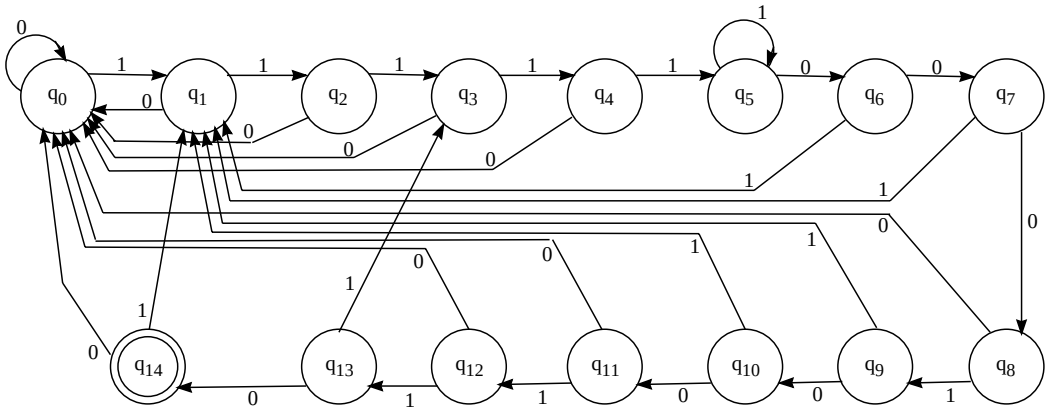

Figura 2.8: AF contando cadenas del etér y puede leer otras cadenas.

El glider A es determinado por el AF de la figura 2.10 y produce un glider A con un mosaico T_3 dada la expresión regular $(111110)^*$. Debemos hacer notar que cada máquina solo puede representar una fase de una estructura dada. Si queremos construir autómatas con la capacidad de reconocer más de una estructura, eventualmente implicará una mayor complejidad en el diagrama de transiciones. Por ejemplo el glider A y el éter es determinado por el AF de la figura 2.11. Nuevamente las palabras aceptadas por este autómata representan una fase de cada estructura, donde la fase es identificada en los estados terminales. Si deseamos representar todas las fases para ambas estructuras (en este ejemplo), entonces el conjunto de estados finales y los elementos en el diagrama de transiciones crece exponencialmente y lo convierte en un problema NP-completo [18, 50].

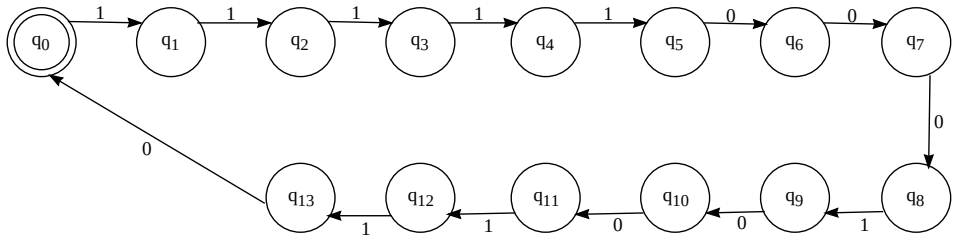

Figura 2.9: AF determinando unicamente cadenas que representan el éter.

Ahora debemos señalar un problema en el cálculo de los diagramas de de Bruijn para

obtener todas las secuencias periódicas que representa cada glider en la Regla 110. El sistema NXLCAU21 actualmente puede calcular diagramas de de Bruijn extendidos hasta diez generaciones (que implica un enorme diagrama de a lo más 1,048,576 vértices). Consecuentemente, tratar de ordenar o intentar una clasificación de todos los ciclos es una tarea muy laboriosa. Además, como podemos ver en la tabla 2.2, los gliders E, $\bar{E}$, F, G, H y los glider guns exceden por varias generaciones el límite de diez generaciones. Para solucionar este problema y determinar todas las expresiones regulares para cada glider de la Regla 110, derivamos todas las fases para cada glider alineando mosaicos T_3^{β}.

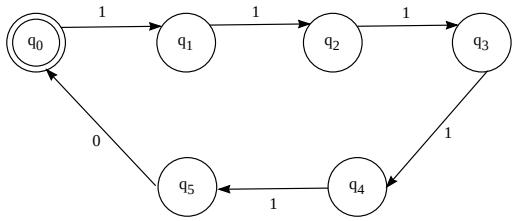

Figura 2.10: AF determinando cadenas que representan el glider $A(\mathrm{f}_1\_1)^*$.

Finalmente, debemos resaltar la utilidad de los diagramas de de Bruijn. Los diagramas de de Bruijn son en realidad AF y los ciclos del diagrama determinan todas las cadenas que son periódicas para la regla de evolución del autómata celular en estudio en su espacio de evoluciones, incluyendo los gliders.

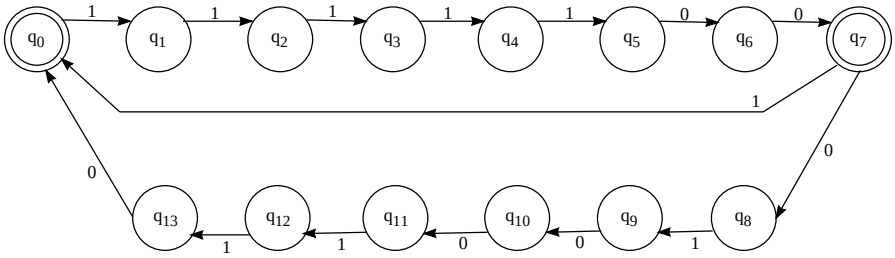

Figura 2.11: AF determinando expresiones $(e(\mathrm{f}_1\_1)+A(\mathrm{f}_1\_1))^*$.

En estos momento podemos realizar una pregunta general, una vez que conocemos cual es la relación entre estas máquina de estados finitos. Buscamos una analogía con la búsqueda de máquinas de Turing universales pequeñas [35]. Entonces la pregunta es: ¿Cuál es el diagrama de de Bruijn o el AF para la Regla 110 que debe reconocer todas las palabras determinadas por los ciclos?

La pregunta no es trivial y recurrimos a un resultado establecido por McIntosh en [31]. Debemos construir el diagrama de subconjuntos a partir de nuestro diagrama de de Bruijn de la Regla 110 (ver figura 2.5).

El diagrama de subconjuntos tiene $2^{k^{2r}}$ vértices, si todas las configuraciones de cierta longitud poseen ancestros entonces todas las configuraciones con extensiones tanto a la izquierda como a la derecha con la misma equivalencia deben tener ancestros. Si éste no es el caso, entonces ellos describen las configuraciones conocidas como Jardín del Edén (cadenas que no tienen ancestros [25]) y no es más que la descripción del camino principal que va del conjunto máximo al conjunto mínimo dentro del diagrama de subconjuntos.

Los vértices dentro del diagrama de subconjuntos estan formados por la combinación de todos los subconjuntos que se pueden formar a partir del número de estados que conforman el diagrama de de Bruijn (conjunto potencia). Por ejemplo para un AC $(2, 1)$ tenemos cuatro fracciones de estados en el diagrama de de Bruijn $\{a\}$, $\{b\}$, $\{c\}$ y $\{d\}$, y de ahí podemos formar todos los subconjuntos posibles: $\{a,\ b,\ c,\ d\}$, $\{a,\ b,\ c\}$, $\{a,\ b,\ d\}$, $\{a,\ c,\ d\}$, $\{b,\ d,\ c\}$, $\{a,\ b\}$, $\{a,\ c\}$, $\{a,\ d\}$, $\{b,\ c\}$, $\{b,\ d\}$, $\{d,\ c\}$, $\{d\}$, $\{c\}$, $\{b\}$, $\{a\}$ y $\{\}$. Dentro de estos subconjuntos se pueden distingir de manera clara las cuatro clases unitarias que se forman, la integración del conjunto vacio garantiza que todos los subconjuntos tengan al menos una imagen, aunque ésta no exista en el diagrama original. Para determinar el tipo de unión que existen entre los subconjuntos se debe revisar el estado en que evoluciona cada estado y de esta manera conocer hacia que estados (subconjunto que lo formen) se puede conectar; de esta manera se construye la siguiente relación para la Regla 110.

vértice	arista con 0	arista con 1
a	a	b
b	ϕ	c, d
c	a	b
d	d	c

Tabla 2.6: Relación entre estados.

De esta manera podemos construir la matriz de subconjuntos donde la dimensión de la misma esta definida por el número de vértices del diagrama y a su vez puede ser particionada a través de sus clases unitarias. Podemos observar que el elemento fundamental para definir alguna función es que cada argumento tiene justo una imagen. Las aristas del diagrama definen una función cuando existe una liga de una clase hacia un vértice; pero si esta condición no se cumple, se debe a que existe una combinación de múltiples aristas a una sola arista entre subconjuntos que implica la existencia de más de una entrada al vértice a través del mismo estado o diferentes estados.

Aunque las aristas de una gráfica como tal no definen una función las aristas entre subconjuntos son siempre funcionales y ésto se debe por la inclusión del conjunto vacio definido para toda la gráfica. Cada clase de aristas definen una función, sea Σ_0 ó Σ_1. El

diagrama de subconjuntos describe la unión de $\Sigma_0 \cup \Sigma_1$, que por si misma no es funcional.

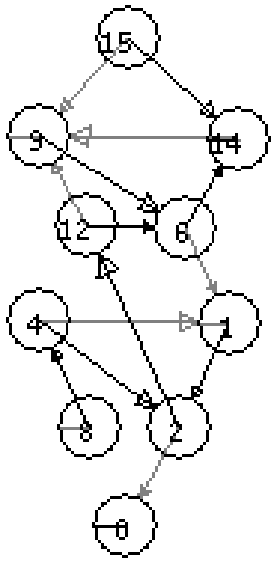

Figura 2.12: Diagrama de subconjuntos para la Regla 110.

Sean a y b vértices, S un subconjunto y $|S|$ la cardinalidad de S; entonces el diagrama de subconjuntos está definido por la siguiente ecuación:

$$\sum_i(S) = \begin{cases} \phi & S = \phi \\ \{b \mid \text{arista}_i\ (a,b)\} & S = \{a\}. \\ \bigcup_{a \in S} \Sigma_i(a) & |S| > 1 \end{cases} \tag{2.2.2}$$

De aquí se desprenden tres propiedades importantes:

1. Si existe una cadena principal del subconjunto máximo al conjunto mínimo, entonces debe existir una cadena similar que salga de algún subconjunto menor al conjunto vacio. Por el contrario si las clases unitarias carecen de aristas al conjunto vacio, aquí no existen configuraciones Jardín del Edén.

2. Existe un cierto residuo del diagrama de de Bruijn, en el sentido de que dado un origen y un destino, siempre debe haber un subconjunto conteniendo el destino accesible y otro subconjunto conteniendo el origen, además el destino puede tener vértices adicionales.

3. El diagrama de subconjuntos puede no estar conectado, si éste es el caso es interesante conocer el subconjunto más grande accesible desde algún subconjunto dado, así como el subconjunto más pequeño.

La función de transición local φ de la Regla 110 tiene una correspondencia inyectiva, conociendo esta correspondencia entonces debemos encontrar rutas en el diagrama de subconjuntos que vayan del conjunto máximo al conjunto vacio. Dos configuraciones minimales que representan Jardín del Edén en la Regla 110 son: (101010)* y (01010)*.

Además de obtener las secuencias Jardín del Edén a través del diagrama de subconjuntos también obtenemos la máquina de estados finitos para el lenguaje regular L_{R110}. Para verificarlo solo es necesario tomar una secuencia de nuestro subconjunto de expresiones regulares Ψ_{R110} y entonces debe existir un camino en el diagrama de subconjuntos iniciando desde el conjunto máximo que determina su existencia.

2.2.2 Fases en la Regla 110

En esta sección discutimos cómo son derivadas, representadas y obtenidas las fases que van a determinar secuencias periódicas en el espacio de evoluciones de la Regla 110.

El mosaico T_3^{β} ilustrado en la figura 2.13 determina cuatro fases o secuencias por renglón: $\mathrm{f}_1 = 1111$, $\mathrm{f}_2 = 1000$, $\mathrm{f}_3 = 1001$, y $\mathrm{f}_4 = 10$ (de aquí en adelante nos referimos al mosaico T_3^{β} simplemente como T_3). De esta manera, la concatenación de las cuatro fases f_i determinan una secuencia (periódica) que representa el éter: $\mathrm{f}_1\mathrm{f}_2\mathrm{f}_3\mathrm{f}_4 = 11111000100110$.

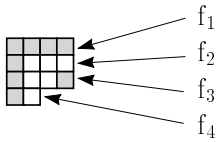

Figura 2.13: Fases f_i en el mosaico T_3.

Siguiendo cada uno de los niveles del mosaico T_3 determinamos que a lo más existen cuatro fases para representar una secuencia periódica. Después derivamos todas las fases posibles para representar el éter en la Regla 110 y las definimos de la siguiente manera: $e(\mathrm{f}_1\_1) = 11111000100110$, $e(\mathrm{f}_1\_2) = 10001001101111$, $e(\mathrm{f}_1\_3) = 10011011111000$, y $e(\mathrm{f}_1\_4) = 10111110001001$.

Generalmente el espacio de evoluciones de la Regla 110 converge al éter desde condiciones iniciales aleatorias a través del tiempo con una probabilidad de 0.57. La condición inicial construida por la expresión $e(\mathrm{f}_1\_i)^*$ $\forall$ $1 \leq i \leq 4$, donde el intervalo es establecido por todas las fases que existen, cubre el espacio de evoluciones de la Regla 110 únicamente con éter. Notemos que cada una de las fases $e(\mathrm{f}_1\_i)$ es una permutación de la primera. Por lo tanto, con fijar una fase es suficiente para establecer nuestra medida. De esta manera, por orden secuencial elegimos las fases f_i_1 para establecer nuestra medida horizontal.

Cook determina dos medidas en el espacio de evoluciones de la Regla 110 [10]: horizontal $\frown^i$ y vertical $\nearrow^i$. Nosotros sólo determinamos una medida horizontal f_i_1.

Las fases f_i_1 tienen cuatro subniveles que son consecuencia de las fases del mosaico T_3 como se muestra en la figura 2.14 (gráfica izquierda) y cada una de estas fases se puede alinear i veces generando todas las fases posibles (gráfica derecha).

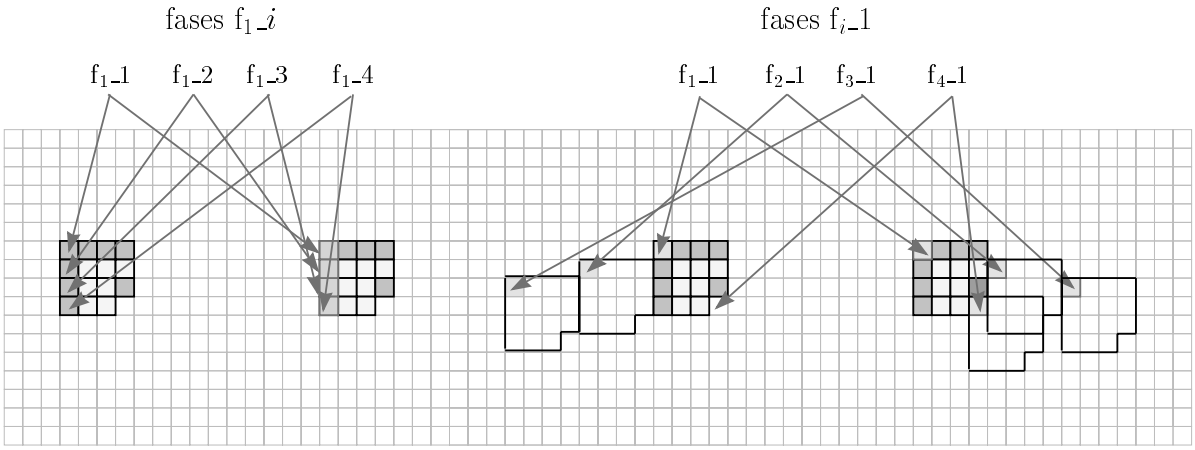

Figura 2.14: Fases f_i_i con el mosaico T_3.

Las fases representan las secuencias periódicas (expresiones regulares para cada glider) de longitud finita del diagrama de de Bruijn. Es importante señalar que una alineación de una fase determina un subconjunto de expresiones regulares y otra alineación determina otro subconjunto de expresiones regulares. De esta manera, tenemos cuatro subconjuntos posibles (tabla 2.7): $\mathcal{F}_1$ (fases en el nivel uno), $\mathcal{F}_2$ (fases en el nivel dos), $\mathcal{F}_3$ (fases en el nivel tres) y $\mathcal{F}_4$ (fases en el nivel cuatro), donde los subconjuntos son disjuntos entre sí para construir condiciones iniciales. La propiedad de expresiones regulares es conservada únicamente en los dominios de cada subconjunto si queremos proyectar dichas estructuras en la dinámica del autómata celular, donde la separación se origina porque debemos recordar que tenemos cuatro permutaciones para representar el éter. Entonces existen cuatro subconjuntos donde las secuencias son permutaciones entre sí y por esa razón con un solo subconjunto es suficiente para construir condiciones iniciales bajo las reglas de expresiones regulares.

fases nivel uno ($\mathcal{F}_1$)	$\rightarrow$	$\{f_1\_1, f_2\_1, f_3\_1, f_4\_1\}$
fases nivel dos ($\mathcal{F}_2$)	$\rightarrow$	$\{f_1\_2, f_2\_2, f_3\_2, f_4\_2\}$
fases nivel tres ($\mathcal{F}_3$)	$\rightarrow$	$\{f_1\_3, f_2\_3, f_3\_3, f_4\_3\}$
fases nivel cuatro ($\mathcal{F}_4$)	$\rightarrow$	$\{f_1\_4, f_2\_4, f_3\_4, f_4\_4\}$

Tabla 2.7: Cuatro conjuntos de fases $\mathcal{F}_i$ en la Regla 110.

La manera para calcular todo el conjunto de cadenas para todos los gliders y siguiendo una alineación es con las fases f_i_1. Para determinar las fases, primero es necesario describir cada glider de la Regla 110 a través de los mosaicos, en su forma y límites. Después fijamos una fase, en nuestro caso tomamos la fase f_i_1 y trazamos una línea horizontal en el espacio de evoluciones empatando dos mosaicos T_3, como se muestra en la segunda ilustración de la Fig. 2.14. De esta manera, la secuencia existente entre los dos mosaicos alineada en cada uno de los cuatro niveles determina una secuencia periódica que representa una estructura en particular en el espacio de evoluciones de la Regla 110. Luego calculamos todas las secuencias periódicas en una fase determinada y este procedimiento enumera todas las secuencias periódicas para representar cada glider.

La variable f_i indica la fase que estamos utilizando y el segundo subíndice i (formando la notación f_1_1) indica qué subconjunto de expresiones regulares $\mathcal{F}_i$ estamos utilizando. Finalmente, nuestra notación propuesta para codificar condiciones iniciales a través de fases se representa de la siguiente manera:

$$\#_1(\#_2, f_i\_1) \tag{2.2.3}$$

donde $\#_1$ representa el glider según la clasificación de Cook (tabla 2.2) y $\#_2$ la fase del glider si el período del glider es mayor que cuatro. Debemos señalar que la asignación que se le dio por letras mayúsculas a $\#_2$ (en el sistema OSXLCAU21 - Fig. A.5) no tiene ningún significado en especial; es únicamente para darle una representación a los diferentes niveles de fases con gliders de períodos módulo cuatro.

Ahora determinamos las fases f_i_1[2] para los gliders A y B como se ilustran en la figura 2.15. Los mosaicos T_3 determinan una fase $\#_1$. En el caso de los gliders A y B sólo es necesario un mosaico T_3 para describir su estructura. En todos los demás gliders son necesarios al menos dos mosaicos T_3.

Siguiendo cada una las fases iniciadas por cada mosaico T_3 tenemos que todas las fases f_i_1 para el glider A son:

- $A(f_1\_1) = 111110$
- $A(f_2\_1) = 11111000111000100110$
- $A(f_3\_1) = 11111000100110100110$

La secuencia es determinada tomando el primer valor de la primera célula del mosaico T_3 de la izquierda hasta empatar una segunda célula que representa el primer valor del segundo

[2]El subconjunto de las expresiones regulares Ψ_{R110} para cada uno de los gliders de la Regla 110 (Apéndice B), sirve como datos de entrada al sistema OSXLCAU21.

mosaico T_3 de la derecha. En la figura 2.15 una célula de color negro indica dónde se acota cada fase.

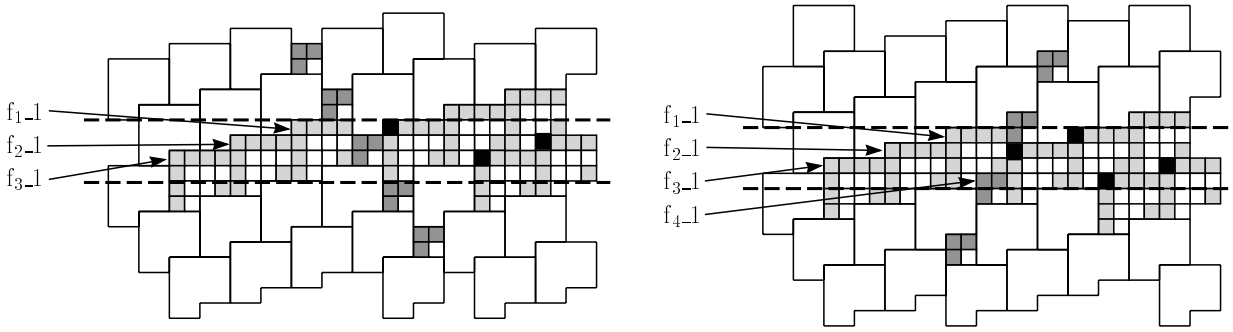

Figura 2.15: Fases f_i_1 en los gliders A y B respectivamente.

En general para toda estructura con velocidad negativa, la fase f_4_1 = f_1_1. Por esa razón no se escribe dicha fase. Cada una de las secuencias periódicas determinadas con los mosaicos T_3 conservan la propiedad de expresión regular al aplicar las reglas básicas. Por lo tanto, las expresiones: ϵ, $A(f_1\_1)$, $A(f_1\_1)$+$A(f_1\_1)$, $A(f_1\_1)$-$A(f_1\_1)$, $A(f_1\_1)$* y $A(f_3\_1)$-$A(f_1\_1)$-$A(f_2\_1)$-$A(f_3\_1)$-$A(f_2\_1)$ son expresiones regulares (denotamos '-' para representar la operación concatenación en nuestras construcciones). Recordemos la codificación en fases, A indica el glider ($\#_1$) y f_i_1 indica la fase.

De esta manera, todas las fases f_i_1 para el glider B son:

- $B(f_1\_1)$ = 11111010
- $B(f_2\_1)$ = 11111000
- $B(f_3\_1)$ = 1111100010011000100110
- $B(f_4\_1)$ = 11100110

La descripción que realizamos con los gliders A y B es proyectada a todos los gliders para determinar todo el conjunto de expresiones regulares base Ψ_{R110}. Ahora discutimos algunas propiedades que se derivan por el mosaico T_3 que representa el éter en la Regla 110 y se reflejan en todo el espacio de evoluciones para cada una de sus estructuras periódicas y no periódicas.

El mosaico T_3 determina tres tipos de pendientes[3] como podemos ver en la figura 2.16:

[3]Si $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$ son dos puntos diferentes cualesquiera de una recta, la pendiente m de la recta es: $m = \frac{y_1 - y_2}{x_1 - x_2}$. Entonces seleccionamos el primer punto (i, j) en el espacio de evoluciones de la Regla 110 dentro de algún mosaico T_3 para cada uno de sus desplazamientos. Si el desplazamiento es de izquierda a derecha, el segundo punto es $(i + 2, j + 3)$. Si el desplazamiento es de derecha a izquierda, el segundo punto es $(i - 2, j + 4)$

pendiente positiva "p^+," pendiente negativa "p^-" y pendiente nula "p^0." Las pendientes p^+ y p^- determinan la velocidad máxima positiva y negativa para todo glider que puede existir en el espacio de evoluciones de la Regla 110.

Si p^+ se tiene un desplazamiento de +2 células en 3 generaciones, entonces la velocidad del éter es $v_{er} = 2/3$. Si p^- se tiene un desplazamiento de -2 células en 4 generaciones, entonces la velocidad del éter es $v_{el} = -1/2$ (como habíamos señalado en la tabla 2.2).

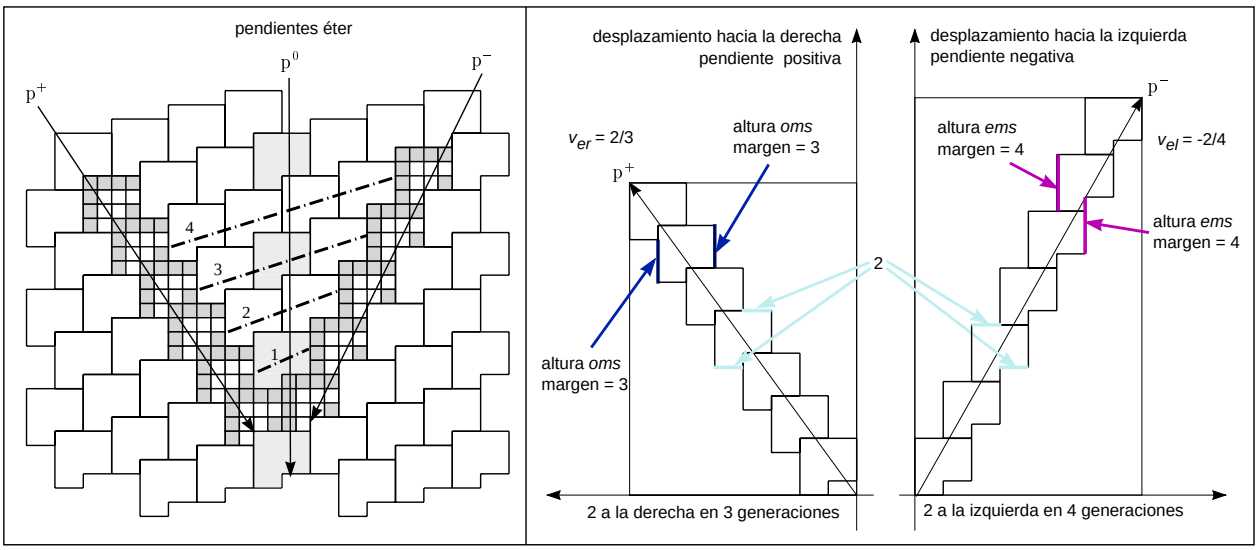

Figura 2.16: Tres tipos de pendientes producidas por el éter.

En el análisis por fases tenemos que el mosaico T_3 determina la existencia de dos márgenes "*oms*" y "*ems*" (ilustración derecha de la figura 2.16) en ambas pendientes y por cada mosaico, que derivan otras propiedades importantes.

Si p^+, entonces existe un margen impar *oms* con una altura de tres células. Si p^-, entonces existe un margen par *ems* con una altura de cuatro células. Los puntos de contacto[4] son determinados por el número de márgenes impares *oms* cuando p^+ o pares *ems* cuando p^-. Finalmente, ambos márgenes (izquierdos y derechos en una estructura periódica) tienen una correspondencia biyectiva (ver tabla 2.2).

Si existen n márgenes *oms* en la parte superior cuando p^+ en un glider, entonces existen n márgenes *oms* en su parte inferior. En el caso contrario, si existen n márgenes *ems* en la parte superior cuando p^-, entonces existen n márgenes *ems* en su parte inferior. En otras

[4]Un punto de contacto [41] indica una región del glider donde éste puede chocar contra otro glider. Por lo tanto, un punto de no contacto implica una región donde el glider no puede ser afectado por algún choque. Pero además los puntos de contacto no sólo existen en estructuras periódicas de la Regla 110, sino que también existen en estructuras no periódicas.

palabras, la existencia de un punto de contacto en un glider implica la existencia de un punto de no contacto en su contraparte. De aquí se desprenden las otras propiedades.

Toda estructura periódica o no periódica que exista en el espacio de evoluciones de la Regla 110 avanza +2 células y retrocede −2 células, entonces toda estructura con p^+ tiene una velocidad $v_g \leq v_{er}$ y si p^- entonces $v_g \leq |v_{el}|$, donde v_g representa la velocidad de un glider g (ver tabla 2.2). Por lo tanto, toda estructura con p^+ avanza con incrementos de v_{er} y retrocede con decrementos de v_{el}. En caso contrario, toda estructura con p^- avanza con incrementos de v_{el} y retrocede con decrementos de v_{er}.

Toda estructura con p^+ puede ser afectada por otra estructura con diferente pendiente (p^0 o p^-), sólo si la primera tiene al menos un margen oms y la segunda tiene al menos un margen ems. En caso contrario, toda estructura con p^- puede ser afectada por otra estructura con diferente pendiente(p^0 o p^+), sólo si la primera tiene al menos un margen ems y la segunda tiene al menos un margen oms.

Finalmente, estas propiedades dan origen a las siguientes ecuaciones. Sea $\mathcal{G}$ el conjunto de todos los gliders en la Regla 110, entonces el desplazamiento de un glider $g \in \mathcal{G}$ se representa de la siguiente manera:

$$d_g = (2 * oms) - (2 * ems). \tag{2.2.4}$$

Toda estructura periódica tiene un período definido por el número de márgenes oms y ems. Por lo tanto, el período de un glider g es derminado por:

$$p_g = (3 * oms) + (4 * ems) \tag{2.2.5}$$

y tiene una velocidad de:

$$v_g = \frac{(2 * oms) - (2 * ems)}{(3 * oms) + (4 * ems)}. \tag{2.2.6}$$

Los choques entre gliders tienen una cota máxima que es determinada por el número de márgenes oms y ems. De esta manera, para un glider arbitrario con oms puntos de contacto y otro glider arbitrario diferente al primero de ems puntos de contacto, tenemos el siguiente número de choques:

$$c \leq oms * ems \tag{2.2.7}$$

donde c representa el número máximo de choques que pueden existir entre dos gliders. Sin embargo, en algunos gliders la cota máxima no se cumple. Depurando la igualdad se tiene que el número de choques que existen entre dos gliders g_i, $g_j \in \mathcal{G}$ donde $i \neq j$, es representada por la siguiente ecuación:

$$c = |(oms_{g_i} * ems_{g_j}) - (oms_{g_j} * ems_{g_i})|. \tag{2.2.8}$$

El procedimiento que utilizamos para codificar condiciones iniciales a través de las fases f_i_1 para manejar choques, determinan dos medidas para representar distancias en el espacio lineal de la Regla 110 (tabla 2.8): mod 4 (por cada mosaico T_3). En este caso tenemos una distancia mínima de cero mosaicos T_3 y una distancia máxima de tres mosaicos T_3 entre gliders. La segunda medida es módulo 14 (por número de células). En este caso tenemos una distancia mínima de 0 células o $4 + 4 + 4 + 2$ células entre gliders. La restricción es determinada por el mosaico T_3.

p^+	p^-	p^0
$f_1\_1 \mapsto 1T_3$(der) - $0T_3$(izq)	$f_1\_1 \mapsto 1T_3$(izq) - $1T_3$(der)	$f_1\_1 \mapsto 1T_3$(izq) - $0T_3$(der)
$f_2\_1 \mapsto 2T_3$(der) - $3T_3$(izq)	$f_2\_1 \mapsto 2T_3$(izq) - $0T_3$(der)	$f_2\_1 \mapsto 2T_3$(izq) - $3T_3$(der)
$f_3\_1 \mapsto 3T_3$(der) - $2T_3$(izq)	$f_3\_1 \mapsto 3T_3$(izq) - $3T_3$(der)	$f_3\_1 \mapsto 3T_3$(izq) - $2T_3$(der)
$f_4\_1 = f_1\_1$	$f_4\_1 \mapsto 0T_3$(izq) - $2T_3$(der)	$f_4\_1 \mapsto 0T_3$(izq) - $1T_3$(der)

Tabla 2.8: Fases determinan distancias mod 4 (por mosaicos T_3).

La importancia de conocer y determinar una distancia en el espacio lineal de la Regla 110 es para tener un control adecuado de las posiciones de los gliders y de esta manera obtener las reacciones deseadas. Ahora analicemos las distancias que establecen las fases (por mosaico) cuando tenemos pendientes p^+, p^- y p^0, tal y como se describe en la tabla 2.8.

En el caso de las secuencias éter, las distancias son las mismas e implica una distancia de 4 mosaicos T_3 que es la distancia máxima por mosaico. Cuando p^+, puede tomarse el glider A como ejemplo. Cuando p^-, puede tomarse el glider B y cuando p^0, puede tomarse algún glider C para verificar las distancias.

Si un glider tiene una p^-, entonces las fases intersectan. Si un glider tiene una p^+, entonces las fases no intersectan. Si la fase f_4_1 no intersecta con la fase f_1_1 entonces se tienen todas las fases f_i_1. Esto representa cuatro secuencias periódicas diferentes. Si la fase f_4_1 intersecta con la fase f_1_1, entonces la fase $f_4\_1 = f_1\_1$.

Si un glider con p^+ choca en sus cuatro fases secuencialmente contra un glider con p^- en sus cuatro fases ubicados en una posición (x, y), entonces los choques se producen en la misma y-posición con distancias idénticas entre cada intervalo. La distancia mínima es importante para generar un choque propio si éste es no-propio. Kenneth Steiglitz determina dos tipos de choques [41] entre gliders en AC:

- *Choque propio.* Los choques se producen en un punto de contacto.

- *Choque no propio.* Los choques se producen si los gliders intersectan en sus secuencias, es decir, sus secuencias intersectan desde condiciones iniciales o durante la evolución.

Regla 110 tiene las dos formas de choques no propios: el primer caso es donde las fases intersectan desde condiciones iniciales y el segundo caso es donde se producen varios gliders en regiones con distancias que no pueden ser extrapoladas.

2.2.3 Proyectando fases a estructuras no periódicas

Las fases no sólo pueden ser determinadas a estructuras periódicas de la Regla 110, también son determinadas en estructuras no periódicas.

La proyección a estructuras no periódicas es realizada del mismo modo que hicimos con las estructuras periódicas, sólo que en el caso de los gliders es más fácil porque éstos tienen un período y, por lo tanto, el número de fases son fijas. Sin embargo, para descomposiciones de regiones caóticas cortas o largas, el número de márgenes *oms* y *ems* varían arbitrariamente. Las regiones caóticas pueden iniciar desde condiciones iniciales. En otros casos son originadas por los choques de dos o varios gliders e incluso por la interacción de una o varias regiones caóticas cercanas.

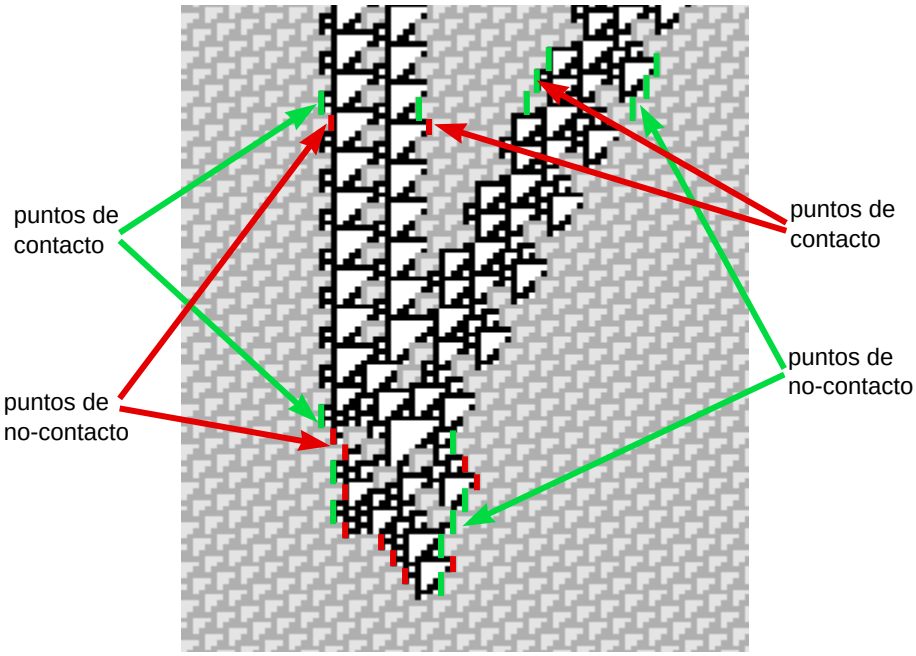

Figura 2.17: Aniquilación de gliders en una descomposición de historia corta.

En la figura 2.17 se ilustra una pequeña descomposición producto del choque entre tres

gliders. La secuencia para este ejemplo es: e*-C_2(A,f$_3$_1)-C_2(A,f$_1$_1)-e-$\bar{B}$(A,f$_2$_1)-e*.[5]

El glider C_2 tiene un margen *oms* y un margen *ems* en cada extremo. El glider $\bar{B}$ tiene 3 márgenes *oms* y 0 márgenes *ems* en cada extremo. La región caótica tiene en la parte izquierda 8 márgenes *oms* y 3 márgenes *ems* en el margen izquierdo, pero en el margen derecho tiene 3 márgenes *oms* y 5 márgenes *ems*. En este caso, la descomposición tiene varios puntos donde otras estructuras pueden interactuar con ella en ambos márgenes. Consecuentemente, para toda estructura no periódica la correspondencia biyectiva entre los márgenes *oms* y *ems* no se conserva, porque no tienen un período.

Concluyendo, toda estructura que exista en el espacio de evoluciones de la Regla 110 debe tener al menos un punto de contacto y no contacto.

Por ejemplo, en la figura 2.17 se pueden ver dos gliders C_2 juntos ($2C_2$) y podemos distinguir los márgenes *ems* y *oms*, conservando el par de gliders sin ser alterados. Al final de la descomposición caótica el último choque es entre un glider A contra un glider B a una distancia de 0 mosaicos T_3. Por esa razón la descomposición no deja residuos al final de su historia. La producción de este choque entre los gliders A y B con distancia 0 puede realizarse con la siguiente expresión: e*-A(f$_1$_1)-B(f$_4$_1)-e*.

2.2.4 Procedimiento para producir choques

El propósito del procedimiento es construir condiciones iniciales en el espacio lineal de la Regla 110 y de esta manera controlar choques en su espacio de evoluciones. Las construcciones son codificadas a través de las fases f$_i$_1 que a su vez determinan el conjunto base de las expresiones regulares Ψ_{R110}. El punto importante es que ofrecemos el primer procedimiento para controlar el espacio de evoluciones en un AC con comportamiento complejo de 1D y de esta manera manejar los choques que deseemos entre todas las estructuras.

Sea Ψ_{R110} el conjunto base de todas las expresiones regulares determinadas por el conjunto de gliders $\mathcal{G}$. Ahora especificamos un subconjunto de expresiones: ϵ, 0, 1, e y $\#_1(\#_2$,f$_i$_1) $\in \Psi_{R110}$ son expresiones regulares bajo las reglas básicas.

Entonces tenemos dos maneras de producir choques entre gliders:

- El primer caso es fijando las fases iniciales en dos gliders g_i, $g_j \in \mathcal{G}$ donde $i \neq j$ y ambos tienen p^+ y p^- diferentes, entonces el intervalo establecido entre ambos gliders es determinado por una secuencia del éter e y con esta condición podemos enumerar todos los choques binarios posibles cambiando únicamente el intervalo del éter, es decir,

[5]De aquí en adelante la fase representada para el éter 'e(f$_1$)' simplemente la representamos como 'e' porque nunca cambia.

manipulando la distancia. Por lo tanto, tenemos que todos los choques entre dos gliders es determinado por la expresión: e^{+}-g_i-e^{*}-g_j-e^{+}. La restricción es que enumera todos los choques únicamente entre dos gliders y no entre paquetes de gliders.

- El segundo caso puede codificar varios gliders iguales o diferentes a la vez, donde la fase y distancia son manipulados específicamente, es decir, podemos cambiar ambos parámetros para obtener un choque en el tiempo y lugar deseado. La ventaja de utilizar este caso, es que el control es total en el espacio de evoluciones, pero la desventaja es que para determinar el choque tenemos que evaluar la producción de la condición inicial para determinar la distancia y fase adecuada para producir algún choque deseado. Es decir, tenemos que construir la codificación paso a paso por aproximación, pero al final veremos que es la mejor opción porque varios choques deben ser codificados cambiando la fase, pero no la distancia (ejemplificamos esta situación en la siguiente sección, cuando aplicamos el procedimiento para construir condiciones iniciales específicas y queremos resolver un problema en particular).

Por ejemplo, si queremos producir un glider a través de choques entre gliders, tenemos que la velocidad de cada glider involucrado es diferente. Por lo tanto, si el choque es simultáneo tenemos que determinar primero la distancia y después la fase adecuada entre ellos para obtener la reacción deseada en el lugar deseado.

A continuación presentamos una serie de pasos para construir condiciones iniciales en la Regla 110 involucrando varios gliders. Debemos señalar que se procede por aproximaciones sucesivas.

1. Determinar el número de gliders g_i donde $i \in \mathbb{Z}^{+}$ y $g \in \mathcal{G}$, que se desean en el proceso.
2. Determinar la fase f_i_1 donde $1 \leq i \leq 4$, en que los gliders deben iniciar.
3. Determinar la distancia para el éter e entre cada glider (si es necesaria).
4. Ejecutar la codificación asignada para evaluar la producción. Si la producción es correcta, termina la asignación. Si no es el caso, entonces:
   (a) Si la distancia es correcta pero la fase no es la adecuada, entonces se realiza una búsqueda recorriendo todas las j-ésimas fases de $\#_2$, donde $1 \leq j \leq p_g$ y p_g es el número de fases posibles determinado por el número de márgenes $ems+oms = p_g$ en un glider g de período j.

(b) Si la fase es correcta pero la distancia no es la adecuada, entonces es necesario calcular el número de configuraciones $ne + \#_1(\#_2,\mathrm{f}_i\_1)$ (mod 4 ó mod 14), determinando si es necesario asignar configuraciones éter o únicamente con la fase de la estructura.

Si la distancia es menor a mod 4 (0, 1, 2 ó 3 mosaicos T_3), entonces no es necesario introducir una secuencia e. En este caso es necesario ajustar la distancia a través de las i-fases f_i_1 del glider g. Si la distancia es mod 14 (4, 4, 4 ó 2 número de células), también sigue el criterio anterior y regresamos el paso 4.

Como mencionamos en la introducción de conceptos fundamentales, nuestro interés no es medir la complejidad en las construcciones que tiene o puede llegar a producir la Regla 110. Nuestro interés es producir comportamientos complejos inferidos desde la interacción de los gliders de la Regla 110 a través de sus choques para algún proceso en particular. La complejidad crece en el espacio de evoluciones de la Regla 110 con respecto al número de gliders involucrados y el tamaño de la configuración inicial, por la cantidad de información contenida en la cadena.

Pero si se desea tener una estimación de la complejidad de una condición inicial para un proceso dado en la Regla 110, basta calcular $I = 2^N$ donde N representa el número de bits en la cadena (pueden utilizarse logaritmos para simplificar la medida o graficar estadísticas). El resultado indica el número de cadenas que se pueden producir en el espacio de evoluciones, estimando su complejidad para el conjunto de todas las cadenas. De esta manera, la complejidad de un autómata celular crece exponencialmente [46].

Finalmente, toda expresión regular w construida a través de fases bajo las reglas de expresiones regulares representa una condición inicial. En la figura 2.18 presentamos el diagrama esquemático para representar choques y fases a través de secuencias periódicas en un espacio de evoluciones.

La relación que existe entre dos ciclos en el diagrama de de Bruijn relaciona dos conjuntos de expresiones regulares. En la figura tenemos que el glider A tiene una conexión con el éter y por otra parte el éter tiene una conexión con el glider B. La secuencia final es una expresión regular asignada en la condición inicial que produce un choque después. El resultado del choque es interpretado en uno o varios gliders δ (no conocemos el resultado y éste puede ser ningún glider). Si los gliders resultantes no chocan, entonces existe otro ciclo en el diagrama de de Bruijn con esos gliders en el espacio. Consecuentemente, tales secuencias están definidas en el conjunto de expresiones regulares de algún glider.

El segundo esquema de la figura representa exactamente lo que es una fase. Por ejemplo, asignamos a la condición inicial una cantidad arbitraria de secuencias éter en ambos extremos

y una fase fija entre los gliders A y B. En consecuencia, una secuencia periódica que está entre secuencias éter y la secuencia no es alterada en su longitud, sino que representará las fases del glider cuando se vaya desplazando a través del espacio de evoluciones.

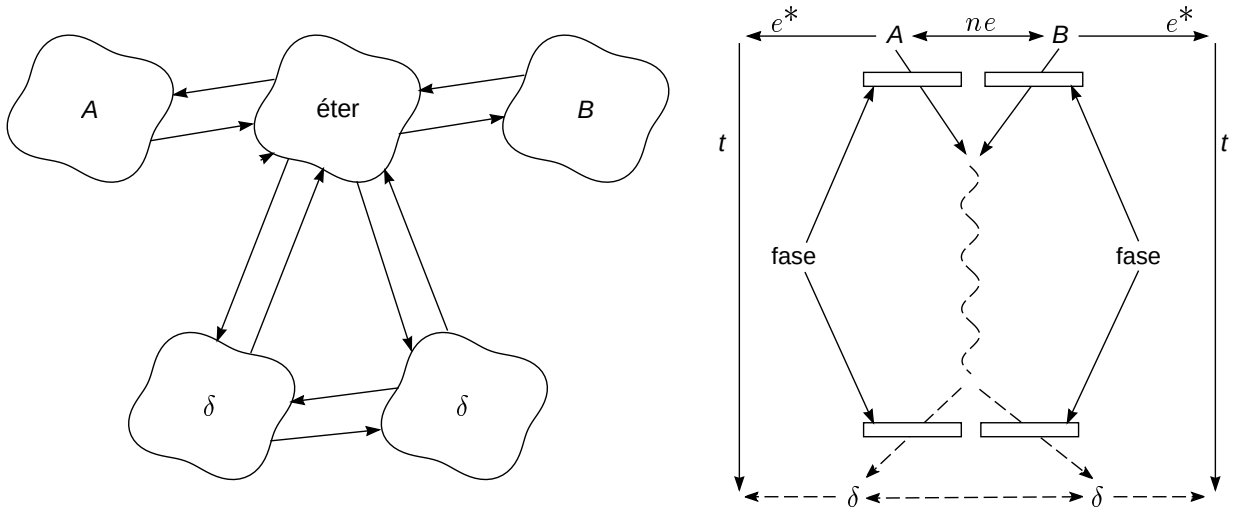

Figura 2.18: Diagrama esquemático representando choques en 1D AC.

Finalmente, determinar que producción final queremos obtener al momento de aplicar el procedimiento puede llevarnos a problemas *indecidibles*, ya que en un momento dado no podemos decidir cuando una serie de composiciones llegarán a la relación deseada. Por otra parte, enumerar todos los choques posibles entre grupos o paquetes de gliders y extensiones de gliders es un problema *intratable*.

2.3 Conclusiones del Capítulo 2

En este capítulo se presentó la estructura básica de la Regla 110, así como sus comportamientos y todos los gliders hasta ahora conocidos. Las fases son explicadas detalladamente, indicándo su origen inducido por el análisis en los diagramas de de Bruijn y mosaicos en la Regla 110. De esta manera, las fases ayudan a determinar una clasificación de secuencias periódicas para obtener el subconjunto de expresiones regulares Ψ_{R110} para todos los gliders (que no era conocido hasta ahora - Apéndice B) y generar el lenguaje regular L_{R110}. Finalmente, una vez obtenido el subconjunto de expresiones regulares proponemos una codificación por fases '$\#_1(\#_2$,f$_i$_1)' para construir condiciones iniciales en el espacio lineal de la Regla 110 y posteriormente determinar el procedimiento para producir choques entre gliders controlando el espacio de evoluciones desde la condición inicial.

Capítulo 3

Aplicando el procedimiento

Ahora debemos aplicar nuestro procedimiento para construir condiciones iniciales a problemas conocidos actualmente en relación a la Regla 110. Consideramos cuatro problemas: la producción de gliders, triángulos grandes, el sistema tag cíclico y operaciones lógicas, todos ellos basados en choques. Algunos de estos problemas pueden ser consultados en otras reglas de evolución en [44, 1, 2, 14].

3.1 Produciendo gliders

Un problema interesante en la Regla 110 es determinar si todos los gliders pueden ser producidos por algún choque (propiedad de cerradura para $\mathcal{G}$). Hasta ahora era desconocido si todos los gliders de la Regla 110 podían ser producidos por algún choque, pero resolvemos este problema encontrando un choque para generar cada glider [22, 23].

Un resultado parcial de nuestra investigación aplicando el procedimiento es que la lista de gliders propuesta por Cook [9, 10] puede ser reproducida por al menos un choque [22, 23]. Se toma como punto de partida todas las producciones binarias [19]. De ahí se tomaron todos los choques binarios que producen un glider en particular. Esta examinación exahustiva demuestra que los gliders D_2, $\bar{B}^n$, $\hat{B}^n$, H y glider gun, no pueden ser obtenidos a través de un choque binario, ya que son producto de múltiples choques.

Los gliders que surgen de manera natural (con mayor frecuencia desde condiciones aleatorias) en el espacio de evoluciones no fueron difíciles de obtener una vez que se realizaron todas las producciones binarias. Cook afirma[1] que no todos los gliders pueden producirse por algún choque, señalando específicamente los gliders $\bar{B}^{n\geq 2}$, $\hat{B}^{n\geq 1}$, y glider gun. Por tanto, extensiones raras efectivamente son difíciles de obtener, pero algunas sí pueden ser reproducidas

[1] Comunicación personal, 2002.

a través de choques. En estos casos se realizaron búsquedas especializadas para cada glider.

En la figura 3.1 se ilustran las producciones para obtener cada uno de los gliders. En el caso del glider $\hat{B}$ el choque es complicado y la sincronización en que deben de llegar cada una de sus partes es única, es decir, no existe otra posibilidad. Como debe ser en un sistema complejo, no es posible predecir su comportamiento desde condiciones iniciales.

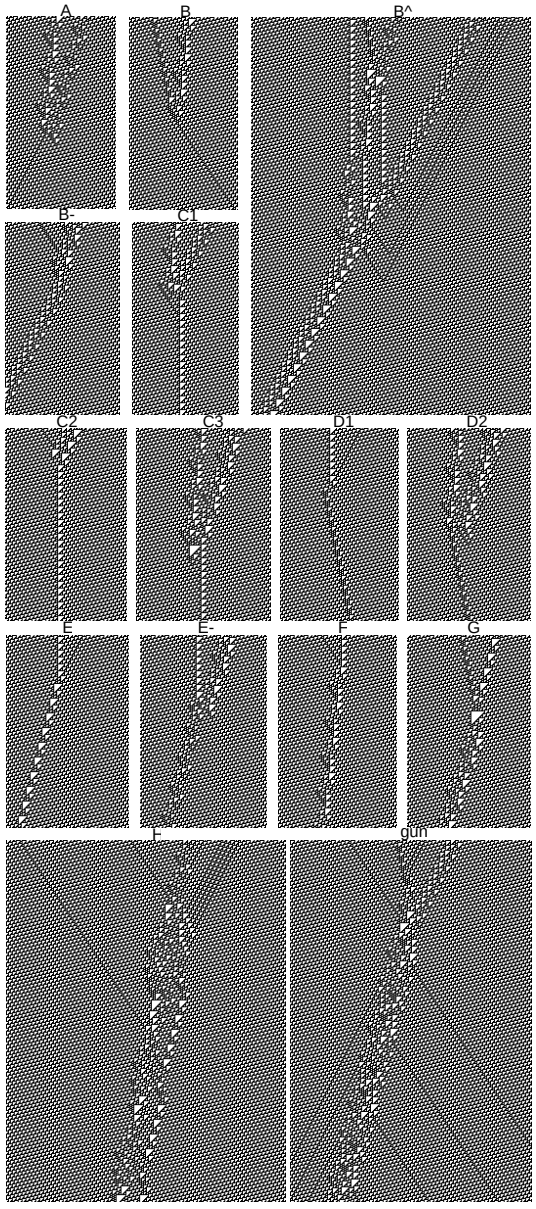

Figura 3.1: Produciendo gliders a través de choques en la Regla 110.

Un punto importante en la producción del glider $\hat{B}$ es que la sucesión de choques debe ser en orden de velocidades para tener un choque propio. Por ejemplo, el glider C_2 tiene una

velocidad de 0 y el glider D_1 de 1/5. Esto implica la existencia del glider D_1 precedida de un glider C_2.

El glider A es calculado con los gliders $\bar{E}$ y F, donde los dos gliders en la mayoría de sus choques producen el fenómeno solitón [41], aunque esto no implica que sean los únicos gliders en la Regla 110 que pueden simular solitones [24].

En la producción del glider G se puede ver que el glider surge desde un mosaico T_{13} aislado. Otra pregunta interesante es conocer si todos los gliders pueden surgir desde un mosaico dado. Por ejemplo, el mosaico T_{10} induce un glider $\bar{E}$, el triángulo T_8 un glider E y el triángulo T_1 los gliders A y B. De esta manera, otro problema abierto es determinar una función suryectiva o biyectiva, tal que $f : T_n \to \mathcal{G}$.

Además, la existencia del glider gun es importante en varios aspectos. El primero es que representa directamente el crecimiento ilimitado en el espacio de evoluciones de la Regla 110 y el segundo es la capacidad que tiene la regla de producir sus propios elementos. En este caso el glider gun surge con más frecuencia que los gliders $\hat{B}$ y H. Sin embargo, la rápida interacción con otras estructuras o regiones caóticas evita una buena formación.

glider	código en fases
A	F(G,f$_1$_1)-e-$\bar{E}$(C,f$_1$_1)
B	D_2(A,f$_1$_1)-e-F(A,f$_1$_1)
$\bar{B}$	A^2(A,f$_1$_1)-e-G(A2,f$_1$_1)
$\hat{B}$	C_2(B,f$_1$_1)-D_1(A,f$_1$_1)-e-$\bar{E}$(B,f$_1$_1)-4e-$\bar{B}$(C,f$_2$_1)-3B(f$_1$_1)
C_1	F(A2,f$_1$_1)-e-$\bar{B}$(A,f$_1$_1)
C_2	A(f$_1$_1)-e-$\hat{B}$(B,f$_1$_1)
C_3	F(G,f$_1$_1)-e-G(E,f$_1$_1)
D_1	C_2(A,f$_1$_1)-e-B(f$_1$_1)
D_2	F(G,f$_1$_1)-e-G(B,f$_1$_1)-B(f$_1$_1)
E	C_3(A,f$_1$_1)-e-B(f$_1$_1)
$\bar{E}$	C_3(A,f$_1$_1)-e-G(E,f$_1$_1)
F	A(f$_1$_1)-e-C_1(A,f$_1$_1)
G	D_2(A,f$_1$_1)-e-E(D,f$_1$_1)
H	A(f$_1$_1)-7e-A(f$_3$_1)-3e-$\bar{E}$(A,f$_1$_1)-B(f$_1$_1)-e-5B(f$_4$_1)
glider gun	A(f$_1$_1)-3e-D_1(C,f$_1$_1)-e-2B(f$_1$_1)-e-$\bar{B}$(f$_1$_1)

Tabla 3.1: Código en fases para producir cada glider de la Regla 110.

La idea de iniciar con los choques binarios en parte fue porque es lo más práctico, aunque esto no quiere decir que es la única manera de generar un glider en particular. Finalmente, la lista de gliders conocidos hasta ahora en la Regla 110 son producidos por al menos un choque [22, 23]. El código en fases para cada producción es presentado en la tabla 3.1 (es entendido que en los extremos de cada expresión concatenamos la cadena e^+).

3.2 Produciendo triángulos grandes

McIntosh plantea dos problemas importantes en el estudio de la Regla 110: el primero es que la Regla 110 es un problema de cubrir el espacio de evoluciones con mosaicos y el segundo es determinar cuál es el mosaico más grande que puede existir en el espacio de evoluciones de la Regla 110 y determinar además si éste es producto de un choque [33].

Búsquedas especializadas establecen condiciones iniciales para construir los mosaicos T_{43}, T_{44} y T_{45}, buscando ancestros para cada mosaico en particular, determinando que no es posible construir mosaicos mayores a $T_{n>45}$ en el espacio de evoluciones de la Regla 110. Este límite fue establecido por Seck Tuoh en agosto del 2001 [33]. Posteriormente, en diciembre del 2002, Cook pudo construir mosaicos cuyo tamaño se encuentra en el intervalo $T_{27\leq n\leq 33}$ a través de secuencias no periódicas en la condición inicial. Cada secuencia determina un mosaico en particular y en algunos casos el cambio de uno o dos bits en la cadena produce un mosaico de tamaño menor o igual al establecido.

Nuestra contribución en este problema es la producción de los mosaicos T_{24}, T_{25}, T_{26}, T_{28}, T_{29} y T_{30} a través de choques entre gliders. Los mosaicos de tamaño menor de 21 son más frecuentes y no es difícil construirlos o inclusive encontrarlos en el espacio de evoluciones a partir de condiciones iniciales aleatorias. En la figura 3.2 presentamos una producción para cada mosaico y en la tabla 3.2 su código en fases para producirlos.

mosaico	código en fases
T_{24}	C_3(B,f$_1$_1)-C_2(A,f$_1$_1)-C_2(A,f$_1$_1)-e-G(A,f$_1$_1)-G(C2,f$_1$_1)
T_{25}	D_2(B,f$_2$_1)-D_2(A,f$_4$_1)-5e-E(B,f$_1$_1)-11B(f$_1$_1)
T_{26}	C_1(A,f$_1$_1)-2e-F(A,f$_1$_1)-e-E(D,f$_1$_1)-2B(f$_1$_1)-2e-6B(f$_4$_1)
T_{28}	C_1(B,f$_1$_1)-C_1(A,f$_4$_1)-C_1(A,f$_1$_1)-C_1(B,f$_1$_1)-e-$\bar{E}$(D,f$_1$_1)-2e-B(f$_3$_1)
T_{29}	A^5(f$_1$_1)-6e-F(B,f$_1$_1)-F(G,f$_1$_1)-B(f$_4$_1)-G(F,f$_4$_1)
T_{30}	A(f$_1$_1)-e-A^5(f$_1$_1)-6e-F(B,f$_1$_1)-F(G,f$_1$_1)-B(f$_4$_1)-G(F,f$_4$_1)

Tabla 3.2: Código en fases para producir grandes mosaicos T_n.

Para obtener un mosaico T_{26} necesitamos un glider C_1, un glider F, un glider E^3 construido con un glider E y dos gliders B, y por último un paquete de $6B$ gliders. En este caso, como en varios otros problemas, el cambio de una estructura aunque sea por un índice cambia totalmente la producción final. Por ejemplo, si el glider C_1 es cambiado por otro glider C_2 o C_3, no se produce el mosaico T_{26}; sólo en casos aislados ayuda a obtener un mosaico más grande. Para obtener un mosaico T_{28} necesitamos cuatro gliders C_1 espaciados, un glider $\bar{E}$ y un glider B para controlar la región caótica de la derecha y generar el mosaico. Si en este caso el glider B no está o no llega en la fase que se necesita, el mosaico T_{28} desaparece completamente, es decir, ni siquiera se produce un mosaico cercano.

Para obtener el mosaico T_{29} necesitamos de un paquete de gliders A^5 que chocan con dos gliders F juntos, pero antes, un glider B interviene con el segundo glider F y al final, un glider G determina el margen derecho que produce el T_{29}.

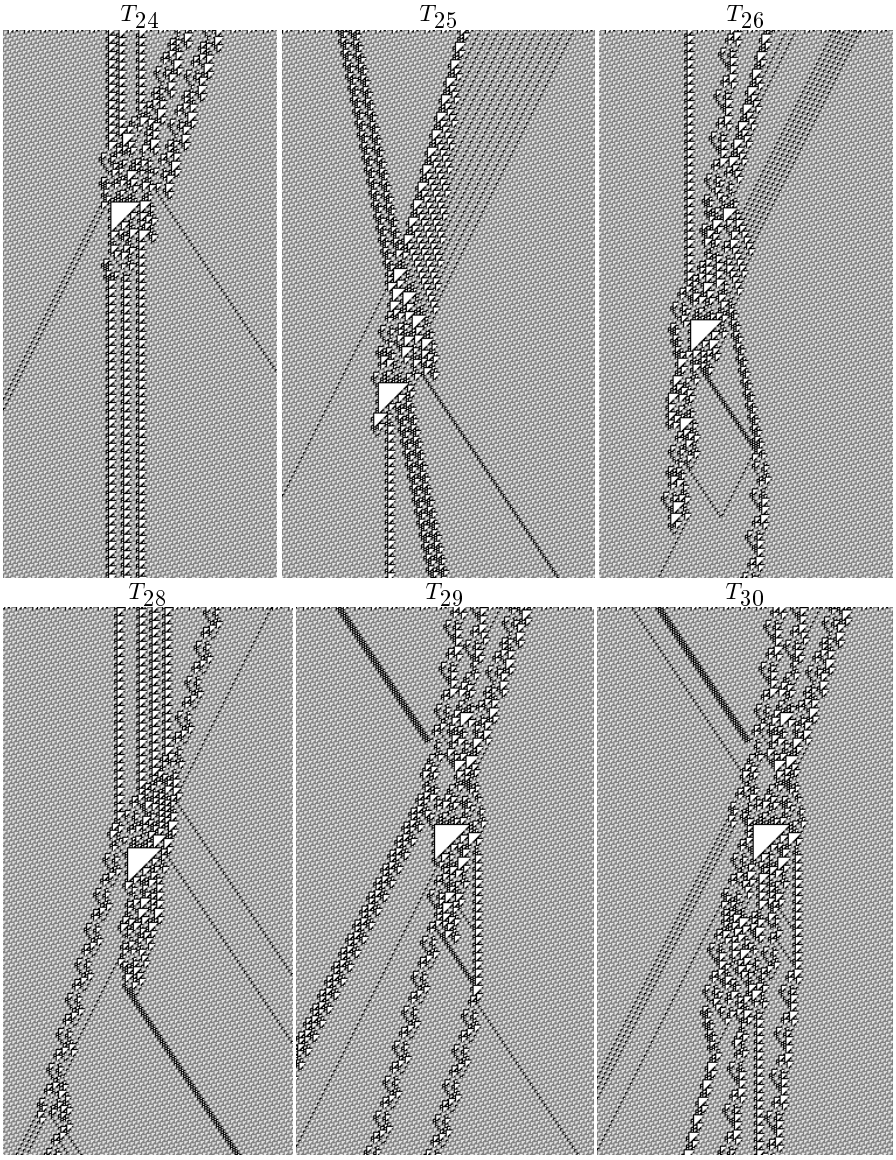

Figura 3.2: Produciendo grandes mosaicos a través de choques.

Debemos mencionar que el mosaico T_{29} fue encontrado en 276 generaciones y al momento de construir su condición inicial un glider fuera de fase produjo accidentalmente el mosaico T_{30}. La secuencia para reproducir el mosaico T_{30} es la misma para el mosaico T_{29}. Sólo es necesario agregar un glider A en la izquierda. Algunas variantes en esta expresión pueden producir los mosaicos T_{16}, T_{20} y T_{22}.

Sin embargo, en los casos de los mosaicos T_{29} y T_{30} se puede decir que tenemos un choque no propio, porque existe un glider B entre un glider F y G que son más lentos. No obstante, habría que determinar primero si existe una producción que origine los gliders B y G para obtener un choque propio.

Finalmente, la producción para los mosaicos T_{27} y $T_{31\leq n\leq 33}$ a través de un choque se desconoce en la actualidad, al igual que el intervalo entre los mosaicos $T_{34\leq n\leq 42}$.

3.3 Reproduciendo el sistema tag cíclico codificado en fases

Ahora presentamos la construcción de la condición inicial para simular el funcionamiento del sistema tag cíclico en el espacio de evoluciones de la Regla 110 a través de fases.

La Regla 110 ganó popularidad a finales de los 90's cuando Cook demostró que dicho autómata es universal [10, 53], simulando el funcionamiento de un sistema tag cíclico (una variante del sistema tag [39]) a través de choques entre grandes bloques de gliders. AC universal ha sido desarrollado y analizado desde sus inicios con von Neumann [46] logrando varias simplificaciones a través de la historia como puede verse en [8, 6, 28, 10].

Antes de continuar debemos señalar que nuestro interés no es discutir la universalidad de la Regla 110, sino construir cada uno de sus componentes y de esta manera la condición inicial a través de fases a fin de reproducir el funcionamiento del sistema tag cíclico en la Regla 110 [21].

Un sistema tag es un sistema canónico normal de Post, donde la máquina es definida por una serie de producciones que determinan las operaciones de borrar y agregar datos a una cinta. Se evalúan los primeros elementos de la cadena, donde al aplicar las producciones iterativamente el sistema puede entrar en un ciclo que representa la solución del problema. El sistema tag es una variante de la máquina de Turing [35] que se usa para realizar procesos computables y cuya universalidad fue demostrada por John Cocke y Marvin Minsky en [35]. A diferencia del sistema tag, el sistema tag cíclico tiene dos reglas de transformación para un mismo valor y éstas se aplican en cada paso periódicamente. Cook propone un par de producciones bases ($0 \to \epsilon$, $1 \to 11$) y ($0 \to \epsilon$, $1 \to 10$), donde ϵ representa la palabra vacía, que se aplica en la simulación con la Regla 110.

En la figura 3.3 se muestra el funcionamiento del sistema tag cíclico en el espacio de evoluciones de la Regla 110 a través de un diagrama esquemático, donde cada línea representa un bloque compuesto por varios gliders [10, 53]. En términos generales tenemos tres partes

importantes en el espacio de evoluciones: los bloques de gliders llegando de la izquierda representan los operadores, los bloques de gliders llegando de la derecha representan los datos que serán transformados y los bloques de gliders centrales ilustrados como columnas representan los datos en la cinta del sistema tag cíclico.

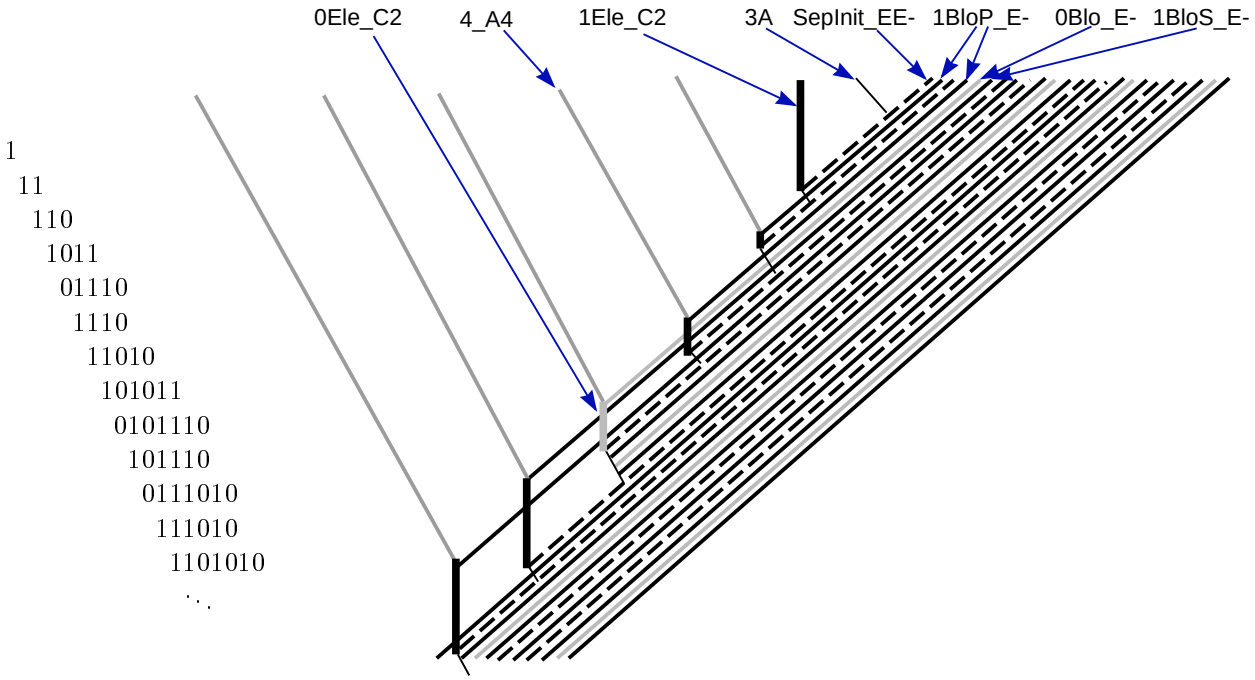

Figura 3.3: Representación del funcionamiento del sistema tag cíclico y su diagrama esquemático especificando cada uno de los componentes necesarios representados por bloques de gliders en la Regla 110.

El diagrama esquemático es muy útil porque nos permite discutir la estructura y funcionamiento de cada componente sin la necesidad de graficarlo. Un problema es representar gráficamente cada bloque de gliders y más complicado es mostrar la evolución del sistema tag cíclico, porque se necesitan millones de células y este proceso es poco entendible en espacios reducidos. Por esa razón, especificamos detalladamente la construcción por fases para cada bloque sin graficarlo.

A continuación explicamos el funcionamiento de cada uno de los componentes en el sistema tag cíclico y su construcción por fases. Si el sistema empieza con un 1 en la cinta, entonces, en un paso, la máquina debe preparar los datos que vienen de la derecha sin transformar, al momento de identifcar el valor actual de la cinta. Si el sistema empieza con un 0 en la cinta, entonces, en un paso, la máquina debe borrar el siguiente bloque de datos que vienen de la derecha.

Al tratar de reproducir esta operación con el mecanismo de gliders de la Regla 110, la operación se cumple porque independientemente de cómo estén estructurados los bloques de datos, éstos no pueden producir ningún elemento en la cinta si no encuentran un valor de 1.

Antes de continuar debemos nombrar cada componente para identificarlos mejor. Cada uno de ellos es identificado en la tabla 3.3.

componente	representación
operadores	4_A^4
datos	1Ele$_C_2$ 0Ele$_C_2$
datos sin transformar	1BloP$_\bar{E}$ 1BloS$_\bar{E}$ 0Blo$_\bar{E}$
datos pre-transformados	1Add$_\bar{E}$ 0Add$_\bar{E}$
guía	SepInit$_E\bar{E}$

Tabla 3.3: Lista de componentes necesarios para construir el funcionamiento del sistema tag cíclico en el espacio de evoluciones de la Regla 110.

La construcción del sistema tag cíclico en la Regla 110 puede dividirse en tres partes principales:

1. La parte periódica de la izquierda: es controlada por los bloques de gliders 4_A^4. Esta parte debe ser estática (en sus repectivos bloques y distancias), porque es la que controla la producción de la información, es decir, son los operadores.

2. La segunda es el centro: que determina el valor donde debe de iniciar el sistema y representa los valores que son introducidos en la cinta, es decir, son los datos.

3. La tercera es la parte cíclica de la derecha: que tiene los datos a procesar representados por bloques de gliders E's, agregando un componente guía[2] que determinará si agrega o borra datos en la cinta, es decir, datos sin procesar.

En la parte izquierda existen 4 paquetes de gliders A^4 que representan un bloque y deben ser iniciados cuidadosamente (como cada uno de los componentes), porque aunque son estáticos tienen sutilezas que deben ser discutidas. El punto delicado para hacer funcionar estos componentes es definir bien sus distancias y fases, porque una fase o distancia diferente

[2]El componente guía es un separador o identificador que determina si son introducidos nuevos datos a la cinta o no. Además, borra cada dato de la cinta.

induce un choque no deseado, es decir, destruye toda la maquinaria. Todos los componentes a su vez siguen esta restricción porque cada uno de los gliders de cada componente deben estar alineados con todos los demás. Para los paquetes de gliders A^4 que vienen de la izquierda hemos comprobado que la distancia es estática entre grupos de ellos, pero sus fases no.

El paquete de gliders A^4 tienen tres fases f_1_1, f_2_1 y f_3_1 (heredado por el glider A). Para definir el primer bloque de gliders 4_A^4 es necesario conocer en que fase debe iniciar cada grupo de gliders A^4. De esta manera, para construir el primer bloque de gliders 4_A^4 se asignan las siguientes fases: $A^4(f_3\_1)$-$27e$-$A^4(f_2\_1)$-$23e$-$A^4(f_1\_1)$-$25e$-$A^4(f_3\_1)$.

Las distancias éter entre cada bloque de gliders 4_A^4 no cambian, son estáticas, pero las fases tienen que rotar para obtener el choque solitón con los gliders $\bar{E}$ que se van generando en el sistema. Entonces la rotación en cada grupo de gliders A^4 también se proyecta al bloque de gliders 4_A^4 y se hace periódica de la siguiente manera: $\{649e$-$\{A^4(f_2\_1)$-$27e$-$A^4(f_1\_1)$-$23e$-$A^4(f_3\_1)$-$25e$-$A^4(f_2\_1)\}$-$649e$-$\{A^4(f_1\_1)$-$27e$-$A^4(f_3\_1)$-$23e$-$A^4(f_2\_1)$-$25e$-$A^4(f_1\_1)\}$-$649e$-$\{A^4(f_3\_1)$ -$27e$-$A^4(f_2\_1)$-$23e$-$A^4(f_1\_1)$-$25e$-$A^4(f_3\_1)\}\}^*$.

Simplificando, si representamos una rotación de fases f_i_1 por P_i para cada bloque de gliders 4_A^4 entonces podemos renombrarlo como: $\{649e$-$4\_A^4(P_2)$-$649e$-$4\_A^4(P_1)$-$649e$-$4\_A^4(P_3)\}^*$. Finalmente, extendemos la representación por fases a bloques o paquetes de gliders, originando una fase para varios gliders involucrados (una meta-fase).

Las distancias entre cada bloque de gliders 4_A^4 fue el último punto construido porque las distancias cortas no permitían una buena construcción. La razón es porque los gliders A viajan más rápido que los gliders E y $\bar{E}$. En consecuencia, los gliders A alcanzan a los datos formados por los gliders C_2 antes de que lleguen los nuevos bloques de datos. Por lo tanto, para nuestro análisis necesitamos al menos 649 configuraciones éter.

La parte central de la condición inicial es determinada por un dato inicial en la cinta que es un 1 y es representado por un paquete de cuatro gliders C_2.

La parte derecha es representada por los bloques de datos sin procesar por los gliders $E^{n=1,4,5}$ y $\bar{E}$. A continuación describimos los bloques que representan datos y el guía. El elemento 1Ele_C_2 representa un 1 en la cinta y es formado por cuatro gliders C_2 espaciados. La secuencia en fases para representar este bloque es: C_2(A,f_1_1)-$2e$-C_2(A,f_1_1)-$2e$-C_2(A,f_1_1)-e-C_2(B,f_2_1). Simplificando, las fases del bloque anterior es representado por: $1\text{Ele}\_C_2(\#_2,f_i\_1)$.

El elemento 0Ele_C_2 representa un 0 en la cinta y es formado por cuatro gliders C_2 espaciados. La diferencia con el elemento anterior es la distancia entre los gliders centrales ($9T_3$-$5T_3$-$7T_3$) y el tercer glider C_2 inicia en otra fase. Por lo tanto, las diferencias en ambos bloques son importantes y se determinan por sus distancias y fases dentro de cada bloque.

El elemento $0\text{Blo}\_\bar{E}$ está formado por 12 gliders $\bar{E}$ y representa un dato con valor 0

sin transformar. Estos deben transformarse cuando chocan contra los gliders C's intermedios (estos gliders C's no son los datos, son otros que se producen durante el proceso). El bloque 0Blo_$\bar{E}$, al transformarse produce el bloque 0Add_$\bar{E}$. La expresión por fases para construir este bloque es: $\bar{E}$(H,f$_1$_1)-2e-$\bar{E}$(C,f$_1$_1)-$\bar{E}$(F,f$_1$_1)-$\bar{E}$(H,f$_2$_1)-2e-$\bar{E}$(A,f$_1$_1)-2e-$\bar{E}$(H,f$_1$_1)-e-$\bar{E}$(C,f$_3$_1)-2e-$\bar{E}$(G,f$_1$_1)-$\bar{E}$(A,f$_4$_1)-$\bar{E}$(C,f$_4$_1)-e-$\bar{E}$(D,f$_3$_1)-e-$\bar{E}$(C,f$_3$_1). La fase y distancia entre cada uno de ellos y en general en todos los componentes tiene que ser exacta, de lo contrario el sistema se destruye por completo.

Para generar un 1 en la cinta es necesario tener dos bloques (lo que no sucede para el bloque 0): componente *primario*-1BloP_$\bar{E}$ y componente *estándar*-1BloS_$\bar{E}$ (el componente primario es ignorado completamente en [53]), donde ambos bloques están formados por 12 gliders $\bar{E}$. El elemento primario se diferencia del estándar porque las distancias de los primeros dos gliders $\bar{E}$ es diferente para cada bloque. Ambos bloques producen el bloque 1Add_$\bar{E}$ aunque en diferentes intervalos.

La razón para utilizar dos bloques que generan un 1 es porque la Regla 110 es asimétrica. Por ejemplo, si sólo se utiliza uno de los dos elementos entonces se obtiene una buena producción en los choques que generan el elemento 1Add_$\bar{E}$, pero al llegar el segundo bloque de gliders 4_A^4 el sistema se descuadra destruyéndolo desde el primer choque.

La secuencia en fases para construir el elemento primario es: $\bar{E}$(H,f$_1$_1)-$\bar{E}$(A,f$_3$_1)-e-$\bar{E}$(A,f$_4$_1)-$\bar{E}$(C,f$_4$_1)-e-$\bar{E}$(D,f$_3$_1)-e-$\bar{E}$(C,f$_3$_1)-2e-$\bar{E}$(G,f$_3$_1)-e-$\bar{E}$(B,f$_3$_1)-$\bar{E}$(E,f$_2$_1)-e-$\bar{E}$(G,f$_4$_1)-e-$\bar{E}$(H,f$_3$_1)-e-$\bar{E}$(G,f$_3$_1). La secuencia en fases para construir el elemento estándar es: $\bar{E}$(H,f$_1$_1)-$\bar{E}$(A,f$_3$_1)-e-$\bar{E}$(A,f$_4$_1)-$\bar{E}$(C,f$_4$_1)-e-$\bar{E}$(D,f$_3$_1)-e-$\bar{E}$(C,f$_3$_1)-$\bar{E}$(E,f$_3$_1)-2e-$\bar{E}$(A,f$_1$_1)-$\bar{E}$(C,f$_4$_1)-$\bar{E}$(F,f$_1$_1)-2e-$\bar{E}$(G,f$_1$_1)-2e-$\bar{E}$(E,f$_3$_1).

El elemento guía SepInit_$E\bar{E}$ es formado por 5 gliders $\bar{E}$ y los gliders $E^{n=5,2,4}$. El elemento guía determina si se agregan más datos a la cinta o son borrados los datos sin transformar dependiendo del valor que encuentre en la cinta; además, borra los datos de la cinta. La secuencia en fases para construir el elemento guía es: E^5(C,f$_1$_1)-e-E^2(B,f$_1$_1)-3e-E^4(C,f$_1$_1)-e-$\bar{E}$(D,f$_4$_1)-e-$\bar{E}$(F,f$_1$_1)-2e-$\bar{E}$(G,f$_2$_1)-$\bar{E}$(F,f$_2$_1)-2e-$\bar{E}$(E,f$_1$_1).

El elemento guía debe ser muy cuidadoso en su manejo porque define dos fases del mismo índice para el glider E^4 aunque éstos son diferentes, porque el glider E tiene 4 fases f$_1$_1 y el glider $\bar{E}$ tiene 8 fases f$_1$_1. En consecuencia, existen dos períodos diferentes en un mismo intervalo ($2p_E = p_{\bar{E}}$). Por lo tanto, si una fase funciona adecuadamente, la otra no debe funcionar porque los gliders $\bar{E}$ estarán fuera de fase.

Los elementos 1Add_$\bar{E}$ y 0Add_$\bar{E}$ se forman por 4 gliders $\bar{E}$ respectivamente. Ambos son producto de los bloques de datos sin transformar. El elemento 1Add_$\bar{E}$ es producto del bloque 1BloP_$\bar{E}$ ó 1BloS_$\bar{E}$. En la reproducción del sistema tag cíclico el bloque primario y estándar producen el mismo elemento 1Add_$\bar{E}$ que es determinado después de un elemento

guía. En el mismo caso, el elemento 0Add_$\bar{E}$ es producto del bloque 0Blo_$\bar{E}$.

Si el elemento guía encuentra un elemento 1Ele_$\bar{E}$ borra este dato de la cinta y agrega los nuevos bloques de datos sin transformar que serán pre-transformados. Si el elemento guía encuentra un elemento 0Ele_$\bar{E}$ borra este dato de la cinta y borra una serie de bloques de datos sin transformar que vienen de la derecha hasta encontrar un nuevo elemento guía.

Finalmente, un elemento 1Add_$\bar{E}$ se transforma en el elemento 1Ele_$\bar{E}$ al chocar contra un grupo de gliders 4_A^4, lo cual representa introducir un 1 en la cinta. El elemento 0Add_$\bar{E}$ se transforma en el elemento 0Ele_$\bar{E}$ al chocar contra un grupo de gliders 4_A^4, lo cual representa introducir un 0 en la cinta.

componente	distancias
1Ele_C_2	9-9-7
0Ele_C_2	9-5-7
1BloP_$\bar{E}$	4-6-2-8-8-2-10-1-2-8-8
1BloS_$\bar{E}$	10-1-2-8-8-2-10-1-2-8-8
0Blo_$\bar{E}$	10-1-2-8-8-8-10-1-2-8-8
SepInit_$E\bar{E}$	4-14-6 y 7-6-9-2-8
1Add_$\bar{E}$	27-21-27
0Add_$\bar{E}$	27-27-27

Tabla 3.4: Distancias (número de mosaicos T_3) entre gliders en cada uno de los componentes.

En la tabla 3.4 se muestran las distancias entre gliders para cada uno de los componentes. Los valores de la segunda columna indican el número de mosaicos T_3 que existen entre gliders. Diferenciar las distancias es una manera práctica de identificar algún componente en el enorme espacio de evoluciones. Con la construcción de cada uno de los componentes, reconstruimos el funcionamiento del sistema tag cíclico presentado en [53, 21]. Nuestra construcción es especificada detalladamente en cada fase y componente. Por tanto, la dividimos en tres partes:

izquierda: ...-217e-4_A^4(F_2)-649e-4_A^4(F_1)-649e-4_A^4(F_3)-649e-4_A^4(F_2)-649e-4_A^4(F_1)-649e-4_A^4(F_3)-216e-

centro: 1Ele_C_2(A,f_1_1)-e-A^3(f_1_1)-

derecha: SepInit_$E\bar{E}$(C,f_3_1)-1BloP_$\bar{E}$(C,f_4_1)-SepInit_$E\bar{E}$(C,f_3_1)-1BloP_$\bar{E}$(C,f_4_1)-0Blo_$\bar{E}$(C,f_4_1)-1BloS_$\bar{E}$(A,f_4_1)-SepInit_$E\bar{E}$(A2,f_2_1)-1BloP_$\bar{E}$(F,f_1_1)-SepInit_$E\bar{E}$(A2,f_3_1)-1BloP_$\bar{E}$(F,f_1_1)-0Blo_$\bar{E}$(E,f_4_1)-1BloS_$\bar{E}$(C,f_4_1)-e-SepInit_$E\bar{E}$(B2,f_1_1)-1BloP_$\bar{E}$(F,f_3_1)-e-

SepInit_$E\bar{E}$(B2,f$_1$_1)-217e-...

obteniendo la secuencia 1110111 y un separador al final con dos solitones. Finalmente, obtenemos una construcción satisfactoria en 57,400 generaciones en un arreglo de 56,240 células en la configuración inicial, implicando un enorme espacio de evoluciones de 3,228,176,000 células.

La simulación del sistema tag cíclico en la Regla 110 es altamente ineficiente y difícilmente puede ser llevado a una aplicación práctica. Sin embargo resulta de gran interés teórico.

El inicio de la operación del sistema tag cíclico en la Regla 110 es con un 1 en la cinta. Después, el paquete de gliders A^3 debe preparar al elemento guía que viene en la parte periódica de la derecha. Este paquete de gliders A^3 pueden llegar juntos o separados.

La primera transformación borra el 1 de la cinta, después el guía borra los datos de la cinta y si encuentra un 1 en la cinta, entonces prepara los siguientes bloques de datos sin transformar que serán agregados a la cinta. Si encuentra un 0 en la cinta, entonces no permite agregar datos en la cinta hasta encontrar otro guía. Toda pareja de gliders $\bar{E}$ que sobran en el proceso son invisibles para el sistema, porque no intervienen en ninguna operación ni destruyen o alteran la maquinaria. Los gliders $\bar{E}$ que sobran, atraviesan como solitones los paquetes de gliders 4_A^4 y los elementos 1Ele_C_2 y 0Ele_C_2.

A diferencia del caso cuando se borra un 1 de la cinta, es decir, cuando se borra un 0 de la cinta, las distancias son más cortas en los primeros choques y se produce un mosaico T_{14} en el proceso, el más grande utilizado en el sistema tag cíclico. La diferencia de distancias provoca un cambio de fase que permite borrar gliders $\bar{E}$ en lugar de generar gliders C's.

El borrado de bloques sin transformar es con la reacción $A^3 \rightarrow \bar{E}$. Cuando el guía borra un 0 de la cinta que produce dos gliders $\bar{E}$ del tipo solitón tal como sucede cuando se borra un 1 de la cinta, todos los demás bloques de gliders $\bar{E}$ que vienen atrás son borrados hasta encontrar el siguiente guía.

La construcción se repite sucesivamente hasta que la máquina encuentre una secuencia periódica o nunca pare. Concluyendo, en términos de fases, el sistema tag cíclico puede codificarse de la siguiente manera:

izquierda: {649e-4_A^4(F$_i$)}*, donde $1 \leq i \leq 3$ en orden secuencial

centro: 246e-1Ele_C_2(A,f$_1$_1)-e-A^3(f$_1$_1)

derecha: {SepInit_$E\bar{E}$(#$_2$,f$_i$_1)-1BloP_$\bar{E}$(#$_2$,f$_i$_1)-SepInit_$E\bar{E}$(#$_2$,f$_i$_1)-1BloP_$\bar{E}$(#$_2$,f$_i$_1)-0Blo_$\bar{E}$(#$_2$,f$_i$_1)-1BloS_$\bar{E}$(#$_2$,f$_i$_1)}*, donde $1 \leq i \leq 4$ y #$_2$ representa una fase en particular.

3.4 Construyendo operaciones lógicas

McIntosh [32] implementa un dispositivo para realizar incrementos o decrementos a través de choques entre gliders en el espacio de evoluciones de la Regla 110. El objeto es construido a partir de los choques básicos que existen entre los gliders A, B y E^n. Para obtener un glider E^{n+1} el glider E^n debe recibir el impacto de un glider B (incremento) y para obtener un glider E^{n-1} el glider E^n debe recibir el impacto de un glider A (decremento).

Actualmente, encontramos otro tipo de producción que produce el mismo fenómeno entre los gliders B, C_2 y E^n, como podemos ver en la figura 3.4.

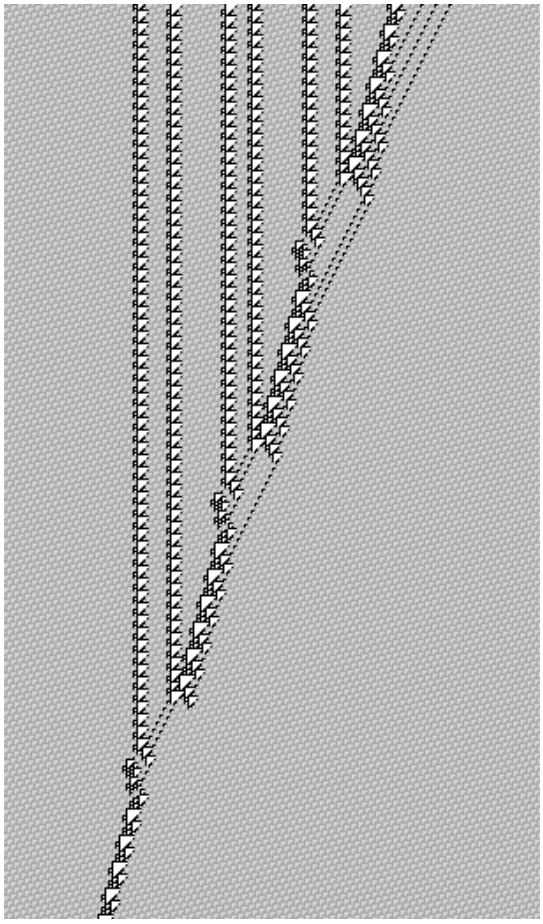

Figura 3.4: Dispositivo implementado en la Regla 110 capaz de realizar incrementos y decrementos basado en choques entre paquetes de gliders.

El funcionamiento es de la siguiente manera: iniciamos con un glider E que es afectado por tres choques contra el glider B produciendo un glider E^4 (incrementos). Después, tres parejas de gliders C_2 disminuyen el índice del glider E^n sucesivamente en E^3 tras el primer

choque, en E^2 después del segundo choque y finalmente en un glider E después del tercer choque. La codificación para este ejemplo puede ser generalizada a través de la siguiente expresión: $\{C_2$(B,f$_1$_1)-C_2(A,f$_1$_1)-e-C_2(A,f$_1$_1)-C_2(B,f$_1$_1)-$e\}$*-E(A,f$_1$_1)-$\{B$(f$_1$_1)$\}$*.

Ahora presentamos la implementación de una compuerta lógica NOT en la figura 3.5 con los gliders A y $\bar{E}$.

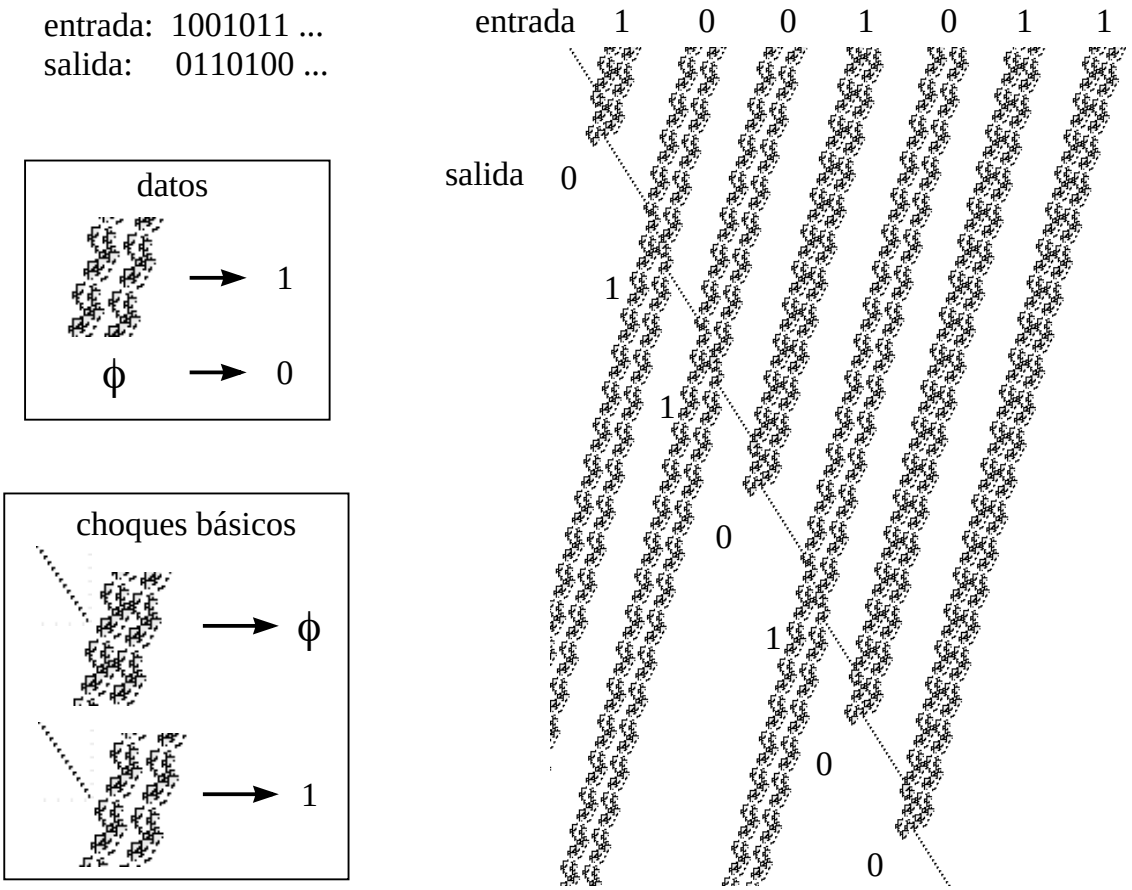

Figura 3.5: Implementando la compuerta NOT en la Regla 110.

La condición inicial es codificada para representar los datos de entrada sin transformar. Después de los choques cada pareja de gliders produce un valor. Si ambos gliders $\bar{E}$ continúan después del choque, entonces representa un 1. En el segundo caso, si ambos gliders $\bar{E}$ son borrados después del choque entonces representa un 0 y de esta manera implementamos una operación lógica NOT en el espacio de evoluciones de la Regla 110. La cadena de entrada es 1001011 y el resultado bajo la operación NOT debe ser 0110100. Finalmente, la codificación en fases para este ejemplo es: A(f$_1$_1)-2e-$\bar{E}$(A,f$_1$_1)-$\bar{E}$(G,f$_4$_1)-2e-$\bar{E}$(A,f$_1$_1)-$\bar{E}$(H,f$_1$_1)-2e-$\bar{E}$(H,f$_1$_1)-$\bar{E}$(G,f$_1$_1)-2e-$\bar{E}$(D,f$_1$_1)-$\bar{E}$(B,f$_4$_1)-2e-$\bar{E}$(D,f$_1$_1)-$\bar{E}$(C,f$_1$_1)-2e-$\bar{E}$(B,f$_1$_1)-$\bar{E}$(H,f$_4$_1)-2e-$\bar{E}$(A,f$_1$_1)-$\bar{E}$(G,f$_4$_1).

La importancia de poder implementar tales dispositivos es para encontrar algún otro mecanismo donde podemos llegar a construir una máquina de Turing, auto-reproducción o el constructor universal en la Regla 110 [24].

3.5 Conclusiones del Capítulo 3

La aplicación del procedimiento para construir condiciones iniciales codificadas en fases, es una herramienta útil y práctica de aplicar para resolver problemas en la Regla 110. Después se atacan cuatro problemas. El primer problema resuelto es la construcción de condiciones iniciales para producir cada uno de los gliders de la Regla 110 a través de choques. El segundo problema es parcialmente resuelto al construir condiciones iniciales para producir diferentes mosaicos grandes a través de choques. El tercer problema fue la reconstrucción de la condición inicial codificada en fases para simular y verificar el funcionamiento del sistema tag cíclico en la Regla 110. El cuarto problema es la implementación de operaciones lógicas: incremento-decremento y compuerta NOT, en la Regla 110.

Capítulo 4

Conclusiones

4.1 Resultados

El resultado y contribución principal de esta investigación es el procedimiento para codificar condiciones iniciales en el espacio lineal de la Regla 110, para determinarlo fue necesario encontrar un lenguaje regular L_{R110} que es construido a través de un subconjunto regular de expresiones regulares Ψ_{R110} (Apéndice B) para todos los gliders de la Regla 110 sin extensiones, donde el lenguaje es conservado en la construcción de las condiciones iniciales. El procedimiento es nuevo en el estudio de AC en una dimensión con comportamiento complejo, porque en la actualidad no se tiene reportado algún otro procedimiento de este tipo y más aún no se tiene reportado la existencia de un lenguaje regular (basado en el sistema de gliders) en algún AC con comportamiento complejo.

El procedimiento toma herramientas ya conocidas para validar sus construcciones a través de los diagramas de de Bruijn, diagramas de subconjuntos, AF, lenguajes, expresiones regulares y mosaicos. El análisis propuesto para determinar el subconjunto de expresiones regulares Ψ_{R110} en todos los gliders de la Regla 110, es el análisis realizado con las fases. De esta manera, el uso de las fases f_i_1 ofrecen una herramienta poderosa para controlar el espacio de evoluciones de la Regla 110 construyendo condiciones iniciales especícas, además de las propiedades y restricciones derivadas del mosaico T_3.

El análisis por fases es implementado en un sistema de computación que llamamos "OS-XLCAU21." El sistema es desarrollado principalmente para manejar o controlar choques entre gliders en la Regla 110 a través de condiciones iniciales construidas con secuencias periódicas del subconjunto de expresiones regulares Ψ_{R110} codificado por las fases. El sistema es construido para un práctico uso sin necesidad de tener experiencia en el manejo de estructuras de la Regla 110. En la actualidad no existe otro sistema equivalente con las

mismas facilidades o características.

Luego aplicamos el procedimiento apoyándonos con el sistema OSXLCAU21[1] para resolver algunos problemas de interés teórico establecidos en la Regla 110.

El primero, es la producción de todos los gliders hasta ahora conocidos de la Regla 110 a través de choques, porque era conocido que no todos los gliders podían ser producto de algún choque. En el AC de 2D de Conway (El Juego de la Vida) existía el mismo problema y después de varios años de estudio por parte de diferentes grupos de investigación se lograron determinar las producciones incluyendo su glider gun [6].

El segundo, es la producción de triángulos grandes iniciada por McIntosh [33] a través de choques. Nuestra contribución es la realización de búsquedas especializadas para cada mosaico y el hallazgo de sus posibles producciones. De esta manera, logramos obtener resultados nuevos en la producción de mosaicos de tamaño: T_{24}, T_{25}, T_{26}, T_{28}, T_{29} y T_{30}, con su respectivo código expresado en fases.

El tercero, es la reconstrucción de los componentes para simular el funcionamiento del sistema tag cíclico [10, 53]. La construcción se logra reconstruyendo cada uno de los componentes a través de las fases [21]. Posteriormente, con los componentes determinados se extendió la codificación de fases a bloques de gliders. Con ésto se logró una construcción satisfactoria con nuestra condición inicial codificada en fases y validando el funcionamiento del sistema tag cíclico en la Regla 110.

El cuarto, es la implementación de operaciones lógicas a través de una compuerta NOT y un dispositivo con la capacidad para realizar incrementos o decrementos, ambos basados en choques.

4.2 Limitaciones

En esta investigación encontramos varias limitaciones que pueden ser discutidas. La construcción de condiciones iniciales a través de las fases es por aproximación, si ésta es para algún propósito en especial. En otro caso, entonces la construcción de todas las condiciones iniciales es un problema combinatorio que involucra al conjunto de expresiones regulares, otra alternativa es aplicar inteligencia artificial o algoritmos genéticos.

Otro problema es que los resultados obtenidos hasta este momento no fueron formalizados teóricamente, principalmente en las propiedades derivadas a través de los mosaicos T_3 que son encontradas en todos los patrones que existen en el espacio de evoluciones de la Regla

[1] Las características y funcionamiento de cada una de sus partes del sistema OSXLCAU21 son explicadas y presentadas detalladamente en el Apéndice A.

110. Sin embargo, los diagramas de de Bruijn [32], la teoría de mosaicos [15], la teoría de conjuntos [47] y la teoría de lenguajes [18]; marcan la dirección para tratar de llegar a dicha formalización en más de un camino.

Finalmente, la proyección de fases (meta-fase) a extensiones de gliders o grandes bloques de gliders es posible en algunos casos. Sin embargo, el conjunto de expresiones regulares crece drásticamente y por ahora es imposible determinar algún criterio para limitar dicho conjunto, pero su creación debe ayudar a asignar fácilmente dichas secuencias periódicas en la construcción de grandes condiciones iniciales para la Regla 110.

4.3 Trabajo por realizar

La investigación que debe realizarse es direccionada para varios problemas en particular.

- Formalizar teóricamente las propiedades de las fases para determinar sus implicaciones en espacios no finitos.
- Proyectar el funcionamiento del procedimiento a otro AC con comportamiento complejo en una dimensión y posteriormente a n-dimensión.
- Demostrar formalmente los límites de velocidad que cualquier estructura pueda tener y demostrar que no existen más gliders u otras extensiones en el espacio de evoluciones de la Regla 110.
- Demostrar que para los gliders $\bar{B}^n \ \forall \ n \geq 3$ y $\hat{B}^n \ \forall \ n \geq 2$, éstos pueden ser producidos a través de choques.
- Demostrar la construcción de los mosaicos T_{27} y $T_{31 \leq n \leq 42}$ a través de choques o secuencias.
- Demostrar que la universalidad de la Regla 110 puede ser construida con otro sistema equivalente Turing o mejor aún, con la misma máquina de Turing.
- En el caso del sistema tag cíclico, implementar algunas operaciones como la secuencia Fibonnaci realizada por Paul Chapman,[2] el balanceo de paréntesis, la suma de números, contadores binarios, etc.

[2]Comunicación personal, Enero del 2003.

- Proyectar el funcionamiento de la Regla 110 en un AC de dos y tres dimensiones, con la finalidad de ayudar a encontrar algunas propiedades que no sean identificadas en una dimensión. En el caso tridimensional debe ser interesante ver el espacio de evoluciones cubierto por tetrahedros determinando el éter y los gliders que ahí puedan existir.

- Realizar la programación de una cuadrícula que permita llenar el espacio de evoluciones con cualquier mosaico T_n sin violar la Regla 110, es decir, actualizando el espacio en todas direcciones cuando se introduce un nuevo elemento.

- Realizar la programación de diagramas de de Bruijn extendidos para más de 10 generaciones.

Finalmente, varios problemas importantes en la Regla 110 continúan abiertos. De entre ellos, destacan tres: demostrar que la Regla 110 es intrínsicamente universal, demostrar que la Regla 110 es capaz de construir su constructor universal y demostrar que la Regla 110 es capaz de soportar auto-reproducción.

Apéndice A

Aplicación OSXLCAU21

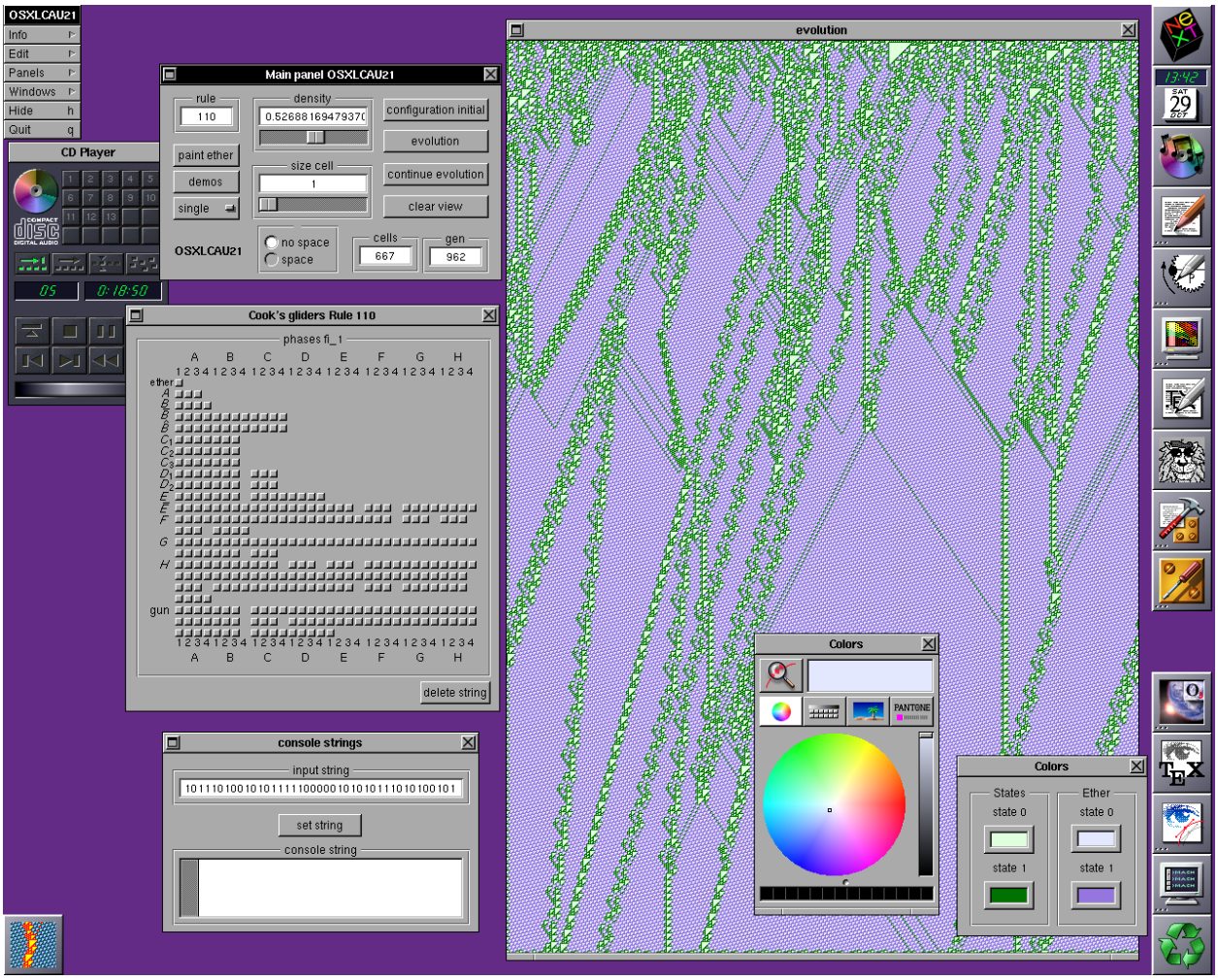

Figura A.1: Sistema OSXLCAU21 disponible desde `http://uncomp.uwe.ac.uk/genaro/OSXCASystems.html`

En este apéndice se describen las funciones de cada una de las partes del sistema OSXLCAU21. La característica principal de nuestra aplicación es el panel de gliders que permite construir condiciones iniciales a través de su representación por fases en el espacio lineal de la Regla 110. El sistema es realizado con la intención de facilitar al usuario su uso y experimentación en el estudio de choques con la Regla 110.

El sistema OSXLCAU21 surge ante la necesidad de construir condiciones iniciales especificando detalladamente la posición de cada glider y el fondo periódico. El programa está escrito en el lenguaje C-Objetivo y actualmente se encuentra disponible para los sistemas OPENSTEP, Mac OS X y Windows (la versión para windows necesita tener instalado el paquete WebObjects para windows).[1]

El uso del sistema OSXLCAU21 es orientado en un ambiente completamente gráfico, dominado por ventanas y botones. Una característica es que el sistema puede trabajar con todas las 256 reglas de evolución de orden $(2, 1)$. Sin embargo, debemos señalar que el manejo de gliders es únicamente para la Regla 110.

A continuación describimos las facilidades que ofrece el sistema así como lo que es posible hacer con ellos. Las tres versiones son realizadas con el mismo ambiente original de OPENSTEP.

A.1 Ventana *Main panel OSXLCAU21*

La ventana principal *Main panel OSXLCAU21* es la que controla la mayoría de las acciones del sistema. Proporciona y establece información útil como podemos ver en la Fig. A.2.

El campo de texto *evolution rule* permite editar directamente la regla de evolución que se desea visualizar. La regla se introduce en notación decimal tomando valores de 0 a 255.

El campo de texto *density* permite definir la densidad de la configuración inicial deslizando la botón de abajo para determinar el valor deseado. Si la densidad se aproxima a cero quiere decir que el estado 0 ocupará más células en la configuración inicial. En el caso contrario, dominará el estado 1. También la densidad puede ser editada directamente en el campo de texto si se desea una cantidad exacta en particular. Por ejemplo, la Regla 110 encuentra su estabilidad estadística en 0.57 [30]. Luego se puede editar esta cantidad directamente en la densidad inicial.

El campo de texto *size cell* permite definir diferentes tamaños en que las células pueden ser vistas en el espacio de evoluciones deslizando el botón que se encuentra abajo. El rango de los

[1]La aplicación y código fuente del programa OSXLCAU21 para las tres versiones son disponibles desde `http://uncomp.uwe.ac.uk/genaro/OSXCA-systems.html` y `http://www.rule110.org/download.html`

diferentes tamaños que podemos manejar para ver las células es de 1 a 10. Particularmente esta opción resulta muy útil cuando se desea obtener una mejor visualización de las células para un choque en particular o alguna estructura.

El campo de texto *cells* muestra el número de células que tiene la configuración inicial y el campo de texto *gen* muestra el número de generaciones calculadas, donde ambos valores son actualizados cada vez que el usuario define un nuevo tamaño en la ventana que muestra la evolución o continua por varias generaciones.

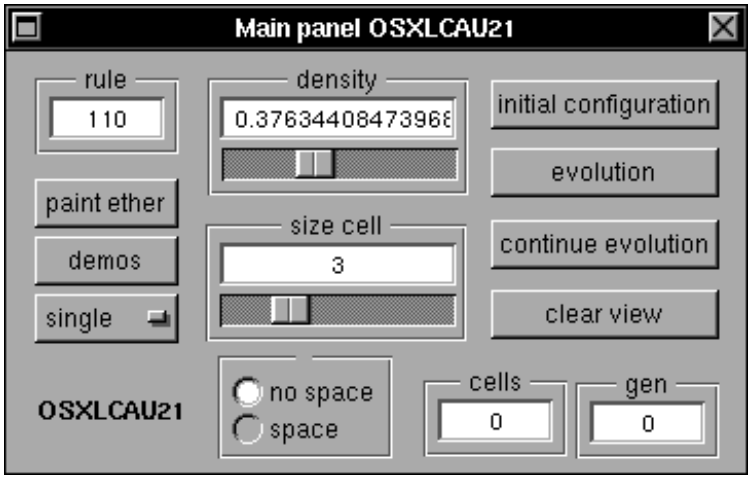

Figura A.2: Panel principal.

El botón *initial configuration* asigna una configuración inicial aleatoria de acuerdo al valor de la densidad inicial y los colores que se tienen definidos en ese momento para cada estado.

El botón *evolution* calcula y muestra la evolución para la configuración inicial que es asignada en ese momento, donde la evolución es graficada por renglón.

El botón *continue evolution* calcula y muestra la continuación de la evolución anterior a partir de la última generación de la primera evolución. Al mismo tiempo, se actualiza el número de generaciones en el campo de texto *gen*.

El botón *clear view* limpia el espacio de evoluciones e inicializa todas las variables del sistema.

El botón *paint ether* colorea el mosaico T_3 en otros colores diferentes al establecido para los estados (si así lo define el usuario). Esto es útil para visualizar mejor los gliders del fondo periódico, choques complicados entre varios gliders o algún mosaico en particular.

El botón con doble opción *space* y *no-space* establecen una pequeña división entre cada una de las células en el espacio de evoluciones. La opción es útil cuando se quiere ver con

detalle el número de células involucradas en una secuencia dada o región, porque se establece una división en todas las células (ver Fig 2.6).

A.2 Ventana *Evolution*

El espacio de evoluciones es controlado por una región gráfica asignada a la ventana *Evolution*. La ventana tiene la característica de ser redimensionable a lo largo y ancho de la pantalla. Un problema importante en el sistema por ahora, es que la configuración inicial está limitada a la resolución de la pantalla.

Un detalle importante es que el sistema no determina el tamaño de la configuración que el usuario introduce. Esto produce estructuras no deseadas en los límites de la configuración. Una alternativa para evitar estructuras no deseadas en la evolución, es ajustar manualmente el ancho de la ventana *Evolution* de manera que la expresion regular que representa el éter termine conrrectamente en la última célula.

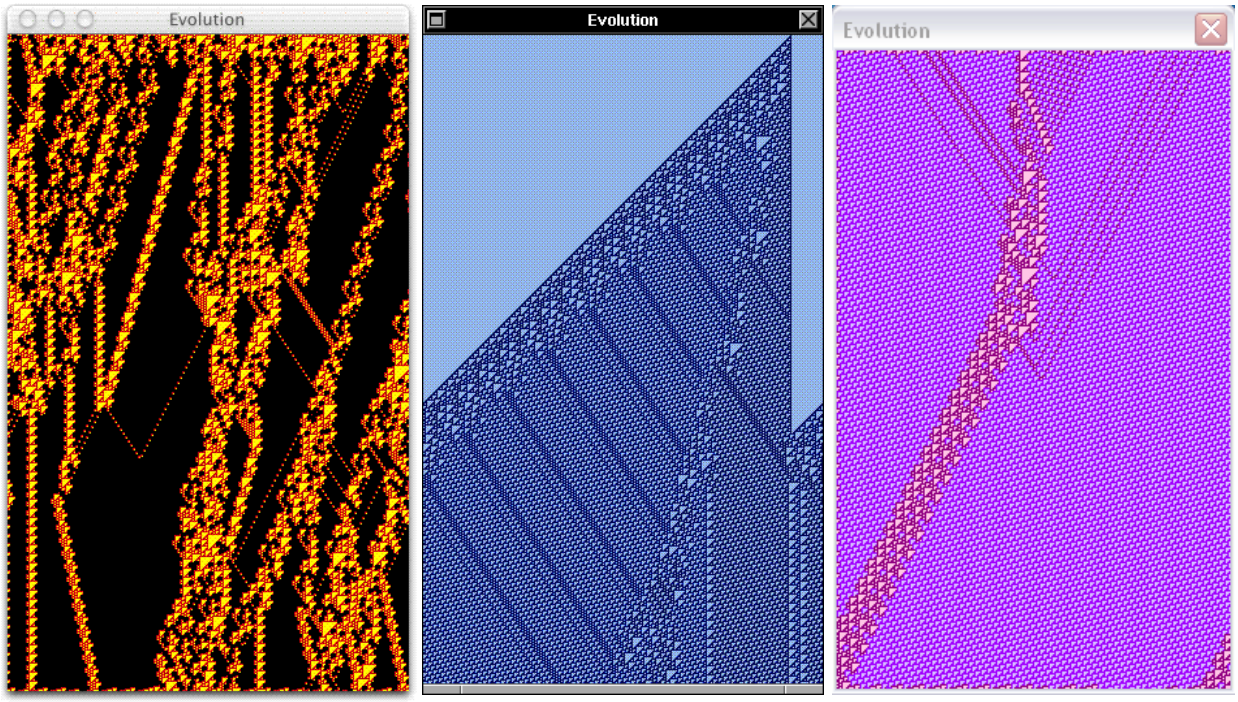

Figura A.3: Espacio de evoluciones: aleatorio, una célula y un choque produciendo el glider $\bar{B}^2$ respectivamente.

En la Fig. A.3 se ilustran tres evoluciones de la Regla 110 para los tres sistemas desarrollados. La primera figura ilustra la evolución de una configuración aleatoria. En este ejemplo

se aplica un solo color para el éter que permite identificar mejor los gliders o simplemente para obtener una figura atractiva.

La segunda figura ilustra la evolución de una célula con el estado 1 y el resto de la configuración inicial con el estado 0. Este ejemplo es interesante porque si el espacio es lo suficientemente grande, es posible ver cómo el margen que se produce, genera glider guns en unas 3,500 generaciones aproximadamente, donde la evolución llega a estabilizarse a través de choques y comportamientos periódicos [23].

La tercera figura es un caso especial que ilustra la construcción de un glider $\bar{B}^2$. Iniciamos la construcción asignando dos configuraciones éter con el panel de gliders. Después se asigna un glider A en fase 2. La ubicación en el panel de gliders funciona como coordenadas. Primero se busca por renglón el glider A. Después se ubica por columna la fase 2. Luego, ésta es la secuencia asignada y así para toda la expresión. En este ejemplo es necesario extraer secuencias que no están en el panel de gliders. Por ejemplo el paquete de gliders A y gliders B tienen que ser asignados desde la consola de cadenas. Este problema se presenta porque las expresiones regulares están para un solo glider y no para paquetes o extensiones de ellos, y ese es otro problema porque el número de veces en que pueden agruparse o extenderse los gliders es ilimitado.

A.3 Ventana *Colors Panel*

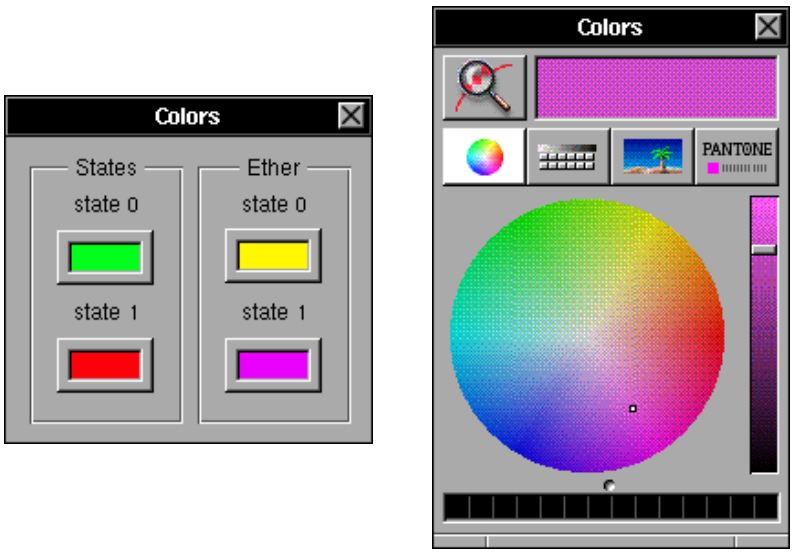

Figura A.4: Panel de colores.

Los colores que asignamos a los estados del AC son determinados con el panel *Colors Panel* (Fig. A.4). Por ejemplo, cuando se desea pintar el éter de otro color se puede definir un color para cada estado del mosaico T_3, donde los pozos de colores (ventana izquierda) invocan al panel *Colors* (ventana derecha) automáticamente cuando se les da un click con el ratón en la orilla del pozo. Esto es una característica original del sistema operativo NeXTSTEP.

Es importante destacar la amplia combinación de colores que ofrece dicho panel, incluyendo la clasificación PANTONE y PANTONE Process, en el caso del sistema NeXTSTEP y OPENSTEP. Además, estos sistemas tienen todas sus salidas en PostScript, lo cual ofrece una mejor resolución para las gráficas, aunque ello implica generar archivos de varios megabytes de tamaño.

A.4 Ventana *Cook's Gliders Rule 110*

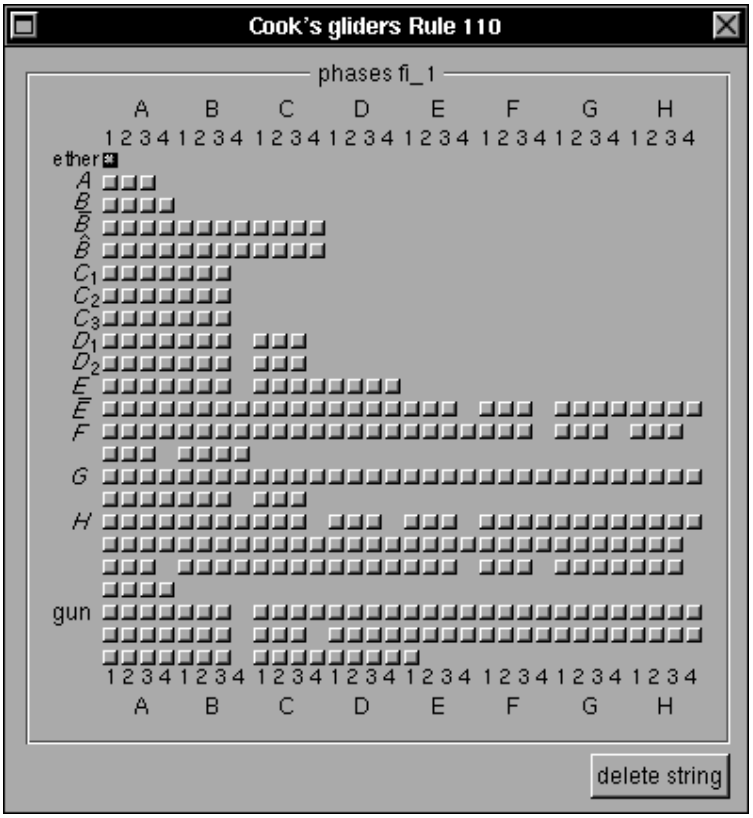

Figura A.5: Panel de fases en los gliders de la Regla 110.

La característica principal de nuestro sistema es el panel de fases para cada uno de los

gliders en la Regla 110 (Fig. A.5), de acuerdo a la clasificación de Cook [10]. Es conveniente señalar que ningún otro sistema cuenta con esta herramienta. Por ejemplo, si se desea introducir una fase en particular, sólo hay que dar un click en el botón de la fase deseada y ese fragmento de configuración se asigna y dibuja inmediatamente en la ventana *Evolution*. Por lo tanto, en este panel se encuentran todas las fases de $\mathcal{F}_1$, para todos los gliders incluyendo el glider gun.

En algunas ocasiones al momento de introducir un glider se puede equivocar la fase que se introduce o simplemente se pueden reordenar algunas expresiones sin necesidad de construir toda la condición inicial otra vez. Para resolver este problema se creó un botón *delete string* que borra la última cadena de configuración introducida. Un problema aquí es que el sistema no guarda la configuración construida.

Analicemos un ejemplo de código para ilustrar la funcionalidad del panel de fases. Por ejemplo, conocemos que la expresión C_3(A,f$_1$_1)-e-G(E,f$_1$_1) produce un glider $\bar{E}$ (ver Tabla 3.1). La manera cómo realizamos la construcción gráfica es la siguiente (recordemos que el panel funciona como un sistema de coordenadas): primero asignamos configuraciones éter e como deseemos. Después, buscamos el glider C_3 por renglón. Luego, la letra A indica que es la primera columna y ahí buscamos la fase f$_1$_1. Después le damos un click a dicha fase. Continuando asignamos una configuración éter, posteriormente buscamos el glider G por renglón y la letra E por columna para localizar la fase f$_1$_1 y le damos un click a dicha fase. Finalmente, asignamos configuraciones éter para finalizar la construcción de la condición inicial.

A.5 Ventana *Console Strings*

Al momento de experimentar la construcción de ciertas configuraciones en particular, se tuvieron que definir cadenas que tenían que ser manipuladas de manera independiente. Por ejemplo, paquetes de gliders que no pueden ser construidos a través de las fases o extensiones que tampoco pueden ser construidas a partir de las fases básicas. Para resolver este problema se tiene una pequeña consola capaz de recibir una cadena en particular y asignarla a la configuración inicial a través de un botón.

En la Fig. A.6 se ilustra la consola de cadenas. Su forma es útil y práctica para configuraciones que no están definidas en el panel de gliders. Se introduce la cadena que se desea en el campo de texto *input string* (el arreglo para la cadena puede soportar hasta 1,500 elementos). Una vez que se tiene la cadena a introducir se presiona el botón *set string*. En la parte de abajo se muestra la cadena que es introducida para verificar que ha sido bien

interpretada y al mismo tiempo se dibuja inmediatamente en la ventana *Evolution*. Si la cadena introducida no es la que se desea por alguna razón, se regresa al panel *Cook's Gliders Rule 110* y se borra la cadena con el botón *delete string*.

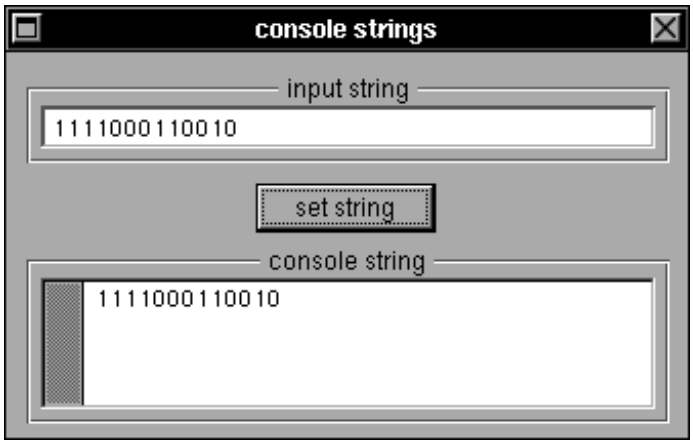

Figura A.6: Panel de cadenas.

Finalmente, debemos señalar que el sistema OSXLCAU21 tiene varias limitaciones que deben ser desarrolladas gradualmente:

1. Controlar la evolución y pararla en el momento que se desee.
2. Crear un BrowserView que permita el manejo de enormes espacios de evoluciones para grandes condiciones iniciales.
3. Editar directamente en el espacio de evoluciones alguna configuración deseada o, en su defecto, si el espacio de evoluciones es cubierto por una evolución dada, entonces que la evolución se actualize con respecto a la nueva célula introducida.
4. Crear un modo de visualización zoom-in y zoom-out.
5. Introducir herramientas de análisis de teoría de gráficas, probabilidad, estadística y matrices.

Apéndice B

Subconjunto de expresiones regulares

A continuación presentamos el conjunto completo $\mathcal{F}_1$ que determina el subconjunto de expresiones regulares Ψ_{R110}, para todos los gliders hasta ahora conocidos en la Regla 110 (código en fases $\#_1(\#_2,\mathrm{f}_i\_1)$ y cadena w).[1] Dando origen al lenguaje regular L_{R110} basado en gliders.

B.1 Éter

$e(\mathrm{f}_1\_1) = 11111000100110$

B.2 Glider A

$A(\mathrm{f}_1\_1) = 111110$
$A(\mathrm{f}_2\_1) = 11111000111000100110$
$A(\mathrm{f}_3\_1) = 1111100010011010011 0$
$A(\mathrm{f}_4\_1) = A(\mathrm{f}_1\_1)$

B.3 Glider B

$B(\mathrm{f}_1\_1) = 11111010$
$B(\mathrm{f}_2\_1) = 11111000$

[1] El subconjunto de expresiones regulares es disponible en un archivo de texto "listPhasesR110.txt" desde `http://uncomp.uwe.ac.uk/genaro/rule110/listPhasesR110.txt`

$B(\mathrm{f}_3\_1) = 1111100010011000100110$
$B(\mathrm{f}_4\_1) = 11100110$

B.4 Glider $\bar{B}$

$\bar{B}(\mathrm{A},\mathrm{f}_1\_1) = 1111100010110111100110$
$\bar{B}(\mathrm{A},\mathrm{f}_2\_1) = 1111100010011111110010111111000100110$
$\bar{B}(\mathrm{A},\mathrm{f}_3\_1) = 1111100010011011000001011111000100110$
$\bar{B}(\mathrm{A},\mathrm{f}_4\_1) = 1111110000111100100110$

$\bar{B}(\mathrm{B},\mathrm{f}_1\_1) = 1111100001000110010110$
$\bar{B}(\mathrm{B},\mathrm{f}_2\_1) = 1111100010001100111011111111000100110$
$\bar{B}(\mathrm{B},\mathrm{f}_3\_1) = 1111100010011001110110111100000100110$
$\bar{B}(\mathrm{B},\mathrm{f}_4\_1) = 1110110111111010000110$

$\bar{B}(\mathrm{C},\mathrm{f}_1\_1) = 1111101111110000111000$
$\bar{B}(\mathrm{C},\mathrm{f}_2\_1) = 1111100011100001000110100111000100110$
$\bar{B}(\mathrm{C},\mathrm{f}_3\_1) = 1111100010011010001100111111011100110$
$\bar{B}(\mathrm{C},\mathrm{f}_4\_1) = 1111100111011000111010$

B.5 Glider $\hat{B}$

$\hat{B}(\mathrm{A},\mathrm{f}_1\_1) = 1111100010110111100110011111111000100110$
$\hat{B}(\mathrm{A},\mathrm{f}_2\_1) = 1111100010011111110010111011000000100110$
$\hat{B}(\mathrm{A},\mathrm{f}_3\_1) = 1111100010011011000001011111011110000110$
$\hat{B}(\mathrm{A},\mathrm{f}_4\_1) = 1111110000111100111001000$

$\hat{B}(\mathrm{B},\mathrm{f}_1\_1) = 1111100001000110010110101100011000100110$
$\hat{B}(\mathrm{B},\mathrm{f}_2\_1) = 1111100010001100111011111111111011100110$
$\hat{B}(\mathrm{B},\mathrm{f}_3\_1) = 1111100010011001110110111000000000111010$
$\hat{B}(\mathrm{B},\mathrm{f}_4\_1) = 1110110111111010000000110$

$\hat{B}(\mathrm{C},\mathrm{f}_1\_1) = 1111101111110000111000000111111000100110$
$\hat{B}(\mathrm{C},\mathrm{f}_2\_1) = 1111100011100001000110100000011000100110$

$\hat{B}$(C,f_3_1) = 1111100010011010001100111110000011100110
$\hat{B}$(C,f_4_1) = 111110011101100010001101

B.6 Glider C_1

C_1(A,f_1_1) = 111110000
C_1(A,f_2_1) = 11111000100011000100110
C_1(A,f_3_1) = 11111000100110011100110
C_1(A,f_4_1) = 111011010

C_1(B,f_1_1) = 11111011111111000100110
C_1(B,f_2_1) = 11111000111000000100110
C_1(B,f_3_1) = 11111000100110100000110
C_1(B,f_4_1) = C_1(B,f_1_1)

B.7 Glider C_2

C_2(A,f_1_1) = 11111000000100110
C_2(A,f_2_1) = 11111000100000110
C_2(A,f_3_1) = 11111000100110000
C_2(A,f_4_1) = 11100011000100110

C_2(B,f_1_1) = 11111010011100110
C_2(B,f_2_1) = 11111000111011010
C_2(B,f_3_1) = 1111100010011011111111000100110
C_2(B,f_4_1) = C_2(B,f_1_1)

B.8 Glider C_3

C_3(A,f_1_1) = 11111011010
C_3(A,f_2_1) = 11111000111111111000100110
C_3(A,f_3_1) = 11111000100110000000100110
C_3(A,f_4_1) = 11100000110

C_3(B,f_1_1) = 11111010000
C_3(B,f_2_1) = 111110001110001100010011 0
C_3(B,f_3_1) = 111110001001101001110011 0
C_3(B,f_4_1) = C_3(B,f_1_1)

B.9 Glider D_1

D_1(A,f_1_1) = 11111000010
D_1(A,f_2_1) = 111110001000111100010011 0
D_1(A,f_3_1) = 111110001001100110010011 0
D_1(A,f_4_1) = 11101110110

D_1(B,f_1_1) = 111110111011111100010011 0
D_1(B,f_2_1) = 111110001110111000010011 0
D_1(B,f_3_1) =111110001001101110100011 0
D_1(B,f_4_1) = D_1(C,f_1_1)

D_1(C,f_1_1) = 11111011100
D_1(C,f_2_1) = 111110001110101100010011 0
D_1(C,f_3_1) = 111110001001101111110011 0
D_1(C,f_4_1) = D_1(A,f_1_1)

B.10 Glider D_2

D_2(A,f_1_1) = 1111101011000100110
D_2(A,f_2_1) = 1111100011111100110
D_2(A,f_3_1) = 1111100010011000010
D_2(A,f_4_1) = 1110001111000100110

D_2(B,f_1_1) = 1111101001100100110
D_2(B,f_2_1) = 1111100011101110110
D_2(B,f_3_1) = 111110001001101110111111000100110
D_2(B,f_4_1) = D_2(C,f_1_1)

$D_2(\mathrm{C},\mathrm{f}_1\_1) = 1111101110000100110$
$D_2(\mathrm{C},\mathrm{f}_2\_1) = 1111100011101000110$
$D_2(\mathrm{C},\mathrm{f}_3\_1) = 1111100010011011100$
$D_2(\mathrm{C},\mathrm{f}_4\_1) = D_2(\mathrm{A},\mathrm{f}_1\_1)$

B.11 Glider E

$E(\mathrm{A},\mathrm{f}_1\_1) = 1111100000000100110$
$E(\mathrm{A},\mathrm{f}_2\_1) = 1111100010000000110$
$E(\mathrm{A},\mathrm{f}_3\_1) = 1111100010011000000$
$E(\mathrm{A},\mathrm{f}_4\_1) = 1110000011000100110$

$E(\mathrm{B},\mathrm{f}_1\_1) = 1111101000011100110$
$E(\mathrm{B},\mathrm{f}_2\_1) = 1111100011100011010$
$E(\mathrm{B},\mathrm{f}_3\_1) = 1111100010011010011111111000100110$
$E(\mathrm{B},\mathrm{f}_4\_1) = E(\mathrm{C},\mathrm{f}_1\_1)$

$E(\mathrm{C},\mathrm{f}_1\_1) = 1111101100000100110$
$E(\mathrm{C},\mathrm{f}_2\_1) = 1111100011110000110$
$E(\mathrm{C},\mathrm{f}_3\_1) = 1111100010011001000$
$E(\mathrm{C},\mathrm{f}_4\_1) = 1110110011000100110$

$E(\mathrm{D},\mathrm{f}_1\_1) = 1111101111011100110$
$E(\mathrm{D},\mathrm{f}_2\_1) = 1111100011100111010$
$E(\mathrm{D},\mathrm{f}_3\_1) = 1111100010011010110$
$E(\mathrm{D},\mathrm{f}_4\_1) = 1111111111000100110$

B.12 Glider $\bar{E}$

$\bar{E}(\mathrm{A},\mathrm{f}_1\_1) = 111110000100011111010$
$\bar{E}(\mathrm{A},\mathrm{f}_2\_1) = 111110001000110011000$
$\bar{E}(\mathrm{A},\mathrm{f}_3\_1) = 111110001001100111011100110001000100110$
$\bar{E}(\mathrm{A},\mathrm{f}_4\_1) = 111011011101011100110$

$\bar{E}$(B,f_1_1) = 111110111111011111010
$\bar{E}$(B,f_2_1) = 11111000111000011100
$\bar{E}$(B,f_3_1) = 1111100010011010001101001100010011 0
$\bar{E}$(B,f_4_1) = 11111001111101110011 0

$\bar{E}$(C,f_1_1) = 11111000101100011101 0
$\bar{E}$(C,f_2_1) = 11111000100111110011 0
$\bar{E}$(C,f_3_1) = 1111100010011011000101111100010011 0
$\bar{E}$(C,f_4_1) = 11111100111100010011 0

$\bar{E}$(D,f_1_1) = 11111000010110010011 0
$\bar{E}$(D,f_2_1) = 11111000100011111011 0
$\bar{E}$(D,f_3_1) = 1111100010011001100011111100010011 0
$\bar{E}$(D,f_4_1) = 11101110011000010011 0

$\bar{E}$(E,f_1_1) = 11111011101011100011 0
$\bar{E}$(E,f_2_1) = 11111000111011111010 0
$\bar{E}$(E,f_3_1) = 1111100010011011100011101100010011 0
$\bar{E}$(E,f_4_1) = $\bar{E}$(F,f_1_1)

$\bar{E}$(F,f_1_1) = 11111010011011110011 0
$\bar{E}$(F,f_2_1) = 11111000111011111001 0
$\bar{E}$(F,f_3_1) = 1111100010011011100010111100010011 0
$\bar{E}$(F,f_4_1) = $\bar{E}$(G,f_1_1)

$\bar{E}$(G,f_1_1) = 11111010011110010011 0
$\bar{E}$(G,f_2_1) = 11111000111011001011 0
$\bar{E}$(G,f_3_1) = 1111100010011011110111111100010011 0
$\bar{E}$(G,f_4_1) = 11111001110000010011 0

$\bar{E}$(H,f_1_1) = 11111000101101000011 0
$\bar{E}$(H,f_2_1) = 11111000100111111000 0
$\bar{E}$(H,f_3_1) = 1111100010011011000001001100010011 0
$\bar{E}$(H,f_4_1) = 11111110000110111001 0

B.13 Glider F

F(A,f$_1$_1) = 111110001011010
F(A,f$_2$_1) = 11111000100111111111000100110
F(A,f$_3$_1) = 11111000100110110000000100110
F(A,f$_4$_1) = 111111000000110

F(B,f$_1$_1) = 111110000100000
F(B,f$_2$_1) = 11111000100011000011000100110
F(B,f$_3$_1) = 11111000100110011100011100110
F(B,f$_4$_1) = 111011010011010

F(C,f$_1$_1) = 11111011111101111111000100110
F(C,f$_2$_1) = 11111000111000011100000100110
F(C,f$_3$_1) = 11111000100110100011010000110
F(C,f$_4$_1) = 111110011111000

F(D,f$_1$_1) = 11111000101100010011000100110
F(D,f$_2$_1) = 11111000100111110011011100110
F(D,f$_3$_1) = 11111000100110110001011111010
F(D,f$_4$_1) = 111111001111000

F(E,f$_1$_1) = 11111000010110010011000100110
F(E,f$_2$_1) = 11111000100011111011011100110
F(E,f$_3$_1) = 11111000100110011000111111010
F(E,f$_4$_1) = 111011100110000

F(F,f$_1$_1) = 11111011101011100011000100110
F(F,f$_2$_1) = 11111000111011111010011100110
F(F,f$_3$_1) = 11111000100110111000111011010
F(F,f$_4$_1) = F(G,f$_1$_1)

F(G,f$_1$_1) = 11111010011011111111000100110
F(G,f$_2$_1) = 11111000111011111000000100110
F(G,f$_3$_1) = 11111000100110111000100000110
F(G,f$_4$_1) = F(H,f$_1$_1)

F(H,f$_1$_1) = 111110100110000
F(H,f$_2$_1) = 11111000111011100011000100110
F(H,f$_3$_1) = 11111000100110111010011100110
F(H,f$_4$_1) = F(A2,f$_1$_1)

F(A2,f$_1$_1) = 111110111011010
F(A2,f$_2$_1) = 11111000111011111111000100110
F(A2,f$_3$_1) = 11111000100110111000000100110
F(A2,f$_4$_1) = F(B2,f$_1$_1)

F(B2,f$_1$_1) = 111110100000110
F(B2,f$_2$_1) = 111110001110000
F(B2,f$_3$_1) = 11111000100110100011000100110
F(B2,f$_4$_1) = 111110011100110

B.14 Glider G

G(A,f$_1$_1) = 11111010011111001110011 0
G(A,f$_2$_1) = 1111100011101100010 11010
G(A,f$_3$_1) = 11111000100110111100111111111000100110
G(A,f$_4$_1) = 1111100101100000000100110

G(B,f$_1$_1) = 1111100010111111000000110
G(B,f$_2$_1) = 1111100010011111000100000
G(B,f$_3$_1) = 11111000100110110010011000011000100110
G(B,f$_4$_1) = 1111110110111000111001 10

G(C,f$_1$_1) = 1111100001111111010011010
G(C,f$_2$_1) = 11111000100011000011101111111000100110
G(C,f$_3$_1) = 11111000100110011100011011100000100110
G(C,f$_4$_1) = 1110110100111111010000110

G(D,f$_1$_1) = 11111011111111011000111000
G(D,f$_2$_1) = 11111000111000011110011010011000100110

$G(\mathrm{D},\mathrm{f}_3\_1) = 1111100010011010001100101111101110011 0$
$G(\mathrm{D},\mathrm{f}_4\_1) = 111110011101111000111010$

$G(\mathrm{E},\mathrm{f}_1\_1) = 111110001011011100100110$
$G(\mathrm{E},\mathrm{f}_2\_1) = 1111100010011111110101101111100010011 0$
$G(\mathrm{E},\mathrm{f}_3\_1) = 1111100010011011000001111111100010011 0$
$G(\mathrm{E},\mathrm{f}_4\_1) = 111111000011000000100110$

$G(\mathrm{F},\mathrm{f}_1\_1) = 111110000100011100000110$
$G(\mathrm{F},\mathrm{f}_2\_1) = 1111100010001100110100001111100010011 0$
$G(\mathrm{F},\mathrm{f}_3\_1) = 1111100010011001110111110001100010011 0$
$G(\mathrm{F},\mathrm{f}_4\_1) = 111011011100010011100110$

$G(\mathrm{G},\mathrm{f}_1\_1) = 111110111111010011011010$
$G(\mathrm{G},\mathrm{f}_2\_1) = 1111100011100001110111111111000100110$
$G(\mathrm{G},\mathrm{f}_3\_1) = 1111100010011010001101110000000100110$
$G(\mathrm{G},\mathrm{f}_4\_1) = 111110011111010000000110$

$G(\mathrm{H},\mathrm{f}_1\_1) = 111110001011000111000000$
$G(\mathrm{H},\mathrm{f}_2\_1) = 1111100010011111001101000011000100110$
$G(\mathrm{H},\mathrm{f}_3\_1) = 1111100010011011000101111100001110 0110$
$G(\mathrm{H},\mathrm{f}_4\_1) = 111111001111000100011010$

$G(\mathrm{A2},\mathrm{f}_1\_1) = 1111100001011001001100111111100010011 0$
$G(\mathrm{A2},\mathrm{f}_2\_1) = 1111100010001111101101110110000010011 0$
$G(\mathrm{A2},\mathrm{f}_3\_1) = 1111100010011001100011111101111000011 0$
$G(\mathrm{A2},\mathrm{f}_4\_1) = 111011100110000111001000$

$G(\mathrm{B2},\mathrm{f}_1\_1) = 1111101110101110001101011001100010011 0$
$G(\mathrm{B2},\mathrm{f}_2\_1) = 1111100011101111101001111111011100110$
$G(\mathrm{B2},\mathrm{f}_3\_1) = 1111100010011011100011101100000011101 0$
$G(\mathrm{B2},\mathrm{f}_4\_1) = G(\mathrm{C2},\mathrm{f}_1\_1)$

$G(\mathrm{C2},\mathrm{f}_1\_1) = 111110100110111100000110$
$G(\mathrm{C2},\mathrm{f}_2\_1) = 1111100011101111100100001111100010011 0$
$G(\mathrm{C2},\mathrm{f}_3\_1) = 1111100010011011100010110001100010011 0$

G(C2,f$_4$_1) = G(A,f$_1$_1)

B.15 Glider H

H(A,f$_1$_1) = 1111100010110000000011111000100110100111111000100110
H(A,f$_2$_1) = 1111100010011111000000011000100110111110100000100110
H(A,f$_3$_1) = 1111100010011011000100000011100110111110001111000110
H(A,f$_4$_1) = 11111100110000011010111110001001100 1000

H(B,f$_1$_1) = 1111100001011100001111111000100110111011001100010 0110
H(B,f$_2$_1) = 1111100010001111010001100000100110111110111101110 0110
H(B,f$_3$_1) = 1111100010011001100111001110000110111110001110011 1010
H(B,f$_4$_1) = 1110111011010110100011111000100110101 10

H(C,f$_1$_1) = 1111101110111111111110011000100110111111111000100110
H(C,f$_2$_1) = 1111100011101110000000001011100110111110000000100110
H(C,f$_3$_1) = 1111100010011011101000000001111010111110001000000110
H(C,f$_4$_1) = H(D,f$_1$_1)

H(D,f$_1$_1) = 1111101110000000110011111000100110000 00
H(D,f$_2$_1) = 1111100011101000000111011000100110111000001100010 0110
H(D,f$_3$_1) = 1111100010011011100000110111100110111110100001110 0110
H(D,f$_4$_1) = H(E,f$_1$_1)

H(E,f$_1$_1) = 1111101000011111001011111000111000110 10
H(E,f$_2$_1) = 1111100011100011000101111000100110100111111100010 0110
H(E,f$_3$_1) = 1111100010011010011100111100100110111110110000010 0110
H(E,f$_4$_1) = H(F,f$_1$_1)

H(F,f$_1$_1) = 1111101101011001011011111000111100001 10
H(F,f$_2$_1) = 1111100011111111101111111000100110010 00
H(F,f$_3$_1) = 1111100010011000000011100000100110111011001100010 0110
H(F,f$_4$_1) = 1110000001101000011011111011110111001 10

H(G,f$_1$_1) = 1111101000001111100011111000111001110 10

$H(\mathrm{G},\mathrm{f}_2\_1) = 11111000111000011000100110001001101011 0$
$H(\mathrm{G},\mathrm{f}_3\_1) = 1111100010011010001110011011100110111111111000100110$
$H(\mathrm{G},\mathrm{f}_4\_1) = 1111100110101111101011111000000000100110$

$H(\mathrm{H},\mathrm{f}_1\_1) = 1111100010111111100011111000100000000110$
$H(\mathrm{H},\mathrm{f}_2\_1) = 1111100010011110000010011000100110000000$
$H(\mathrm{H},\mathrm{f}_3\_1) = 1111100010011011001000011011100110111000001100010011 0$
$H(\mathrm{H},\mathrm{f}_4\_1) = 1111110110001111101011111010000011100110$

$H(\mathrm{A2},\mathrm{f}_1\_1) = 1111100001111001100011111000110001101 0$
$H(\mathrm{A2},\mathrm{f}_2\_1) = 1111100010001100101110011000100110100111111100010011 0$
$H(\mathrm{A2},\mathrm{f}_3\_1) = 1111100010011001110111101011100110111110110000010011 0$
$H(\mathrm{A2},\mathrm{f}_4\_1) = 1110110111001111101011111000111000011 0$

$H(\mathrm{B2},\mathrm{f}_1\_1) = 1111101111110101100011111000100110010 00$
$H(\mathrm{B2},\mathrm{f}_2\_1) = 1111100011100001111110011000100110111011001100010011 0$
$H(\mathrm{B2},\mathrm{f}_3\_1) = 1111100010011010001100001011100110111110111011100110$
$H(\mathrm{B2},\mathrm{f}_4\_1) = 1111100111000111101011111000111001110 10$

$H(\mathrm{C2},\mathrm{f}_1\_1) = 1111100010110100110011111000100110101 10$
$H(\mathrm{C2},\mathrm{f}_2\_1) = 1111100010011111110111011000100110111111111000100110$
$H(\mathrm{C2},\mathrm{f}_3\_1) = 1111100010011011000001110111100110111110000000100110$
$H(\mathrm{C2},\mathrm{f}_4\_1) = 1111110000110111001011111000100000000110$

$H(\mathrm{D2},\mathrm{f}_1\_1) = 1111100001000111101011110001001100000 0$
$H(\mathrm{D2},\mathrm{f}_2\_1) = 1111100010001100110001111100100110111000001100010011 0$
$H(\mathrm{D2},\mathrm{f}_3\_1) = 1111100010011001110111001100010110111110100001110011 0$
$H(\mathrm{D2},\mathrm{f}_4\_1) = 1110110111010111001111111000111000110 10$

$H(\mathrm{E2},\mathrm{f}_1\_1) = 1111101111110111110101100000100110100111111100010011 0$
$H(\mathrm{E2},\mathrm{f}_2\_1) = 1111100011100001110001111110000110111110110000010011 0$
$H(\mathrm{E2},\mathrm{f}_3\_1) = 1111100010011010001101001100001000111110001111000011 0$
$H(\mathrm{E2},\mathrm{f}_4\_1) = 1111100111110111000110011000100110010 00$

$H(\mathrm{F2},\mathrm{f}_1\_1) = 1111100010110001110100111011100110111011001100010011 0$
$H(\mathrm{F2},\mathrm{f}_2\_1) = 1111100010011111001101110110111010111110111011100110$

$H(\mathrm{F2,f_3\_1}) = 11111000100110110001011111011111101111100011100111010$
$H(\mathrm{F2,f_4\_1}) = 11111100111100011100001110001001101011 0$

$H(\mathrm{G2,f_1\_1}) = 1111100001011001001101000110100110111111111000100110$
$H(\mathrm{G2,f_2\_1}) = 1111100010001111101101111100111101111100000000100110$
$H(\mathrm{G2,f_3\_1}) = 1111100010011001100011111100010110001110001000000 0110$
$H(\mathrm{G2,f_4\_1}) = 1110111001100001001111100110100110000 0$

$H(\mathrm{H2,f_1\_1}) = 1111101110101110001101100010111110111000001100010011 0$
$H(\mathrm{H2,f_2\_1}) = 1111100011101111101001111110011110001110100001110011 0$
$H(\mathrm{H2,f_3\_1}) = 1111100010011011100011101100001011001001101110001101 0$
$H(\mathrm{H2,f_4\_1}) = H(\mathrm{A3,f_1\_1})$

$H(\mathrm{A3,f_1\_1}) = 1111101001101111000111110110111110100111111100010011 0$
$H(\mathrm{A3,f_2\_1}) = 1111100011101111100100110001111110001110110000010011 0$
$H(\mathrm{A3,f_3\_1}) = 1111100010011011100010110111001100001001101111000011 0$
$H(\mathrm{A3,f_4\_1}) = H(\mathrm{B3,f_1\_1})$

$H(\mathrm{B3,f_1\_1}) = 1111101001111111010111000110111111001000$
$H(\mathrm{B3,f_2\_1}) = 1111100011101100000111110100111110001011001100010011 0$
$H(\mathrm{B3,f_3\_1}) = 1111100010011011110000110001110110001001111101110011 0$
$H(\mathrm{B3,f_4\_1}) = 1111100100011100110111100110110001110 10$

$H(\mathrm{C3,f_1\_1}) = 1111100010110011010111110010111111001 10$
$H(\mathrm{C3,f_2\_1}) = 1111100010011111011111110001011110000101111110001001 10$
$H(\mathrm{C3,f_3\_1}) = 1111100010011011000111000001001111001000111100010011 0$
$H(\mathrm{C3,f_4\_1}) = 1111110011010000110110010110011001 00110$

$H(\mathrm{D3,f_1\_1}) = 1111100001011111000111111011111011101 10$
$H(\mathrm{D3,f_2\_1}) = 1111100010001111000100110000111000111011111100010011 0$
$H(\mathrm{D3,f_3\_1}) = 1111100010011001100100110111000110100110111000010011 0$
$H(\mathrm{D3,f_4\_1}) = 1110111011011111010011111011111010001 10$

$H(\mathrm{E3,f_1\_1}) = 1111101110111111000111011000111000111 00$
$H(\mathrm{E3,f_2\_1}) = 1111100011101110000100110111100110100110101100010011 0$
$H(\mathrm{E3,f_3\_1}) = 1111100010011011101000110111110010111110111111110011 0$

H(E3,f$_4$_1) = H(F3,f$_1$_1)

H(F3,f$_1$_1) = 11111011100111110001011110001110000010
H(F3,f$_2$_1) = 1111100011101011000100111100100110100000111100010011 0
H(F3,f$_3$_1) = 1111100010011011111100110110010110111110000110010011 0
H(F3,f$_4$_1) = H(G3,f$_1$_1)

H(G3,f$_1$_1) = 11111000010111111011111110001000111011 0
H(G3,f$_2$_1) = 1111100010001111000011100000100110011011111100010011 0
H(G3,f$_3$_1) = 1111100010011001100100011010000110111011111000010011 0
H(G3,f$_4$_1) = 11101110110011111000111110111000100011 0

H(H3,f$_1$_1) = 11111011101111011000100110001110100110 0
H(H3,f$_2$_1) = 1111100011101110011110011011100110111011100010011 0
H(H3,f$_3$_1) = 1111100010011011101011001011111010111110111011110011 0
H(H3,f$_4$_1) = H(A4,f$_1$_1)

H(A4,f$_1$_1) = 11111011111101111000111110001110111001 0
H(A4,f$_2$_1) = 1111100011100001110010011000100110111010111100010011 0
H(A4,f$_3$_1) = 1111100010011010001101011011100110111110111110010011 0
H(A4,f$_4$_1) = 11111001111111111010111110001110001011 0

B.16 Glider gun

gun(A,f$_1$_1) = 11111010110011101001100101111100000100110
gun(A,f$_2$_1) = 11111000111111011011101110111100010000110
gun(A,f$_3$_1) = 11111000100110000111111011101110010011000
gun(A,f$_4$_1) = 11100011000011101110101101110011000100110

gun(B,f$_1$_1) = 11111010011100011011011111111011011100110
gun(B,f$_2$_1) = 11111000111011010011111011100000011111010
gun(B,f$_3$_1) = 11111000100110111111011000111010000011000
gun(B,f$_4$_1) = gun(C,f$_1$_1)

gun(C,f$_1$_1) = 11111000011110011011100001110011000100110

gun(C,f_2_1) = 1111100010001100101111101000110101110011 0
gun(C,f_3_1) = 1111100010011001110111100011100111111 1010
gun(C,f_4_1) = 1110110111001001101011000 00

gun(D,f_1_1) = 1111101111110101101111111100001100010011 0
gun(D,f_2_1) = 1111100011100001111111100000010001110011 0
gun(D,f_3_1) = 1111100010011010001100000010000011001 1010
gun(D,f_4_1) = 1111100111000001100001110111111100010011 0

gun(E,f_1_1) = 1111100010110100001110001101110000010011 0
gun(E,f_2_1) = 1111100010011111110001101001111101000011 0
gun(E,f_3_1) = 1111100010011011000001001111101100011 1000
gun(E,f_4_1) = 1111110000110110001111001101001100010011 0

gun(F,f_1_1) = 1111100001000111111001100101111101110011 0
gun(F,f_2_1) = 1111100010001100110000101110111100011 1010
gun(F,f_3_1) = 1111100010011001110111000111101110010011 0
gun(F,f_4_1) = 1110110111010011001110101101111100010011 0

gun(G,f_1_1) = 1111101111110111011101101111111100010011 0
gun(G,f_2_1) = 1111100011100001110111011111100000010011 0
gun(G,f_3_1) = 1111100010011010001101110111000010000011 0
gun(G,f_4_1) = 1111100111110111010001100001111100010011 0

gun(H,f_1_1) = 1111100010110001110111001110001100010011 0
gun(H,f_2_1) = 1111100010011111001101110101101001110011 0
gun(H,f_3_1) = 1111100010011011000101111101111111011010
gun(H,f_4_1) = 1111110011110001110000001111111100010011 0

gun(A2,f_1_1) = 1111100001011001001101000001100000010011 0
gun(A2,f_2_1) = 1111100010001111101101111100001110000011 0
gun(A2,f_3_1) = 1111100010011001100011111100010001101 0000
gun(A2,f_4_1) = 1110111001100001001100111110001100010011 0

gun(B2,f_1_1) = 1111101110101110001101110110001001110011 0
gun(B2,f_2_1) = 1111100011101111101001111101111001101 1010

gun(B2,f_3_1) = 1111100010011011100011101100011100101111111111000100110
gun(B2,f_4_1) = gun(C2,f_1_1)

gun(C2,f_1_1) = 1111101001101111001101011110000000100110
gun(C2,f_2_1) = 1111100011101111100101111111010000000110
gun(C2,f_3_1) = 1111100010011011100010111100000101100000 0
gun(C2,f_4_1) = gun(D2,f_1_1)

gun(D2,f_1_1) = 1111101001111001000011111000001100010011 0
gun(D2,f_2_1) = 1111100011101100101100011000100001110011 0
gun(D2,f_3_1) = 1111100010011011110111110011100110001101 0
gun(D2,f_4_1) = 1111100111000101101011100111111100010011 0

gun(E2,f_1_1) = 1111100010110100111111111010110000010011 0
gun(E2,f_2_1) = 1111100010011111110110000000111111000011 0
gun(E2,f_3_1) = 1111100010011011000001111000000110000100 0
gun(E2,f_4_1) = 1111110000110010000011100011001100010011 0

gun(F2,f_1_1) = 1111100001000111011000011010011101110011 0
gun(F2,f_2_1) = 1111100010001100110111100011111011011101 0
gun(F2,f_3_1) = 1111100010011001110111110010011000111111 0
gun(F2,f_4_1) = 1110110111000101101110011000011100010011 0

gun(G2,f_1_1) = 1111101111110100111111101011100011010011 0
gun(G2,f_2_1) = 1111100011100001110110000011111010011111 0
gun(G2,f_3_1) = 1111100010011010001101111000011000111011000111000100110
gun(G2,f_4_1) = 1111100111110010001110011011110011010011 0

gun(H2,f_1_1) = 1111100010110001011001101011111 0010
gun(H2,f_2_1) = 1111100010011111001111101111110001011110001001 10
gun(H2,f_3_1) = 1111100010011011000101100011100000100111100100110
gun(H2,f_4_1) = 1111110011111001101000011011001 0110

gun(A3,f_1_1) = 1111100001011000101111100011111101111110001001 10
gun(A3,f_2_1) = 1111100010001111100111100010011000011100000010011 0
gun(A3,f_3_1) = 1111100010011001100010110010011011100011010000110

gun(A3,f_4_1) = 11101110011111011011111010011111000

gun(B3,f_1_1) = 11111011101011000111111000111011000100110001001 10
gun(B3,f_2_1) = 11111000111011111100110000100110111100110111001 10
gun(B3,f_3_1) = 11111000100110111000010111000110111110010111110 10
gun(B3,f_4_1) = gun(C3,f_1_1)

gun(C3,f_1_1) = 11111010001111010011111000101111000
gun(C3,f_2_1) = 11111000111001100111011000100111100100110001001 10
gun(C3,f_3_1) = 11111000100110101110110111100110110010110111001 10
gun(C3,f_4_1) = 11111110111111001011111101111111010

gun(D3,f_1_1) = 11111000001110000101111000011100000
gun(D3,f_2_1) = 11111000100001101000111100100011010000110001001 10
gun(D3,f_3_1) = 11111000100110001111100110010110011111000111001 10
gun(D3,f_4_1) = 11100110001011101111101100010011010

gun(E3,f_1_1) = 11111010111001111011100011110011011111110001001 10

Lista de Figuras

Lista de Tablas

Bibliografía

[1] Andrew Adamatzky, *Computing in Nonlinear Media and Automata Collectives*, Institute of Physics Publishing, Bristol and Philadelphia, 2001. (ISBN 0-7503-0751-X)

[2] Andrew Adamatzky (Ed.), *Collision-Based Computing*, Springer, 2002. (ISBN 1-85233-540-8)

[3] S. Amoroso and G. Cooper, "The Garden-of-Eden theorem for finite configurations," *Proceedings of the American Mathematical Society*, 1970.

[4] Michael A. Arbib, *Theories of Abstract Automata*, Prentice-Hall Series in Automatic Computation, 1969.

[5] Yaneer Bar-Yam, *Dynamics of Complex Systems*, Perseus Books, 1997. (ISBN 0-201-55748-7)

[6] Elwyn R. Berlekamp, John H. Conway and Richard K. Guy, *Winning Ways for your Mathematical Plays*, Academic Press, 1982 (ISBN 0-12-091152-3) vol. 2, chapter 25.

[7] Michael Brin and Garrett Stuck, *Introduction to Dynamical Systems*, Cambridge University Press, 2002. (ISBN 0-521-80841-3)

[8] E. F. Codd, *Cellular Automata*, Academic Press, Inc. New York and London 1968.

[9] Matthew Cook, "Introduction to the activity of rule 110" (copyright 1994-1998 Matthew Cook), `http://w3.datanet.hu/~cook/Workshop/CellAut/Elementary/Rule110/110pics.html`, January 1999.

[10] Matthew Cook, "Universality in Elementary Cellular Automata," *Complex Systems*, Volume 15, Number 1, pp. 1-40, 2004.

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

[12] J. Demongeot, E. Golès, and M. Tchuente (Eds.) *Dynamical Systems and Cellular Automata*, Academic Press Inc. (London Ltd.), 1985. (ISBN 0-12-209060-8)

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

[14] David Griffeath and Cristopher Moore, *New Constructions in Cellular Automata*, (Santa Fe Institute Studies on the Sciences of Complexity) Oxford University Press, 2003. (ISBN 0-1951-3717-5)

[15] Branko Grünbaum and G. C. Shephard, *Tilings and Patterns*, W. H. Freeman and Company, New York 1987. (ISBN 0-7167-1193-1)

[16] Wim Hordijk, Cosma Rohilla Shalizi and James P. Crutchfield, "Upper Bound on the Products of Paticle Interactions in Cellular Automata," *Physica D* **154**, 240-258, 2001.

[17] Lyman P. Hurd, "Formal Languaje Characterizations of Cellular Automaton Limit Sets," *Complex Systems* **1**, pp. 69-80, 1987.

[18] John E. Hopcroft and Jeffrey D. Ullman, *Introduction to Automata Theory Languajes, and Computation*, Addison-Wesley Publishing Company, 1987. (ISBN 0-201-02988-X)

[19] Genaro Martínez and Harold V. McIntosh, "ATLAS: Collisions of gliders like phases of ether in Rule 110," `http://uncomp.uwe.ac.uk/genaro/papers.html`, August 2001.

[20] Genaro Martínez, Harold V. McIntosh, Juan C. Seck Tuoh Mora and Sergio V. Chapa Vergara, "Determining a regular language by gliders structures called phases f_i_1 in Rule 110," submitted to *Journal of Cellular Automata*, August 2006.

[21] Genaro Martínez, Harold V. McIntosh, Juan C. Seck Tuoh Mora and Sergio V. Chapa Vergara "Reproducing the cyclic tag systems developed by Matthew Cook with Rule 110 using the phases f_i_1," by submit to *International Journal of Unconventional Computing*.

[22] Genaro Martínez, Harold V. McIntosh and Juan C. Seck Tuoh Mora, "Production of gliders by collisions in Rule 110," *Lecture Notes in Computer Science* **2801**, pp. 175-182, 2003.

[23] Genaro Martínez, Harold V. McIntosh and Juan C. Seck Tuoh Mora, "Gliders in Rule 110," *International Journal of Unconventional Computing*, Vol. 2, Num. 1, pp. 1-49, January 2006.

[24] Genaro Martínez, Harold V. McIntosh, Juan C. Seck Tuoh Mora and Sergio V. Chapa Vergara, "Rule 110 objects and other construccions based-collisions," *Journal of Cellular Automata*, by publish, 2006.

[25] Genaro Martínez, "Grados de Reversivilidad en Autómata Celular Lineal," Tesis de Licenciatura, Facultad Acatlán, Universidad Nacional Autónoma de México, 1998.

[26] Christopher G. Langton, "Self-Reproduction in Cellular Automata," *Physica D* **10**, pp. 135-144, 1984.

[27] Kristian Lindegren, Cristopher Moore and Mats Nordahl "Complexity of Two-Dimensional Patters," Working Paper, Santa Fe Institute, Santa Fe, New Mexico, May 30, 2000.

[28] Kristian Lindgren and Mats G. Nordahl, "Universal Computation in Simple One-Dimensional Cellular Automata," *Complex Systems* **4**, pp. 229-318, 1990.

[29] Wentian Li and Mats G. Nordahl, "Transient behavior of cellular automaton rule 110," *Physics Letters A* **166**, 335-339, (1992).

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

[31] Harold V. McIntosh, "Linear cellular automata via de Bruijn diagrams," `http://delta.cs.cinvestav.mx/~mcintosh/oldweb/pautomata.html`, 1991.

[32] Harold V. McIntosh, "Rule 110 as it relates to the presence of gliders," `http://delta.cs.cinvestav.mx/~mcintosh/oldweb/pautomata.html`, January 1999.

[33] Harold V. McIntosh, "A Concordance for Rule 110," `http://delta.cs.cinvestav.mx/~mcintosh/oldweb/pautomata.html`, 2000

[34] Michael Brin and Garrett Stuck, *Introduction to Dynamical Systems*, Cambridge University Press, 2002. (ISBN 0-521-80841-3)

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

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

[37] Edward F. Moore, "Machine models of self-replication," American Mathematical Society, *Proceedings of Symposia in Applied Mathematics* **14**, pp. 17-33, 1962.

[38] U. Pesavento, "An implementation of von Neumann's self-reproducing machine," *Artificial Life* **2**, 337-354, 1995.

[39] Emil L. Post, "Formal reductions of the general combinatorial decision problem" *American Journal of Mathematics*, vol. 65, pp. 197-215, 1943.

[40] William Poundstone, *The Recursive Universe*, William Morrow and Company, Inc. New York, 1985. (ISBN 0-688-03975-8)

[41] James K. Park, Kenneth Steiglitz and William P. Thurston, "Soliton-like behavior in automata," *Physica D* **19**, 423-432, 1986

[42] Claude E. Shannon, "A Mathematical Theory of Communication," *Bell Systems Technical Journal* **27**, pp. 379-423 623-658, 1948.

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

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

[45] Alan M. Turing, "On Computable numbers, with an application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, Ser. 2, vol. 42, pp. 230-265, 1936. Corrections, Ibid, vol 43, pp. 544-546, 1937.

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

[47] Burton H. Voorhees, *Computational analysis of one-dimensional cellular automata*, World Scientific Series on Nonlinear Science, Series A, Vol. 15, 1996. (ISBN 981-02-2221-1)

[48] Andrew Wuensche and Mike Lesser, *The Global Dynamics of Cellular Automata*, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley Publishing Company, 1992. (ISBN 0-201-55740-1)

[49] Stephen Wolfram, "Universality and Complexity in Cellular Automata," *Physica D* **10**, pp. 1-35, 1984.

[50] Stephen Wolfram, "Computation Theory on Cellular Automata," *Communication in Mathematical Physics*, volume 96, pp. 15-57, November 1984.

[51] Stephen Wolfram, *Theory and Aplications of Cellular Automata*, World Scientific Press, Singapore, 1986. (ISBN 9971-50-124-4 pbk)

[52] Stephen Wolfram, *Cellular Automata and Complexity*, Addison-Wesley Publishing Company, 1994.

[53] Stephen Wolfram, *A New Kind of Science*, Wolfram Media, Inc., Champaign, Illinois, 2002. (ISBN 1-57955-008-8)

[54] Andrew Wuensche, "Classifying Cellular Automata Automatically," *Complexity*, vol. 4, no. 3, pp. 47-66, 1999.

[55] Andrew Wuensche, "Self-reproduction by glider collisions; the beehive rule," *Alife9 Proceedings*, pp. 286-291, MIT Press, 2004.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL IPN
DEPARTAMENTO DE
INGENIERíA ELÉCTRICA

El comité certifica que han revisado el trabajo realizado y recomiendan a la Sección Computación aceptar esta tesis titulada "**Procedimiento para producir comportamientos complejos en Regla 110**" por **Genaro Juárez Martínez**, cumpliendo los requerimientos para obtener el grado de **Doctor en Ciencias**.

Asesor: _____
Harold V. McIntosh

Asesor: _____
Sergio V. Chapa Vergara

Comité revisor: _____
Germán González Santos

_____
José O. Olmedo Aguirre

_____
Carlos A. Coello Coello

_____
Manuel González Hernández