UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

UNIVERSALIDAD EN AUTOMATAS CELULARES

T E S I S

QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACION

P R E S E N T A:

EMMANUEL GARCES MEDINA

FACULTAD DE CIENCIAS
UNAM

DIRECTOR DE TESIS:

DR. PEDRO EDUARDO MIRAMONTES VIDAL

2005

VNIVERSIDAD NACIONAL
AVFNOMA DE
MEXICO

ACT. MAURICIO AGUILAR GONZÁLEZ
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito:

Universalidad en Autómatas Celulares

realizado por Emmanuel Garcés Medina

con número de cuenta 09957441-3 , pasante de la carrera de
Lic. en Ciencias de la Computación.
Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis Propietario	Dr. Pedro Eduardo Miramontes Vidal
Propietario	Dr. Germinal Cocho Gil
Propietario	M. en C. José Antonio Neme Castillo
Suplente	M. en C. José de Jesús Galaviz Casas
Suplente	Dr. Francisco Hernández Quiroz

Consejo Departamental de Matemáticas

Dr. Francisco Hernández Quiroz

FACULTAD DE CIENCIAS
CONSEJO DEPARTAMENTAL
DE
MATEMATICAS

Agradecimientos

A todas las personas que revisaron este trabajo.
En especial al doctor Pedro Miramontes por iniciarme
en la dinámica no lineal de los sistemas complejos.

A mi familia,
A mis amigos.

Universalidad en autómatas celulares

Emmanuel Garcés Medina

Índice general

Capítulo 1

Introducción

La complejidad

La naturaleza ha dotado al mundo de una extraordinaria variedad de diseños, configuraciones, estructuras, patrones, mecanismos, funciones, etcétera. Todos estos trabajan acoplados de alguna manera con otros y desarrollan tareas específicas, algunos crecen, se adaptan, se reproducen y otros construyen cosas nuevas. Estos mecanismos no son sencillos pues generalmente constan de muchas partes y contienen varios niveles de organización de muchos tamaños que son generalmente difíciles de definir. Ante toda esta complejidad, el estudio de un sistema en su totalidad resulta difícil y tedioso, tanto al investigar cómo funcionan las cosas como al hacer una clasificación de las partes del sistema. Sin embargo, ese no es el único objetivo de la ciencia. También existe interés por conocer los principios generales que rigen la emergencia de la complejidad.

La complejidad de la naturaleza es importante en muchos sentidos, sobre todo porque da la capacidad de construir y representar cosas, pero hay más allá. Para un artista la complejidad es sinónimo de diversidad de formas

que permiten que un cuadro pintado con muchos matices y trazos sea más apreciable comparado con el dibujo de una simple figura geométrica. Para un sociólogo, una sociedad compleja es manifestación de gran actividad social, lo mismo para un psicólogo que examina las conductas de una persona y un economista que analiza las fluctuaciones de la bolsa de valores. Los comportamientos complejos implican dinamismo y diversidad en los fenómenos. Además dan robustez en los sistemas y amplían el horizonte de las posibilidades.

La complejidad aparece en todas las disciplinas. La biología estudia a los seres vivos, todos ellos están asociados con manifestaciones altas de complejidad debido a que las estructuras sencillas no son capaces de soportar el menor indicio de vida. La química hace uso de las estructuras moleculares complejas para construir sustancias que nos hacen la vida más fácil, una molécula sencilla no sirve de mucho porque su estructura simple no se puede acoplar fácilmente con otras. La historia nos muestra la riqueza compleja de acontecimientos del hombre a través de los años para luego aprender de ello, si la historia no fuese compleja no tendría caso estudiarla porque no habría mucha distinción entre los distintos sucesos. La filosofía nos brinda el pensamiento complejo que la humanidad ha desarrollado y que necesita de la diversidad de corrientes para fortalecer la discusión. En resumen, todas las disciplinas se fortalecen con estructuras ricas en complejidad para existir.

La complejidad puede manifestarse tanto en la forma de las cosas como en la manera en que estas se comportan. Así por ejemplo, una manzana en movimiento manifiesta mayor complejidad que una manzana en reposo, pero menos complejidad que un árbol. Es difícil dar una medidad exacta de la cantidad de complejidad asociada a algo. En este trabajo no se discuten los intentos

por hacer mediciones rigurosas a la complejidad, pero sí se asume una idea intuitiva de lo que esto representa.

Para hablar de la complejidad nos hacemos valer de niveles de descripción. Usualmente, los niveles microscópico y macroscopico son los que importan, pero poco se sabe acerca de la transición desde micro hasta macro estructuras. Es esa región intermedia o mesoscópica la que nos puede dar mucha información acerca de cómo se crean las cosas y como adquieren sus propiedades. En el presente trabajo se muestran algunos de los mecanismos que permiten el surgimiento de nuevas características a partir de simples reglas de interacción.

La emergencia de la complejidad

La naturaleza en nuestro organismo se puede clasificar por niveles de descripción. Desde el nivel químico hasta la conformación de sistemas como la respiración, circulación o digestión, pasando por el nivel celular, de tejidos y órganos. Los elementos de cada nivel mantienen características que en niveles inferiores no tienen sentido y que en niveles superiores carecen de relevancia. Por ejemplo, la auto replicación que ocurre con las células nunca se lleva a cabo con las proteínas ni con otras biomoléculas, pero también es irrelevante el estudio de la auto reproducción celular al explicar el funcionamiento de los órganos. Al origen de propiedades que tienen sentido en un nivel de complejidad y que no aparecen en niveles inferiores se le llama *emergencia de patrones.*

Muchos de los sistemas complejos adquieren sus características emergentes a pesar de que ninguno de sus componentes conoce la existencia de los demás, excepto los más próximos a él. Por ejemplo, en un hormiguero cada hormiga

sólo interacciona con aquellas hormigas que están a su alrededor y sin embargo, el resultado final de esta convivencia local da lugar a una actividad compleja colectiva e inclusive a formas de organización creadas automáticamente. El ejemplo del hormiguero es típico cuando se habla de fenómenos de autoorganización, que también suele surgir en otras sociedades de insectos sociales como las abejas y más aún en las sociedades humanas.

La emergencia de patrones implica también emergencia de complejidad. En la naturaleza es suficiente la contemplación de las manchas de una cebra, una mariposa o un pez para entender el grado de complejidad en las formas que se puede lograr y uno se puede explicar como estas pigmentaciones se producen siendo el resultado de la dinámica de muchas sustancias químicas que reaccionan. Los mismo pasa con las demás partes del organismo. Más aún, la naturaleza ha permitido la existencia de seres que además de todo lo anterior son capaces de organizarse en sociedades, adquirir conciencia, lenguaje, inteligencia, etcétera. Estas características son en principio causadas por motivos biológicos, pero el entorno en que un organismo interactúa forma también parte de la dinámica que propicia la aparición de tales propiedades. La cultura de una comunidad, las normas sociales, la educación, son ejemplos de factores ambientales que estimulan el desarrollo de otro tipo de características que nos hacen ser humanos.

El concepto de *patrón* debe entenderse ampliamente como algo que podemos distinguir de lo demás. Dicho esto, existen patrones espaciales, patrones temporales, patrones de comportamiento (no confundir con funcionalidad) y cualquier otro que se nos ocurra. Los patrones espaciales aparecen en todos lados. Basta voltear la vista a cualquier objeto para distinguirlo de los demás.

Los patrones temporales son apreciables al avanzar el tiempo, por ejemplo, el ritmo del corazón que exhibe movimientos de contracción y relajación irregulares. Un patrón de comportamiento nos dice el modo en que se comportan las cosas y que podemos etiquetar como leyes. Por ejemplo, la ley de gravedad se puede distinguir de las leyes electrostáticas, etcétera. Además podemos también pensar en los patrones de conducta que identifican los psicólogos, los patrones de comportamiento social que clasifican los sociólogos, los patrones históricos que buscan los historiadores, los patrones temporales de los indicadores que analizan los economistas, los patrones pintados en un cuadro que admiran los artistas, etc. La identificacóon de patrones (de aquí en adelante también llamados configuraciones o estructuras) tiene lugar en cualquier rama, disciplina o actividad humana. Una clasificación general de los tipos de patrones junto a un estudio general de la forma en que interactúan tales patrones podría ayudarnos a entender mejor cosas que son aparentemente distintas y que se estudian con disciplinas diferentes.

El mundo biológico es el ejemplo inmediato de emergencia de patrones. La vida misma es una clara evidencia de que pueden surgir cosas bastamente extraordinarias a partir de principios básicos. Pero la biología no es la única disciplina que nos enseña a contemplar el surgimiento de nuevas estructuras. La física de partículas describe quizás el más pequeño de los niveles de organización que se conozca en el universo. Para entender esto debemos partir de las partículas más elementales conocidas: los quarks, y de como estos se juntan para formar a los electrones en un siguiente nivel de organización. De los electrones pasamos a los átomos y luego a las moléculas para saltar a la química. En esta rama de la ciencia nos importa la forma en que interactúan los conglomerados moleculares para generar sustancias, que a niveles más altos

pueden formar patrones espaciales (como las manchas del tigre). En la psicología y la sociología se pueden distiguir también evidencias de la generación espontánea de formas complejas. Ambas disciplinas no se pueden reducir a neuronas o personas respectivamente, lo que cada una estudia es el resultado de la interacción de enormes cantidades de sus respectivas unidades atómicas.

En principio, cada nivel de organización inferior determina al siguiente nivel. Esto quiere decir que los patrones de un nivel junto con la interacción que ellos tienen generan nuevos patrones y nuevas reglas de interacción para el siguiente nivel y así sucesivamente. Es decir, que los niveles inferiores dan las condiciones de frontera para los niveles superiores.

Hay varios caminos para que la emergencia de patrones ocurra. Muchas veces tenemos un conjunto homogéneo de cosas interactuando y en unos instantes se rompe la homogeneidad del conjunto, en otras palabras, la simetría que imperaba se rompe para dar lugar a formas que podemos distinguir. A este proceso se le conoce como ruptura de simetría. Un fenómeno en la biología que ejemplifica lo anterior es el crecimiento de un embrión a partir de la unión de dos células reproductoras. Inicialmente la célula fecundada se duplica iteradamente para formar una esfera homogénea, la blástula, pero pasado cierto tiempo, la interacción entre células más el medio provocan una ruptura de simetría dando lugar a un cambio de forma de la esfera y a una especialización de las células para formar tejidos y órganos.

Las rupturas de simetría están asociadas a un cambio cualitativo del sistema. Un cambio se presenta casi instantáneamente cuando el sistema se encuentra en estado crítico, esto es, cuando el sistema se encuentra transitando entre

dos estados distintos. Por ejemplo, el momento en que el agua deja de serlo para convertirse en vapor o cuando un líquido deja de mostrar patrones regulares para convertirse en turbulencia o una sociedad en el momento que estalla una revolución. En ese momento, dentro del sistema, ocurren fluctuaciones de muchos tamaños. Muchas de esas fluctuaciones podemos medirlas y registrarlas en un histograma para darnos cuenta que están distribuidos siguiendo una *ley de potencia*, es decir, una distribución del tipo a^b, lo que implica independencia de escalas y autosemejanza como en los fractales.

Cuando un sistema se encuentra en estado crítico también se maximiza su capacidad de portar información. En ese momento la complejidad del sistema aumenta gracias a que nuevas formas y nuevas dinámicas tienen lugar. También la conectividad del sistema se incrementa, del tal forma que pequeños cambios en un elemento del sistema pueden repercutir en otra parte muy rápidamente. A estas manifestaciones se les puede sacar mucho provecho. Esta tesis habla acerca de como podemos tomar ventaja de tal comportamiento de un sistema dinámico muy particular.

Todos los sistemas que se han presentado para ejemplificar la emergencia de patrones, autoorganización, ruptura de simetría, caos, etc., son conocidos como sistemas dinámicos complejos. No existe una definición clara y formal para este concepto, pero podemos intuirlo. Un sistema complejo es aquel cuyas partes interactúan de manera no trivial, o bien, que la consecuencia de la interacción no es proporcional a la causa que lo provoca. Cuando la causa es proporcional se dice que la interacción es lineal, por ejemplo, la aceleración de un objeto es proporcional a la fuerza que lo empuja. En un sistema complejo esto no pasa, es como si la fuerza aplicada a un objeto tuviera una manifesta-

ción errática.

Actualmente gran parte de la ciencia ha adoptado el punto de vista reduccionista, el cual asegura que todo puede ser explicado en función de sus partes. Pero todo el estudio de los sistemas complejos contradice este enfoque. Las ciencias y cualquier otra disciplina se pueden estudiar de otra manera: Identificando su complejidad, analizando la emergencia de estructuras, encontrando el origen de las dinámicas que rigen su funcionamiento y buscando analogías entre procesos equivalentes (universalidad). Tal enfoque elimina las barreras que existen entre las disciplinas. Este tendido de puentes rompe con un enfoque separado de las disciplinas y nos lleva a terrenos de la transdisciplina, aún vagamente explorados.

La universalidad

Cuando un sistema dinámico se encuentra en estado crítico, este adquiere tal actividad que es posible encontrar equivalencias con otros sistemas dinámicos en el mismo estado. Se le llama comportamiento universal aquel que encontramos equivalente en todos los sistemas, independientemente de la naturaleza de cada uno. El concepto de universalidad es una de las principales facetas con que la que se reconoce a un sistema complejo en estado crítico. Constituye la noción que nos permite encontrar equivalencias entre sistemas de distinta naturaleza y que sin embargo bajo ciertos aspectos se comportan de manera similar.

Esta tesis dirige su atención a un tipo especial de sistema dinámico complejo llamado autómata celular. Un autómata celular es sistema dinámico similar a un conglomerado de células o celdas, y cuya configuración está determinada

por el estado individual de sus componentes. Cada celda del autómata celular evoluciona en función de los estados de sus celdas adyacentes y de su propio estado. Los parámetros que alteran este sistema complejo dinámico son las reglas de evolución de las celdas. Al hacer una modificación en estos parámetros podemos construir autómatas celulares en estado crítico y encontrar en ellos una cierta clase de universalidad que nos permite usar a los autómatas celulares como máquinas capaces de realizar computación, lo que implica también capacidades de construcción y auto reproducción de otras configuraciones.

El presente trabajo trata sobre la universalidad computacional de los autómatas celulares, es decir, la capacidad que algunas de estas máquinas tienen para eventualmente simular una computadora. Dicha propiedad universal de cómputo también se sospecha inherente a los hormigueros o a cualquier otro sistema dinámico en estado crítico (la vida inclusive).

En los siguientes capítulos, se comienza por definir formalmente el concepto de autómata celular, la manera en que estos se comportan y como puede emerger complejidad de estos. Más adelante, se fija la noción de computabilidad acercando los modelos formales más conocidos, entre ellos la máquina de Turing, que es el paradigma dominante en la teoría de la computación. En el último capítulo muestro los trabajos de tres científicos sobre universalidad computacional en autómatas celulares, dando en detalle considerable la construcción de sus modelos. Por último se discute sobre la capacidad de auto reproducción y construcción de los autómatas celulares.

Capítulo 2

Autómatas celulares

2.1. Introducción

Los autómatas celulares (AC) fueron introducidos por John Von Neumann [22] y Stanislaw Ulam para modelar procesos biológicos tales como la autorreplicación de organismos. El objetivo de estos científicos era encontrar un mecanismo idealizado que generase comportamientos complejos a partir de reglas o interacciones simples entre elementos iguales y cercanos espacialmente. Un autómata celular es entonces, un mecanismo matemático, cuyos componentes son elementos discretos que evolucionan a través del tiempo de manera discreta mediante reglas locales. El propio nombre de los autómatas celulares delata su naturaleza; un autómata celular es una máquina cuyos elementos consisten en celdas o células adyacentes. En estos modelos el tiempo avanza de forma cuantizada. Por cada clic del reloj, las celdas modifican su valor de acuerdo a reglas locales previamente establecidas. Tales reglas sólo tienen dominio local, es decir sólo actúan en la vecindad de alguna celda. El resultado del autómata en el transcurso del tiempo es la evolución en paralelo de todas las celdas bajo la acción de las reglas actuando localmente.

Si uno tuviese frente a sí el estado actual de un AC, uno vería a todas las celdas cambiar con el paso del tiempo y seguramente sería complicado concebir el grado de complejidad emergente. Sin embargo, los AC se suelen representar de manera gráfica al describir todos los estados de las celdas respecto al tiempo, de esta manera es fácil encontrar atractores simples, periódicos, patrones complejos o caos. Justamente esta tesis apela mucho a aquellos AC donde el comportamiento dista mucho de ser estable o periódico, es justamente la complejidad la que genera la mayor cantidad de fenómenos en la naturaleza. En particular se tratará con aquellos que son capaces de imitar a las computadoras.

Los resultados que se obtienen al representar la evolución de un AC gráficamente son interesantes e inesperados, sobre todo cuando las reglas de evolución no son lineales. Tales propiedades han llevado a la teoría de autómatas celulares a ser una de las más importantes en las ciencias de la computación, los sistemas complejos y muchas otras ramas de la ciencia.

El uso de los AC es extenso. Actualmente se usan como base para modelar fenómenos físicos, químicos y biológicos; fluidos, gases, formación de galaxias, trenes de ondas y formación de patrones. Su estudio ha permitido entender los procesos de autoorganización, comportamientos emergentes y cooperación en colectivo.

La simplicidad de los autómatas celulares es una ventaja que permite hacer simulaciones computacionales con facilidad, y cuyas propiedades formales pueden ser demostradas matemáticamente.

El propósito de esta tesis es mostrar que existen autómatas celulares que bajo ciertas condiciones se comportan de manera similar a una máquina universal de Turing. Esta hipótesis se ha manejado mucho en la literatura, principalmente con Von Neumann, Ulam y Stephen Wolfram.

2.2. El autómata celular elemental

El autómata celular elemental consiste de una hilera de celdas, que bien puede ser representada como un arreglo unidimensional y donde cada celda asume sólo uno de dos valores, 0 y 1 por ejemplo. Dadas estas condiciones, cualquier arreglo unidimensional binario es un estado inicial donde comienza la ejecución de algún autómata celular elemental.

Las reglas para el autómata celular elemental actúan localmente sobre vecindades de radio 1 (primeros vecinos). Es decir, que el valor de una celda se modifica sólo en función del valor de ella misma y de las dos celdas adyacentes.

Supongamos que el tamaño de la hilera de celdas del AC (arreglo unidimensional) es n, la notación para el AC elemental que usaremos es la siguiente:

$$X_0^t\ X_1^t\ X_2^t \ldots X_{n-2}^t\ X_{n-1}^t$$

Donde $X_i^t \in \{0, 1\}$ es el valor de la i-esima celda del AC elemental en el tiempo t.

Las reglas para el AC elemental se definen con una función φ cuyo dominio es el conjunto de las vecindades de radio 1 en el arreglo unidimensional $\{X_i^t\}$

y cuya imagen es el conjunto $\{0,1\}$.

$$\varphi : \{0,1\}^3 \longrightarrow \{0,1\}$$

El mecanismo del AC elemental opera de la siguiente manera: Para cada elemento del arreglo unidimensional se aplica φ a su vecindad y el valor obtenido se sustituye en el lugar del elemento original. Hay que notar que la vecindad de radio 1 está bien definida para todas las celdas, excepto para la primera y la última. Debido a esto, la hilera de celdas se cierra en sí misma y se opera sobre un arreglo unidimensional circular. Dicho esto, la función que define las reglas del AC queda de la siguiente manera,

$$X_i^{t+1} = \varphi(X_{i-1}^t, X_i^t, X_{i+1}^t)$$

Donde la condición de frontera habitual es:

$$X_{-1}^t = X_{n-1}^t \text{ y } X_n^t = X_0^t$$

La función φ se puede escribir explícitamente con una tabla que describa el resultado de la función para cada uno de los posibles valores de su dominio. Y como tal conjunto consiste en todas las tercias con valores 0 o 1, entonces el tamaño de la tabla debe ser $2^3 = 8$. Por lo tanto, el número total de funciones posibles para el AC elemental es igual al número total de tablas de tamaño 8. Y como la función sólo puede tomar dos valores, entonces el número total de tablas es igual a la cantidad de arreglos de tamaño 8 y cuyas celdas sólo pueden tomar dos valores, es decir $2^8 = 256$. Por esta razón, todos los AC elementales pueden ser estudiados exhaustivamente por una computadora.

Si sólo hay 256 posibles reglas para un AC elemental, entonces podemos hacerles referencia mediante algún número. La forma estándar y bien conocida de hacer esto es fijarnos en los valores de la función al listarlos en la tabla de tamaño 8. Por ejemplo, supongamos que nuestra función tiene asociada la siguiente tabla:

Vecindad	φ
000	1
001	0
010	1
011	1
100	0
101	1
110	0
111	1

Entonces, el número en base dos formado por los valores de la función tomados de arriba hacia abajo y colocados de izquierda a derecha es, $10110101_2 = 181_{10}$. Y como esta es la única asignación que genera tal número, entonces decimos que el ejemplo anterior ilustra la tabla de la regla 181 para el autómata celular elemental.

Aún siendo muy pequeño el espacio de posibles AC elementales, existe gran diversidad que vale la pena explorar. En las siguientes imágenes se muestran algunos ejemplos. La figura 2.1 muestra las configuraciones espaciales de un autómata celular que genera una estructura ordenada conocida como el triángulo de Sierpinski. La figura 2.2 muestra el desarrollo del mismo autóma-

ta pero con una condición inicial aleatoria que sin embargo, a pesar de esto, sigue conservando regularidades en su evolución.

En la figura 2.4 se aprecia la evolución de un autómata celular cuya estructura parece no cambiar mucho conforme avanza el tiempo. La figura 2.3 muestra los patrones que se generan durante la evolución de un autómata celular cuya celda central varia manera indistinguible del azar y donde se exhibe una asimetría lateral en una estructura que crece hacia ambos lados.

La figura 2.5 muestra el desarrollo de otro autómata celular donde se aprecia un patrón complejo que cuelga sobre un fondo de triángulos ordenados sumergidos en una estructura que crece hacia la izquierda.

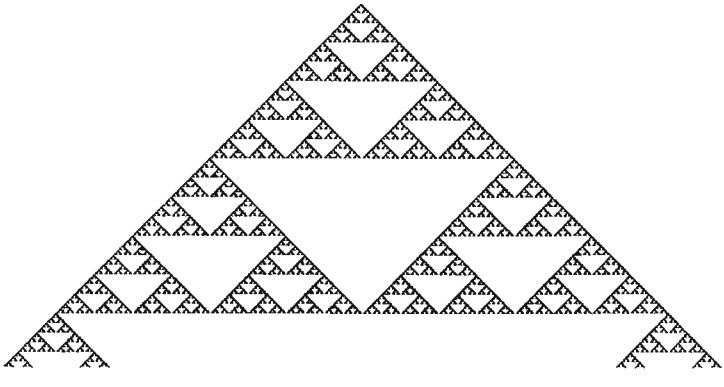

Figura 2.1: Evolución del autómata celular elemental con la regla 22 a partir de un estado inicial con todas las celdas en 0, excepto una, la central. El patrón formado por los estados del autómata a través del tiempo es el triángulo de Sierpinski.

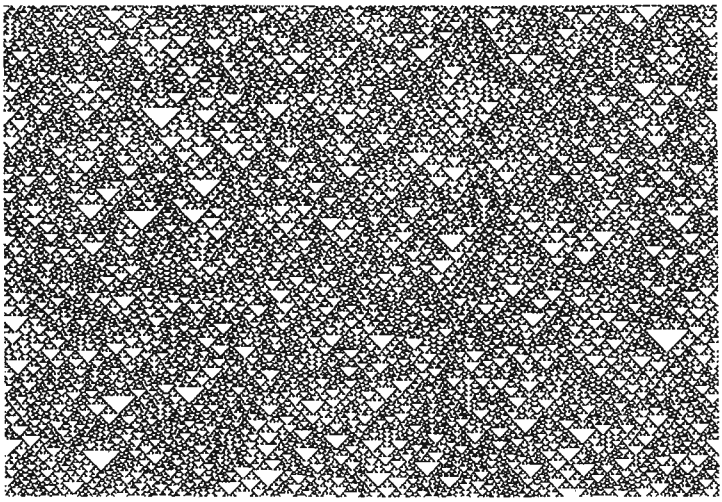

Figura 2.2: Evolución del autómata celular elemental con la regla 22 a partir de un estado inicial totalmente aleatorio.

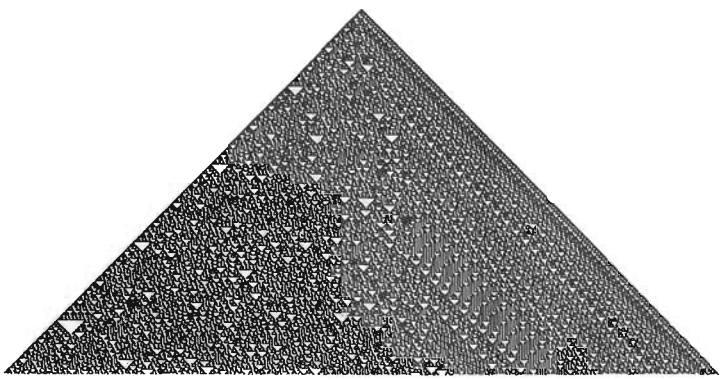

Figura 2.3: Evolución del autómata celular elemental con la regla 86 a partir de un estado inicial con todas las celdas en 0, excepto una, la central.

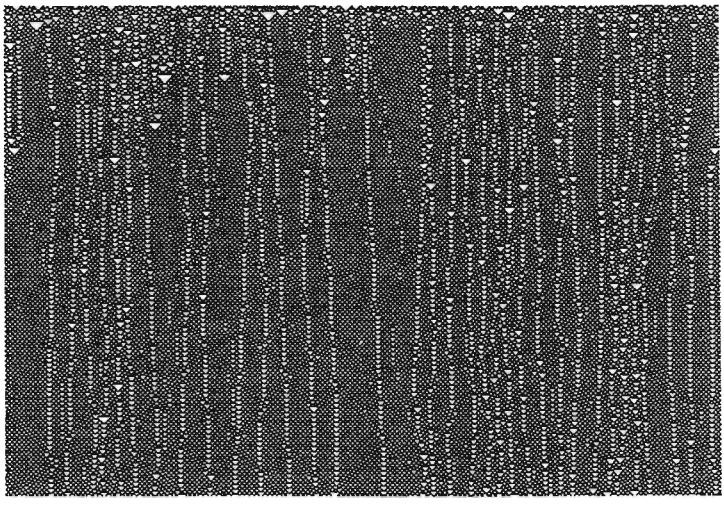

Figura 2.4: Evolución del autómata celular elemental con la regla 54 a partir de un estado inicial totalmente aleatorio.

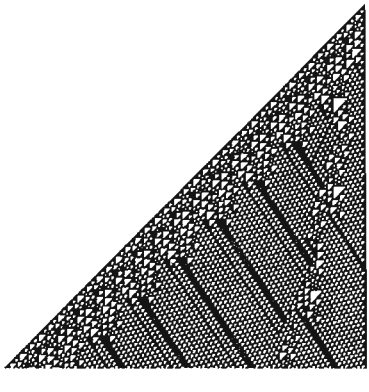

Figura 2.5: Evolución del autómata celular elemental con la regla 110 a partir de un estado inicial con todas las celdas en 0, excepto una, la central.

2.3. Autómatas celulares bidimensionales

El autómata celular elemental es el modelo más simple que se pueda construir. Cualquier variación de este producirá comportamientos distintos. Si se cambia el tamaño de la vecindad o se da más libertad a los valores de las celdas, entonces el autómata será cualitativamente diferente.

Otro tipo de autómatas celulares que son comúnmente estudiados son aquellos cuyas celdas se acomodan en un arreglo bidimensional y las reglas locales actúan sobre vecindades igualmente bidimensionales. Comúnmente se usan las vecindades de Moore y Von Neumann, pues ambas son análogas a aquellas que se usan en los AC elementales.

Para efectos prácticos, el arreglo bidimensional de estos autómatas se cierra en sí mismo formando un cilindro, en analogía con un anillo cerrado del AC elemental. Aunque las condiciones de frontera pueden ser de lo más variadas.

El empleo de los autómatas celulares bidimensionales es más extenso comparado con los AC elementales. La razón es que muchos procesos de la naturaleza son mapeables a superficies. Como ejemplos se incluyen los sistemas de reacción y difusión, crecimiento de cristales y flujos. Tales fenómenos generalmente se modelan empleando sistemas de ecuaciones diferenciales parciales, pero la implementación numérica de estas se obtiene con métodos que usan mecanismos muy parecidos a los de un autómata bidimensional.

2.4. Clasificación de Wolfram

En 1984, Stephen Wolfram publicó un artículo llamado **Universalidad y complejidad en autómatas celulares** [33]. En este documento, Wolfram conjeturó la posibilidad de clasificar los autómatas celulares en categorías de acuerdo a su comportamiento. Una clasificación con tal propósito tiene que asumir categorías y agrupar aquellos AC que tengan el mismo comportamiento cualitativo. En palabras de Wolfram **Esta universalidad implica que muchos detalles de la construcción de un autómata celular son irrelevantes para determinar su comportamiento cualitativo. Por lo tanto, sistemas físicos y biológicos complejos tienen que caer en las mismas clases de universalidad tal como los modelos matemáticos idealizados provistos por autómatas celulares.**

Wolfram supone la existencia de cuatro clases de AC. Las cuatro clases caracterizan cualitativamente los atractores que emergen de la dinámica de los AC. En la clase I los atractores son puntos límite, es decir el autómata converge a un estado final del cual no cambia debido al determinismo de las reglas. La clase II considera a los AC cuyos atractores son ciclos límite, o periódicos. Todos los autómatas celulares que evolucionen en un espacio limitado perteneceran a esta clase, pues la finitud del espacio fase implica necesariamente repetición de estados en algún momento. La clase III caracteriza aquellos autómatas celulares cuyo atractor es caótico, son aquellos que no presentan periodicidad alguna y son sensibles a las condiciones iniciales. Muchas veces el comportamiento de los autómatas de esta clase parecen tender a la aleatoriedad. Por último, la clase IV, la que nos interesa, es justamente la que caracteriza a aquellos AC que exhiben el comportamiento más complejo. Un equilibrio entre el orden y el caos.

De igual manera, las distintas clases de Wolfram describen distintos niveles de predicción. Evidentemente, para la primera clase el estado final está completamente determinado. Para la segunda clase, el valor de una celda está determinado por alguna región de la condición inicial del autómata. Para la tercera clase, el valor de una celda depende de un número creciente de celdas iniciales, de tal forma que valores iniciados aleatoriamente hacen tender al caos. En algunos casos, las celdas de un autómata de clase 3 se pueden determinar mediante algoritmos. Finalmente, la cuarta clase de autómatas celulares necesita de algoritmos igual o más complejos que el autómata mismo para determinar el valor de alguna celda. La predicción en esa clase de autómatas únicamente se logra haciendo la simulación explícita y esa es la razón para asociar la irreducibilidad de un proceso de cómputo con la evolución del autómata.

La sensibilidad a condiciones iniciales es diferente en cada clase. En el primer caso la dinámica del autómata converge a un solo estado final independientemente de los cambios aplicados a las condiciones iniciales. En el segundo caso, los cambios en la condición inicial solo afectan una determinada zona del sistema. En el tercer caso, la sensibilidad a condiciones iniciales es típica de un sistema caótico. Los cambios aplicados inicialmente eventualmente afectan a todo el sistema conforme avanza el tiempo. En el cuarto caso, los cambios también se dispersan por el sistema pero de manera esporádica, igualmente compleja e intermedia entre el comportamiento de la clase II y la clase III.

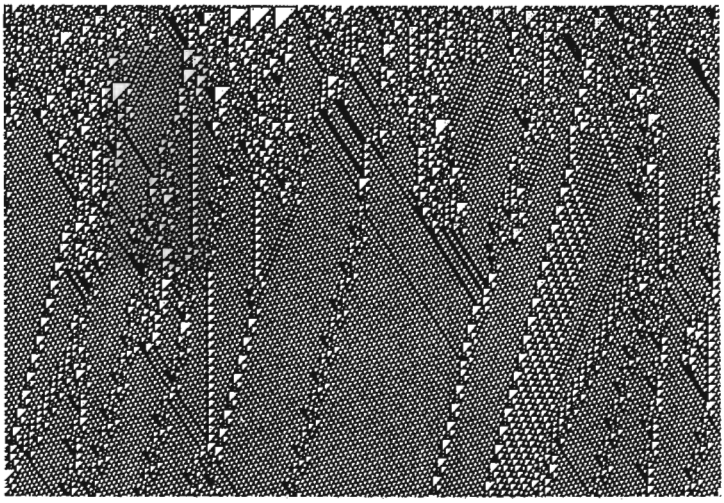

Figura 2.6: Evolución del autómata celular elemental con la regla 110 a partir de un estado inicial con todas las celdas iniciadas aleatoriamente. El mecanismo exhibe tanto estructuras que se propagan como zonas con aparente aleatoriedad.

2.5. Autómatas celulares de clase IV

Mucho se ha hablado en la literatura de que un sistema en transición partiendo de un estado estable a un estado caótico exhibe comportamientos complejos, son aquellos instantes justo en la frontera entre el orden y el caos donde se aprecian la más diversas manifestaciones de complejidad y se maximizan las capacidades del sistema para portar información. Los autómatas celulares son sistemas complejos que no eluden estas afirmaciones, en tal caso, los casos más interesantes y aquellos donde la complejidad es mayor, son aquellos donde ni el orden ni el caos se revelan de manera exclusiva. De acuerdo a la clasificación de Wolfram, los autómatas celulares que están en el borde del caos son los que pertenecen a la clase IV.

En la sección anterior se ha mostrado la manera en que evolucionan algunos autómatas celulares considerados en la clase IV. En ellos se pueden ver algunos patrones espaciales periódicos, algunos otros son patrones que persisten y otros desaparecen. En resumen, la riqueza de estructuras que pueden surgir es amplia, además, el tipo de interacciones que ocurren entre tales estructuras es variado de igual forma. Sin embargo, a pesar de todo lo anterior, sólo existe un número limitado de estructuras que nos podrían ser útiles para almacenar información o hacer cómputo. Por otro lado, puesto que estamos tratando con un sistema determinista, toda la evolución del autómata celular depende de la condición inicial, por lo que debemos conocer la manera en que cambia el comportamiento del autómata cuando cambian las condiciones iniciales.

En los últimos años han habido numerosos intentos para proveer bases cuantitativas a la clasificación de Wolfram o para proponer esquemas alternativos de clasificación. Un primera aproximación se debe a Cullick II y Yu, quienes han intentado un acercamiento teórico de computación como una base de la clasificación la cual coincide con Wolfram para los autómatas celulares totalisticos. En esta clasificación todos los autómatas computacionalmente universales pertenecen a la clase 4.

El otro intento de acercamiento a la cuantificación de las clases de Wolfram lo hace Langton y sus colaboradores, quienes hicieron un estudio estadístico de las reglas k, r espaciales de AC. Ellos usaron varias cuantificaciones de teoría de la información tales como la entropía de Shannon e información mutualista.

Dada una tabla de reglas, Langton introdujo un parámetro λ, definido como la fracción de reglas en la tabla que son mapeadas a un estado no estacionario.

Las reglas con un valor bajo λ cercanos a cero pertenecen o bien a la clase 1 o 2, mientras que aquellos con un alto valor de λ cercano a 1 pertenecen a la clase 3. Para las reglas de la clase cuatro se espera que el parámetro λ ocurra en valores intermedios.

A una colección de reglas generadas aleatoriamente, cada una con un valor de λ barriendo el invervalo $0,1$ se le llama una lambda-cadena. Si ahora pensamos en el espacio abstracto donde cada punto representa una regla de autómata celular, entonces una lambda-cadena particular podría ser vista como una curva dentro del espacio de reglas. Lo que Langton notó fue que en la mayoría de las curvas la transición de las reglas de clase 2 a las reglas de clase 3 era discontinua, sólo ocurria que algunas curvas en el espacio de reglas en las cuales la transición de la clase 2 a la clase 3 era suave. Esas curvas en efecto pasaban por las reglas de la clase 4.

En términos más cuantitativos, la entropía de Shannon calculada sobre las lambda-cadenas mostraba un salto en algun valor de λ (que dependía de la cadena escogida), desde valores cercanos a 0 hasta valores cercanos a 1. El valor de la información mutua en diferentes puntos de tales curvas no difirió sustancialmente de 0. Sin embargo, para lambda-cadenas del tipo anterior, la entropía mostraba una transición relativamente suave desde valores cercanos al 0 hasta valores cercanos al 1. La información mutua por otro lado, mostró un pico para esas lambda-cadenas. Langton concluyó que la estructura del espacio de reglas aparentaba lo siguiente:

Hay una fase ordenada de la clase 1 y 2 reglas de AC separadas, en general, desde el desorden o fase caótica de las reglas de la clase 3 por una transición de

primer orden. Sin embargo, en la vecindad de esta frontera de fase caen zonas de reglas complejas de clase 4. Si el espacio de reglas es visitado cruzando esas zonas de reglas complejas, la transición de orden a caos es una transición de segundo orden.

La variación de lambda produce un espectro en el comportamiento de los autómatas celulares cuya correspondencia con la clasificación de Wolfram nos diría que la clase uno (puntos fijos) transita hacia la clase dos (periodico), y esta hacia la clase cuatro (complejo) para finalmente transitar a la clase tres (caótico) como se muestra en el siguiente diagrama.

I(Puntos fijos) $\rightarrow$ II(Periodico) $\rightarrow$ IV(Complejo) $\rightarrow$ III(Caótico)

Debido a la naturaleza no continua de los parámetros que intervienen en un autómata celular, no es posible hablar de transiciones suaves entre los cambios de fase que se manifiestan. Sin embargo, cualquiera que sea el criterio para clasificar a los autómatas celulares, es claro que existen transiciones de fase desde comportamientos regulares hacia caóticos y en algunos momentos intermedios hay manifestaciones de alta complejidad como en cualquier otro sistema dinámico complejo. Los autómatas de clase IV de Wolfram o los de clase III son los que nos interesan en este trabajo. Tal como se mencionó en la introducción, la complejidad es la condición que se necesita para poder construir cosas.

Capítulo 3

La propiedad universal de cómputo

3.1. Procesos computables

Desde los tiempos de la calculadora de Pascal y la máquina analítica de Babagge se ha especulado sobre la existencia de funciones o algoritmos que puedan ser procesados por alguna máquina o el ser humano con la única restricción de que estos deben de arrojar resultados no importando el tiempo que se lleve realizar tal tarea. También se conoce la existencia de funciones que no pueden ser resueltas ni por máquinas o humanos. La primer clase de funciones se llaman computables o efectivamente computables y la otra no computables.

Una función efectivamente computable deber poder ser descrita mediante instrucciones de longitud finita usando un número finito de símbolos. Un programa de computación es un ejemplo de tales instrucciones que no pueden ser infinitas, pues el lenguaje de computación es finito. Si la descripción de una función fuera de longitud infinita entonces debería existir un proceso interno

infinito capaz de leer las instrucciones, lo cual contradice la noción de computabilidad.

Existen diversos formalismos que intentan clasificar las funciones computables. Algunos de esos modelos formales sirven para construir explícitamente alguna función, entre ellas la más común es la máquina de Turing. Otros modelos sólo formalizan las relaciones entre funciones y sus componentes, como el cálculo lambda. los modelos de la primera clasificación nos son útiles porque manejan explícitamente la información que se esta tratando y son relativamente faciles de construir con la computadora o cualquier otro dispositivo que nos facilite el almacenamiento, flujo y potencialidad para transformar la información.

La manera en que el autor de este trabajo considera a una función computable, es como aquella capaz de modificar cierta estructura abstracta a partir de condiciones iniciales a través de flujos de información. El resultado final entonces se debe interpretar como el resultado de la función. Tal estructura abstracta inicial puede ser de cualquier tipo. Por ejemplo, un procesador modifica su estado físico a partir del flujo de electrones y fuerzas electromagnéticas. La manera en que tal flujo se lleva a cabo es siempre la misma, pues las leyes de la física nunca cambian. Sin embargo, el estado final de un procesador depende directamente de la configuración inicial en que este se encuentre. Pero, aunque las leyes de la física rigen todo el universo, no sólo estas pueden servirnos para hacer cómputo. Pues cualquier persona puede hacer cálculos matemáticos en una hoja y no depender directamente del material del lápiz o el papel. En tal caso las condiciones iniciales del proceso son ciertos datos anotados y la función de cómputo la lleva a cabo el cerebro humano.

Con la idea de función computable que se ha propuesto, cualquiera que haya estudiado sistemas dinámicos encontrará una analogía inmediatamente. Pues, al final de cuentas hacer computación equivale a hacer evolucionar una cierta condición inicial a través de ciertas reglas. Las condiciones iniciales son la descripción del algoritmo junto con los datos de entrada y las reglas de evolución equivalen a la descripción del algoritmo mismo.

Sin embargo las funciones computables terminan su ejecución en un tiempo finito por definición. Por lo tanto, a pesar de la aparente analogía que existe entre un sistema dinámico y un proceso de cómputo, los sistemas dinámicos no tienen porque terminar en tiempo finito, de hecho. Existen aquellos que son periódicos y que nunca terminan.

Por tanto, debemos hacer una restricción sobre los sistemas dinámicos, de tal forma que la analogía tenga sentido, es decir, debemos encontrar la forma de poder detener un sistema dinámico cuando su estado nos sea satisfactorio. La forma más simple de hacer esto es agregar una condición de finalización a la función de evolución. Tal condición debe de depender de un conjunto de estados en el espacio fase del sistema dinámico.

Además, hay otra importante restricción que debemos hacer a los sistemas dinámicos. Esta se refiere a la continuidad, pues, en general un sistema dinámico puede pasar de un estado a otro de manera continua, es decir, para cada intervalo de tiempo existe una transición que lleva un estado a otro de manera continua. Un proceso de cómputo está determinado por las funciones de transición de estados, tales funciones transforman estructuras de datos en

otras y tales cambios requieren de un tiempo cuantizado a invervalos, es decir, no existe ley alguna que modifique un estado en otro de manera continua.

Por tal motivo, la analogía entre sistemas complejos y funciones computables debe considerar la continuidad de los estados de transición y de la discusión anterior podríamos concluir que existe una equivalencia entre sistemas dinámicos finitos con estados discretos y funciones computables.

3.2. El modelo de la bola de billar.

Un ejemplo de modelo que exhibe características de computación es el modelo de la bola de billar. Este es un sistema mecánico no disipativo y conservativo clásico, donde la posición, velocidad, masa y tiempo son variables reales. El modelo consiste de dos esferas idénticas. En el sistema se dice que se tiene un '1' en el punto del espacio y del tiempo donde se encuentra la bola, y '0' en otro caso. Tal convención nos dice que el número de '1"s nunca puede cambiar debido a que el sistema es conservativo.

Un circuito lógico se puede simular como una colisión de esferas. La ruta que sigue una bola depende de la presencia o no de una colisión, esto crea un proceso de decisión. Para hacer esto, considérense los puntos A y B de la figura 3.1.

Si hay dos bolas presentes al mismo tiempo en los puntos A y B, entonces estas colisionarán y tomarán el camino externo hacia los dos puntos AB. Si A=0 y B=1, entonces la bola que viene del punto B no colisionará con otra bola y terminará por llegar al punto AB (no A y B). Esto se convierte en un

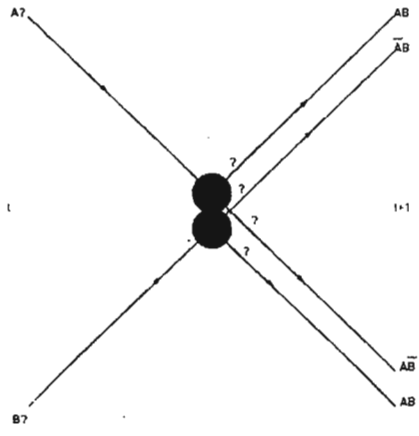

Figura 3.1: Un circuito lógico en el modelo de la bola de billar.

circuito con dos entradas y cuatro salidas.

Para que las bolas permanezcan en el sistema se introducen los espejos en el modelo. Los espejos son usados para direccionar señales y retardarlas para ejecutar las operaciones lógicas. Se escoge una latíz para que las colisiones entre bolas con espejos se den en los vértices. Todas las colisiones ocurren en ángulos rectos, así que una unidad de tiempo después de una colisión, las bolas aún siguen en la latíz.

El modelo se puede implementar usando un autómata celular bidimensional. Para cada celda se asigna un valor 0 o 1 y se deja evolucionar de acuerdo a las reglas locales. Sin embargo, el efecto de translación directa no es fácil de implementar. Se usan 6 estados y una vecindad de 17 celdas. Para esto, Margolus construyó un esquema que usa una vecindad llamada vecindad de Margolus (figura 3.2) la cual actualiza un bloque de cuatro celdas simultáneamente en vez de una sola.

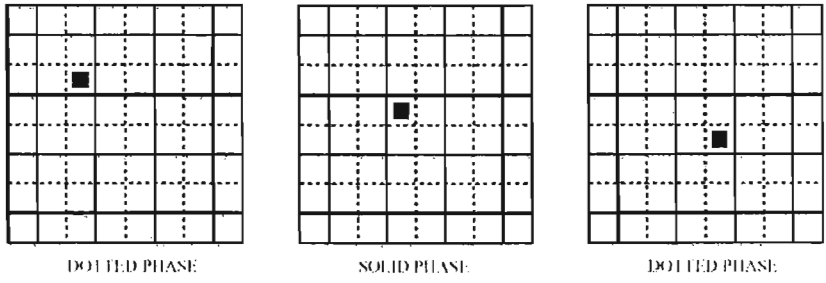

Figura 3.2: La vecindad de Margolus.

La forma en como esto se hace se muestra en la figura 3.2. La actualización se lleva a cabo en dos fases: primero los bloques dentro de las líneas sólidas se actualizan, después los bloques dentro de las líneas punteadas se actualizan, ambos de acuerdo a la misma regla local. En la figura una partícula que se mueve a través de la latíz está completamente determinada por la regla local que actúa sobre la vecindad de Margolus. Las reglas locales que pueden implementar el modelo de la bola de billar se muestran en la figura 3.3 (Se omiten las reglas simétricas).

La figura 3.4 muestra como se implementa el modelo de la bola de billar usando el esquema anterior. Las colisiones se pueden implementar usando pares de unos, usando la regla (c) durante la colisión. En la figura, las rutas que los siguen originalmente los unos se muestran con líneas punteadas. Nótese que las reglas conservan el número de unos para cada bloque de cuatro celdas.

El modelo de la bola de billar es útil para construir una máquina de cómputo universal. Si se pudiesen acomodan los elementos necesarios que sirven para

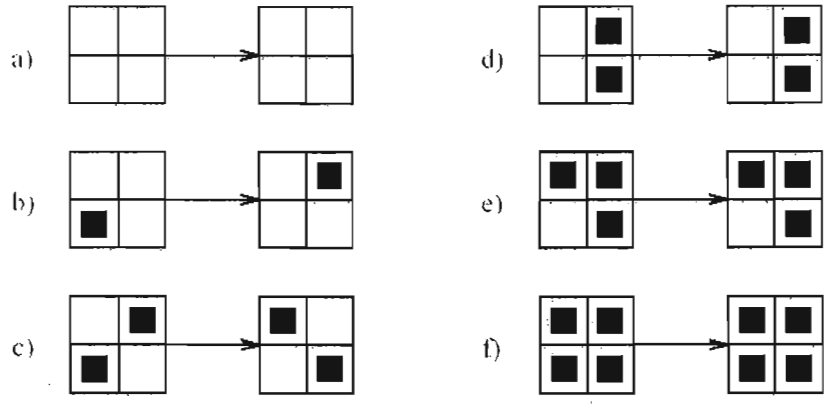

Figura 3.3: Las reglas que implementan el modelo de la bola de billar.

hacer circuitos lógicos, junto con mecanismos de almacenamiento de información entonces se puede demostrar la universalidad computacional de este modelo.

3.3. Lenguajes y gramáticas formales

Para definir la noción de lenguaje formal se comienza precisando el concepto de alfabeto, el cual es cualquier conjunto no vacio y finito. Una cadena en el alfabeto A es un conjunto ordenado $\alpha = a_1, a_2, a_3, ..., a_n$. Con cada a_i elemento de un alfabeto A. También se introduce la cadena vacía ϵ. La longitud de la cadena es el número de elementos de esta. La longitud de ϵ es 0.

Un lenguaje formal es un subconjunto de $A^* = A^+ \text{ U } \epsilon$, donde A^+ denota el conjunto de todas las cadenas posibles sobre un alfabeto A. Por ejemplo, si A es el conjunto de todas las palabras del diccionario, entonces una cadena es una expresión formada con palabras, y de todas ellas, sólo algunas tienen

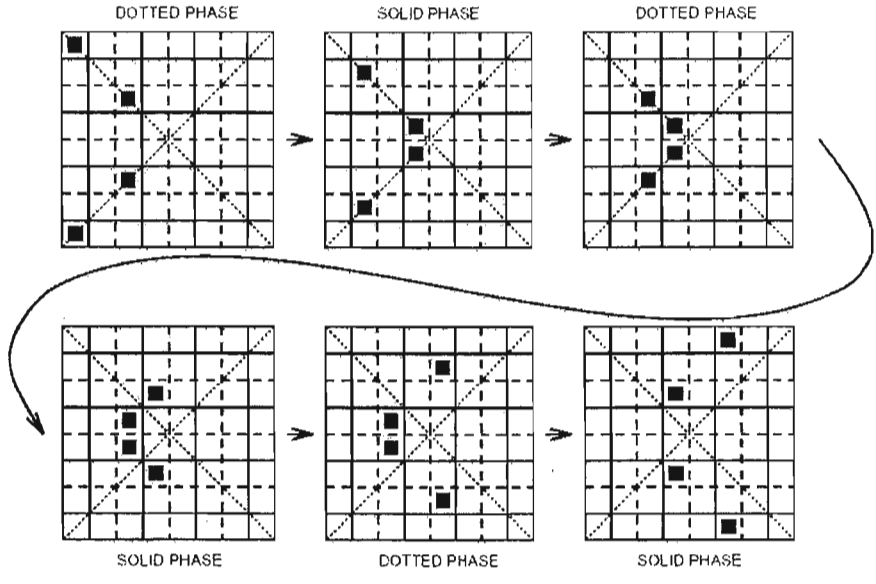

Figura 3.4: Colisión de dos pares de unos.

sentido en un lenguaje.

La forma en que podemos describir un lenguaje sobre un alfabeto A es muy diversa. En 1959, Noam Chomsky desarrolló una clasificación de gramáticas a través del tipo de transformaciones que se tienen que aplicar a los enunciados de un lenguaje. Tales transformaciones deben ser capaces de generar todos los enunciados válidos del lenguaje.

Una gramática formal (o gramática de Chomsky) consiste de un alfabeto A, un alfabeto abstracto B. Un elemento b de B que es el símbolo inicial y un conjunto finito de producciones de la forma $\alpha \rightarrow \beta$ donde α es un elemento de $(AUB)^+$ y β es un elemento de $(AUB)^*$.

3.4. Problemas de decisión

Una clase importante de problemas en ciencias de la computación son los problemas de decisión. Son aquellas preguntas cuya respuesta es si o no, y pueden ser vistos como un lenguaje que contenga todas las preguntas cuya respuesta es afirmativa. Es decir, un problema de decisión se puede simplicar en un lenguaje. Si el dominio del problema es numerable, entonces el lenguaje que lo simplifica también lo es.

Un lenguaje es aceptable si existe algún proceso computacional que se detenga y acepte todas las preguntas como parte del lenguaje. Para las otras preguntas, el proceso puede no detenerse o no aceptar. Si un problema de decisión es aceptable y numerable, entonces existe un proceso computacional que puede numerar todas las preguntas para cual la respuesta es si. Por tanto, los problemas de decisión asociados a lenguajes aceptables también se suelen llamar computacionalmente enumerables.

Un lenguaje es decidible si existe un proceso computacional que acepta el lenguaje y además se detiene con todas las entradas posibles. Tal lenguaje también es llamado computable. Todos los lenguajes decidibles son aceptables, mas no lo contrario, por tanto, el conjunto de los lenguajes decidibles es un subconjunto del conjunto de los lenguajes aceptables.

Los procesos computacionales que se usan para demostrar la computabilidad de un lenguaje están basados teóricamente en varios modelos de computación. Uno de ellos es la máquina de Turing, otros modelos son el cálculo lambda, las funciones recursivas y los sistemas de Post.

En 1930, Kurt Gödel estableció la existencia de proposiciones matemáticas que son no son decidibles a partir de un sistema formal. En particular, su primer teorema establece que *En cualquier sistema axiomático (sistema formal de matemáticas) consistente suficientemente fuerte para permitir que uno haga aritmetica básica, uno puede construir un enunciado acerca de los números naturales que no pueda ser probado ni refutado dentro de ese sistema.* Un ejemplo de una proposición con esas características pudiera ser la conjetura de Goldbach que afirma que todo número natural par es suma de dos números primos. Hasta la fecha no se conoce ninguna prueba que apruebe o refute su validez, pero tampoco se ha probado que no se pueda probar.

Un sistema axiomático es aquel con un conjunto de axiomas y un conjunto recursivo de teoremas que el sistema puede generar de manera efectiva mediante procesos computables es decir, una máquina de Turing. Un enunciado para el cual no exista manera de probar o refutar su validez no es parte del sistema axiomático, por lo que se le dice incompleto. El primer teorema de incompletes de Gödel dice que cualquier sistema formal de matemáticas suficientemente fuerte es o bien inconsistente o incompleto. Existen otros problemas de decisión derivados de inconsistencias lógicas, tales como la paradoja de Russel.

3.5. El cálculo lambda

El cálculo lambda fue introducido por Alonzo Church [6] en 1934. Su trabajo es la base formal para los lenguajes funcionales actuales y para la descripción formal de las funciones recursivas. Una expresión E en el cálculo lambda se define como:

E ::= C | V | (E_1E_1) | (\v$E_1$) | (E_1)

Donde, C es un conjunto de constantes que incluye a los nombres de las funciones. V es un conjunto finito de variables y los siguientes términos son un conjunto de expresiones lambda, que pueden ser alguna de las siguientes: Una constante de C, una variable de V, una aplicación de una expresión E_2 a una expresión E_2(E_1 es el operador y E_2 es el operando), una abstracción \v.E1 que involucra una variable v de V y una expresión E_1, unos paréntesis (E_1).

Una reducción beta (\v.E_1)E_2 $\rightarrow$ E_1[E_2/v] denota que E_2 es sustituido por todas las ocurrencias libres de v en E_1.

Una reducción eta \v.Ev $\rightarrow$ E se hace cuando v no es libre en E.

3.6. Funciones recursivas.

El concepto de función recursiva se debe a Gödel y Herbrand. La clase de las funciones recursivas y las definidas por el cálculo lambda son equivalentes. Una función recursiva parcial es aquella que puede tener un ciclo infinito para algún valor de su dominio. Una función recursiva, también llamada Total recursiva es aquella que siempre retorno un valor para todas las entradas de su dominio.

Las funciones recursivas primitivas son un subconjunto de las funciones totales recursivas con la restricción de que la recursión primitiva sea usada un número finito de veces usando la función cero y la función predecesor.

La recursión primitiva se define como:

$f(x_1,...,x_n) = g(x_1,...,x_{n-1})$ si $x_n = 0$ y $f(x_1,...,x_n) = h(x_1,...,x_n, f(x_1,...,x_{n-1}, x_{n-1}))$ si $x_n > 0$

donde, g y h son funciones primitivas recursivas.

3.7. Sistemas formales de Post

Un sistema formal de Emile Post [24] es una cuatupla Pi = (C, V, A, P) donde,

C es la unión de las constantes no terminales con las constantes terminales. V es un conjunto finito de variables. A es un subconjunto finito de C^* llamado axiomas. P es una lista finita de producciones de la forma $x_1v_1x_2 \ldots x_nv_nx_{n+1} \rightarrow y_1w_1y_2 \ldots y_nw_ny_{n+1}$ donde x_i y y_i son elementos de C^*, v_i y w_i son elementos de V con la restricción de que cada variable en la derecha debe aparecer en la izquierda.

Los sistemas de etiquedado (Tag systems) son sistemas formales de Post y fueron introducidos en 1921. Un sistema de etiquetado es un conjunto de reglas que especifican un número fijo de elementos para ser removidos del principio de una secuencia y otro conjunto de elementos para ser pegados al fin de la secuencia, y que están basados en los elementos removidos del principio. Por ejemplo, el sistema de etiquetado v = 1 que se muestra en la figura 3.5, en el cual los cuadros negros representan 1 y los blancos 0. Las reglas de transición

son $(1,...) \rightarrow (...,0,1)y(0,...) \rightarrow (...,1,0)$. El estado inicial es 1.

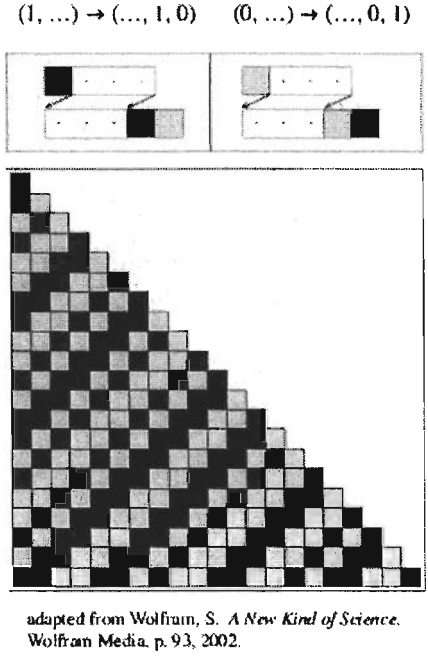

Figura 3.5: Sistema de etiquetado v = 1.

3.8. Máquinas de registros

Una máquina contadora (counter machine) o máquina de registros (register machine) consiste en un conjunto fijo de registros de datos e instrucciones para operar sobre estos. Otros nombres con que se les conoce son máquinas contadoras y máquinas de programas. Una de las máquinas más sencillas que se han estudiado es aquella con dos registros y dos operaciones (incrementar y decrementar).

Las máquinas de registros fueron inicialmente estudiadas por Shepherdson y Sturgis [27] y Minsky [19]. Este último diseñó un autómata celular bidimensional de cuatro estados capaz de simular una máquina de registros.

Formalmente una máquina de registros consisten en una colección de registros etiquetados $R_1, R_2, R_3...$ cada uno de los cuales contiene un número natural $r_1, r_2, r_3....$ Las instrucciones para la máquina son:

1. Para cada número natural n. Una instrucción Cero Z(n) que pone 0 en R_n y deja sin alterar los demás.

2. Para cada número natural n. Una instrucción Sucesor S(n) que incrementa el valor del registro R_n y deja sin alterar a los demás

3. Para cada número natural m y n. Una instrucción de Transferencia T(m,n) que transfiere el valor r_n en el registro R_m y deja sin alterar a los demás

4. Para cada número natural m, n y q. Una instrucción de Salto J(m,n,p) que causa la ejecución de la instrucción q-esima cuando $r_m = r_n$.

Un programa para la máquina de registros es una secuencia de instrucciones $I_1, I_2, I_3...$ con algun arreglo de registros como condición inicial.

3.9. Máquinas de Turing

En 1936, Alan Turing introdujo un autómata finito con la capacidad de resolver funciones computables de acuerdo a su noción de computabilidad. Tal mecanismo rebasó la capacidad de los autómatas conocidos anteriormente pues el tamaño del conjunto de los lenguajes que era capaz de reconocer era mucho

mayor. Además, su máquina también adquirió la capacidad de transformar cualquier cadena de símbolos en otra siguiendo cualquier tipo de instrucciones. Fue pues, el modelo que formalizó la noción de cómputo y es la referencia más común que actualmente se tiene para este concepto.

El funcionamiento general de una máquina de Turing (MT) consiste en transformar una cierta entrada de datos en otra. En su forma más elemental, una MT transforma una cadena finita de símbolos, letras o colores. De aquí en adelante me referiré a la MT en su forma más elemental, aunque, desde luego existan modelos más complejos. La máquinaria básica de una MT consiste básicamente en una cabeza lectora y escritora de símbolos, y una cinta unidimensional de longitud semi-infinita donde se almacena la cadena a reconocer o las instrucciones de algún proceso.

También se agrega a la máquina un estado interno que cambia para cada paso del tiempo. Los símbolos de la MT pertenecen a un alfabeto finito, el cual contiene un símbolo blanco o vació. También existe un alfabeto para etiquetar al conjunto de estados. Dada la definición anterior, debemos considerar que los alfabetos de símbolos y estados son finitos.

El mecanismo de una MT (figura 3.6) consiste en lo siguiente. Se coloca la cabeza de lectura y escritura sobre un símbolo de la cinta y se inicializa el estado actual como el inicial. La MT evoluciona a través del tiempo de manera discreta, de esta manera, para cada clic del reloj se actualiza el estado interno actual y se mueve la cabeza lectora sobre la cinta de símbolos, los cuales también se actualizan. Todos estos movimientos dependen del estado actual y del símbolo actual donde se localice la cabeza lectoescritora. Entonces el funcio-

namiento de la MT está determinado por una función definida como sigue:

$$\varphi : \text{SxA} \longrightarrow \text{SxBx}\{I, D\}$$

Donde S es el conjunto de estados, A es el alfabeto de entrada, B es alfabeto de salida, y el conjunto I,D denota el movimiento que debe hacer la cabeza, es decir, moverse a la derecha (D) o izquierda (I).

Las reglas de una máquina de Turing no tienen que contemplar todos las combinaciones posibles de estados internos y cadenas de símbolos. Pues el avance de una MT puede ser detenido de alguna de las siguientes dos formas: Encontrar el estado final o caer en una configuración donde no exista regla alguna que nos permita seguir avanzando. Fijados estos acuerdos estamos en condiciones de fabricar nuestra primera MT, no sin antes hacernos algunas observaciones que todo programador debe hacerse, como tener cuidado de no crear ciclos infinitos y asegurar que cualquier entrada válida tenga que ser una condición inicial que nos garantize llegar a un resultado.

Nótese que el alfabeto de entrada de una MT debe estar contenido en el alfabeto de salida. Esos conjuntos no son iguales necesariamente, pero la contención se debe de cumplir siempre. Además, se debe incluir en el alfabeto de entrada a un símbolo o color que denote un espacio en blanco y otro que indique inicio y fin de la cadena de datos. Los alfabetos que etiquetan a los elementos que conforman una máquina de Turing deben ser finitos, pues nuestra definición de función computable es válida para estructuras de datos que inicialmente pueden ser descritas con un número finito de colores, de igual forma, el número de estados en los que puede estar la MT es finito debido a

Figura 3.6: Esquema de una máquina de Turing.

que la existencia de un número infinito de estos sería una contradicción a la afirmación de que el mecanismo deba tener fin en algun momento.

A continuación se muestran algunos ejemplos que muestran la evolución de un estado inicial que se ha sometido a las reglas de una MT, así como la transición que experimenta el estado inicial hasta llegar a un estado final o inexistente.

Las reglas que se muestran en la tabla de abajo corresponden a la implementación de un sumador simple. Dadas dos cadenas de unos y separadas por un cero, la salida será una cadena cuya longitud es la suma de las dos anteriores. Por ejemplo, la cadena 101 se convierte en 11 y 110111 en 11111.

Una máquina de Turing también nos permite reconocer y aceptar lenguajes formales tal como lo hace un autómata de estados finitos. En el siguiente ejemplo se muestran las reglas para una MT que nos permite reconocer el

Estado	0	1
q0	(q1,1,D)	(q0,1,D)
q1	(q2,0,I)	(q1,1,D)
q2	(q2,0,I)	(q3,0,I)
q3	(halt,0,D)	(q3,1,I)

Cuadro 3.1: Reglas para una MT que implementa un sumador simple

lenguaje L = $\{0^n1^n|n > 0\}$. En este ejemplo se usan dos símbolos que no aparecen en la cadena de entrada pero que ayudan a la MT para llevar a cabo su tarea, además se emplea un símbolo o color blanco para reconocer el fin de la cadena. El estado que indica el fin del proceso es q4, y el estado inicial q0.

Estado	0	1	X	Y	B
q0	(q1,X,D)	-	-	(q3,Y,D)	-
q1	(q1,0,D)	(q2,Y,I)	-	(q1,Y,D)	-
q2	(q2,0,I)	-	(q0,X,D)	(q1,Y,I)	-
q3	-	-	-	(q3,Y,D)	(q4,B,D)
q4	-	-	-	-	-

Cuadro 3.2: Reglas para una MT que reconoce el lenguaje L = $\{0^n1^n|n > 0\}$

La máquina de Turing es un mecanismo determinista, pues en ninguna de las reglas de evolución de los estados se incluyen arbitrariedades o procesos aleatorios. Por lo tanto es seguro que toda condición inicial tendrá el mismo destino. La observación se hace pues se quiere enfatizar que es posible construir máquinas de Turing que involucren procesos estocásticos, al igual que existen autómatas celulares con las mismas características, tales máquinas salen del

objetivo principal de esta tesis, la cual, como se ha intentado hacer notar, fundamenta su razón de ser en el determinismo.

Las MT son en sí mismas dispositivos matemáticos muy interesantes, pues nos permiten implementar funciones computables del tipo que enunció Church. Y si sabemos que una MT es en sí un proceso computable también, entonces debe existir una meta máquina de Turing con la peculiaridad de que el proceso que simula es justamente otra máquina de Turing. A estos meta artefactos se les conoce como máquinas universales de Turing, y son también aquellas las cuales se van a tratar en los siguientes párrafos. En 1937, en su artículo seminal, Turing dió a conocer la primera construcción de una máquina universal de Turing.

Como es de esperarse, las MT tienen la misma capacidad que los modelos de computación anteriores. Se puede programar con ellas cualquier producción de gramaticas formales, sistemas formales de Post y máquinas de registros. También se puede formalizar su funcionalidad con el cálculo lambda o con funciones recursivas.

En 1956, Shannon dió a conocer la prueba de que dos colores son suficientes. De ahí en adelante sólo se usarían esos. En 1962, Minsky descubrió una máquina universal de Turing con 7 estados y cuatro colores. Tal máquina convertida a dos colores necesitaría de 43 estados.

Estado	q0	q1	q2	q3	q4	q5	q6
0	(q0,0,I)	(q1,2,D)	(q2,0,H)	(q4,2,D)	(q2,2,I)	(q2,3,I)	(q5,2,D)
1	(q1,1,I)	(q1,3,D)	(q2,3,I)	(q6,1,I)	(q4,3,D)	(q5,3,D)	(q6,1,D)
2	(q0,0,I)	(q0,0,I)	(q2,2,I)	(q3,2,I)	(q4,2,D)	(q5,2,D)	(q6,0,D)
3	(q0,1,I)	(q5,2,D)	(q3,1,I)	(q3,1,I)	(q4,1,D)	(q5,1,D)	(q1,0,D)

Cuadro 3.3: Reglas para una la máquina universal de Turing de Minsky, donde qo denota el símbolo blanco, y h es el estado de halt.

3.10. La tesis de Church

La definición de procedimiento efectivo es intuitiva, no es una noción formal. Así que no hay forma de probar teoremas con esta. Sin embargo, las máquinas de Turing nos dan la potencialidad de demostrar la efectividad computacional de alguna función en el sentido de Church.

La proposición de Church es una tesis pues no existe manera de verificar su validez. Se puede contradecir al encontrar una función que se compruebe computable y que sin embargo no pueda ser ejecutada por una máquina de Turing. Sin embargo, se puede aceptar que el conjunto de las funciones computables en el sentido de Church son aquellas que pueden ser ejecutadas por una máquina de Turing. Lo que no sabemos es si este conjunto está completo o no.

Las evidencias que han sustentado a la tesis de Church son: a) Cada función efectivamente computable que ha sido investigada ha resultado también serlo mediante una máquina de Turing. b) Todos los métodos conocidos u operaciones para obtener nuevas funciones efectivamente computables a partir de funciones efectivamente computables son métodos paralelos a la construcción de máquinas de Turing a partir de máquinas de Turing). Todos los intentos

por analizar la clase de funciones computables han sido equivalentes en el sentido que todas han seleccionado a la misma clase de las funciones, es decir las computables por una máquina de Turing. Debido a la gran variedad de análisis, esta evidencia se considera la más fuerte.

Capítulo 4

Universalidad en autómatas celulares

Previo a la conjetura de Wolfram existieron trabajos importantes acerca de la capacidad de cómputo universal de los autómatas celulares. John Von Neumann y E. F. Cood fueron los principales investigadores de esta área. Sus publicaciones manejan la noción de computación universal y construcción universal. Este último término se refiere a la capacidad de ciertas configuraciones para construir otras configuraciones.

Segun Wolfram, algunos AC pueden ser interpretados como computadoras, en el sentido que estos pueden almacenar y ejecutar un programa de igual manera (no con la misma eficiencia necesariamente). Esta interpretación sugiere que la condición inicial de un AC representa al programa y datos almacenados. El resultado de la evolución del AC a través del tiempo da como resultado la salida del programa, es decir, el resultado de la computación. Dicho esto, las reglas del autómata celular representan las operaciones de una computadora.

Langton conjeturó que las clases de autómata celular sugeridas por Wolfram tienen correspondencia con las clases de computabilidad de la siguiente manera:

I(Halting) → II(Halting) → IV(Indecidible) → III(No halting)

Sobre la aparición de indecibilidad en la transición de fase hacia la clase cuatro, Langton decía que en ese momento se presenta el fenómeno de desaceleración crítica donde las transiciones puede diverger a infinito. Sobre la propiedad universal de cómputo en la misma transición de fase, Langton opinaba que .. *La existencia de computación universal es explicada por el hecho de que la dinámica de los sistemas físicos en la vecindad de transiciones críticas exigen una divergencia en su suceptibilidad, esto es, en su sensibilidad a pequeños detalles en su estructura interna y perturbaciones externas. Esta autosensibilidad en la vecindad de una transición crítica es manifestada en computadoras universales como su programabilidad...*

Así como existen máquinas universales de Turing capaces de simular otra máquina de Turing, también existen autómatas celulares cuyos mecanismos permiten simular el comportamiento de otro autómata celular. Por lo que basta encontrar un sólo autómata celular capaz de simular una máquina de Turing para poder hacer la biyección entre los autómatas celulares que pueden simular otros y máquinas universales de Turing.

Von Neumann fue el primero en diseñar un autómata celular capaz de soportar cómputo universal en particular y en general construcción universal (última sección del capítulo).

4.1. Cómputo universal en el autómata de Smith

En 1971 Alvy Ray Smith [29] diseñó de manera simple un autómata celular bidimensional con la capacidad de hacer cómputo universal. El diseño consiste en simular cualquier máquina de Turing dada, y en particular, alguna máquina universal de Turing, a través de los mecanismos de un autómata celular. La simulación se hace en tiempo real, es decir, que existen una correspondencia de uno a uno entre los pasos de una máquina de Turing y el autómata de Smith.

El funcionamiento del autómata celular de Smith (ACS) es el siguiente. Supongamos que tenemos una máquina de Turing con m estados y n símbolos. El ACS consiste en un arreglo bidimensional infinito con un número de estados k para cada celda, donde k es el máximo entre m y n más uno, k=max(n,m)+1.

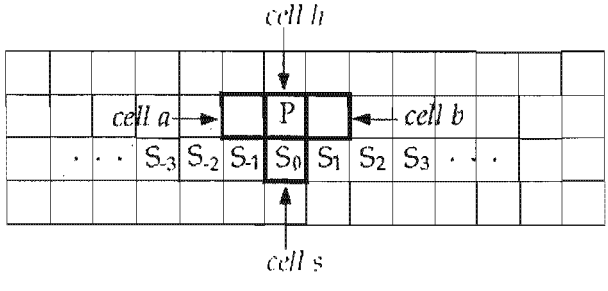

Figura 4.1: La configuración del autómata celular de Smith

El estado inicial del autómata consiste en asignar un estado nulo en todas celdas excepto en un renglón de longitud finita. Inicialmente, los estados de tal renglón son asociados a los símbolos de la cinta de la máquina de Turing.

Por otro lado, existe una celda que representa a la cabeza lectora de la cinta

y cuyo estado representa el estado de la máquina de Turing (Es por eso que sólo se necesitan max(n,m)+ 1 estados). La celda que representa a la cabeza lectora se localiza justamente en la celda que está arriba del símbolo que se va a leer.

La actualización de los estados del ACS depende de una vecindad de 7 elementos, puesto que para todos los casos, las celdas que se actualizan no dependen de las celdas superior izquierda y superior derecha, pues estas no contienen información relevante para la máquina de Turing.

Las reglas de evolución del ACS son muy similares a las de la máquina de Turing que emula. Para cada incremento del tiempo sólo se actualizan las celdas cercanas a la celda que imita la cabeza lectoescritora. En la figura 4.2 se muestran parte de las reglas de transición que hacen trabajar al autómata. Un movimiento se representa como (u,v) = pXq, que significa leer el símbolo u en el estado v, escribir un nuevo símbolo p, ir al estado q y mover la cabeza hacia X, donde X es I o D (izquierda o derecha).

El autómata celular de Smith está regido por los mismos mecanismos de una máquina de Turing, es una adaptación hecha adhoc para las reglas de un autómata celular. Pero esta trampa es válida, demuestra la relativa flexibilidad que nos brindan los AC para con mecanismos matemáticos cuyo comportamiento es discreto en estados y discreto en el tiempo.

En una dimensión, con p representando el tamaño de la vecindad y q el número de estados, Smith encontró las siguientes máquinas universales: 2x40, 3x18, 6x7, 8x5, 12x3, 14x2. Siendo esta última una máquina con sólo dos celdas

por vecindad. Smith descubrió también algunos teoremas interesantes acerca de la simulación de máquinas de Turing con autómatas celulares.

Cell c	Neighborhood of c	Next state of c	conditions
s	P S_{-1} $S_0^{\rightarrow}$ S_1 0 0 0	p	$S_0 = u$ $P = v$
h	0 0 $P^{\rightarrow}$ 0 S_{-1} $S_0^{\rightarrow}$ S_1	0	in all cases
a	0 0 $0^{\rightarrow}$ P S_{-2} $S_{-1}^{\rightarrow}$ S_0	0 q	if $X = R$ if $X = L$
b	0 P $0^{\uparrow}$ 0 S_0 $S_1^{\uparrow}$ S_2	q 0	if $X = R$ if $X = L$

Figura 4.2: Algunas reglas de transición del autómata de Smith.

4.2. Cómputo universal en el autómata de Banks

Muchos de los autómatas celulares que presentan propiedades universales de cómputo basan su construcción en la simulación de componentes lógicos parecidos a los de una computadora electrónica.

En 1970, Roger Banks [2] mostró que la regla 4005091440 con dos estados y cinco vecinos para el autómata bidimensional, también reproduce los mismos componentes lógicos. Esta regla, a diferencia del juego de la vida, necesita de un espacio fase infinito al igual que la regla 110. Una descripción más detallada de la construcción de este autómata celular se muestra en los siguientes párrafos.

En su tesis doctoral, Roger Banks propone una computadora universal de tres estados y cinco vecinos por celda. Aunque dos estados son suficientes para universalidad, Roger expone un autómata celular para ilustrar las técnicas empleadas. Los siguientes párrafos muestran parte del trabajo que Banks escribe en su tesis doctoral [2].

Los tres estados se representaran por los símbolos 2,1 y 0 que es el espacio en blanco. El primero de muchos elementos necesarios es una especie de cable. Una ensamblado especial junto a una terminal del cable sirven para lograr el objetivo. La motivación para crear esos elementos es el deseo de mantener una analogía con el cableado físico de una máquina de estados finitos hecha de cables, transistores, etc.

Para que la señal *10* se propague hacia la derecha del circuito, las reglas de transición de la figura 4.3 son necesarias.

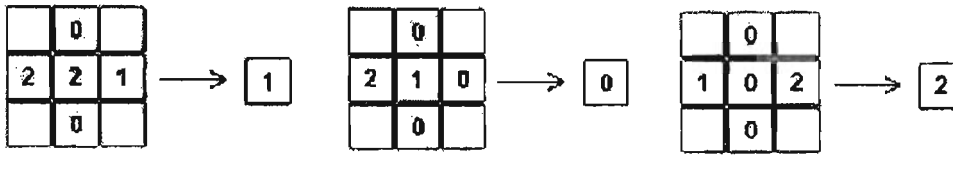

Figura 4.3: Reglas de transición para la propagación.

También es necesario contar con un cable ramificado para mandar señales a más de un lugar. El ensamblado tiene la propiedad que hace que cuando una señal entre, esta misma salga de los tres brazos de la configuración. La figura 4.4 muestra las reglas que se tienen que agregar.

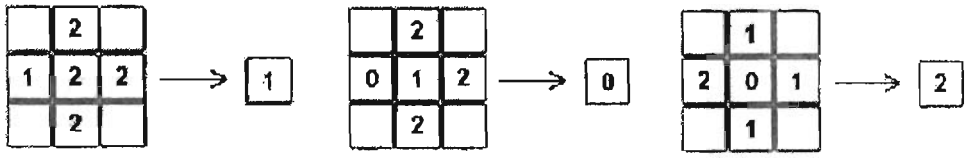

Figura 4.4: Reglas necesarias para el flujo de la señal en el elemento de ensamblaje.

El elemento terminal es simplemente un cable truncado. Una señal que alcance el fin del cable se extinguirá sin cambiar la longitud del cable. Esta terminal requiere que la siguiente regla sea agregada.

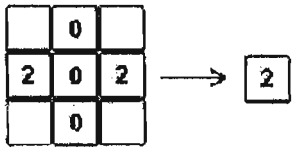

Figura 4.5: Regla de transición necesaria para la operación de la terminal.

Observando que sucede con el elemento de ensamblaje con tres cuando tres entradas llegan simultáneamente, se ve que las tres entradas son mutuamente aniquiladas. Esto resulta de las reglas de transición anteriores. Cuando dos entradas llegan simultáneamente, las dos siguientes reglas de transición se necesitan para que la operación de dos entradas y dos salidas pueda ser lograda.

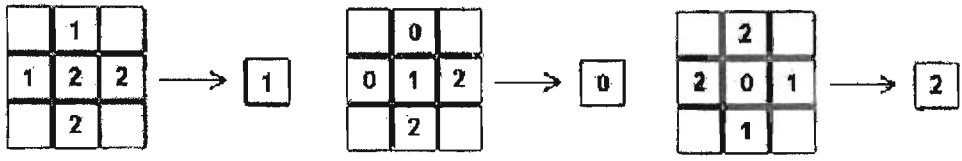

Figura 4.6: Reglas de transición necesarias para efectuar la operación de dos entradas y dos salidas.

Si se tiene un diodo (una entrada de una sola vía) entonces la operación de dos entradas y dos salidas de la figura de arriba podría hacerse con una función OR. Los diodos podrían ser usados para prevenir que las señales viajen fuera de las entradas. Si los cables son representados por líneas, las curvas por ángulos rectos y las terminales por círculos, entonces el diodo se configura de la siguiente manera:

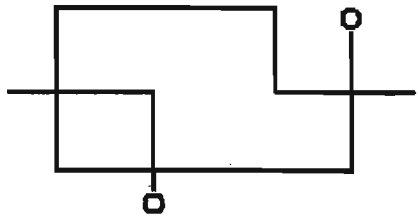

Figura 4.7: El diodo y su correspondiente símbolo.

La razón que justifica la asimetría es explicada al seguir su operación. Considérese primero una señal que entra desde la derecha la cual puede terminarse o seguir fluyendo.

Finalmente, todas las señales resultantes llegaran simultáneamente hacia la parte izquierda del ensamblado y serán eliminadas como se dijo previamen-

te. Además, una señal que entra desde la izquierda tendrá éxito al atravesar pues al menos dos señales llegarán a algún ensamblaje al mismo tiempo. Con el diodo, la operación OR puede ser estructurada enviando dos señales y haciéndolas pasar por el diodo para luego encontrarlas en una intersección.

Otra configuración útil es el reloj o emisor periódico de señales. Una señal circulando en un ciclo enviará una señal fuera del alambre lateral por cada intervalo de tiempo. Ahora, con dos relojes y un diodo, la función NOT (o inversor) puede ser configurada.

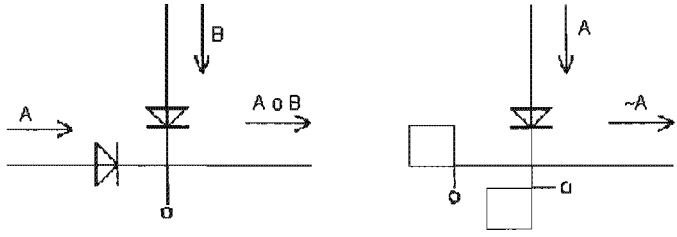

Figura 4.8: La implementación del or y el not.

Cuando una señal entra a A, no hay señal de salida debido a la propiedad de aniquilación de las tres entradas. Pero, en la ausencia de una señal, los dos relojes generarán una salida debido a la propiedad dos entradas/dos salidas del ensamblado. Los relojes deben ser sincronizados con los tiempos permisibles para las señales que estén en el ensamblado.

Las funciones OR y NOR forman un par universal de elementos lógicos. Hay sólo un elemento faltante que es el cruzado. Dado esto, la estructura lógica ya puede ser *cableada*.

El propósito del exceso de curvas en cada bloque del cruzado es hacer un retraso. Esto es, la señal B que entra desde la parte inferior debe fluir de tres maneras. Pero las tres rutas hacia la salida A forzan un aniquilamiento de señales debido a la propiedad de eliminado de tres entrada simultaneas. El cruzado requiere que las entradas A y B nunca lleguen simultáneamente. Si uno asume un espaciado mínimo de suficiente tamaño entre señales, entonces la siguiente configuración puede siempre ser usada para asegurar la no simultaneidad de entradas.

Un objetivo ha sido ser capaz de configurar en un espacio finito cualquier máquina de estados finitos con esos elementos. En particular, ya que podemos transmitir una señal binaria desde cualquier zona del espacio hacia cualquier otra, y como las señales pueden ser combinadas a través de funciones lógicas universales tales como el OR y NOT, entonces, es claro que todos los componentes necesarios para construir cualquier computadora de propósito general están presentes dentro de las configuraciones de estados en el arreglo anterior.

Alternativamente, el autómata de ocho estados de Codd [7] o el autómata de veintinueve estados de Von Neumann podria ser simulado. Entonces un arreglo de celdas de Codd simuladas podria actuar de la misma manera que Codd describe. Muchas otras configuraciones con tres estados han sido declaradas universales. Los elementos más comunes incluyen elementos lógicos universales, cableado, terminales y cruzados. Algunos tienen menos reglas de transición. Diferentes configuraciones de diodos y cruzados tienen que ser obtenidas con base en las propiedades del ensamblado.

4.2.1. Comentarios

Esta es la descripción muy general que hace Banks en su tesis doctoral. En ella se resalta la posibilidad de construir mecanismos universales a partir de la construcción de mecanismos con dominio local tales como operaciones lógicas o de conteo. Y que al colocarlas en conjunto con mecanismos que permitan el flujo de información entonces podemos crear macro estructuras de diverso tipo, en particular alguna que sea computacionalmente equivalente a una máquina de Turing, o de cualquier otro tipo. El modelo de Banks sugiere que el mundo físico de los circuitos, compuestos por cables, transistores, etc, puede ser modelado de manera muy sencilla por un autómata celular, y si se tiene la suficiente paciencia, poder construir estructuras análogas a los elementos reales de una computadora. La construcción de un circuito también es un modelo de computación. Sus operaciones principales comprenden operaciones lógicas, codificadores, decodificadores, multiplexores, desmultiplexores, sumadores y multiplicadores.

En este autómata celular no existen indicios claros de la existencia de emergencia de patrones. Los que se construyen se crean con reglas cuya función es descrita explícitamente. En este modelo no hay necesidad de esperar a ver que tipo de nuevas estructuras y reglas surgen de manera emergente. Lo único que se debe hacer es acoplar los elementos construidos para crear máquinas.

En cuanto a la clasificación de Wolfram, el autómata celular de Banks es de clase IV porque la dinámica de los circuitos dista mucho de ser periódica o caótica. La complejidad sigue siendo la manifestación más importante de los mecanismos con capacidad de cómputo.

4.3. Cómputo universal en el juego de la vida

Uno de los autómatas celulares más difundidos en la literatura es el Juego de la Vida de Conway. Tal autómata se diseñó a finales de los 60's y principios de los 70's cuando Conway intentaba simplificar el autómata celular previamente introducido por Von Neumann. Su autómata celular se popularizó en la columna científica de Martin Gardner [9] en 1970. En un principio, Conway quería buscar el autómata celular más simple que mantuviera la propiedad universal de cómputo. Decidió usar dos estados, muerto o vivo, y cuatro reglas sencillas:

- Si una celda viva tiene menos de dos vecinos, entonces muere.
- Si una celda viva tiene más de tres vecinos, entonces muere.
- Si una celda muerta tiene sólo tres vecinos, entonces adquiere vida.
- En cualquier otro caso, todas las celdas permanecen en su estado original.

Este autómata celular también genera comportamientos complejos, y, tal como Conway lo probó, también presenta la propiedad universal de cómputo. Los indicios de la prueba de la universalidad del juego de la vida fueron trabajados por Berlekamp, Conway, y Guy en 1982, y un año después por Gosper.

4.3.1. Los gliders

El autómata del juego de la vida presenta estructuras que se repiten de manera periódica al igual que la regla 110 del autómata celular simple y otros autómatas de clase 4. También existen patrones que aparecen y desaparecen, y entre todos ellos existe interacción de igual manera. A veces unos chocan y se

convierten en otros, ventaja que puede ser aprovechada para hacer predicados.

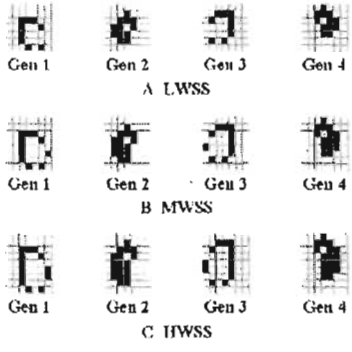

Figura 4.9: Patrones de naves espaciales.

Existe otro tipo de patrones oscilatorios especiales que se mueven de manera diagonal, un cuadro cada cuatro generaciones. A tal tipo de estructuras se les llama glider, y tales constituyen el principal componente para simular mecanismos como una máquina de Turing. Existe una familia de patrones que se mueven de manera ortogonal cada dos generaciones, a tales objetos dinámicos se les conoce como Light Weight Space Ship (LWSS), Medium Weight Space Ship (MWSS) y Heavy Weight Space Ship. En alusión a esos nombres, el nombre genérico de esas estructuras se les llama Naves espaciales (figura 4.9). Existe un patrón particular descubierto por Bill Gosper llamado pistola de gliders, el cual genera gliders de 30 generaciones. Actualmente se conocen pistolas que generan gliders por generaciones mayores a 14.

Otro elemento que emergen del mecanismo del juego de la vida además de la pistola de gliders es el aguijón de abeja reina. Un aguijón de abeja reina es capaz de avanzar y retroceder, enviando pequeños grupos de celdas en cada retorno. Un pistola de gliders se compone de dos aguijones de abeja reina,

al chocar ambos producen un glider y al agregar dos bloques más, se puede generar un envió continuo de gliders.

A continuación se muestran las ideas que Paul Rendell [26] escribió acerca de la construcción de un sumador usando las características de computación facilitadas por los gliders del juego de la vida. A partir del sumador, Rendell diseñó una máquina de estados finitos para finalmente llegar a la construcción de una máquina de Turing.

El sumador, que se muestra en la figura 4.10 es un dispositivo notable construido por David Buckingam y Marck Niemec en los años ochenta. Básicamente, el dispositivo suma dos cadenas binarias representadas por gliders y coloca el resultado en una tercera cadena de gliders. La figura 4.10 muestra a las celdas vivas en negro. Las manchas alrededor se han agregado para mostrar el agrupamiento de esas celdas en patrones tales como pistolas de gliders de periodo 60. Las líneas muestran las rutas de los gliders. Los números son codificados por la presencia o ausencia de un glider cada 60 generaciones. El código es una representación binaria con el bit menos significativo enviado primero.

El mecanismo del sumador binario se integra de dos etapas. La primera de ellas tiene dos salidas. La suma de dos bits con el mismo valor y algún residuo. Los resultados de la primera etapa se transfieren a la segunda de tal manera que el residuo se transfiere al siguiente sumando parcial. Estos cálculos pueden generar un nuevo residuo el cual se transfiere de regreso al primer proceso y así sucesivamente.

La ventaja de dividir el proceso en dos radica en que cuando el primero

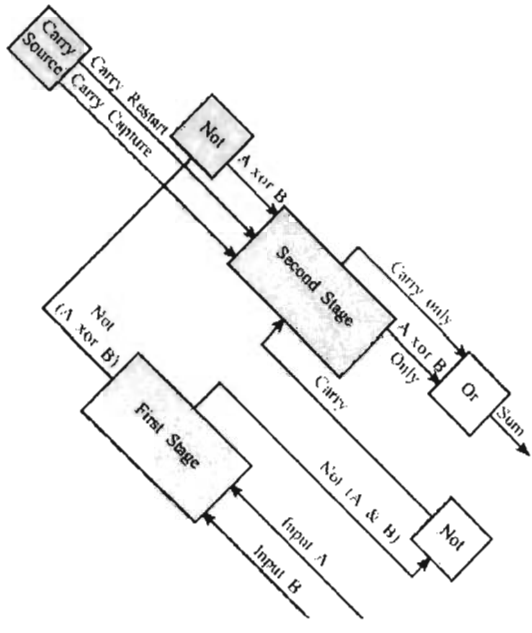

Figura 4.10: Esquema del sumador que implementa la suma binaria de dos cadenas de gliders. El resultado también es una cadena de gliders.

genera un residuo, entonces este arroja cero como resultado y el segundo proceso sólo genera un residuo cuando el primero arroja algo distinto de cero.

La primera etapa del proceso incluye un glider que borra otros dos para generar las dos salidas. La segunda etapa contiene la reacción de contragolpe, esta es aquella que convierte dos glider en un comedor, esta también convierte a un glider y un comedor en otro glider.

Existe otra importante característica del juego de la vida que nos permite crear una máquina universal, esta es la memoria de bloque deslizante, la cual fue construida por Dean Hickerson [12] en 1990. Esta es la implementación

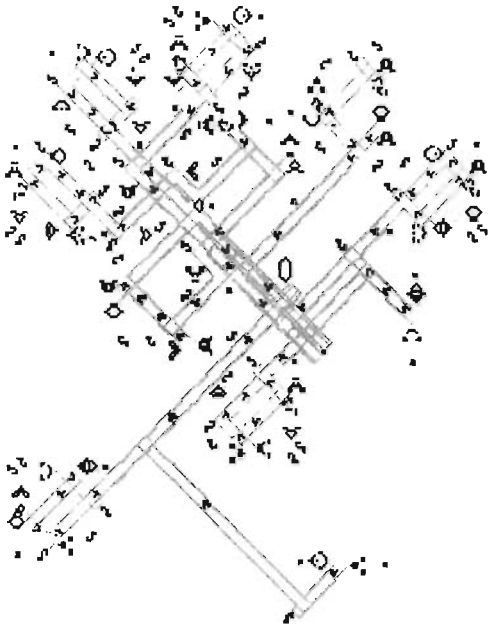

Figura 4.11: Configuraciones del sumador en el autómata del juego de la vida. Las zonas sombreadas muestran agrupaciones de patrones tales como una pistola de gliders de periodo 60. Las lineas muestran la ruta de los gliders.

práctica del concepto descrito por Conway para construir una máquina universal de cómputo basada en el trabajo de Minsky. Minsky empleó el concepto de máquina de registros, el cual tiene la propiedad de almacenar cualquier número positivo. La figura 4.13 muestra la construcción hecha por Hickerson. En la imagen se muestran los agrupamientos de patrones que sirven para diseñar una pistola de gliders de periodo 120.

El diseño incorpora cierto número de pistolas de periodo 120 para generar descargas de gliders que mueven al bloque y para hacer la prueba del cero. Estas pistolas se construyen usando varios arreglos de dos pistolas de periodo 30 con un glider que va y viene usando la reacción de contragolpe. El bloque

de memoria deslizante usa una descarga de dos gliders para decrementar el contador moviendolo diagonalmente un espacio. Luego se usan tres gliders para incrementarlo separandolos y un glider adicional se manda a través de la estructura durante el decremento. Finalmente, este es borrado por la operación de decremento cuando el bloque llega a cero.

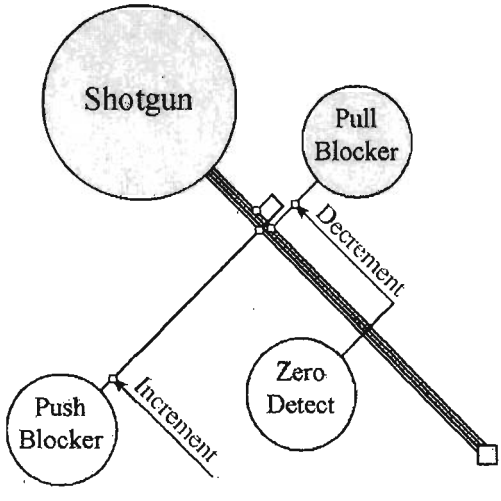

Figura 4.12: Esquema de la memoria de bloque deslizante.

La celda de memoria es otra configuración generada por juego de la vida con potencial para ser usada en una máquina de cómputo universal (Figura 4.14). Esta fue construida por Paul Rendell a principios de los 80. Rendell diseñó la celda para ser construida dentro de un patrón de matrices de celdas similares. El contenido de la celda se obtiene mediante la colisión de un MWSS que atraviesa la parte posterior de la celda y un LWSS subiendo por otro lado. La estructura que resulta de esta colisión interactúa con un pentadecatlon. Esto produce un glider que abre un agujero de ocho gliders en la entrada de la celda. La entrada es la pistola de gliders que dispara desde arriba hacia la

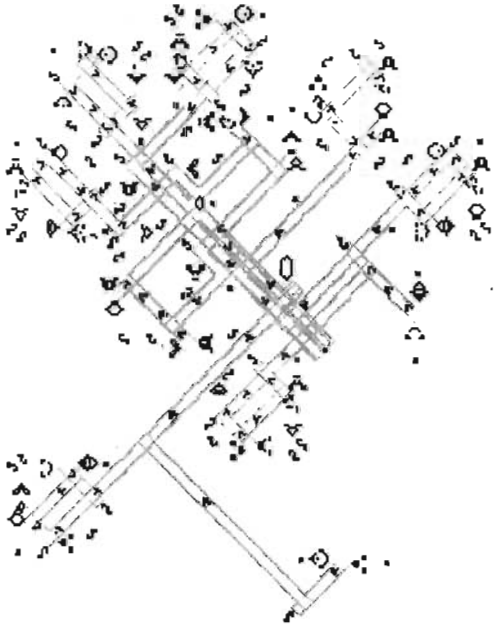

Figura 4.13: Configuraciones de la memoria de bloque deslizante.

derecha y bloquea la salida de la celda.

El núcleo de la celda es el patrón de flujos que se muestran en la figura A.3. Este patrón usa dos aguijones de abeja reina para reflejar las corrientes de gliders. Estos están colocados opuestamente de tal forma que la abeja reina que refleja el glider que viene de la izquierda, estabiliza el que refleja la señal de entrada proveniente de la derecha. En el proceso, señal de entrada es duplicada. Para la celda de memoria, tres buckaroos se usan para hacer un ciclo que regrese una salida hacia la entrada de tal forma que el patrón en el ciclo se repita indefinidamente.

Un buckaroo (como lo define Rendell) es un aguijón de abeja reina estabilizado al final por un comedor, de tal forma que este puede interactuar con un

glider y producir otro glider en ángulos rectos desde el original.

Figura 4.14: Celda de memoria.

Paul Rendell hace una descripción con buen nivel de detalle de la construcción de una máquina de Turing a partir de la dinámica del juego de la vida. Los siguientes párrafos corresponden al mismo artículo que Rendell escribió para mostrar la capacidad del juego de la vida para soportar cómputo universal.

Para iniciar la construcción de la máquina de Turing se usa la celda de memoria que se describe en la sección anterior. Esta puede ser usado como base para cualquier máquina de cómputo universal que se construya a partir de la máquinaria que proporciona el juego de la vida. La celda de memoria puede ser construida dentro de una unidad de memoria que contenga varias celdas. El sumador provee la habilidad de incrementar la dirección que apunta a la siguiente instrucción de la unidad de memoria y la memoria de bloque deslizante es un registro de contador capaz de almacenar valores de cualquier tamaño.

La figura A.1 del apéndice muestra el diagrama de una máquina de Turing. La máquina de estados finitos contiene la unidad de memoria, construido a partir de las celdas de memoria que se describen en la sección anterior. En cada ciclo de la máquina de Turing, la unidad de memoria envía su salida al detector de señales y al stack. El detector de señales separa el siguiente estado de la salida y lo envía a través de un ciclo de retardo de regreso a la máquina de estados finitos donde formará parte de la dirección del siguiente ciclo.

4.3.2. La máquina de estados finitos

La máquina de estados finitos (Figura 4.15) contiene una unidad de memoria que se construye con un arreglo de celdas de memoria descritas en la sección anterior. Tiene dos entradas. Una que viene del detector de señales que representa el siguiente estado y es usado para seleccionar un renglón. La otra entrada viene de uno de los stacks que es el símbolo leído y usado para seleccionar una columna. La estructura al pie de la columna seleccionada genera un LWSS y la estructura al final del renglón seleccionado genera un MWSS. Dichas naves viajan a traves desde la matriz y cada una colisiona en la celda seleccionada. La salida de la celda seleccionada es recogida por una flota de ocho LWSS's enviada a lo largo del renglón seleccionado la cual eventualmente es recogida por otra flota de ocho LWSS enviada después de la columna final.

Para alimentar la dirección del renglón al final de cada renglón se usa un disparador o pistola de naves espaciales MWSS de periodo 30. Esta pistola fue diseñada por Dieser Leithner y varias personas más. La selección de una columna se muestra en la figura 4.17. Los gliders de la pistola de gliders a la derecha son destruidos por el MWSS de la cadena de localización, pero sobre-

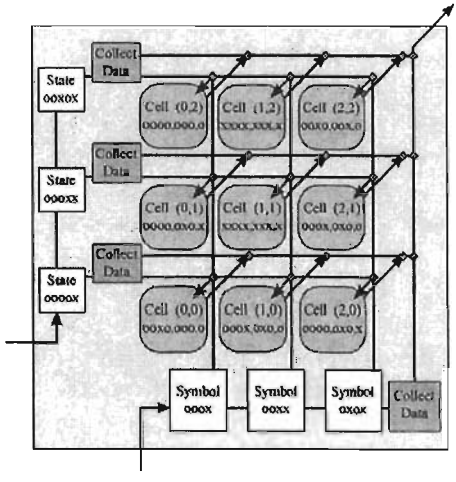

Figura 4.15: El Esquema de la máquina de estados finitos que consiste de una unidad de memoria y dos entradas. Una representa el siguiente estado y la otra el símbolo leido.

viven si algún MWSS se pierde. La estructura resultante es comparada con el contenido de la celda de memoria que está en la parte superior. El resultado de la colisión es percibida por la salida de otra pistola de gliders, los gliders de la pistola son destruidos debido a error de compatibilidad. Cuando no se destruyen, los gliders forman el resto de un latch set reset en el centro de la configuración.

La parte del reset es la salida invertida de una pistola de gliders de periodo 240 (Figura A.14 del apéndice). La salida del latch es detectada al final del ciclo de localización por otra pistola de periodo 240 en la derecha. Si el glider de la derecha no es destruido por la salida del latch, éste acciona el mecanismo de la parte de abajo para formar un MWSS.

El mecanismo para seleccionar una columna (Figura 4.17) es muy similar al equivalente para seleccionar un renglón. La diferencia reside en que el generador de MWSS's final usado para selección de renglones es remplazado por un generador de LWSS's.

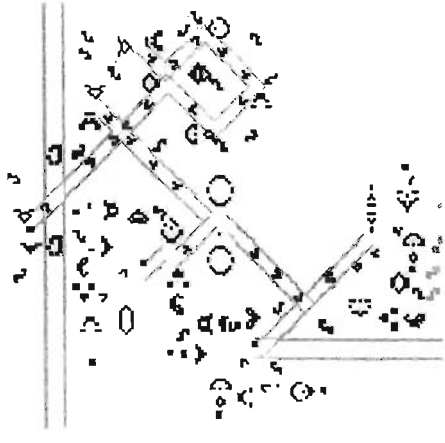

Figura 4.16: Selección de un renglón.

El diseño del latch usado en la selección de renglón y columna explota los dos tipos de colisión de dos cadenas de gliders de periodo 30 que se encuentran a noventa grados y fuera de fase. Los huecos en una cadena cambian el estado del latch de tal forma que la cabeza de sus gliders interactúan con la cola de la otra cadena de gliders.

Por otro lado, los boquetes en la otra cadena de gliders cambian el modo de regreso. En esta versión, ambos modos son establecidos con un pentadecathlon (un oscilador de periodo quince). Uno de estos modos del latch produce los gliders de salida. La figura 4.18 muestra algunas configuraciones del latch.

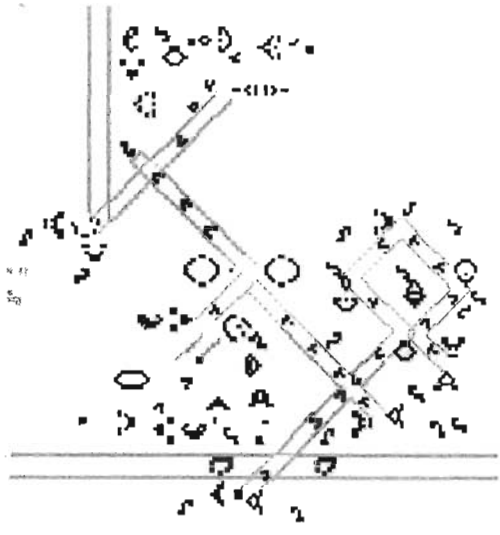

Figura 4.17: Selección de una columna.

La figura 4.19 muestra la estructura usada para recoger la salida de la celda de memoria seleccionada. En la figura, el MWSS generado por el comparador de dirección de renglón, es detectado y usado para hacer un agujero de ocho gliders en la cadena de gliders que bloquea la salida de la pistola de LWSS de periodo 30. Esto libera ocho LWSS's que recolectan la información de la celda de memoria seleccionada en alguna parte bajo el renglón.

De igual manera, en la figura 4.20 se puede apreciar la variación del diseño usado para seleccionar los LWSS's sobrantes al final del renglón seleccionado y transferir la información dentro del stack. La estructura es accionada directamente desde el MWSS de la dirección de la columna y la pistola de periodo 240 es usada para detectar la etiqueta de la *dirección presente* que es adherida a los números de renglón y columna de tal forma que el valor cero puede ser usado.

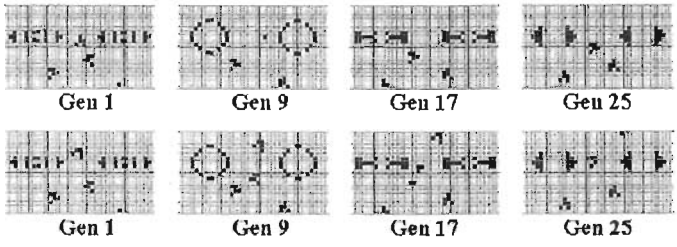

Figura 4.18: Algunas configuraciones del latch set reset usando pentadecatlones. El renglón superior muestra el modo de reset y el renglón inferior muestra el modo de set para el latch.

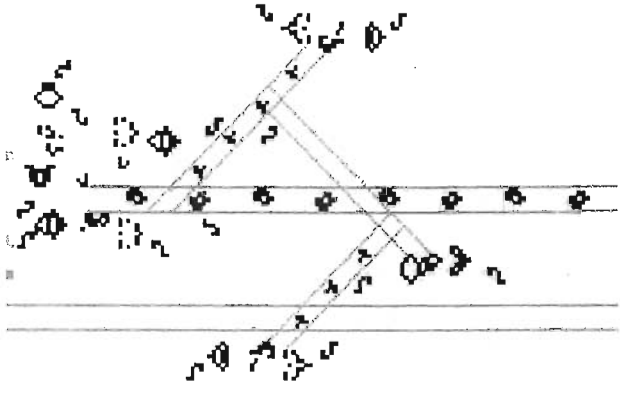

Figura 4.19: Colector de datos del renglón.

La cinta de estados de la máquina de Turing se construye a partir de dos stacks pues para mover la cinta sobre la cabeza de lectura y escritura es necesario un stack que lleve a cabo una operación *pop* y otro para hacer una operación *push*. De esta manera las piezas de la cinta no se representan con el símbolo actual sobre ella. La misma construcción del autómata celular reemplaza este símbolo llevando su representación hacia uno de los stacks al principio de cada ciclo de lectura.

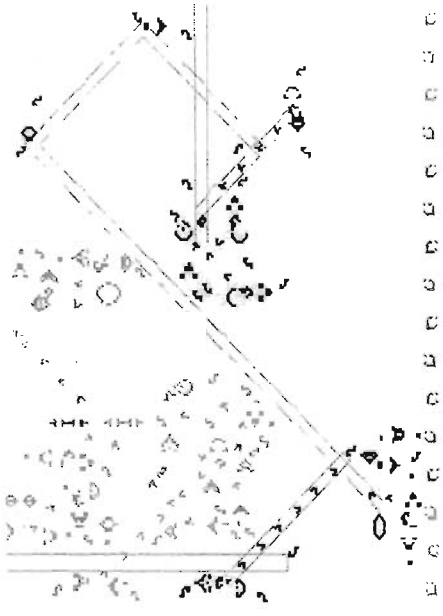

Figura 4.20: Colector de datos de la columna.

El esquema del stack se muestra en la figura A.4 del apéndice. El diseño se escoge así para usar la reacción de contragolpe y atrapar a los gliders, representando el símbolo entre dos cadenas de gliders opuestas. Para retardar los gliders de símbolos durante la operación push, un arreglo de pentadecatlons se usa para crear una ruta de convolución hacia la siguiente celda. Un arreglo similar es usado para la ruta de la operación pop con el propósito de mantener la alineación.

En figura A.5 del apéndice se muestra un ejemplo de dos celdas ligadas por el mecanismo de retardo. La figura muestra los patrones durante su desarrollo. La lógica para duplicar la cadena de gliders manteniendo los símbolos en una celda no está completa y no ha sido aplicada a las paredes externas. La celda de la parte superior en la figura está a punto de vaciar su símbolo a través de

un boquete de cuatro gliders de ancho. El agujero que deja pasar el primero en el fondo de la celda puede ser visto usando la señal de control de la izquierda.

La lógica para controlar la transferencia de información sobre y fuera de cada stack (push y pop) se muestra esquemáticamente en la figura A.6 La primera etapa se etiqueta como *conversión de control.* Una construcción ligeramente diferente se usa para cada stack, de tal manera que uno hace un push donde el otro hace un pop.

En la figura A.7 del apéndice se muestran las configuraciones del stack anterior, la cual incluye el mecanismo lógico de *control izquierdo* que se muestra en el esquema del stack. La salida de la máquina de estados finitos sube desde la mitad derecha de las configuraciones del stack donde golpea a una pistola de gliders la cual invierte la señal. Una pistola de periodo 240 recoge esta señal invertida y la deja pasar sólo si la salida de la máquina de estados finitos tiene un glider presente en la posición que indica una operación push para este stack. Si el glider recolectado continúa, entonces se convierte en la entrada de un flujo. Una salida del flujo se convierte en el control de push para el stack y el otro borra el glider de la *señal presente.* Si la operación es un push, entonces el glider de la *señal presente* no es eliminado e inicializa la operación pop.

La figura A.8 del apéndice muestra la configuración de la implementación *conversión de control* para la parte inferior del stack. Esta capa es un poco diferente respecto a la anterior, pues esta hecho para que el glider de la *señal presente* que baja desde la izquierda se convierta en el control del operador push.

En la parte superior izquierda del stack se muestra la forma en que se crea la señal de control para el lado izquierdo de la parte superior del stack. Los agujeros en la señal de control que suben hacia el lado izquierdo de la parte superior del stack crean aberturas en las paredes de las celdas del stack para que los símbolos salgan durante un pop y entren durante un push. El control del pop fluye hacia fuera de tal forma que una copia se dirige hacia el control del stack de la derecha y la otra copia hace un agujero de cuatro gliders en el control del stack de la izquierda.

La operación push necesita tres copias tal como se aprecia en el esquema del stack (Figura A.6). Una va al control de la parte derecha del stack derecho, uno más va hacia la entrada que recibe información dentro del stack (Figura 4.21 del apéndice), y otro va hacia la estructura en la parte baja central de la parte superior izquierda del stack, el cual hace los tres agujeros requeridos para los gliders del símbolo que entra a la celda de stack.

Este perforador de tres agujeros es ahora un bit más grande que puede ser hecho con dos flujos, pero su impacto visual lo maquilla. Está hecho de tres pistolas de periodo 120 sincronizadas de tal manera que cada una pone un agujero en el control de stack pero las salidas de las tres son bloqueadas por otra corriente de gliders. El control de pop hace un agujero de tres gliders en éste para permitir que pasen. En la figura A.6 del apéndice se muestra el control del stack derecho el cual es muy similar. El glider de control del push hace un agujero de cuatro gliders en la señal de control para permitir que los gliders de símbolo salgan de las celdas del stack y el glider de control del pop activa a un perforador de tres agujeros para hacer los hoyos de acceso.

La entrada que aloja el símbolo dentro del stack (Figura 4.21) alimenta los gliders de símbolo en cada ciclo. Esto se efectúa con un pedazo de un ciclo de retraso que se muestra en la parte inferior derecha. Un glider del control de stack llega durante una operación push y hace un agujero de tres gliders en una corriente bloqueadora de gliders para permitir a los gliders de símbolo entrar sólo en el ciclo de pop. Esos gliders hacen un agujero en otra cadena bloqueadora de gliders. En este momento, la cadena bloquea la salida de tres pistolas de periodo 20 las cuales estas alineadas y sincronizadas para inyectar los gliders símbolo dentro del stack. El control de stack normal tendrá asegurado que pared de la celda del stack tenga agujeros para permitir entrar a los gliders símbolo.

Para obtener los gliders de símbolo fuera del stack durante una operación pop se usa un pequeño truco. La figura 4.22 muestra tal configuración. Una pistola de periodo 120 en la parte inferior derecha es normalmente bloqueada por la pared de la celda del stack. Esto tiene dos funciones. Primero, el agujero que hace, junto con el agujero hecho por algún símbolo crea un patrón de cuatro gliders el cual es ideal para direccionar la máquina finita de estados. Este agujero extra se convierte en la etiqueta de *dirección presente.* En segundo lugar, durante una operación pop, los cuatro agujeros que se requieren para sacar a los tres gliders también dejan salir a la pistola de periodo 120. Esta pasa frente al stack donde hace un agujero de cuatro gliders de ancho en una cadena bloqueadora de gliders. El patrón de gliders que se dejó pasar es la salida del stack.

Los gliders en la celda de stack son destruidos por tres copias de una estructura oscilante de periodo ocho conocido como bloqueador. Esta estructura

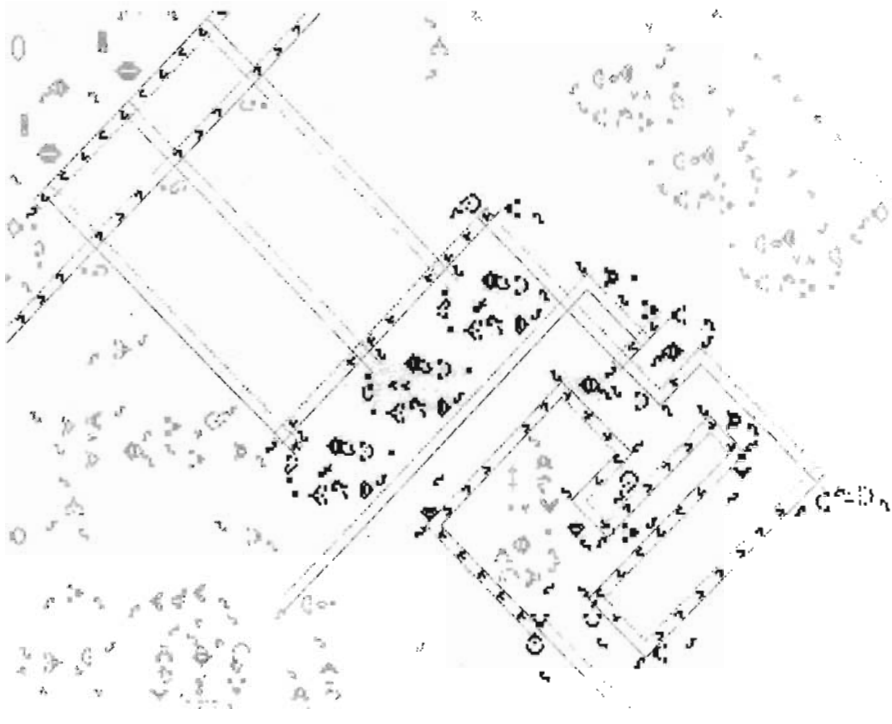

Figura 4.21: Entrada de símbolos del stack.

también es usada para la pistola de periodo 120 (Figura A.13 del apéndice). El que se usa en el stack es una pistola de periodo 60 con el bloqueador puesto para eliminar la mitad de aquel. La salida de ambos stacks es combinada mediante una simple reacción invertida y retroalimentada a la máquina finita de estados para formar parte de la dirección.

4.3.3. El acoplamiento de la máquina de estados finitos con los stacks

Para conectar a la máquina de estados finitos con los stacks necesitamos que la información proveniente de la máquina finita de estados tenga que ser fracturada para alimentar los stacks y el siguiente estado. El siguiente estado

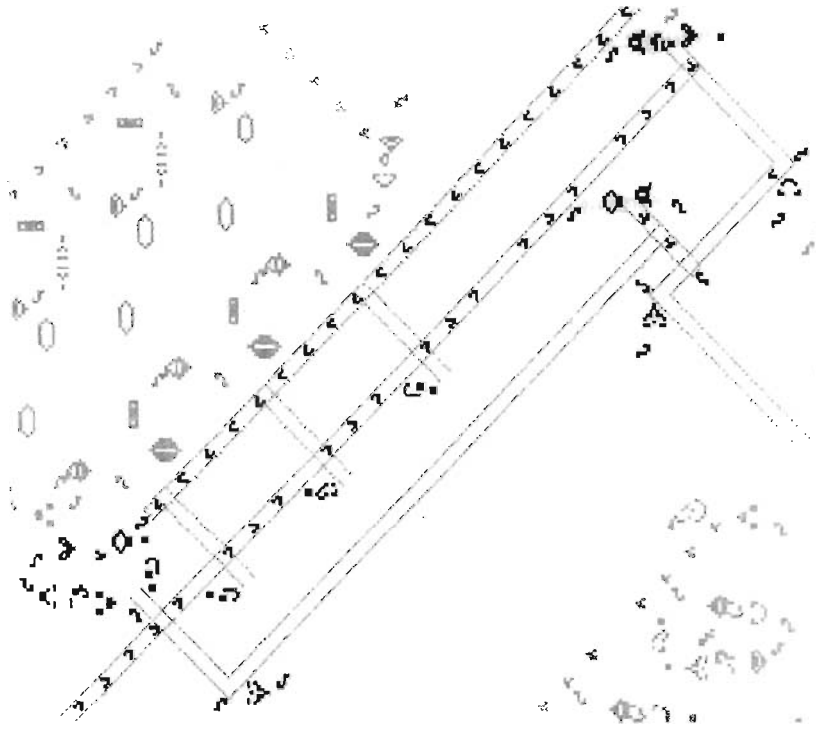

Figura 4.22: Salida del stack.

debe ser regresado a la máquina finita de estados al momento que un símbolo es obtenido de uno de los stacks. La máquina finita de estados es accionada para generar una salida recibiendo direcciones que coincidan con las direcciones de renglón y columna. Aquella dirección debe ser sincronizada con el ciclo de tiempo de 240 generaciones de las celdas de memoria que mantienen la dirección y están arregladas de tal forma que un valor cero no es usado como una dirección. Una solución similar fue adoptada para la operación de accionar los stacks. Esto implica construir un detector de señales que pueda detectar un valor no cero en cualquier momento. El resultado se muestra en la figura A.10.

Una mecanismo que se asemeja a un latch set reset esta localizado en el corazón del detector. El latch es similar al mostrado anteriormente. Sin embargo, este usa su propia salida para resetearse a sí mismo después de 240 generaciones. El diseño usa un aguijón de abeja reina que refleja gliders en la misma dirección que el ventilador de flujos.

Una modalidad del latch previene a la abeja reina de reflejar un glider mientras que el otro modo no. El ciclo de retroalimentación negativa que reajusta el latch también contiene un ventilador de flujos. Nuevamente para hacer que las cosas encajen, la reacción de inversión tiene que ser estabilizado con un pentadecatlon. La otra parte del ventilador bloquea la salida de una pistola de periodo 240. Si el latch detecta cualquier glider en periodo 240 entonces la salida de esta pistola no es bloqueada y se convierte en el glider de *la señal presente.*

La figura A.11 del apéndice muestra la siguiente etapa del acoplamiento. La información original de la máquina de estados finitos y la salida del detector de señales pasan hacia cada stack con otra copia de la información empezando un ciclo largo de regreso a la máquina de estados finitos. Esto es modificado en la parte baja del mecanismo mediante el uso de la salida del detector de señales para crear la marca de la presente dirección para la localización del renglón de la máquina de estados finitos.

En la figura A.12 del apéndice se aprecian las configuraciones que hacen formal la dirección del siguiente estado borrando tres gliders. Esto es realizado usando una pistola de periodo 240 para crear un agujero de tres gliders de ancho, invirtiendo el resultado y borrando los tres gliders principales en cada

marco. Esto deja la etiqueta de *la dirección presente* seguida por el siguiente estado. Las otras salidas del distribuidos de señales van hacia cada stack. Ambos stacks toman un glider de *señal presente* y la información de la celda de memoria. Aquellas son entradas para la parte superior izquiera e inferior de stack.

Esto completa la descripción de Rendell sobre la arquitectura de la configuración de una máquina de Turing. Tal construcción está en si misma configurada con información en la máquina de estados finitos para representar una máquina de Turing especifica y el stack está ajustado con datos iniciales para el proceso de esta máquina.

En el trabajo de Rendell se muestra la implementación de una máquina de Turing sencilla. Esta duplica un patrón de 1's. Las flechas muestran la transición entre estados con el símbolo que inician la transición en la cola de la flecha y el símbolo a escribir y dirección de movimiento en la cabeza de lectura en la mitad de la flecha.

La máquina empieza con dos 1's en la cinta a la derecha de la cabeza lectora y esta toma 16 ciclos para detenerse con cuatro 1's en la cinta. En el ejemplo, la transición de comienzo se acciona por una pistola de gliders de periodo 240 colocada detrás de una cadena bloqueadora de gliders. Esto es sincronizado del tal forma que cuando el glider bloqueado es borrado, este inserta la instrucción en la ruta tomada por gliders del stack.

Rendell señala que en efecto, una máquina universal de Turing necesita de una máquina de estados finitos más grande que la simple presentada en el ejemplo y que la arquitectura del mecanismo mostrado en su trabajo es limi-

tado a una máquina de 16 estados y 8 símbolos. Tal limitación es construida en la forma en que los contenidos de las celdas de memoria son interpretados. Las celdas de memoria mostrados anteriormente sólo mantienen 8 bits de información. Esos son interpretados como 4 bits para el siguiente estado, 3 bits para el símbolo a escribir y 1 bit para la dirección de movimiento. Un valor cero detiene la máquina.

Minsky mostró que una máquina universal de Turing puede ser construida con sólo 7 estados y cuatro símbolos. Así que la arquitectura básica no es una restricción. Agregar renglones y columnas a la arquitectura de la máquina de estados finitos descrita se hace de manera directa. La única dificultad es que el tiempo del ciclo de la máquina debe ser modificado en múltiplos de 204 generaciones. Sin embargo hay una variedad de técnicas para hacer ajustes de sincronización, del tal forma que sea fácil superar esto.

Una limitación de la configuración descrita aquí es que la máquina tiene que ser provista con suficiente cinta en blanco para llevar a cabo sus cálculos. Esto es un requerimiento para todas las máquinas de Turing. Se puede argumentar que el juego de la vida de Conway tiene un espacio vacío teóricamente ilimitado a su disposición y que un diseño que pueda hacer uso de esto sería una mejora. El autor (Paul Rendell) está trabajando actualmente en una configuración que se pueda mover a través del espacio vacío, construyendo celdas de stack en blanco mientras avanza, para extender el stack más rápido de lo que la máquina puede usar. Construir una configuración dinámica razonablemente densa usando colisiones presenta una muy serie de problemas muy diferentes que aquellos superados en la construcción de la máquina de Turing presentada aquí.

En 2002, Chapman [5] dió una construcción explícita de una máquina de Turing Universal basada en el modelo de Rendell. Esta construcción demostró finalmente que el juego de la vida es capaz de soportar cómputo universal.

4.3.4. Comentarios

El autómata del juego de la vida es muy sencillo de describir, sólo bastan unos renglones para definir la tabla de reglas de transición que constaría de 512 renglones. Pero, tal sencillez contrasta con la riqueza de objetos complejos que se crean al paso que el autómata evoluciona, tales pueden ser estáticos o dinámicos, los primeros objetos nos sirven para representar información, ya sea un programa o datos iniciales. Los segundos nos ayudan a transferir la información. Las colisiones entre gliders son útiles para construir predicados y tomar decisiones. En todo eso radica la capacidad del juego de la vida para ser usado como material para construir una máquina de Turing, y por ende, ser declarado universal.

El juego de la vida es un ejemplo claro de complejidad emergente. Esta cualidad es la que brinda una gran riqueza de formas y dinámicas que pueden ser utilizadas como herramientas, siendo los gliders la materia prima más común. El trabajo de Rendell es un trabajo casi artesanal, semejante a un artista que sabe construir formas a partir de otras más elementales. En los párrafos anteriores se muestró de manera muy somera los pasos necesarios para construir un mecanismo, pero sin duda es posible hacer otro tipo de construcciones computacionalmente equivalentes usando los mismos materiales. El autómata celular nos genera herramientas y nosotros les damos un uso.

4.3.5. Una implementación semicontinua del juego de la vida

La construcción expuesta anteriormente demuestra la capacidad para efectuar cómputo del juego de la vida. En este apartado se mostrará una implentación desarrollada por el autor de este trabajo, la cual consiste en un autómata celular de muchos estados con celdas que varían de forma semicontinua espacialmente y cuyo comportamiento es cualitativamente equivalente al que se aprecia en el juego de la vida.

Los estados de este autómata celular son representados por valores numéricos entre 0 y 1. La dinámica de este autómata celular consiste en lo siguiente: En el primer paso se deben sumar las cantidades contenidas en una vecindad de Moore excepto la celda central, para las cuales se ha superado el umbral h=0.5, llamemos s a esta cantidad. El siguiente paso consiste en aplicar las reglas del juego de la vida como sigue:

- Si la celda central esta por arriba del umbral y $2 \leq s \leq 3$ entonces la celda toma el valor de $s/3$, es decir, queda por debajo del umbral.
- Si la celda central esta por arriba del umbral y $1 \leq s < 2$ entonces la celda toma el valor de $s/2$, es decir, queda por arriba del umbral.
- Si la celda central esta por debajo del umbral y $2 \leq s \leq 3$ entonces la celda toma el valor de $s/3$, es decir, queda por arriba del umbral.
- En otro caso, si el promedio de todas las celdas vecinas es menor al umbral entonces la celda central toma el valor de este promedio, si no, la celda central toma la mitad de este promedio, es decir, de cualquier manera el valor de la celda queda menor al umbral.

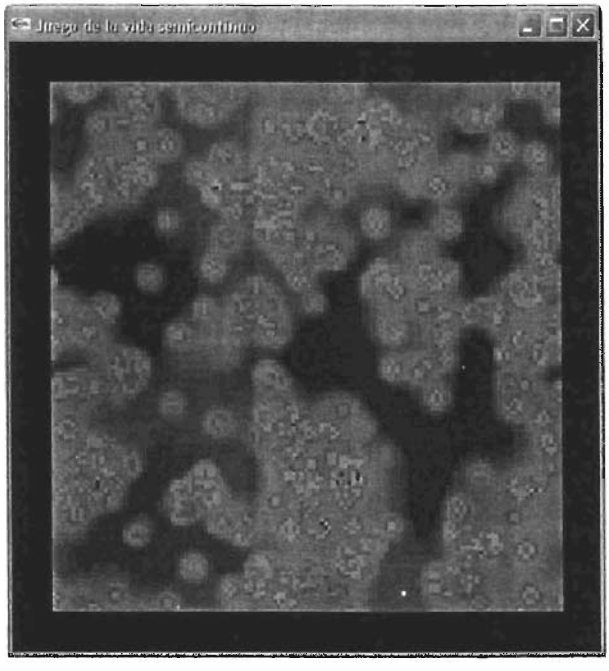

Figura 4.23: Las configuraciones del juego de la vida semicontinuo.

Esta implementación muestra a un autómata celular con capacidad universal de cómputo con mayor robustez que los AC tradicionales pues el mecanismo acepta condiciones iniciales con cierto nivel de ruido y que no afectan a la convergencia final del sistema.

4.4. Cómputo universal en el autómata regla 110

La regla que se asocia al número 110 de acuerdo al sistema de identificación de Wolfram (Capítulo 2) es particularmente interesante debido al intrincado comportamiento del autómata celular que la implementa. Este AC genera pa-

trones que evolucionan e interactúan dinámicamente así como los gliders del juego de la vida. Tal comportamiento fue en principio estudiado en 1992 por Wentian Li y Mats G. Nordalh quienes realizaron estudios estadísticos que ilustraron el comportamiento de la regla 110.

Al apreciar los patrones que emergen en el espacio de celdas durante la evolución del AC con regla 110, se puede percibir un fondo extendido y dinámico formado por triángulos isósceles de diversos tamaños al que Cook llamó éter por razones obvias, y que es una de tantas configuraciones estables que aparecen bajo la evolución de la regla. Dada esta comparación, los gliders que surgen en el autómata celular pueden ser vistos como dislocaciones o defectos del éter contrastando con el juego de la vida, donde los gliders surgen sin necesidad de un medio. Pero tal característica hace difícil el análisis, pues la interacción de las estructuras del éter y los gliders pueden generar otras que visualmente son muy parecidas.

En 1991, Wolfram escribió: *El contorno general de que tiene que ser hecho ha sido bastante claro, pero hay un inmenso número de detalles a ser manejados. Y le pedí a un joven asistente mío llamado Matthew Cook que los investigara. Sus resultados iniciales fueron animosos, pero después de unos cuantos meses el se mostró enormemente convencido de que la regla 110 nunca sería probada universal. Yo insistí, sin embargo, el seguía intentando, y sobre los siguientes muchos años el desarrollo un sistema de diseño ayudado por la computadora para trabajar con estructuras en la regla 110. Usando esto, en 1994 él fue exitosamente capaz de buscar los elementos principales para la prueba. Muchos detalles fueron cubiertos sobre el siguiente año, algunos errores fueron corregidos en 1998, y la versión específica que anoto abajo fue*

construida en 2001. Como muchas pruebas de universalidad, la prueba final que él encontró es conceptualmente absolutamente directa, pero está llena con muchas detalles elaborados. Y entre esos detalles es ciertamente posibles que algunos errores aún permanezcan. Pero....

Para probar la universalidad de la regla 110 a la que se refiere Wolfram primero es necesario describir otro tipo de máquina formal, conocida como sistema de etiquetado (Tag System, comúnmente denotado por v o b), y también un mecanismo relacionado denominado sistema de etiquetado cíclico, muy parecido al sistema introducido por Post.

Los resultados sobre sistemas de etiquetado de Post no se conocieron ampliamente hasta su publicación años más tarde (1943). Post aparentemente estudió todo acerca de tales sistemas que incluían borrado y adición de no más de dos elementos en cada paso y concluyó que ninguno de ellos producía algún comportamiento complejo. Sin embargo, al trabajar con reglas que removían tres elementos en cada paso, descubrió que regla particular variaba considerablemente con las condiciones iniciales. En general, si el número de los que se agrega nunca es mayor que el número de los que se eliminan, el comportamiento resultante será al menos cíclico. Esto es, si se permiten adiciones hasta cierto tamaño se obtendrá un comportamiento complejo interesante.

Los sistemas de etiquetado también enfrentan un problema de decisión parecido al halting problem de las máquinas de Turing. Esta decisión se basa a partir de alguna condición inicial dada arbitrariamente y si aplicadas las reglas se llega a una palabra de longitud menor que el número de elementos removidos del principio.

Minsky probó que una máquina de Turing puede ser representada como un sistema de etiquetado y probo la existencia de universalidad e indecisión de halting en sistemas de etiquetado, el resultado fue mejorado por Cook, Misky y Wang. Wang también consideró una clase de sistema opuesto al de etiquetado que basó en un sistema de retraso. Maslov probó que para cualquier $v \geq 2$, hay un sistema de etiquetado tal que ambas formas enunciadas por Post no tiene solución.

Un tag System Cíclico es aquel donde n reglas de etiquetado se aplican a un sistema en orden secuencial hasta empezar de nuevo por la primera regla. En un tag system cíclico, cada regla de etiquetado es tal que un patrón es añadido si y sólo si el primer elemento del patrón actual es 1, independientemente si el primer elemento es 1 o 0, este es borrado.

Los elementos necesarios para probar la universalidad de la regla 110 parten de la existencia de estructuras dinámicas parecidas a los gliders del juego de la vida. Tales estructuras nos sirven para diseñar una máquina con la capacidad de soportar cómputo universal. Para el caso del juego de la vida, se tiene construida una máquina de Turing basada en una máquina de registros propuesta por Minsky, y se ha descrito de manera general el procedimiento para llevar a cabo esa tarea. Que dicho sea de paso, no constituye una demostración formal, pero si propone una manera en que demostrar.

En el caso del autómata con regla 110, Cook hizo la construcción de un sistema de etiquetado cíclico con la propiedad universal de cómputo a partir de la dinámica de las interacciones de los gliders que se conocen para la regla 110.

La dinámica de interacciones de los gliders se aprecia claramente sobre el éter de Cook que esta formado por mosaicos triangulares lo cuales se pueden clasificar de acuerdo a MacIntosh [16] como T_n, donde n es el número de ceros en su renglón superior, siendo estos completamente bordeados o semibordeados. Los triángulos semibordeados se complementan con las celdas del borde de triángulos adyacentes en las aristas horizontales y verticales.

4.4.1. Los gliders

En la evolución del autómata celular elemental con regla 110 se pueden apreciar muchos tipos de gliders, pero sólo unos cuantos son los que tienen potencial para interactuar y son aquellos que están asociados al éter de Cook. Las configuraciones de gliders parten del hecho de que estos están conformados por mosaicos triangulares. Esta clasificación se puede encontrar en el trabajo de MacIntosh, y sus descripciones espaciales se encuentran en el apéndice.

Existen reglas de paridad que se aplican a las deformaciones del éter, la más utilizada es la paridad binaria simple. Esto es, los gliders son o bien pares o impares, y combinaciones de gliders suman sus paridades. La paridad resultante debe ser conservada cuando ocurre una colisión.

Cuando ocurre una colisión entre gliders impares entonces el resultado es un glider par o un glider nulo, o se produce otro par de gliders impares que podrían ser los originales que solamente cambiaron de lugar, esto forma ondas viajeras que fueron descubiertas por MacIntosh al analizar sistemáticamente los choques binarios entre los gliders existentes y pueden ser usados para la transmisión de información desde los extremos de un bloque de estructuras.

A través de análisis experimentales, se ha revelado la existencia de gliders que se han cuantificado de acuerdo a posibles cambios de periodicidad. Otros experimentos revelan el resultado de colisiones entre gliders, sin embargo, debido a la enorme cantidad de estos, el trabajo se hace difícil. Se han descubierto combinaciones de solitones como los gliders C, F y Ebar.

Otras estructuras absorben gliders a semejanza de un agujero negro y otros son emisores como las pistolas de gliders. En la actualidad sólo una pistola de gliders es conocida, la que genera gliders A's y B's en cantidades similares. También se conoce la existencia de estructuras en la regla 110 que nos permiten hacer predicados. Hay procedimientos que dividen un flujo en dos partes, una de las cuales es borrada por el predicado. Con esto se pueden tomar la decisión de cuando borrar un predicado.

Las colisiones que nos interesan para construir el sistema de etiquetado cíclico son aquellas que nos permitan eliminar, traspasar barreras y hacer posible la toma de decisiones. Es por eso que el glider Ebar toma un rol importante en la función de traspasar barreras a manera de solitón u onda viajera.

El sistema de etiquetado cíclico consta de una parte estática compuesta de gliders C, en especial gliders C2, jugando el rol de bit debido a su relativo espaciamiento entre ellos. Los gliders C2 tienen un papel activo y pasivo. El modo pasivo debe permitir que todos los gliders Ebar pasen libremente mientras avanzan de izquierda a derecha. Debido a que tal encuentro requiere un contacto entre la interacción del Ebar y el triángulo T6 del glider C2, entonces el espaciamiento entre C2's debe ser importante en esta relación. Solamente

puede variar incluyendo uno o más celdas unitarias extras del EBar. La figura A.15 del apéndice muestra las cuatro posibilidades para una colisión entre el glider C2 y el glider EBar, una de ellas es solitonica, la cual define características para el sistema de etiquetado.

En la misma figura A.15, la primera colisión muestra un comportamiento solitonico, dejando pasar el glider C2 a través del Ebar. En la segunda, un glider C1 se genera después de la colisión, y el glider Ebar se convierte en un glider tipo F. Esto da la posibilidad de tener un mecanismo para intercambiar dos pares. La tercera colisión de la figura muestra una típica colisión C-E, la cual no es importante en este contexto. La última colisión constituye el mecanismo por el cual los predicados son implementados en el sistema de etiquetado cíclico.

Los gliders C1 y C3 juegan un papel secundario en el sistema de etiquetado cíclico, sin embargo no dejan de ser importantes en el proceso de construcción del sistema. Los C3's se alternan con pares de C1 cuando es requerida la transmisión de una hilera de EBars. En la figura A.16 del apéndice se muestran las cuatro colisiones C3-EBar. Dos se disipan en grupos de B, uno de ellos en tetadra. La cuarta interacción juega un papel crítico en el sistema de etiquetado Cíclico, pues este alterna un par de C2 con C3 para crear una membrana semipermeable que permite pasar a un Ebar a través de este.

Otro requerimiento para que el sistema de etiquetado cíclico pueda funcionar es la colisión C1-C1. La separación entre gliders C1 no puede ser suficiente para que la interacción con EBar pueda ser realizada de manera individual. En la figura A.17 del apéndice se muestran las cuatro posibles colisiones C1-C1-

Ebar. Esas colisiones tienen que producir residuos impares, al menos un glider impar. La cuarta colisión es solitonica debido a que el Ebar atravieza dejando al par de C1's más separados que anteriormente. La tercera colisión también es solitonica, dejando pasar al EBar y expulsando un podador de gliders A. La segunda colisión deja un simple glider D1. La primera colision es la más importante, porque elimina al último EBar para cerrar el ciclo que empiezó con el C3, y prepara el camino para el siguiente bit o la señal de stop en la cadena de Ebar's.

Una operación esencial para el sistema de etiquetado cíclico consiste en borrar estructuras, ya sean Gliders EBars dinámicos o gliders C estáticos. Una colisión interesantes es la que ocurre entre un podador de gliders A y un EBar para producir un glider D, el cual colisiona con otro EBar para restaurar al podador de A's original.

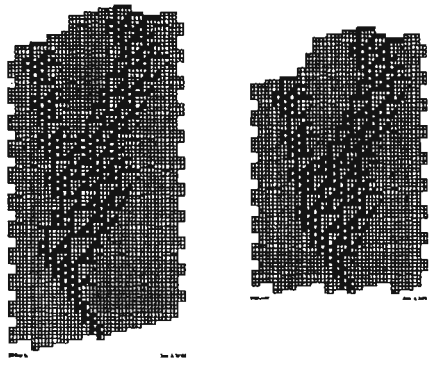

Figura 4.24: Las dos colisiones D1-EBar que pueden participar en el borrado alternado.

La demostración formal de que la regla 110 posee capacidades de cómputo universal requiere de una construcción muy complicada. Sin duda, este

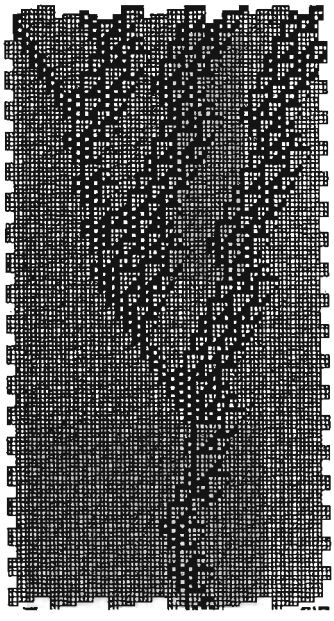

Figura 4.25: Tres gliders Ebar moviéndose son borrados alternadamente, primero por un podador de A's, luego por un glider D, y por último de nuevo por un podador de A's.

autómata celular exhibe configuraciones muy complejas, pero también difíciles de manejar. Los resultados mostrados a continuación son el resultado de muchos años de trabajo de varios científicos al tratar de encontrar y luego clasificar la enorme cantidad de estructuras que emergen a partir de las reglas de este autómata celular. Quizás este esfuerzo represente el comienzo de un intento por encontrar maneras para lograr demostrar la universalidad de otros autómatas celulares.

El argumento principal usado para construir una máquina universal en este autómata sigue estando respaldado por la existencia de objetos dinámicos conocidos como gliders. Algunos otros son estáticos y son claramente usados

para codificar programas, información, etc. Otros son dinámicos, y se usan para el transporte y la transformación de la información.

4.4.2. La lectura de un predicado

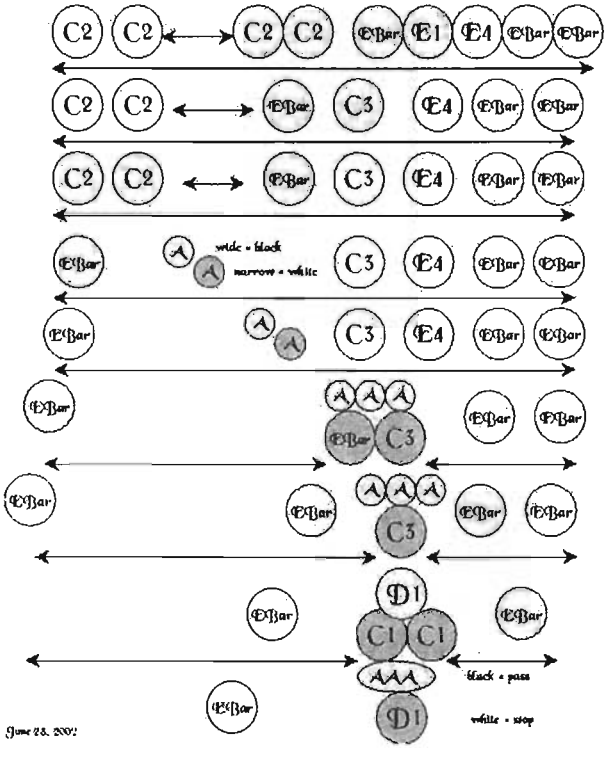

Figura 4.26: Las nueve etapas que se siguen en la lectura y ejecución de un predicado.

El glider que representa los datos debe ser estático, otro más, uno dinámico se usa para transportar la información que debe atravesar el grupo de instrucciones y detenerse hasta el final para entrar en un predicado y convertirse o no en un nuevo elemento de los datos.

La lectura de un predicado y su consecutiva respuesta (soltar un glider D1 o un eliminador de gliders A) involucra nueve etapas que se muestran en la figura 4.26. Las primeras tres etapas corresponden a la lectura de un elemento estático. El elemento estático está compuesto por varios gliders C2, dos de ellos tienen una separación variable para diferenciar entre valores negro o blanco del sistema cíclico. Un valor blanco corresponde a gliders ampliamente separados, un valor negro corresponde a gliders estrechamente separados.

Las siguientes cinco figuras muestran las cinco principales etapas que participan en la lectura y ejecución de un predicado para el sistema de etiquetado cíclico.

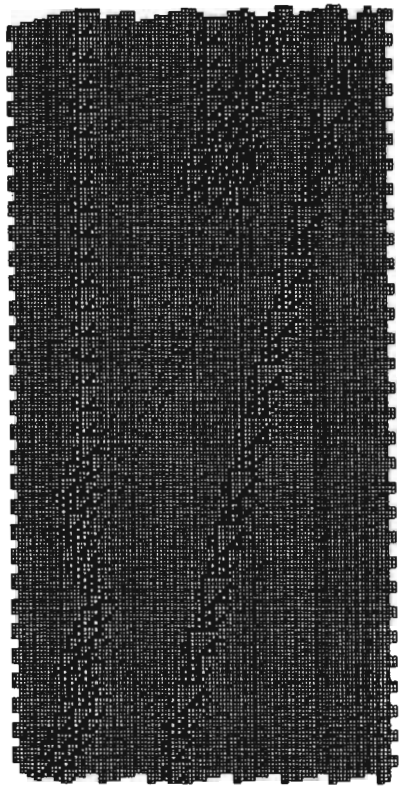

Figura 4.27: En la primera etapa, un par de C2 crea un Ebar y un C3. Ambos Cs están separados por quince mosaicos del éter acomodados en dirección noreste.

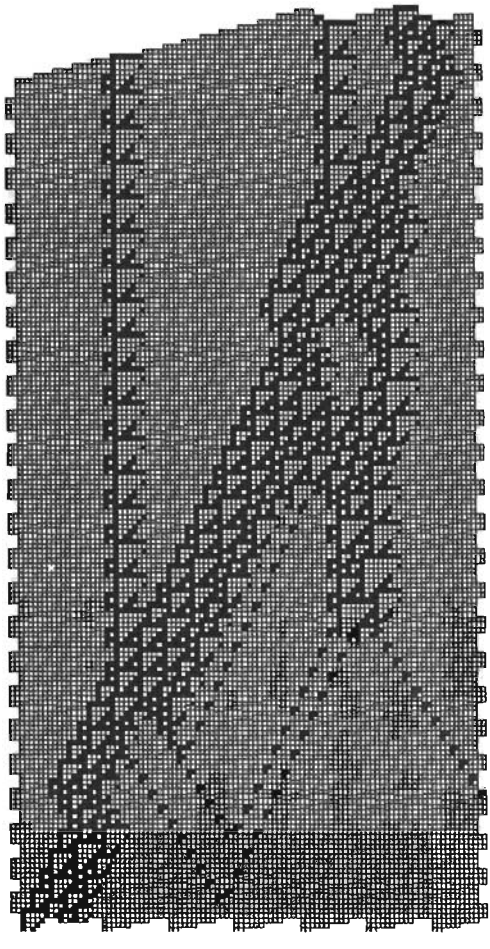

Figura 4.28: En la segunda etapa se usan el tercer y el cuarto C2 estático que se leen de derecha a izquierda. Un glider A se emite hacia la derecha, el momento de su emisión depende de que tan lejos se encuentre el tercer glider C2. También se crea un BBar que es detenido por el cuarto C2 para convertirse en un EBar. El tercer y cuarto glider C2 están separados por 19 mosaicos del éter. Dentro de estos mecanismos se eliminan mutuamente tres A's y tres B's. En esta etapa el resultado del predicado ya está determinado, lo determina la posición del glider A emitido hacia la derecha.

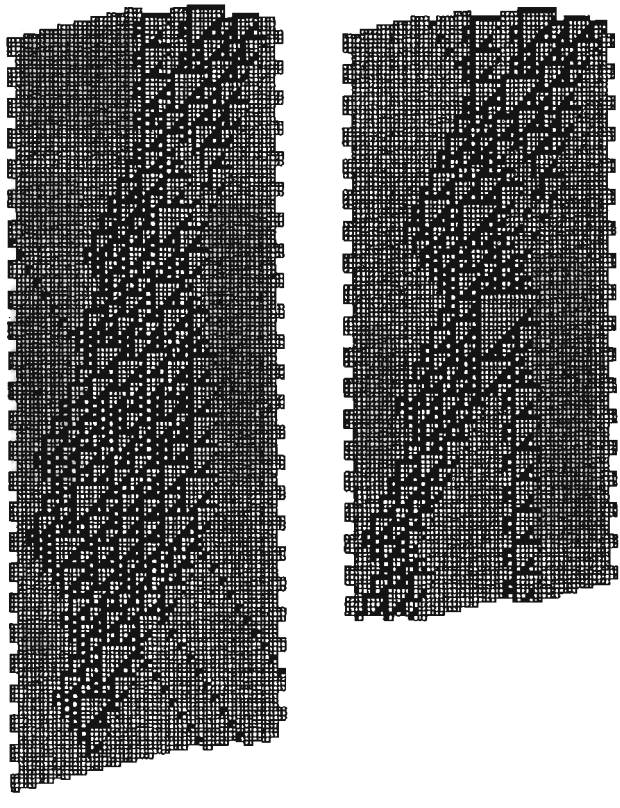

Figura 4.29: En la tercera etapa se genera el resultado del predicado dependiendo del momento en que llega el glider A. En la izquierda se muestra la interacción provocada por la lectura de un elemento negro. En la derecha se hace lo correspondiente para el elemento blanco. El resultado de cada interacción es una tripleta de A's o un C3 respectivamente.

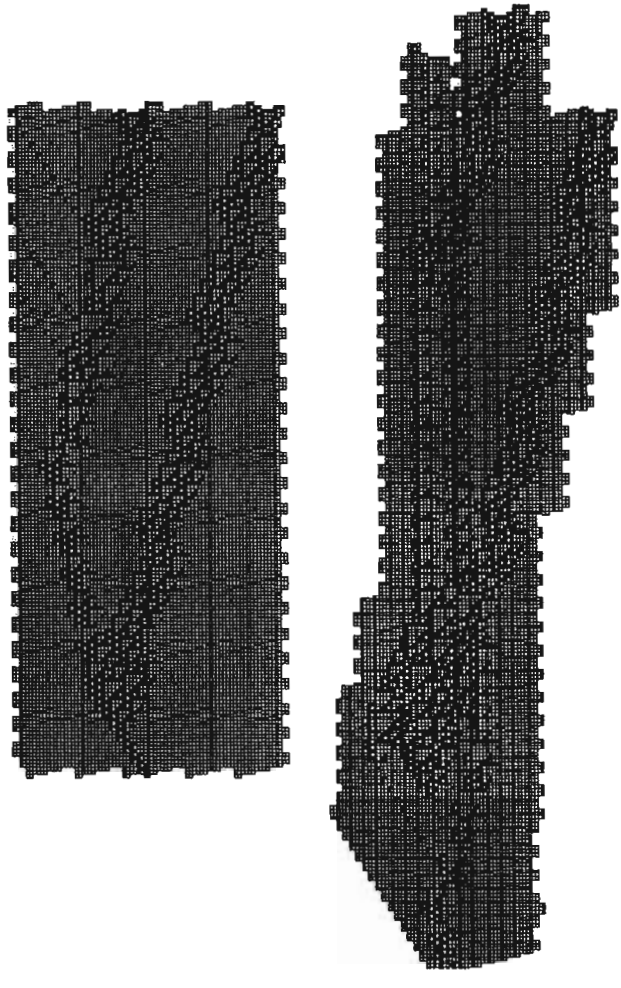

Figura 4.30: En la cuarta etapa dos Ebars que viajan desde la derecha se encuentran con la tripleta de A's (izquierda) o un C3 (derecha), creando un A-Podador o un D1 para el predicado negro y blanco respectivamente. Este resultado tiene que ser implementado al chocar con la cadena de Ebars, así que se necesita otro par de gliders para invertir los roles del podador y D1, y crear una estructura de datos que permita que algunos Ebar puedan atravesar.

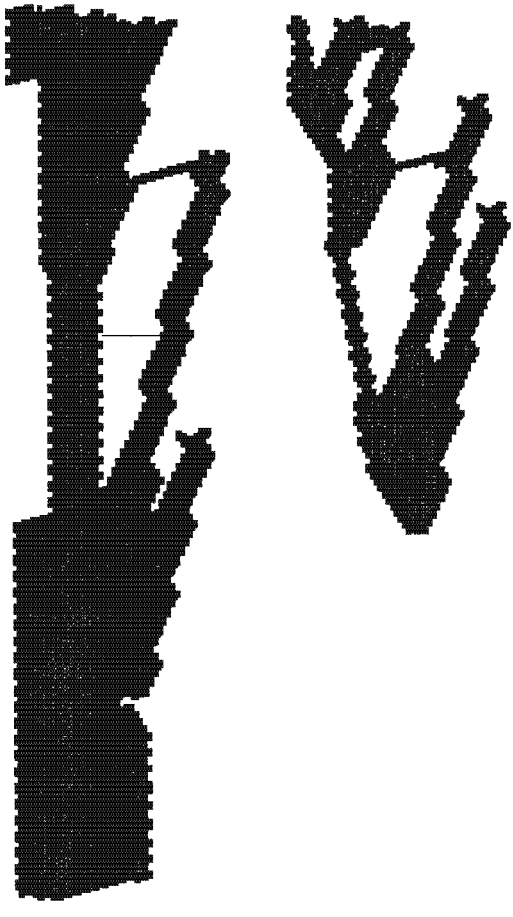

Figura 4.31: En la quinta etapa el predicado negro debe preparar un filtro, mientras que el predicado blanco debe comenzar a borrar Ebars. Estas acciones deben estar sincronizadas junto con la cadena de gliders Ebar.

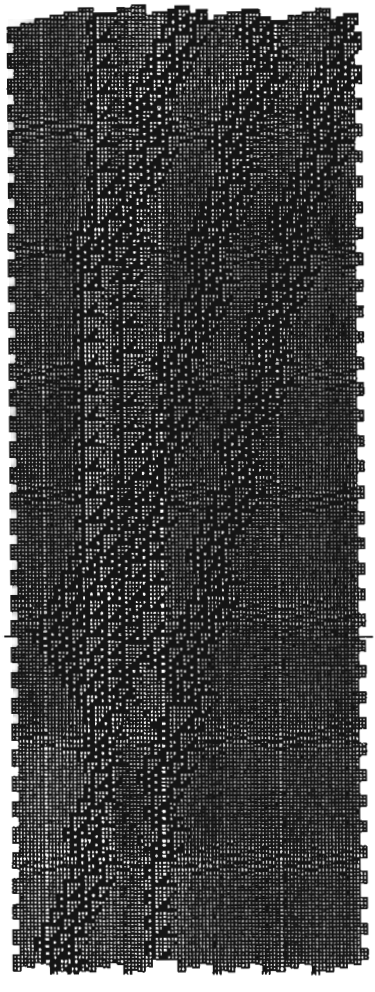

Figura 4.32: En el filtro, sólo un Ebar pasan a través de las interacciones de tres Ebar con un C3 y un par de C2's.

Un importante punto que se debe considerar es notar que el glider C3 no aparece en la interacción A-podador,D1, Ebar. Las combinaciones que son útiles para esta etapa del sistema de etiquetado cíclico son la dupla C1 y C1-C2 que aparecen cuando un E3 colisiona con un Ebar. El par alternado dupla C1, C3 interactúa con los Ebars que llegan desde la derecha, sin embargo, sólo uno de ellos tiene éxito al atravesar. Afortunadamente el proceso de borrado también usa el mismo espaciamiento que el proceso de filtrado, de tal forma que el desgaste de gliders puede ser compensado insertando más gliders dentro de la cadena.

El ciclo alternado dupla-C1,C3 debe empezar y terminar justo donde el contacto con el Ebar no sea separable. Cuando el predicado se ha realizado, se debe de parar el ciclo pues no se tiene que borrar todo. Para llevar a cabo esto se necesita un par E5-E2 que al colisionar quedan convertidos en una pareja Ebar-E1 para iniciar un nuevo predicado completando así el ciclo del sistema de etiquetado.

4.4.3. Comentarios

Las páginas anteriores mostraron la capacidad del autómata con regla 110 para construir configuraciones dinámicas y estáticas emergentes con propiedades muy diversas. Más que eso, se mostró que tales objetos adquieren funcionalidad que puede ser explotada para representar y transferir información, tal como ocurre con los glider C Y Ebar. Las configuraciones emergentes resultan ser atractores dinámicos que interactúan, algunos lo hacen de manera robusta, es decir, no importa la manera en que colisionen, el resultado será el mismo. Otros son más sensibles y su manejo debe hacerse de manera cuidadosa. Cualquiera que sea el tipo de interacción, vale la pena el análisis.

La prueba de que la regla 110 es universal demuestra que no es necesario contar con un sistema cuyas reglas subyacentes sean complicadas para que la universalidad emerja. Antes de que se conociera la capacidad de la regla 110 para emular otro sistema universal se conocían autómatas celulares con la misma propiedad cuyas reglas eran muy complicadas pues era necesario almacenar en ellas la información de los procesos que se querían emular. Por ejemplo, la segunda regla más sencilla para un autómata celular universal consiste en un autómata de primeros vecinos y siete estados cuyo objetivo consiste simular el comportamiento de cualquier otro autómata celular y desplegar espacialmente patrones cualitativamente iguales a los del autómata celular emulado. La prueba de que este mecanismo funciona es clara pues existe evidencia visual de que el autómata hace lo correcto. Sin embargo, la regla 110 no despliega patrones claros que hagan evidente su capacidad universal aunque computacionalmente el resultado sea equivalente al del autómata celular universal más complicado.

Una diferencia importante entre el anterior modelo y el juego de la vida es la dimensión espacial. En el juego de la vida el transporte de información se puede lograr evitando llegadas simultaneas, pues un espacio bidimensional permite trazar caminos entre dos puntos siempre y cuando estos no esten rodeados por obstáculos. En una dimensión no hay manera de evitar obstáculos, por lo tanto el flujo de información tiene que convertirse en una onda tipo solitón para poder atravesar el espacio perturbando estructuras sin causarles alguna modificación.

La construcción de un sistema de etiquedado cíclico usando las estructuras dinámicas y estáticas de la regla 110 es bastante complicada. Parte desde la

identificación de los gliders y sus colisiones hasta el armado mismo del modelo. Debido a esta riqueza dinámica no es difícil creer que exista una construcción que pueda simular a una máquina de Turing unidimensional, los elementos están ahí y sólo se requiere habilidad para construir.

En la naturaleza sucede algo semejante a lo que se hace con las configuraciones de la regla 110 y el juego de la vida. Las estructuras que se forman expontáneamente a partir de dinámicas en estado crítico son aprovechadas por la naturaleza, en palabras del doctor Germinal Cocho *La botica de la naturaleza no tiene de todo, y de lo que hay, si a la naturaleza le sirve lo toma....*

4.5. Cómputo universal en otros autómatas celulares

El potencial de los autómatas celulares para generar estructuras dinámicas y estáticas complejas que nos sirvan para construir modelos de computación depende solamente de la habilidad que se tenga para poder construir modelos. Han existido varios intentos por demostralo.

A mediados de los sesenta, Edgar Codd [7] propuso un sistema similar al de Von Neumann y mostró que este podría ser construido con ocho posibles estados por cada celda. Sin embargo, sus reglas no poseían alguna dirección preferencial, en vez de eso tenían una rotación preferencial. Sus reglas permitían al autómata enviar una señal simétrica desde una pistola vertical de tal manera que cuando la señal alcanzaba el fin de la pistola entonces aparecía una esquina en la parte derecha de esta, como si la pistola se doblara.

En 1970, Ricard Shoup [28] consideró el diseño de un circuito integrado para usarlo en espacios celulares sin tomar en cuenta la noción de universalidad, pues sólo estaba interesado en hacer eficiente la implementación práctica de las operaciones.

En 1961, Hennie y Moore estudiaron las limitaciones y capacidades de autómatas celulares sin que les importara alguna función en particular. Amoroso, Liblien y Yamada en 1969 expresaron la definición de un marco matemático para la formulación de preguntas concernientes a arreglos uniformes de máquinas de estados finitos.

Codd también mostró que dos estados son suficientes para universalidad si se permite que la vecindad crezca enormemente. En particular, mostró que un arreglo de dos estados con ochenta y cinco vecinos cada uno podría simular su máquina universal de ocho estados.

Wolfram construyó un autómata celular de 19 estados y vecindad de tamaño 20, que podía simular la dinámica de cualquier autómata celular unidimensional con dos estados y vecindad de tres celdas. Debido a la complejidad de tal autómata, este no se puede simular a sí mismo, pero si al correspondiente con regla 110, por lo tanto, la construcción del autómata de Wolfram daría paso una máquina con cómputo universal. En la figura 4.33 se muestra la simulación de la regla 30 y 90 respectivamente realizada por el autómata de Wolfram.

En 1990, Lindgren y Nordahl [15] construyeron un autómata celular universal con siete estados que emulaba la máquina universal de Turing de Minsky,

y demostraron que en general una regla con s + k + s estados podría implementar una máquina de Turing con s estados y k colores.

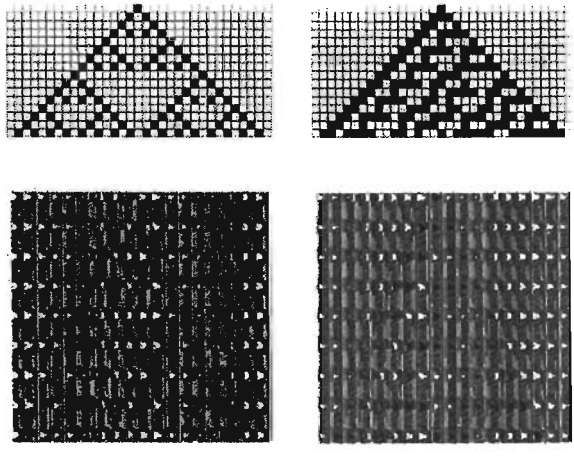

Figura 4.33: El autómata universal de Wolfram.

4.6. Autómatas celulares auto reproducibles

Más allá de la capacidad de procesar cualquier función computable, existen autómatas celulares con características de auto reproducción, concepto que denota la capacidad de alguna configuración para crear copias idénticas de si misma. Uno de los sistemas auto replicables más sencillos es la siguiente línea de un programa en C que cuando se ejecuta se imprime a sí mismo.

```
main() {char q=34,n=10,*a="main()
{char q=34,n=10,*a=%c%s%c; printf(a,q,a,q,n);}%c"; printf(a,q,a,q,n);}
```

Christopher Langton [14] estaba interesado en construir una máquina de propósito general cuyas propiedades fueran equivalentes a las de la vida en el

sentido de su capacidad de auto reproducción. El escribió *Es altamente inverosímil que las recientes moléculas autorreproducibles, de las cuales se supone todos los organismos vivos son derivados, tengan dotes de construcción universal, y no hubiéramos querido eliminar aquellas de la clase de verdaderas configuraciones autorreproducibles.*

Langton buscaba la más sencilla configuración existente que pudiera reproducirse a si misma. Von Neumann y Codd satisficieron este requerimiento mediante el modelo de autómata celular de dos formas: Primero trataron la información como instrucciones a ser interpretadas, tal como las instrucciones de una programa de computadora. Segundo, trataron a la información como datos sin interpretar, así como los números en una base de datos son copiados y almacenados en un registro dentro de la computadora.

Langton diseñó un conjunto dinámico al que llamó *loops* o ciclos, los cuales se colocan en un fondo de celdas con estado cero. Un loop de Langton se asemeja a un cuadrado con una pequeña cola en uno de sus vértices. El modelo está basado en las instrucciones que Codd simplificó del modelo complejo de Von Neumann. Los loops de Langton tienen tres capas parecidas a un cable plano. Las capas exteriores o envoltura son cadenas de celdas en estado 2. Estas actúan como celdas protectoras de las celdas principales, las cuales contienen la información necesaria para la reproducción. Por cada generación, las celdas en la capa interna rigen su dinámica a través de reglas que afectan el estado de sus vecinos y también propagan señales dentro de la cadena genética en el interior.

Langton quería hacer un arreglo de varios estados de tal manera que siguiendo una tabla de reglas especifica, la cola se convirtiera en un dispositivo

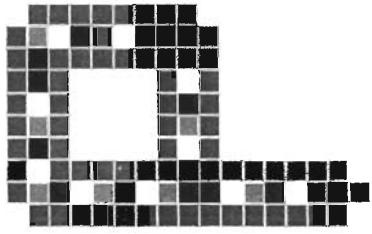

Figura 4.34: Loop de Langton.

de construcción. Esta debería empujarse hacia fuera hasta alcanzar cierto tamaño, girar por cada esquina y repetir el proceso hasta completar el cuadrado. Una vez que la forma exterior del nuevo loop se asemejara al loop padre, el flujo de la capa interior continuaría después. Tal fluido debería estar compuesto por los estados de las celdas portando datos. Cuando la información fuera completamente trasferida al loop descendiente, los dos loops debieran separarse. Momentos después, las señales cambiarian la configuración del núcleo del loop hijo de tal manera que este estuviera en las mimas condiciones que el loop original. Por lo tanto, el hijo sería una copia exacta del padre.

Este comportamiento reproductivo, según Langton, debiera estar ligado al de los organismos vivos. En la reproducción de loops, existe un genotipo que está compuesto por las celdas del núcleo conteniendo código genético que es se copia a la siguiente generación. También existe un fenotipo, que es la configuración codificada por las instrucciones que producen un nuevo organismo.

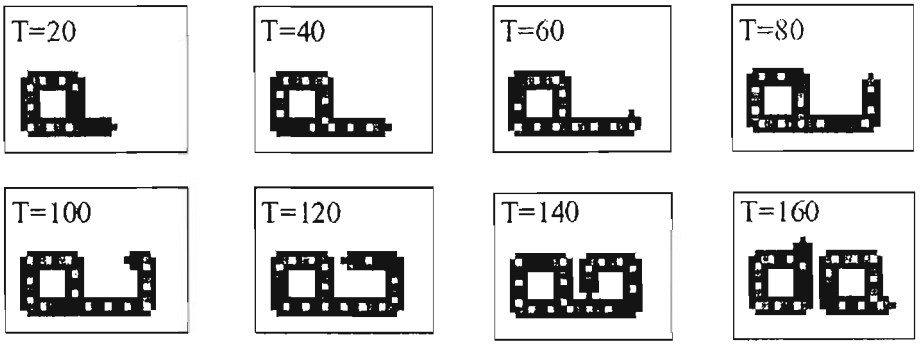

Figura 4.35: Auto replicación del loop de Langton.

4.7. El constructor universal de Von Neumann

John Von Neumann [22] pensaba que la reproducción en los organismos vivos era algo más que la simple transferencia y reimplementación de algún tipo de información. De tal forma pensó que una computadora podría simular lo mismo al transferir información hacia otra computadora en una nueva generación. El modelo que diseñó se basó en un autómata celular, y llego a la conclusión de que un sistema con auto reproducción debería tener la habilidades de cómputo universal y constructor universal. Una implementación con tales características puede ser encontrada en el trabajo de Pasavento [23].

Un constructor es un dispositivo capaz de reproducir cualquier configuración que pueda ser descrita con un número finito de elementos y cuyas instrucciones de construcción también puedan ser descritas de la misma manera. Un constructor universal es aquel con la habilidad de reproducir cualquier configuración incluyéndose a sí mismo, en tal caso, decimos que el constructor universal es auto reproducible.

La composición básica del autómata de Von Neumann consiste en:

1. Un constructor universal A que puede construir otro autómata de acuerdo a una cinta de instrucciones I.

2. Un copiador B que puede hacer una copia de la cinta de instrucciones I.

3. Un controlador C que combina A y B para que permita a A construir un nuevo autómata de acuerdo a I, a B copiar las instrucciones de I y juntarlos al nuevo autómata que lo separa del sistema A + B + C.

4. Un autómata D que consiste de A, B y C.

5. Una cinta de instrucciones I_D que describa la forma de construir a D.

6. Un autómata E que consiste de D + I_D.

Von Neumann y Codd trabajaron con los siguiente premisas:

1. Las máquinas celulares se configuran inicialmente en un renglón finito del espacio celular. Es decir, sólo un número finito de celdas con algún estado existen en la configuración inicial.

2. La construcción de nuevas configuración construidas por otras configuraciones se lleva a cabo dentro de una región del espacio inicialmente vacía.

El constructor universal de Von Neumann es una extensión del concepto lógico de máquina de cómputo universal. Pues al poder describir las instrucciones que se pueden aplicar sobre una cinta finita, también podemos implementar una máquina de Turing.

Von Neumann propuso una máquina realizada con base a un autómata celular bidimensional con dos elementos centrales: Una computadora universal y un constructor universal (Figura 4.36). La computadora universal contiene un programa que dirige el comportamiento del constructor universal. El constructor universal se usa para construir igualmente otra computadora universal y otro constructor universal. Una vez terminado esto, la nueva computadora universal se programa copiando el programa contenido en la computadora universal original y entonces la ejecución puede volver a empezar.

El constructor universal de Von Neumann que se muestra en la figura 4.36 consiste de un autómata celular con 29 estados y una vecindad de cinco celdas (de Von Neumann). El diseño de Von Neumann se extiende en un espacio celular bidimensional infinito de 200,000 celdas. Consta de un brazo que puede moverse y una extremidad que puede ser usado para cambiar el estado de la celda en la cual está puesto. Cuando el brazo se mueve progresivamente mientras el estado de la celda de la extremidad cambia, se crean configuraciones específicas en la región bidimensional del autómata celular.

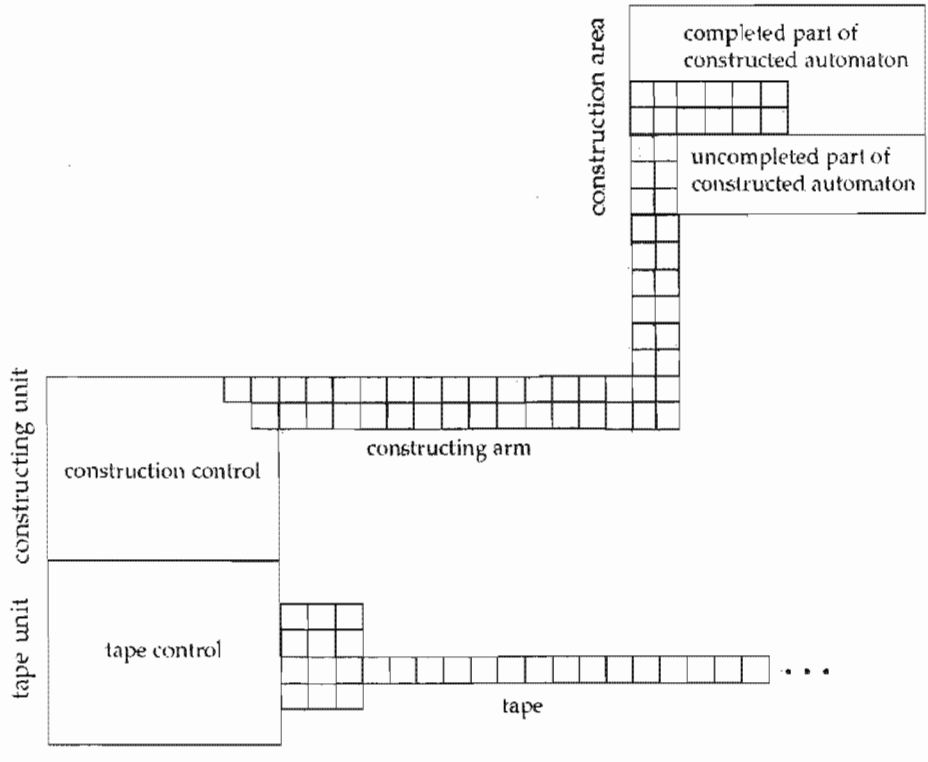

Figura 4.36: Constructor universal de Von Neumann.

El funcionamiento básico del constructor universal está basado en la noción de Turing para computación, en la que una máquina lee una cinta de memoria y ejecuta instrucciones que dependen de los estados y símbolos de la máquina. Sin embargo, la diferencia radica en que la implementación de Von Neumann posee la capacidad de construir ya que su cinta de instrucciones describe como hacer un nuevo autómata de cualquier tipo.

La discución del trabajo de Von Neuman nos lleva a otra. El constructor es un dispositivo que puede producir cualquier máquina arbitraria incluyendose a sí misma, esto implica que la complejidad de la máquina progenitora puede

ser menor que la complejidad de la máquina descendiente. El resultado es que existe un incremento de la complejidad que no ocurre en la experiencia cotidiana pues cuando se construye un mecanismo generalmente se usan artefactos de mayor complejidad. Von Neumann demostró que el camino contrario es posible, y amplió las posibilidades de entender la forma en que los organismos evolucionan si los consideramos como máquinas.

La construcción de dispositivos con mayor complejidad no se puede asociar a los modelos de auto reproducción que se puede encontrar en modelos sencillos como el de Burks [3] incluse Langton. Para Burks, la complejidad se puede medir con la habilidad para soportar cómputo universal. Ahora bien, Herman [11] mostró que el autómata de Burks mezclado con otro producia un autómata de poca complejidad y con las mismas capacidades que el constructor de Von Neumann. Ante esto, Herman rechazó el criterio de Burks para la noción de complejidad. Lo que abre la discución acerca de si la complejidad de un mecanismo es condición necesaria para que este pueda ser auto reproducible, universal y constructor.

Para Langton, el criterio de Burks no puede ser útil porque en los organismos vivos hay evidencias de auto reproducción no trivial (compleja). Si el criterio fuera la habilidad de hacer cómputo universal entonces se podría estar en un caso semejante al autómata de Herman, donde la auto reproducción es trivial (simple). Una alternativa al criterio de Burks se basa en identificar dos partes fundamentales en los organismos que se auto reproducen: genotipo y fenotipo. El genotipo es, segn Langon, la información almacenada estáticamente (o casi) en algún lugar y que puede ser transferida o copiada hacia los descendientes. El fenotipo es la parte dinámica del organismo, la que se

encarga de interpretar al genotipo, siendo la acción de simplemente copiar un caso particular. El constructor de Neuman exhibe ambas partes, mientras que el de Herman no.

4.8. Un autómata auto reproductor

En un espacio de dos dimensiones es posible construir más configuraciones que en un espacio unidimensional, además un autómata celular en tal espacio necesita la definición de más reglas de interacción puesto que el tamao de la vecindad de una celda se incrementa. Este aumento de funcionalidad nos permite construir cosas con mayor claridad que la permitida por un autómata simple.

En los capítulos anteriores se discutieron los trabajos de varios científicos que lograron estructurar máquinas universales basándose en mecanismos propios de autómatas celulares bidimensionales y uno unidimensional (regla 110). También se mostró brevemente el trabajo de Langton que propone un autómata celular capaz de auto reproducir una configuración inicial finamente detallada y cuya dinámica se asemeja al flujo y desarrollo de una cadena de genes de algún organismo. El mecanismo de Langton sugiere que la auto reproducción se puede llevar a cabo desde la dinámica interna de algun organismo con la ayuda de una cantidad de información que fluye al exterior y que a la larga evolucionará para formar otra configuración similar.

Pero la forma en que el loop de Langton se auto reproduce no es la única evidencia de este comportamiento para un espacio bidimensional o algún otro. El autómata elemental con regla 90 genera copias de celdas no vacias, donde

cada celda se copia a si misma siguiendo un patrón de Sierpinski. Cada copia de cada celda no se translapa con las demás debido a la propiedad aditiva de la regla. El resultado final consiste en que cualquier configuración inicial se va a copiar a sí misma siguiendo el patrón de Sierpinski, es decir, cada cierto intervalo de tiempo aparecen dos copias identicas de la configuración inicial, luego cuatro, después 2, 4, 8, 2, 4, 8, 2, 4, 8, 16, etc dependiendo del número de celdas no vacías que surjan en cada renglón conforme el triángulo crezca.

El mecanismo es poco claro con un autómata celular unidimensional, pero es posible extenderlo a dos dimensiones, la implementación es propuesta por el autor de este trabajo y consiste en aplicar por turnos la regla 90 a cada triada de celdas de cada uno de los cuatro lados de una vecindad de Moore (nueve celdas). Es decir, en el primer intervalo de tiempo se aplica la regla 90 a la parte superior de la vecindad de Moore y se coloca el resultado en la casilla central. En el segundo paso se aplica la misma regla a la parte derecha, luego a la parte inferior y finalmente a la parte izquierda. Esta evolución en cuatro tiempos se puede reducir a uno incrementando el número de estados a ocho y modificando las reglas de interacción. Hecho esto obtenemos un autómata celular bidimensional de ocho estados con la capacidad de auto reproducir cualquier configuración inicial.

Las siguientes imágenes muestran tres pantallas de un ejemplo de evolución del autómata celular propuesto pero con una variante: el número de colores de la configuración inicial es ocho y la regla 90 se aplica usando Xor con los bits de las celdas correspondientes.

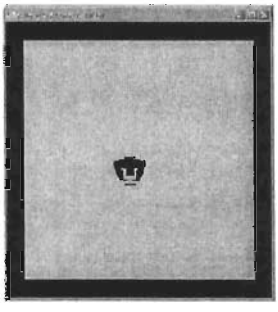

Figura 4.37: Estado inicial del autómata celular.

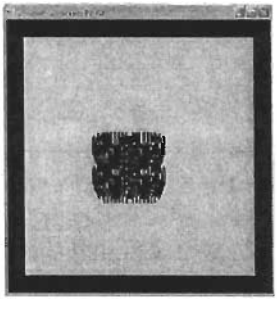

Figura 4.38: Estado intermedio del autómata celular.

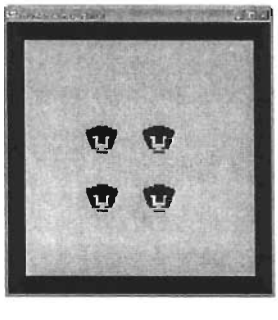

Figura 4.39: Estado intermedio del autómata celular despues de lograr replicar el estado inicial.

Capítulo 5

Comentarios y discusión

El principal objetivo de este trabajo es demostrar que algunos autómatas celulares pueden ser utilizados para construir modelos de computación universal. Durante el desarrollo de los temas que se incluyen también se dieron evidencias que nos ayudan a reafirmar la validez del bien conocido hecho de que la complejidad puede emerger a partir de leyes sencillas. Se ha mostrado el comportamiento espacial de algunos autómatas celulares, muchos de los cuales manifiestan estructuras muy complejas e impredecibles. Todos esos modelos han sido bien estudiados y se han descubierto propiedades interesantes que siguen siendo asombrosas. Sin embargo, el objetivo principal de este trabajo no es exclusivamente mostrar estructuras espaciales complejas, sino dar a conocer la existencia de emergencias de comportamiento que surgen a partir de emergencias dinámicas y espaciales.

Quizás uno de los fenómenos más conocidos que ejemplifican lo anterior es justamente la capacidad de ciertos autómatas celulares para soportar cómputo universal. Otro ejemplo es el potencial de otros autómatas celulares para auto reproducirse o construir otras configuraciones. En los siguientes párrafos

discutiré las conclusiones a las que se han llegado después de conocer muchas de las ideas que han llevado a varios científicos a afirmar la existencia de tales autómatas. Se empieza por esbozar brevemente la motivación inicial que se tuvo como preámbulo y se discutirán las conclusiones después.

Las razones que motivaron el desarrollo de este trabajo fueron varias. Una de ellas fue la curiosidad por saber en que parte de un organismo se localizan las instrucciones que lo hacen funcionar y también la manera en que esto se lleva a cabo. Por ejemplo, si alguna especie de planta se desarrolla de alguna manera particular, entonces las instrucciones necesarias para que tal desarrollo se lleve a cabo deberían de estar codificadas desde la semilla. Tales instrucciones deberían de transmitirse de alguna manera hacia todas las partes de la planta, cada una donde le corresponde para efectuar las funciones propias de los componentes. Y si las instrucciones finales son muy complejas, entonces estas debieran de haber evolucionado en el camino desde que salen de la semilla hasta que llegan a la hoja.

El estudio de los sistemas complejos fue útil para entender que muchas cosas en la naturaleza adquieren propiedades y comportamientos de manera emergente y que no es necesario que un organismo adquiera todas sus capacidades y estructura desde el principio de su desarrollo. Hay fenómenos de la naturaleza que se crean de manera aparentemente espontánea y sin necesidad de que existan instrucciones que indiquen al sistema como actuar de manera explícita. Pero esos fenómenos no se crean de la nada, existen leyes que sustentan su formación. Así pues, las partes que conforman un sistema pueden interactuar localmente y crear patrones con diferente complejidad que a la vez interactúan con otros para seguir creando estructuras de complejidad distinta

y que inicialmente no estaban determinadas por las interacciones locales iniciales. También se comprendió que los patrones emergentes pueden no dar lugar a comportamiento útil, son sólo el resultado de la intensa actividad que se lleva a cabo en niveles inferiores de magnitud y en muchos casos la interacción con el medio ambiente es también importante. Además, si queremos entender como se generan tales estructuras, debemos estudiarlas, ya sea con analogías o a partir de la busqueda de principios generales.

El curso de redes neuronales, biocomputación y otros impartidos por el doctor Pedro Miramontes dieron fuertes argumentos para convencerse que tanto una planta como todos los demás organismos vivos son sistemas complejos y que las funciones que estos llevan a cabo pueden aparecer de manera emergente, es decir, un organismo trabaja a base de emergencias de comportamiento. Una emergencia de este tipo se obtiene cuando un sistema puede obtener resultados equivalentes a los de alguna función sin necesidad de que un algoritmo se haya codificado explícitamente en las partes del sistema. Este concepto contrapone la idea de los algoritmos convencionales, aquellos donde existe un método previamente planeado para resolver un problema especifico o hacer alguna simulación. Aunque es compatible con el paradigma de los algoritmos genéticos, pues un *algoritmo emergente* necesitaría ser sometido a interacciones simples al igual que la selección natural, mutación o reproducción, pero este sería solo un caso particular. Quizás la evolución de las especies pueda ser mejor comprendida si entendemos como emergen propiedades de comportamiento y estructurales a partir de interacciones simples.

La variedad de emergencias estructuales y de comportamiento también debe estar incluida en el mecanismo de la vasta funcionalidad de los organis-

mos vivos. Quizás podríamos entender la manera como surge el complicado mecanismo de una célula, y como a partir de esta pueden surgir órganos y extremidades, cada uno con una función distinta. Todas estas partes a su vez forman otras más grandes, heredando funciones en cada nivel de organización y creando mecanismos de superior complejidad. También podríamos entender como han evolucionado los organismos a traves de tiempo y como ha surgido la cooperación y dependencia entre muchas especies. Los autómatas celulares y los sistemas complejos en general no dan las respuestas directamente, pero nos ayudan a entender el surgimiento de fenomenos complejos a partir de procesos sencillos y nos ofrecen nuevas formas de ver al mundo como un todo y no solo como la suma de varias partes.

Otro de los propósitos de esta tesis es hacer énfasis sobre una importante cualidad que figura en muchos procesos de la naturaleza: la universalidad. Este concepto establece que muchos fenómenos del universo tienden a exhibir comportamientos cualitativamente equivalentes debido a la similaridad en que sus componentes se relacionan y que sin embargo, son independientes del medio material que los constituye. A los procesos que comparten esta cualidad se les suele agrupar en una clase de universalidad dinámica y comparten una propiedad protegida de la materia.

5.0.1. Sobre la complejidad

Todos los sistemas descritos matemáticamente o aquellos que existen en la naturaleza exhiben cierto grado de complejidad. Actualmente no es posible asignar una medida al nivel de complejidad que cierto sistema posee pero en ciertos casos si es posible decidir si un sistema es cualitativamente más comple-

jo que otro. Como ejemplo, dada la condición inicial más simple, el autómata con regla 51 exhibe estructuras menos complejas que el autómata con regla 90 pues esta última genera triángulos organizados de varios tamaos y la primera solo alterna los estados de cada celda. En termino medio podríamos ubicar al comportamiento de la regla 109 que exhibe patrones periódicos que se dispersan a lo largo del espacio.

Una de las cualidades de los mecanismos con propiedad universal es que pueden emular cualquier otro mecanismo, y por lo tanto exhibir comportamientos de cualquier tipo de complejidad. Un mecanismo universal no solo puede ser clasificado como aquel que puede alcanzar el máximo grado de complejidad, si no que puede alcanzar cualquier nivel de complejidad que exista. En tal caso, la regla 110 del autómata celular elemental sería capaz de emular cualquier mecanismo complejo que no sea un autómata celular necesariamente y de ahí generalizar el resultado a todos los procesos de la naturaleza donde la complejidad es muy diversa.

Lo anterior implica que esencialmente toda la complejidad que logran los procesos de la naturaleza puede crearse también usando cualquier sistema universal o como un proceso computacional de sofisticación equivalente. Este principio fue enunciado por Wolfram y se conoce actualmente como Principio de Equivalencia Computacional. El principio implica que no existen diferentes niveles de sofisticación computacional, si no que solamente hay uno el cual comprende el mayor grado de sofisticación posible y que todos los procesos que no son obviamente simples son alcanzados con este. Este principio no ha sido enunciado formalmente, es decir, se defiende su validez usando palabras de nuestro lenguaje sin ninguna ecuación matemática.

El principio de equivalencia computacional da lugar, en términos generales, a que existe una intrínseca relación entre sistemas que exhiben complejidad alta y procesos computacionales. Entonces, casi cualquier sistema que exhiba complejidad no simple o sistemas en el borde del caos o en estado crítico deberían en principio ser capaces de efectuar computación universal. La sentencia de este principio tiene enormes consecuencias en la forma de ver al mundo. En primer lugar sugiere que todos los fenómenos del universo podrían tener subyacente a ellos alguna regla que sea la responsable de generar toda la complejidad asociada. Dicha regla tendría que ser sobre todo universal. Si el principio es válido entonces cualquier sistema complejo en estado crítico puede ser capaz de emular toda la complejidad del universo incluyendo auto reproducción, cooperación, adaptación, evolución, inteligencia, computación universal, etc. No todos estos sistemas exhiben tales propiedades, pero el potencial para hacerlo debería ser inherente a ellos.

Entonces, desde un punto de vista computacional, casi todos los fenómenos del universo son equivalentes y por lo tanto, potencialmente pueden adquirir el mismo nivel de complejidad. Dado esto, un sistema en estado crítico como una tormenta en turbulencia, una sociedad en revolución, el agua en ebullición, la vida, etc. tendría la capacidad de adquirir inteligencia por sí mismo o alguna otra propiedad asociada a la complejidad si existiesen las condiciones y el tiempo adecuados.

5.0.2. Sobre los componentes de un sistema y su comportamiento

Se ha puesto mucho énfasis durante los temas que se tocan en esta tesis acerca de la gran importancia que hay en la formación de estructuras o configuraciones. Sobre todo, de la necesidad de estas para almacenar y transmitir información. La noción de información no existiría si no fuese posible romper la homogeneidad de un medio para plasmar en el distorsiones que nosotros interpretamos como datos y que nos dan información. Una medida de la información contenida en cierta configuración es la entropía, la cual en términos generales cuantifica el grado de desorden de un sistema. Si un sistema está muy desordenado como un gas entonces no es posible construir configuraciones que sean capaces de soportar datos sea cual sea la interpretación que queramos darle al desorden. Por otro el extremo, un sistema totalmente ordenado es también incapaz de representar información.

Los autómatas celulares presentados en esta tesis y que tienen la capacidad de soportar computación universal no actúan a semejanza de un gas cuyas partículas se mueven aleatoriamente, tampoco generan estructuras homogéneas. El concepto más acertado para denotar su comportamiento es complejidad. Aquella que parece estar como el término medio entre el orden y el desorden. La complejidad es el ingrediente principal que se necesita para que un autómata celular pertenezca a la clase cuatro, según Wolfram. Es por eso, que su conjetura es muy intuitiva. Suena lógico pensar que para que un mecanismo tenga la posibilidad de desarrollar cómputo de propósito general es necesario que exista un buen manejo de la información, y los sistemas que se clasifican como complejos, o clase cuatro para autómatas celulares tienen la capacidad para ello. La parte más difícil de esta conjetura es demostrar que

todos los autómatas de clase cuatro o en régimen complejo adquieren también la capacidad de computación universal.

Ya se ha discutido en este trabajo que la propiedad universal de cómputo es una característica protegida de la materia, dicho eso, lo importante en un sistema para decir que este es computacionalmente equivalente a una máquina de Turing, es la forma en que sus elementos interactúan, es decir, la manera en que se crean las configuraciones y estructuras necesarias para soportar el proceso de cualquier función computable. Pero no podemos dejar de lado la importancia que tienen los estados internos elementales que nos permiten discernir entre estructuras. En el caso de los autómatas celulares, por ejemplo, representamos a tales estados como celdas con cierta localización espacial y cierto valor asociado, luego, mediante reglas que uno determina se actualizan los estados dentro del autómata. Sin embargo se puede precindir de estados elementales en algunos casos. Por ejemplo, en el juego de la vida existen gliders que podríamos reetiquetar para formar un nuevo conjunto de estados elementales y un mecanismo que se derive de las reglas del autómata celular del juego de la vida para formar un nuevo tipo de interacción. En tal caso, nos olvidaríamos de que un glider es en realidad un conjunto organizado de celdas y que su movimiento se rige por reglas de transición de un autómata celular. Habria nuevas reglas que emergen de las del autómata y que trabajan sobre elementos llamados gliders.

El nuevo mecanismo basado en el comportamiento de gliders sigue dependiendo tanto de estados elementales como de maneras de comportarse mediante reglas de interacción. De igual manera, un autómata celular se puede construir basándose en celdas que otro mecanismo más *elemental* construyó y

con reglas que el mismo mecanismo elemental transformó. Si seguimos este camino tendremos que llegar necesariamente al punto en que los componentes elementales no puedan dividirse más y por lo tanto, las leyes de interacción que trabajen a ese nivel también deben ser las más elementales. A partir de ahí podría construirse todo.

Tal pareciera que sigue existiendo una separación entre componentes de un sistema y reglas de evolución, pero estos conceptos están muy relacionados. Existe una dualidad entre los componentes de un sistema y su comportamiento. Los componentes de un sistema son sólo el medio en el que las leyes de interacción hacen su trabajo, hay una analogía entre las partículas de materia y las leyes físicas sobre ellas, siendo el comportamiento más elemental conocido hasta ahora el de la onda. Si existe una dualidad entre onda y materia, es considerable hacer la misma suposición para los sistemas dinámicos. Y aunque no sepamos cual sea la partícula más pequeña que existe, deberíamos conocer la manera en que estas cosas interactúan con sus similares para dar pie a la formación de más cosas.

En un sistema muy elemental, los componentes del sistema tal vez son sólo el medio en el que las leyes físicas trabajan. Por ejemplo, el espacio es un medio donde podemos aplicar leyes para curvearlo y podríamos crear curvas de muchos tipos, cada una con propiedades diferentes que son consecuencia directa de su forma y de las leyes que las rigen. En un autómata celular diríamos que el medio es el espacio infinito de celdas, cada una con alguna propiedad de un número finito de propiedades a las que llamamos estados. En niveles superiores de organización, las nuevas reglas que emergen no sólo dependen de las reglas en un nivel inferior, sino que también dependen de la estructura de los com-

ponentes que sustentan su funcionamiento, de igual manera, una estructura de componentes en un nivel de organización no sólo depende de su similar en un nivel anterior, si no que también depende de las leyes que funcionan un nivel antes. Por ejemplo, los glider de un autómata deben su forma espacial al tipo de celdas que lo compone y a las reglas de transición del autómata celular. De igual forma, un glider se mueve diagonalmente debido a las mismas razones.

Entonces, tanto reglas o leyes de evolución como los componentes o las partes que conforman un sistema son resultado de una intrincada relación que existe en niveles inferiores de organización. En niveles altos de organización una regla emergente carecería de sentido sin una estructura de componentes que la soporte y viceversa. La primera afirmación es clara, porque la definición de una regla, función, o ley es una declaración de como deben de comportase los objetos, sin ellos no tiene sentido definir una regla. La segunda afirmación también es cierta, pues una configuración de elementos es un objeto inútil sin la existencia de alguna interpretación que haga válida su existencia. Y la interpretación depende de la dinámica del objeto la cual depende de las reglas de comportamiento. En analogía con las partículas físicas, podríamos decir que una onda sólo tiene sentido si existe algún medio material por la cual propagarse, y una partícula tiene sentido si existe manera de tener conocimiento de ella, la manera en que esto se logra es observándola al conocer como interactúan con las demás partículas, por ejemplo con fotones de luz. Lo anterior no quiere decir que las partículas no existan sin observador, lo que se quiere dar a entender es que la importancia de las formas que una configuración de partículas pueda tener radica en la capacidad que estas tengan al interactuar.

El cómputo universal también tiene sentido cuando existe una interpreta-

ción de las estructuras que intervienen en el mecanismo. En tal caso, la interacción en una máquina universal debe ser la máxima posible. Todo el sistema se debe encontrar altamente correlacionado en el sentido que cualquier cambio efectuado por un componente del sistema, eventualmente afectara a todo lo demás. Esto maximiza también la capacidad de compartir y almacenar información pues un sistema altamente correlacionado puede mantener y disipar estructuras con relativa facilidad. Además, en un sistema con tales características existen estructuras emergentes de todos los tamaos de magnitud que sin duda dan origen también a comportamientos de todo tipo que eventualmente son útiles para construir maquinarias con funcionalidad con cualquier tipo de complejidad.

5.1. Trabajo a futuro

El estudio de objetos matemáticos dinámicos con propiedades constructoras, auto replicables y cómputo universal es relativamente reciente. Todos los modelos revisados tienene al menos algo en común, son sistemas dinámicos que transitan en estados lejos del equilibrio, lejos del orden, y lejos del desorden. Exhiben estructuras complejas que eventualmente se convierten en portadoras de información y constructoras de mecanismos. Todos esos modelos necesitan de condiciones iniciales muy específicas que generalmente son muy difíciles de definir.

Para una máquina universal, las condiciones iniciales describen tanto el programa a ejecutar como los datos iniciales para su ejecución. Hasta el momento no existe alguna máquina universal cuyas condiciones iniciales no tengan que

ser minuciosamente descritas por algún agente externo. En los modelos basados en la regla 110 y el juego de la vida, las condiciones iniciales consisten en gliders, pistolas de gliders, etc, que deben ser colocados con precisión pues si hubiese un pequeño error en su colocación, el resultado final sería completamente diferente al deseado y la dinámica carecería de sentido. Algunos gliders del autómata con regla 110 exhiben cierto grado de robustez, pues el resultado de su colisión es independiente del lugar donde esta ocurra, pero otros son altamente sensibles a la posición donde la colisión ocurre. Quizás existan autómatas celulares con gliders cuyas colisiones sean robustas o en otros términos, que las colisiones representen dinámicas con un único atractor.

En la naturaleza, los procesos computacionales son robustos. No es necesaria una colocación precisa de las piezas iniciales para echarlos a funcionar. El comportamiento caótico de los sistemas dinámicos restringe la construcción de modelos robustos, pero en estudios posteriores podríamos proponer esquemas de cómputo universal menos sensibles a condiciones iniciales y más robustos. Estos modelos también nos podrían servir como base para estudiar patrones de evolución que también exhiban características de computación sin la existencia de condiciones iniciales descritas con precisión.

Una propuesta para lograr lo anterior es definir condiciones de frontera sobre los estados de un autómata celular. Esto quiere decir, dividir el espacio de todas la posibles configuraciones del autómata celular (espacio fase) en zonas y considerar a estas como los estados verdaderos del sistema. Este ejercicio pareciera similar a la tarea de tomar promedios haciendo alusión a la termodinámica o física estadística, pero el método es más general, pues los criterios de zonificación del espacio fase pueden ser muy diversos e incluso dinámicos.

Por otra parte, es importante investigar la existencia de autómatas celulares universales que no basen su construcción única en la existencia de gliders. Existen otras formas de transmitir y almacenar información, una depende de la correlación que existe entre el estado de una celda y las celdas de su vecindad en tiempos anteriores. Esta correlación aumenta de tamaño al aumentar el tiempo. Sin embargo, el estudio de este tipo de máquinas debería ser mucho más complicado, pues una sola celda puede codificar parte de la información de datos muy distintas. Y datos muy distintos pueden compartir varias celdas al mismo tiempo. En un modelo basado en gliders no ocurre esto, en este caso, cada dato es representado por un conjunto bien definido de configuraciones dinámicas.

Una forma de tratar la transmisión de información en autómatas celulares es buscando analogías con sistemas distribuidos. En un sistema distribuido, la información se distribuye por todo el sistema y en particular, ningún componente codifica algo específico. Si acoplaramos un sistema distribuido con un sistema robusto como una red neuronal, entonces podríamos construir modelos distribuidos, robustos y factibles de computación universal.

Otra propuesta de trabajo para el futuro consiste en estudiar autómatas celulares continuos universales. La importancia de esto radica en explicar la posible transición de fase que da lugar al comportamiento universal mediante cambios continuos en los parámetros del autómata mediante análisis de bifurcaciones.

La hipótesis de que cualquier sistema dinámico en estado crítico posee ca-

pacidades de computación universal puede ser en principio demostrada para autómatas celulares, y quizás, obtener de esa demostración el camino que nos pueda conducir a la demostración general. Tal vez se puedan encontrar similitudes entre los gliders y alguna otra estructura dinámica de cualquier otro sistema dinámico complejo en estado crítico. Tal vez existan otras estructuras que puedan transportar información y que no son apreciables a simple vista. Es posible que al encontrar fluctuaciones de todos los tamaños que siguen una ley de potencias se pueda aprovechar esa diversidad de estructuras para construir con ellas lo que uno quiera aunque siempre acotado con las cosas inesperadas que pudieran surgir. Y quizás no sólo existan fluctuaciones espaciales, si no que también dinámicas en el sentido de que existan mecanismos de todos los tamaños con capacidades distintas para transformar cosas. Aún queda mucho camino por recorrer.

Bibliografía

[1] Avinash Dhar, Porus Lakdawala, Guantam Mandal, Spenta R. Wadia. Universal Cellular autómata and class 4. Tata Institute of Fundamental Research, India.

[2] Banks Edwin Roger. Information Processing and transmission in cellular autómata (Tesis doctoral). Massachusetts Institute of Technology, 1971.

[3] Von Neumann's Self-Reproducing Automata, Essays on Cellular Automata Burks, pp. 3-64. 1970.

[4] Casey Lee, Emergence and Universal Computation, 2003.

[5] Chapman, P. Life Universal Computer. http://www.igblan.com/ca/.

[6] Church, Alonzo, The Calculi of Lambda Conversion, en 'Annals of Mathematical Studies', No.6, Princeton University Press, Princeton N.J. 1941.

[7] Codd E. F., Cellular autómata, Academic Press, New York, 1968.

[8] Gacs, P. Reliable Cellular Automata with Self-Organization., J. Stat. Physics 103, 45-267, 2001.

[9] Gardner, M. Wheels, Life, and Other Mathematical Amusements. New York: W. H. Freeman, pp. 230 and 248-249, 1983.

[10] Herken Rolf, The Universal Turing Machine. A Half-Century Survey. Springer-Verlag Wien New York. Second Edition.

[11] Herman, G. T.,On Universal Computer-Constructors, Information Processing Letters 2, 61-64, 1973.

[12] Hickerson, Dean. Dean Hickerson's Game of life page. http://www.math.ucdavis.edu/ dean/life.html

[13] Klaus Sutner. Computing in Cellular Automata. Carnegie Mellon University. 2003.

[14] Langton, Christopher. 'Self-Reproduction in Cellular Automata', Physica 10D, 135-144, 1984.

[15] Lindgren, K. and Nordahl, M. G. Universal Computation in Simple One-Dimensional Cellular Automata. Complex Systems 4, 299-318, 1990.

[16] MacIntosh, Harold V., Rule 110 as it Relates to the Presence of Gliders. Instituto de Ciencias, Universidad Autónoma de Puebla, México, 2001.

[17] MacIntosh, Harold V., "Rule 110 is Universal!" A Gamin's Guide To Goldbergean Gadgeteering, 2002.

[18] Minsky, M.L. Computation: Finite and Infinite Machines, Prentice-Hall Series in Automatic Computation, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1967.

[19] Minsky, M. L. Recursive Unsolvability of Post's Problem of 'Tag' and Other Topics in Theory of Turing Machines. Ann. Math. 74, 437-455, 1961

[20] Mitchell Melanie, Computation in Cellular Automata: A Selected Review en NonStandar Computations. 1998.

[21] Mitchell Melanie, Is the Universe a Universal Computer?. Science Magazine, October 4, 2002.

[22] Von Neumann, John, Theory of Self Reproducing Automata, (edited by Arthur W. Burks), University of Illinois Press, 1966.

[23] U. Pasavento, An Implementation of Von Neumann's Self-Reproducing Machine, Artificial Life. 337-354, 1995.

[24] Post, E. L. Formal Reductions of the General Combinatorial Decision Problem. Amer. J. Math. 65, 197-215, 1943.

[25] Reggia A. James, Chou Hui-Hsien, Lhon Jason D. Cellular Automata Models of Self-Replicating Systems in Advances in Computers, M. Zelkowitz (ed). AcademiC Press. New York 1998.

[26] Rendell Paul. Turing Universality of the Game of Life. AA Book. 2001.

[27] Shepherdson, J. C. and Sturgis, H. E. Computability of Recursive Functions. J. Assoc. Comput. Mach. 10, 217-255, 1963.

[28] Richard G. Shoup. Programmable Cellular Logic Arrays. PhD thesis, Carnegie-Mellon University, Computer Science Department, March 1970.

[29] Smith, A. R. III. Simple Computation-Universal Cellular Spaces. J. Assoc. Comput. Mach. 18, 339-353, 1971.

[30] Wim Hordijk, Dynamics, Emergent Computation and Evolution in Cellular Automata, 1994.

[31] Wolfram Stephen, Cellular Automata as Simple Self-Organizing Systems, 1982.

[32] Wolfram Stephen, A New Kind of Science, 2002.

[33] Wolfram Stephen, Theory and Applications of Cellular Automata, World Scientific Publishing, 1986.

[34] Wolfram Research Page. http://mathworld.wolfram.com/

Apéndice A

Las siguientes imágenes fueron tomadas del trabajo de Rendell [26].

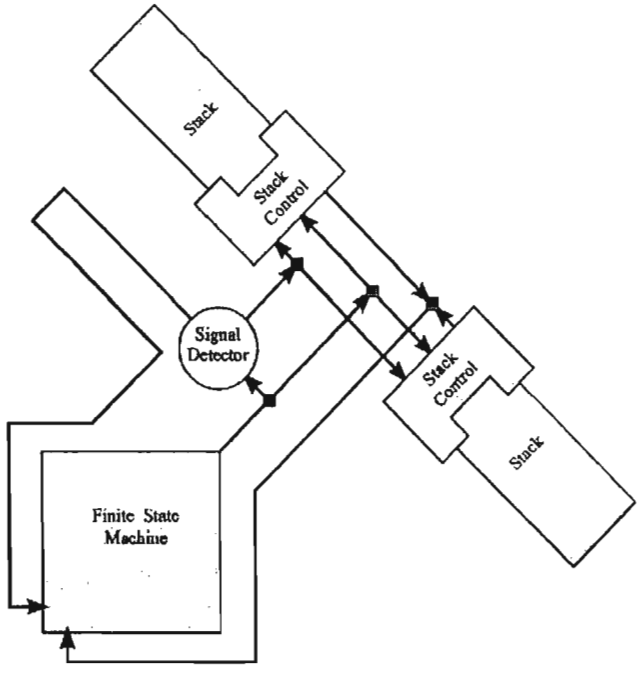

Figura A.1: Esquema de una máquina de Turing implementada en el autómata del juego de la vida.

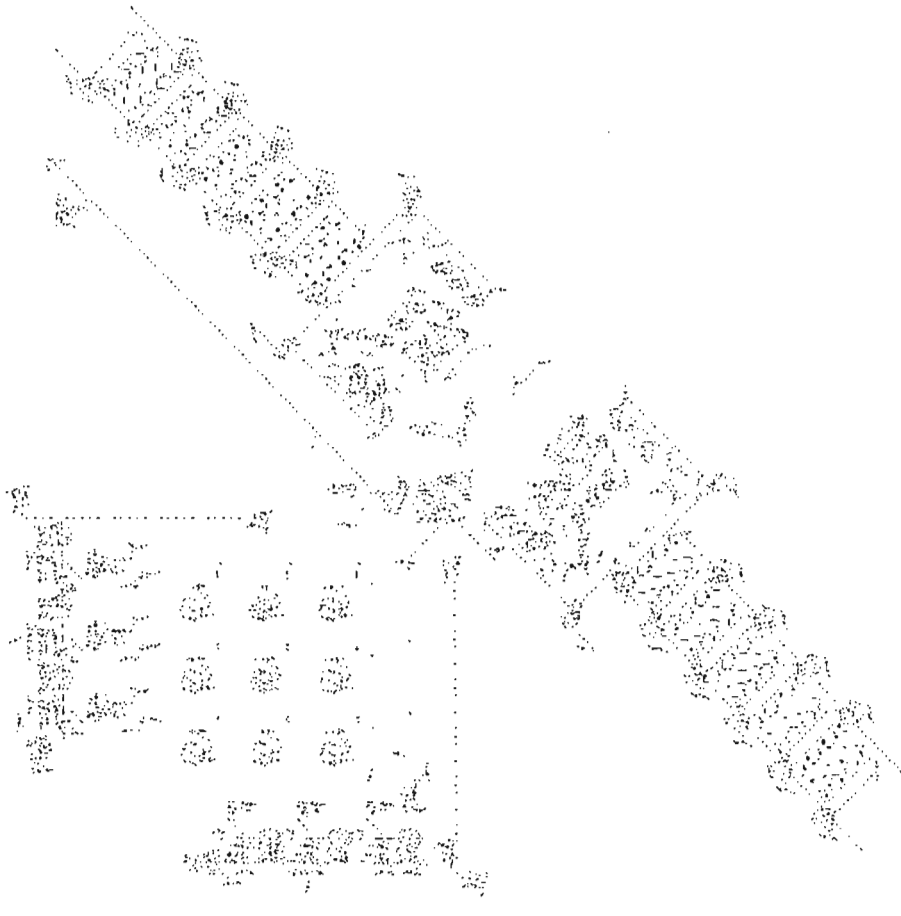

Figura A.2: Configuraciones de una máquina de Turing implementada en el autómata del juego de la vida.

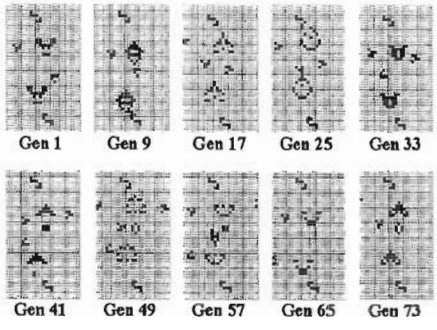

Figura A.3: El mecanismo de flujo que usa dos aguijones de abeja reina para reflejar cadenas de gliders. En el proceso, la señal de entrada es duplicada.

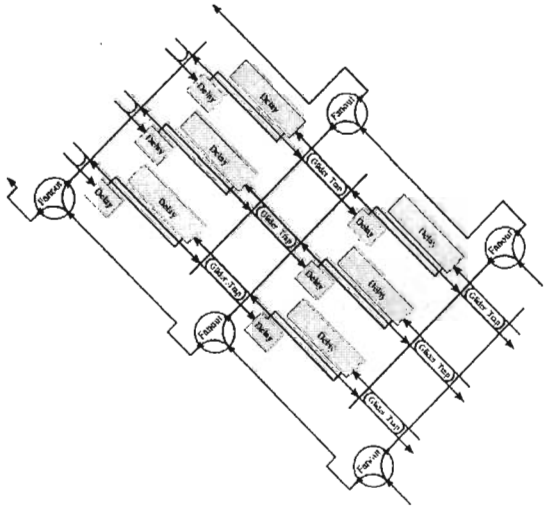

Figura A.4: Esquema de la celda del stack.

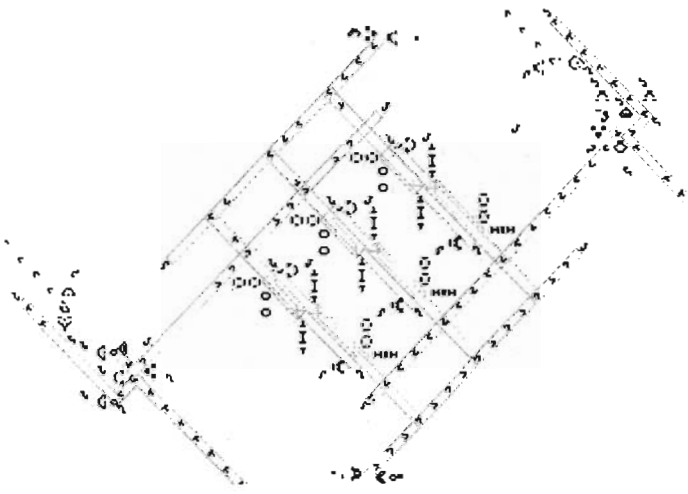

Figura A.5: Configuraciones de la celda del stack.

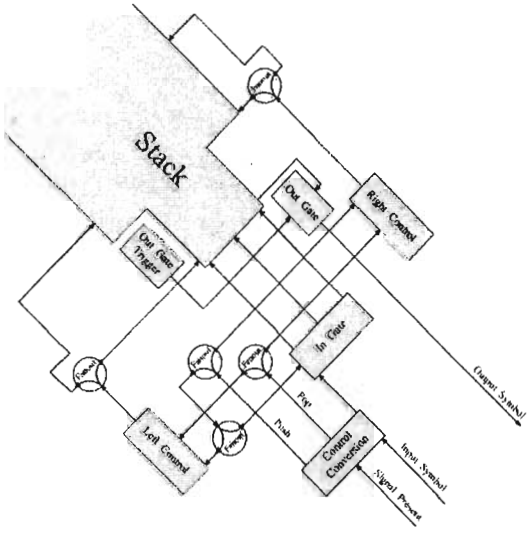

Figura A.6: Esquema del control del stack.

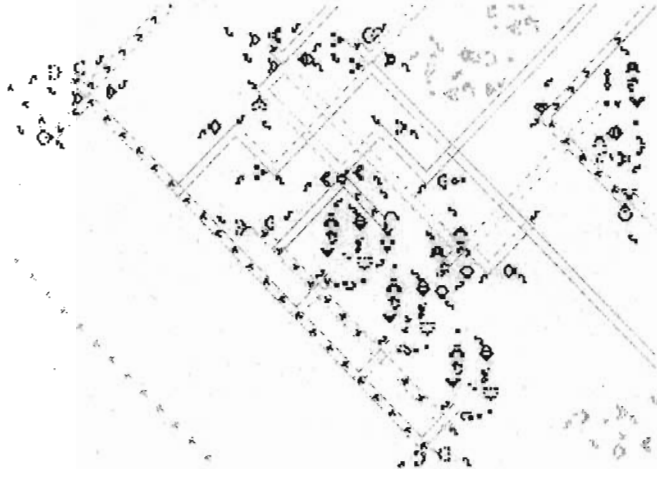

Figura A.7: Parte superior izquierda del control del stack.

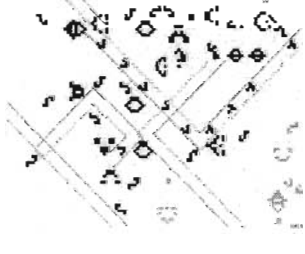

Figura A.8: Parte inferior del control del stack.

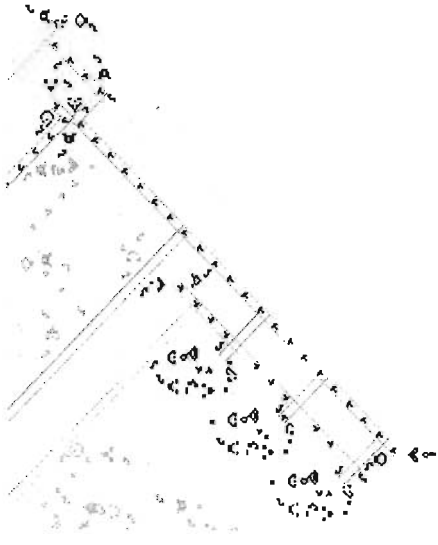

Figura A.9: Parte derecha del control del stack.

Figura A.10: Detector de señales.

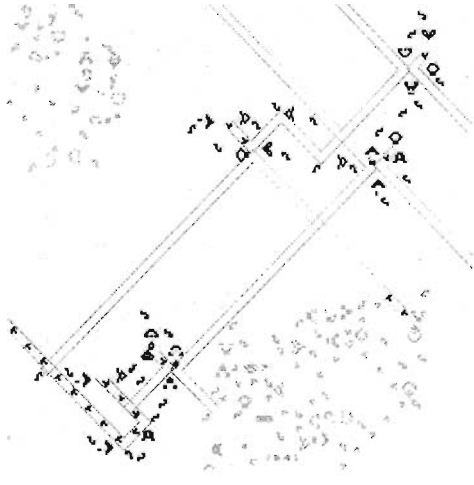

Figura A.11: Distribuidor de señales.

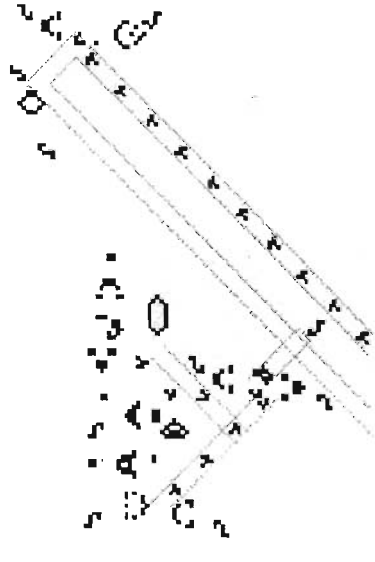

Figura A.12: Retraso del siguiente estado.

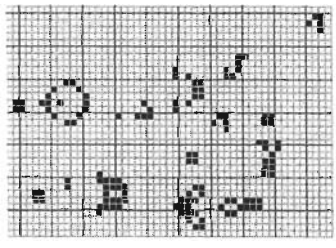

Figura A.13: Pistola de gliders de periodo 120.

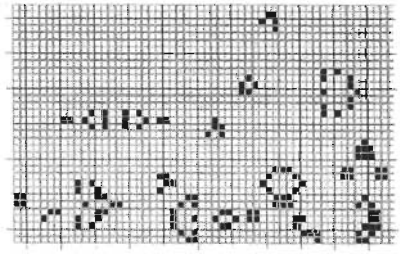

Figura A.14: Pistola de gliders de periodo 240.

Las siguientes figuras fueron tomadas de la página de Macintosh [17].

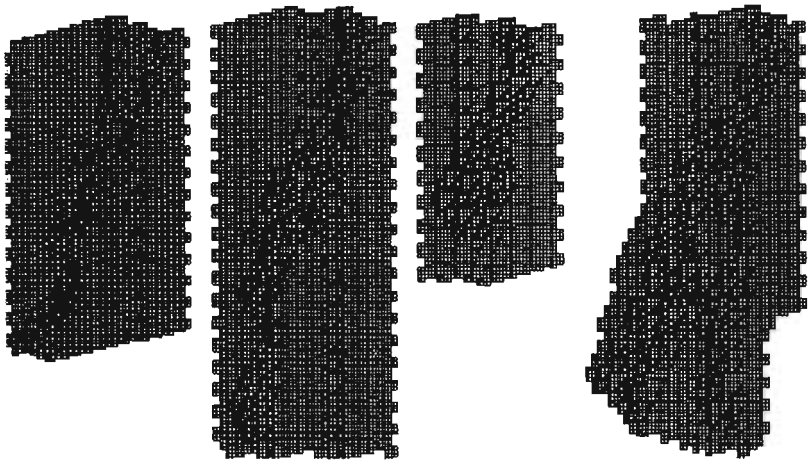

Figura A.15: Las cuatro colisiones C2-EBar. Una de ellas es solitonica. Otra genera un par C3 que eventualmente juega un papel importante en el filtrado de cadenas EBar.

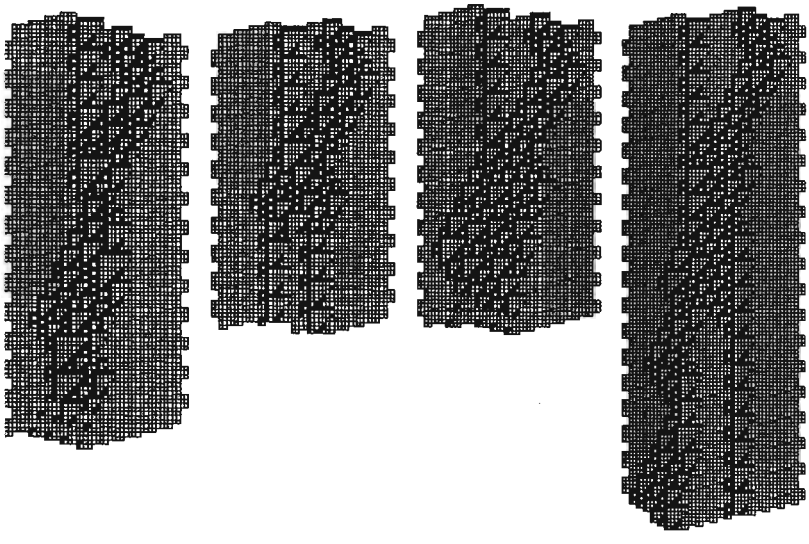

Figura A.16: Las cuatro colisiones C3-EBar. Una de ellas es solitonica y produce un par C1.

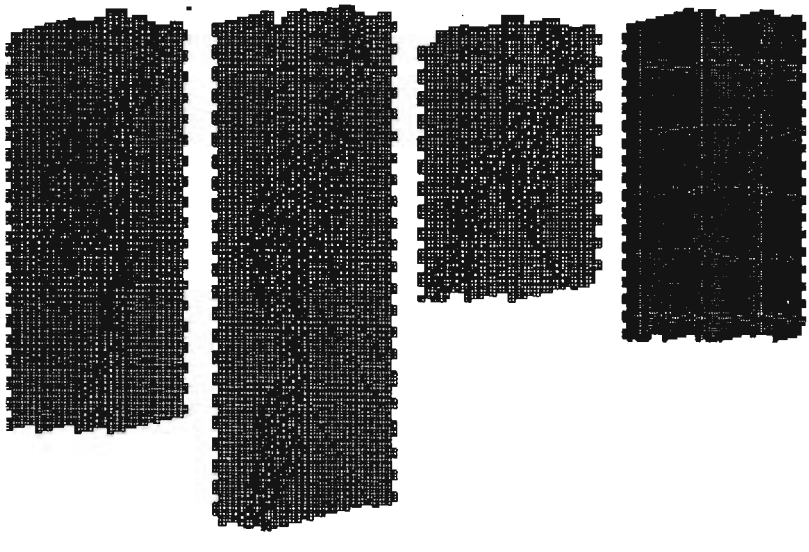

Figura A.17: Las cuatro colisiones C1-C1-EBar.