Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Unidad Zacatenco. Departamento de Ingeniería Eléctrica
Sección de Computación

"Máquina celular de computación basada en mosaicos regla 110"

Tesis que presenta:

Abdiel Emilio Cáceres González
para obtener el grado de Doctor en Ciencias
en la especialidad de Ingeniería Eléctrica
con opción en Computación.

Director de Tesis:
Dr. Sergio Victor Chapa Vergara

México D.F. 23 de Febrero de 2005

A mis padres, quienes me han enseñado cómo se obtiene
y de dónde viene la verdadera sabiduría.

A mis hermanos, con quienes comparto esas enseñanzas.

Centro de Investigación y de Estudios Avanzados - IPN,
Departamento de Ingeniería Eléctrica-Sección computación
Unidad México D.F.
Av. Instituto Politécnico Nacional 2508 esquina con Av. Ticomán
Delegación Gustavo A. Madero. C.P. 07350, México D.F., México.
Dirección electrónica: acaceres@computacion.cs.cinvestav.mx
URL: http://computacion.cs.cinvestav.mx/~acaceres

De manera especial agradezco al Dr. Harold V. McIntosh por su dirección y guía en mi formación académica, al Dr. Jaime Rangel Mondragón, al Dr. Sergio V. Chapa Vergara, al Dr. Francisco Rodríguez Henríquez, al Dr. José Oscar Olmedo Aguirre y al Dr. Germán González Santos por revisar este trabajo y darme sus puntos de vista; al Dr. Guillermo Benito Morales Luna por sus sugerencias y a la Sra. Sofia Reza por ayudarme en todos esos detalles administrativos estudiantiles; a todos mis amigos, entre ellos Nareli, José Manuel y Whats. También a las autoridades del CONACyT por el apoyo financiero y a las autoridades del CINVESTAV por el soporte de infraestructura y recursos académicos. Y agradezco a mis padres y hermanos que incondicionalmente me apoyan siempre.

Resumen. A mediados de la década de 1980, Steven Wolfram propuso que en las evoluciones del autómata celular lineal elemental regla 110, se podría obtener un comportamiento global equivalente a hacer computación universal. Esta conjetura se originó por la aparente interacción entre los patrones gráficos generados en las evoluciones de ese autómata celular, la conjetura fue demostrada recientemente por Matthew Cook, elaborando un intrincado y complejo sistema de choques de gliders, simulando de ese modo los datos de entrada, el intercambio de información y los resultados de los cálculos hechos. A partir de entonces se ha tomado la regla 110 como un objeto de estudio.

En esta tesis se toman los patrones emergentes en las evoluciones de la regla 110 para generar mosaicos, con los que es posible dar un sentido algorítmico al proceso de hacer cubrimientos del plano con estos mosaicos. Se descubre el origen de los patrones gráficos y se verifican las condiciones necesarias para que estos patrones gráficos sean llamados mosaicos.

Los mosaicos considerados ofrecen interesantes regularidades que permiten la definición de herramientas para manejarlos; se estudia entonces la relación de estos patrones gráficos con problemas como el "máximo número de besos" y el 18^{vo} problema de Hilbert. Estudiar estas regularidades será útil para posteriormente crear otro tipo de reglas.

Cambiamos el punto de vista a un manejo simbólico y definimos un alfabeto de símbolos. Con esos símbolos construimos palabras y aprovechamos las regularidades encontradas para crear reglas de agregación, lo que nos da el sentido algorítmico deseado. Aplicar estas reglas provoca que el cubrimiento cambie en cada paso de tiempo, definiendo un comportamiento dinámico llamado "evoluciones de Post". Hacia el final de la tesis se describe la construcción de las máquinas celulares de computación basada en mosaicos.

Palabras y frases claves. Autómatas celulares, Decibilidad, Computabilidad, Mosaicos, Espacios métricos Computación simbólica

Abstract. In the middle of the 80's, Steven Wolfram proposed that evolutions of elementary linear cellular automata rule 110, are sufficient to get an equivalent global behavior to perform universal computation. That conjecture is the result of the apparent interaction between triangles generated into evolutions of rule 110; the conjecture has been recently demonstrated by Matthew Cook, elaborating a complex system of gliders, simulating the exchange of information, data inputs and the computation output. Since this approach the rule 110 is a research field.

In this thesis, we take the patterns that emerge in the rule 110 evolutions in order to generate tiles, and to give an algorithmic sense to the process of make full or partial covers of the plane. We formalize the origin of the graphical patterns and we verified the necessary conditions to call "tiles" to that rule 110 triangles.

The considered tiles offer interesting regularities, that allow to define tools to handle these rule 110 figures, in order to study the relation among these graphical patterns with other kind of problems like "the kissing number" and 18th Hilbert's problem. To study these regularities it is useful to create aggregation rules.

We change to a symbolic viewpoint of rule 110 triangles. An alphabet of symbols is defined, with this alphabet we can make words and we took advantage of regularities to create aggregation rules, which give us the desired sense of algorithm. Applying these rules, the cover changes in each step of time, defining a dynamic behavior called "Post's evolutions". Towards the end of the thesis, we describe the construction of the cellular computation machines based on tiles.

Contenido

Índice de Figuras

Índice de Cuadros

Introducción

1. Introducción

La computación universal en autómatas celulares ha sido un campo de estudio que surge desde la década de los 60's con la concepción misma del modelo[66]. Sin embargo no fue sino hasta la década de 1970 cuando se quizo mostrar las condiciones necesarias para que un autómata celular fuera capaz de hacer computación universal [61, 38, 39, 56].

En su inicio, la teoría de autómatas celulares fue pensada con el propósito de modelar la auto-reproducción en algunos sistemas biológicos, von Neumann propuso un modelo matemático de autómata celular definido en un arreglo bidimensional de 200,000 unidades de cómputo interconectadas, capaces de estar en uno de 29 posibles estados[66]. Paralelamente surge el concepto de computación universal con autómatas celulares, puesto que el modelo se rige por las reglas de computación universal dadas anteriormente por personas como Alan M. Turing [63, 64] y Emil L. Post [41, 42, 43]. La complejidad que el modelo muestra es un mal necesario, propiciando entonces la búsqueda de condiciones mínimas necesarias para que un autómata celular sea capaz de hacer computación.

Este problema que fue iniciado y planteado por von Neumann y continuado por Burks[66], fue simplificado y extendido por Codd, Banks, Smith y otros, a fines de la década de 1960 y principios de la década de 1970[7].

En los siguientes años se dieron varias propuestas, algunos científicos e ingenieros del MIT[1] desarrollaron un modelo que efectivamente mostraba la auto-reproducción, pero sin hacer computación universal[27, 62].

Ahora nos encontramos en un punto donde se tienen varias interrogantes y se abren nuevos problemas, que pueden ser vistos desde diferentes perspectivas: dinámica simbólica y sistemas complejos; computabilidad y complejidad; y computación simbólica. Por consiguiente, se desprenden varios problemas fundamentales por atacar.

- Clasificación de autómatas celulares

[1] Massachusetts Institute of Technology

- Auto-reproducción y computación universal
- Comportamiento emergente

1.1. Regla 110 como tema de estudio. Hemos encontrado interesante que, en los autómatas celulares lineales elementales, que serán estudiados en el siguiente capítulo; y en el caso particular de la regla 110, las evoluciones de sus configuraciones globales cuando son extendidas en el tiempo, muestran patrones gráficos con una admirable regularidad como se muestra en el ejemplo de la figura 1.

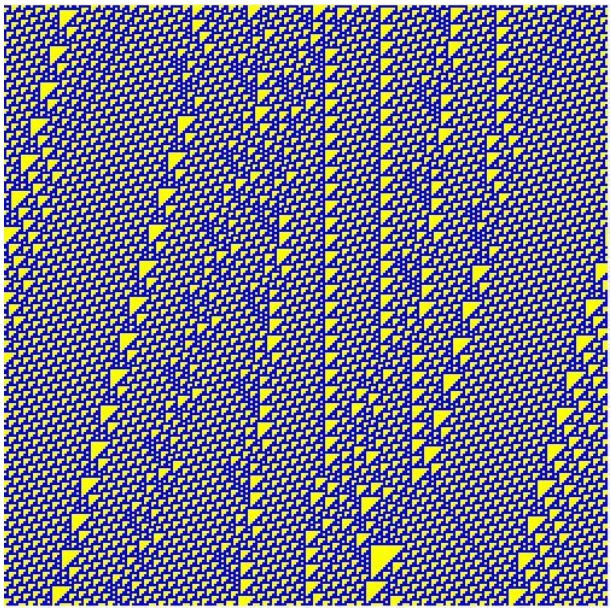

Figura 1. Evolución del autómata celular lineal regla 110 con 250 células en 250 generaciones.

Estas observaciones nos llevaron a suponer que los patrones gráficos generados, se podrían relacionar con el problema de correspondencia de Post; con el problema de cubrir el plano con mosaicos, por ende el saber si los patrones gráficos generados de manera emergente en las evoluciones de la regla 110 son capaces de cubrir el plano y describir algún tipo de computación y si esta clase de computación se puede relacionar con la computación convencional, es decir la que es basada en funciones Turing computables.

1.2. Objetivo fundamental.

(1) Mostrar que es posible definir procedimientos computacionales con los patrones gráficos generados por el autómata celular regla 110.

Para demostrar la capacidad de hacer computación, normalmente se sigue una de las siguientes dos direcciones:

(1) Considerar la función de transformación global como una máquina que toma como entrada una configuración global del autómata celular, y como salida, produce otra configuracion global que se considera como el resultado de la computación.
(2) Los patrones emergentes en las evoluciones del autómata celular interactúan, simulando alguno de los modelos conocidos que tienen la capacidad de hacer computación.

En nuestro trabajo, seguimos la segunda dirección. Consideramos los patrones gráficos regulares, que se generan de manera emergente en los diagramas espacio-temporales de las evoluciones del autómata celular lineal conocido como "regla 110", y entonces se hace referencia a la teoría de mosaicos [65, 40, 19, 47, 58].

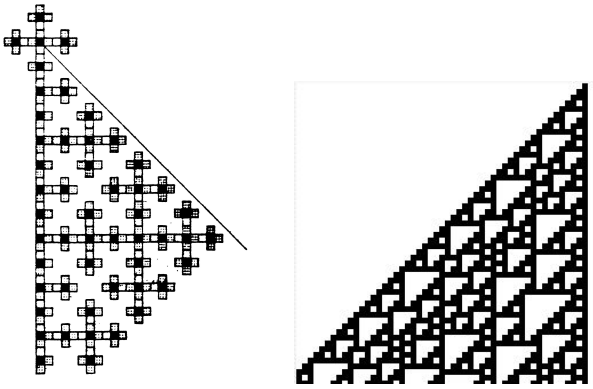

Figura 2. Izq.: Patrón de cruces "maltesas" creado por Ulam en 1962 [65];
Der.: Regla 110 con configuración inicial de una célula de estado 1

1.2.1. *Alcances de esta tesis.* Esta sección se ha incluído para describir desde un punto de vista global, los tópicos que se estudian y aquellos que hicieron falta por cubrir.

Se ha logrado dar un conjunto bien definido de símbolos, tomados de regiones de las evoluciones del autómata celular regla 110; se define una manera de manipularlos con la que se descubren nuevas regularidades, estas son los mosaicos propios, reticulados y la relación del determinante de un reticulado con el número de células en un mosaico propio; por otra parte, también se ha logrado dar un procedimiento para agregar mosaicos, que es legal desde el punto de vista de la regla 110 y proporciona un sentido algorítmico, el sentido algorítmico es iniciar un proceso de agregación de mosaicos con un mosaico como dato inicial, y terminar el proceso de agregación con un cubrimiento que se toma como el resultado del algoritmo.

Se da una métrica para que el resultado del algoritmo pueda ser comparado con los elementos de un conjunto de soluciones de manera que se pueda decir si el resultado del algoritmo es aceptado o rechazado. Se establece una relación de los cubrimientos producidos por algoritmos con las interacciones de los triángulos en la regla 110.

1.3. Cubrimientos con mosaicos. El problema general de mosaicos es saber si existe un cubrimiento del plano (normalmente euclideano) con mosaicos, y si existe, determinar las condiciones necesarias para que exista. Nosotros consideramos las figuras que se generan en las evoluciones del autómata celular lineal regla 110, y haremos un estudio para determinar en qué condiciones cubren el plano.

La idea de cubrir el plano con mosaicos no es nueva, basta mencionar los trabajos de Post con el problema de correspondencia[44, 46]; y el ejemplo de la conjetura de Wang con el problema del dominó [4].

Vamos a describir algunos temas relacionados que tienen que ver con la teoría de mosaicos, y que posteriormente tendrán relación con los triángulos que estudiaremos, estos temas sirven como antecedentes para estudiar el espacio celular extendido en el tiempo, formando un espacio de 2 dimensiones, que es donde surgen estos patrones gráficos regulares.

Este trabajo se fundamenta en tres enfoques de estudio principales, cuyos resultados nos dan las bases para describir la teoría que aquí presentamos.

(1) Cubrimientos con mosaicos
(2) El problema de la palabra
(3) El problema de computabilidad

Los puntos anteriores se describen con más detalle en las siguientes secciones.

Una manera de decir “cubrimiento del plano con mosaicos” es describir una partición de un espacio infinito ($\mathbb{Z}_n \times \mathbb{N}$ en nuestro caso) en subespacios de un número finito de piezas de diferente forma.

Los cubrimientos del plano pueden ser periodicos o aperiodicos, dependiendo de las reglas que se observen al cubrir ese espacio. Si existe una simetría, como es nuestro caso, entonces podremos formar una malla. También hay mucho interés en estudiar los cubrimientos aperiodicos con un conjunto finito de mosaicos.

Estos tipos de crubrimientos se han relacionado con ciertas estructuras cristalinas [60], y con varias ramas de las matemáticas como sistemas dinámicos, dinámica simbólica y otras.

El cubrimiento del espacio mediante mosaicos congruentes[2] es un problema importante, tratado desde diferentes perspectivas, entre ellas:

- El 18^{vo} Problema de Hilbert [24], el empaquetamiento de esferas y el máximo número de besos [8].
- El número de Heesch [22].
- El problema del dominó [4].

1.3.1. *El 18^{vo} Problema de Hilbert, el empaquetamiento de esferas y el máximo número de besos.* En 1900, David Hilbert proporcionó una lista de 23 problemas matemáticos en el Congreso Internacional de Matemáticos en París. En Particular, el 18^{vo} problema tiene relación con grupos cristalográficos, dominios fundamentales y el problema de empaquetamiento de esferas [57].

En 2000, un escritor de Nueva Zelanda[21] puso a prueba el 18^{vo} problema de Hilbert, mencionandolo a un vendedor de frutas y verduras. Una parte del problema era que una persona cualquiera pudiera entenderlo.

El escritor le dijo -¿Crees que exista una mejor manera de amontonar las naranjas que ponerlas en una pirámide? Después de algunos intentos quedó convencido [el vendedor] de que esa era la mejor manera, pero terminó diciendo: -Si, creo que es la mejor manera, pero ahora tengo un problema con las alcachofas.

El problema de las naranjas ya habia sido observado muchos años antes por Isaac Newton y David Gregory, al enunciar el problema del máximo número de besos [8], que consiste en determinar el número máximo de esferas de dimensión $d \in \mathbb{Z}^+$ que pueden tocar de manera simultánea a otra esfera colocada en el centro.

Como veremos más adelante, en el capítulo 3, los triángulos se pueden agrupar al rededor de otro triángulo, observando ciertas condiciones que tiene mucha relación con la manera en que son construidos estos triángulos.

El problema a resolver en esa parte será determinar el máximo número de triángulos que se pueden colocar al rededor de algún triángulo, conservando en todo momento la validez de la regla de evolución local del autómata celular regla 110.

1.3.2. *Número de Heesch.* El problema de Heesch es averiguar cuantas veces se puede rodear por completo una figura con copias de esa misma figura, de manera que si se puede

[2]Decimos que dos figuras son congruentes si una se puede transformar en la otra por medio de una isometría.

cubrir el plano con copias congruentes de esa figura, entonces se podrá cubrir un número infinito de veces, de otro modo el número de Heesch para esa figura debe ser finito.

Al construir el cubrimiento, se debe observar que los traslapes estan prohibidos, pero las rotaciones y traslados son permitidos. Los triángulos, cuadrados y hexágonos regulares pueden cubrir todo el plano, y por consiguiente tienen un número de Heesch infinito. Luego, todas las figuras que tienen un número de Heesch infinito, pueden cubrir el plano [14]

La manera de construir el cubrimiento es poner la primera corona a la figura. Una corona es el conjunto de figuras que rodean a un cubrimiento. La primera corona es el conjunto de figuras que comparten al menos un punto de la frontera en común con el centro del cubrimiento. La segunda corona es entonces, el conjunto de figuras que comparten al menos un punto de la frontera con las figuras de la primera corona, y así en adelante.

Nuestra relación con el problema de Heesch, es buscar una manera de construir figuras a partir de los triángulos emergentes en las evoluciones de la regla 110, de tal manera que esas figuras tengan un número de Heesch infinito.

1.3.3. *El problema del dominó.* Con las fichas del dominó se han hecho diferentes estudios matemáticos, como el que hizo Philip J. Davis[3], con varios conjuntos de diferentes dominoes. Algunas de los problemas que plantea son: El problema de la cadena simple y el problema de la cadena circular.

El problema de la cadena simple se resuelve al dar una secuencia de fichas que formen un camino de longitud igual a la cardinalidad del conjunto de fichas. El problema de la cadena circular es solamente una extensión del primero, en donde la última ficha debe tocar a la primera.

Sin embargo, nuestro interés es mayor en la Conjetura de Wang [67]. Esta conjetura fue enunciada en 1961 declarando que si un conjunto de mosaicos pueden cubrir el plano, entonces siempre se deben acomodar de manera periodica. El interés en este tema comenzó cuando en 1966 Robert Berger [4] refutó la conjetura, dando un conjunto de 20,426 mosaicos, posteriormente se redujo a 13 y el interés en esta área es por dar el conjunto mínimo de mosaicos que cubren el plano de manera aperiodica.

Nuestro trabajo se relaciona con la Conjetura de Wang (el problema del dominó), en que debemos encontrar una manera sistemática de obtener fichas de manera que cubran el plano de manera periodica, y luego proporcionar mecanismos de construcción para crear cubrimientos del plano de manera aperiodica.

[3]En el libro *The Mathematical Experience* Birkhauser Boston; Study ed edition (July 1, 1995); escrito junto con Reuben Hersh de la Universidad de Nuevo México.

El cubrimiento aperiodico lo estudiaremos desde el punto de vista de las **reglas de agregación**, que es un término nuevo que hemos creado, derivado de las reglas de sustitución en la teoría de algoritmos, esto porque el significado de la agregación es más preciso que el de la sustitución.

1.4. El problema de la palabra. Dada la regularidad de los patrones gráficos, es posible definir operaciones algebraicas, de manera que se puede definir un semigrupo, como se estudiará en los siguientes capítulos.

Uno de los temas importantes en el estudio de semigrupos es el problema de la palabra para un conjunto A de palabras. El problema de la palabra puede establecerse cuando tenemos un conjunto de ecuaciones que relacionan dos palabras del conjunto A, y dos palabras ω_1 y ω_2 que son elementos del mismo conjunto A de palabras.

El problema a resolver es determinar si al iniciar con ω_1 una serie de transformaciones de acuerdo al conjunto de ecuaciones, se puede obtener, al final de aplicar esas transformaciones la palabra ω_2.

Un segundo problema que es una extención del problema de la palabra es determinar si una palabra ω_3, ocurre en algún momento determinado de las aplicaciones de esas escuaciones, es decir, que ocurra como una palabra intermedia entre ω_1 y l apalabra final ω_2.

En esta tesis, vamos a considerar los triángulos como símbolos de un alfabeto, de manera que podremos formar palabras, en el mismo sentido que en la teoría de lenguajes formales. Así, una palabra es una secuencia de símbolos, pero también van a representar un cubrimiento del plano. Este estudio nos permitirá definir criterios para determinar la validez de un cubrimiento obtenido, mediante la aplicación de las reglas gramaticales definidas.

Vamos a ver qué condiciones son necesarias para determinar si es posible que se pueda dar una respuesta afirmativa al problema de la palabra orientada a los mosaicos que son generados por la evoluciones del autómata celular regla 110.

1.5. El problema de computabilidad. Dentro de este campo de estudio, dedicaremos un espacio para estudiar el problema de la detención del proceso constructivo (*halting problem*).

El problema de la detención lo podemos enunciar considerando nuestro enfoque de los mosaicos del siguiente modo: Dados un algoritmo basado en mosaicos $\mathcal{A}$ y un cubrimiento

p como dato de entrada, ¿podremos asegurar que el proceso constructivo de cubrimientos se va a terminar?

Figura 3. La Máquina Universal. Autor: Jin Wicked *Imagen usada con permiso del autor.* `http://www.jinwicked.com/`

En este documento vamos a establecer las condiciones necesarias para asegurar que se podrá terminar el proceso. Se verá que a pesar de que se cumplan esas condiciones, pueden ocurrir otros elementos de decisión que hacen que este problema no pueda asegurar llegar a una solución satisfactoria al final de la ejecución del algoritmo $\mathcal{A}(p)$.

2. Contenido general

El contenido de esta tesis está dividido en tres bloques. El primero de ellos es el capítulo 1 que es introductorio; el segundo bloque lo constituyen los capítulos 2 y 3, en donde se estudian las propiedades algebraicas de los patrones estudiados, determinando así las características sintácticas de las construcciones simbólicas; y el tercer bloque lo integran los capítulos 4 y 5, que formalizan la idea de construcción de subcubrimientos basados en la agregación de mosaicos, dando así un sentido algorítmico que involucra el concepto de computación.

- En el capítulo 1 daremos una definición de los autómatas celulares, que de manera especial mostratemos con ejemplos el caso de la regla 110, que es precisamente la regla de la cual obtenemos estos patrones gráficos que estudiaremos más adelante.

Como vamos a definir un nuevo tipo de computación, estudiaremos tres modelos de autómatas celulares que se han usado para demostrar que es posible hacer computación (y computación universal) empleando autómatas celulares. Estos modelos son característicos porque siguen uno de dos enfoques:

(1) La capacidad de hacer computación universal surge como consecuencia natural del modelo, pues fueron creados precisamente para hacer computación universal.

(2) La capacidad de hacer computación universal surge como un comportamiento inesperado que es observable en cantidades macroscópicas de los elementos que interactúan en el sistema espacio-temporal.

- En el capítulo 2 veremos que la regla 110 no es la única que genera ese tipo de figuras, sino que pertenece a un grupo de 4 reglas. De manera que tomaremos la regla 110 como representante de ese grupo.

  Estudiaremos la manera de crear triángulos de la regla 110, la relación entre la regla de evolución local y cada parte del triágulo con que directamente se relaciona. Daremos una descripción formal de los triángulos.

  Tomando el conjunto de las figuras que son tomadas de las evoluciones de la regla 110, vamos a crear una operación, la operación de concatenación gráfica. Entonces el conjunto de triángulos junto con la operación de concatenación gráfica forman un semigrupo. También veremos algunas restricciones que hacen que no cualquier concatenación sea considerada como válida, formando así el conjunto de los cubrimientos prohibidos.

- En el capítulo 3 nos daremos cuenta que las figuras que el autómata celular regla 110 son mosaicos, porque cumplen las condiciones que han sido dadas desde el punto de vista topológico.

  Ya que tenemos mosaicos y una regla de concatenación gráfica, vamos a definir una manera de moverlos en el espacio, definiendo así espacios de traslados, generando mallas. Aquí es donde veremos los mosaicos propios, que son los que hacen cubrimientos completos periódicos del espacio.

Con este tipo de cubrimientos completos hay cosas interesantes por descubrir, porque cubren el espacio haciendo crecer la zona cubierta de una manera similar a la manera que lo hacen algunos cristales.

- En el capítulo 4 La existencia de un semigrupo nos dará la oportunidad de estudiar el problema de la palabra [**10**, **11**, **12**] considerando luego, un conjunto de reglas de transformación. A estas reglas de transformación las llamaremos "reglas de agregación", que será una manera diferente de hacer cubrimientos del plano.

  Otros tópicos importantes que se verán en este capítulo son una manera de comparar cubrimientos, de hecho, daremos dos criterios, uno para decidir que los cubrimientos considerados son suficientemente parecidos y el otro criterio nos dará la igualdad de ellos; y daremos la definición de las "evoluciones de Post" que dan la dinámica en el uso de las reglas de agregación.

- En el capítulo 5 Definimos un algoritmo tomando un subconjunto de las reglas de agregación que definen un esquema (capítulo 4) y ordenando los elementos de ese conjunto. De manera que veremos una máquina de computación teórica basada en mosaicos, definida por un algoritmo determinado, que toma como dato de entrada un cubrimiento y calcula otro cubrimiento. El resultado de esta máquina basada en mosaicos es quizás un nuevo cubrimiento que se compone por los triángulos definidos en nuestro alfabeto.

  Con los algoritmos se pueden definir algunas operaciones de conjuntos, en particular con la unión de algoritmos se busca crear un algoritmo que sea capaz de calcular todos los cubrimientos de ambos algoritmos originales. Otras operaciones son la contención y la intersección.

  Estudiaremos cómo funciona la máquina celular de computación basada en mosaicos, y las condiciones que son necesarias para que el proceso computacional termine.

Capítulo 1

Autómatas celulares y computabilidad

1. Resumen

En este capítulo nos vamos a enfocar al estudio de los autómatas celulares y su relación con la computación universal. Se inicia con una introducción que describe de manera muy general tres modelos que relacionan autómatas celulares con universalidad desde tres enfoques diferentes.

Después se describe de manera general cómo funciona cada uno de esos modelos, iniciando con el modelo original de von Neumann, luego con el modelo de Alvi Ray Smith III y finalmente el modelo *life* de Conway.

El objetivo de estudiar estas aproximaciones es hacer notar que con los autómatas celulares, cuando son manipulados de manera especial, se puede dar una interpretación de un proceso algorítmico, aunque como veremos en los siguientes capítulos, nuestra aproximación de proceso algorítmico es diferente.

2. Introducción

Las tres perspectivas mencionadas son en primer lugar el modelo original de von Neumann , a mediados de la década de 1960 [66], que fue creado para hacer computación universal al modelar una máquina de Turing.

En segundo lugar el modelo de Alvi Ray Smith III a principios de la década de 1970 [49, 50] quien se basa en las funciones parciales recursivas y de esa manera crea su modelo.

Finalmente en tercer lugar, el modelo que originalmente describió Conway también en 1970, pero demostrado por Buckingham hacia 1978 [5], este es el modelo que es más sorprendente, porque no era el propósito original que tuviera capacidad de simular una máquina de Turing, ni simular la actividad de las compuertas lógicas fundamentales, sin embargo, mediante la interacción de las céluas a través del tiempo, se ha podido demostrar que es posible dar una configuración inicial del modelo *life* que hace computación universal.

Posteriormente daremos una definición de autómatas celulares y describiremos cómo se obtienen los triángulos, que en el resto de la tesis serán de suma importancia.

3. Definiciones generales de la teoría de autómatas celulares

Los autómatas celulares son sistemas dinámicos discretos que involucran cuatro elementos de construcción:

(1) El espacio celular
(2) Los estados de las celulas
(3) La configuración de vecindades
(4) La regla de evolución local

El espacio celular: Los autómatas celulares están definidos por un arreglo de células de dimensión d, llamado espacio celular; donde cada uno de los elementos del arreglo se denomina *célula*, que es un término tomado de las ciencias biológicas como algunos otros tales como "evolución", "vida", "muerte"; que son adecuados para comprender el comportamiento individual de los elementos.

El espacio celular, en principio y tal como fue definido originalmente, es infinito en todas sus direcciones, y posteriormente debido principalmente a las limitaciones prácticas, el espacio celular puede estar acotado.

Para poder simular el espacio celular infinito, se ha decidido tomar condiciones periodicas de frontera, esto es, que la célula siguiente a la última es la primera, y la célula anterior a la primera es la última (ver la figura 5 en la página 19). Cuando $d = 1$ el espacio celular es un anillo, en $d = 2$ se forma un espacio toroidal.

El conjunto de estados de las células: El estado que cada célula tendrá en el siguiente paso del tiempo está determinado por el estado actual de ella misma y de las células vecinas más cercanas en un radio de vecindad predeterminado.

La notación usada en este trabajo para identificar cada uno de los autómatas celulares, es la misma que ha sido definida en [33] y [68]. Un autómata celular lineal está definido por el par (k, r), donde k es el número de estados para cada célula y r es el radio de vecindad. Los autómatas celulares que usaremos en este trabajo son los acl(2,1) que significa: "autómata celular lineal de dos estados y radio de vecindad de uno". Un ejemplo de ello se muestra en la figura 4.

La configuración de vecindades: La manera en que las células actúan dentro del espacio celular, está determinada por cómo son afectadas por el entorno que las rodea. A este entorno se le llama vecindad.

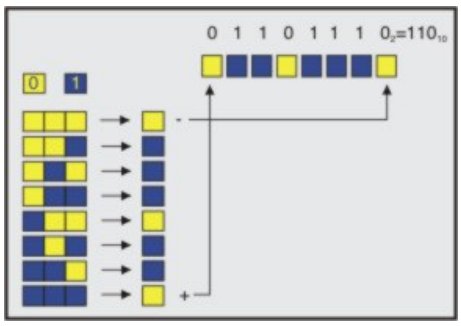

Figura 4. Configuración de vecindades en un acl(2,1)

Así, la vecindad de una célula, llamada célula en transición, o célula central x_c, es el conjunto de células cercanas a ella a una distancia $d(x_c, x_i) \leq r$, además de la propia célula central. En el caso que estudiamos ahora, la vecindad es formada por el conjunto $V = \{x_i | d(x_c, x_i) = 1\} \cup \{x_c\}$.

La regla de evolución local: Cada célula del espacio celular, toma su valor en dependencia de la configuración de su vecindad. Si K es el conjunto de estados, con $|K| \neq \emptyset$, la regla de evolución local f_l está definida como:

$$f_l : K^{2r+1} \to K$$

Donde $2r + 1$ es el tamaño de la vecindad definida para el autómata celular lineal, en nuestro caso, tomaremos un radio de vecindad de $r = 1$.

0	$000 \to$	α_0
1	$001 \to$	α_1
2	$010 \to$	α_2
3	$011 \to$	α_3
4	$100 \to$	α_4
5	$101 \to$	α_5
6	$110 \to$	α_6
7	$111 \to$	α_7

Cuadro 1. Con cada $\alpha_i \in \{0, 1\}$. $k^{2^{2r+1}}$ es el número total de reglas. $256 = 2^{2^3}$ reglas de evolución en los acl(2,1)

La regla de evolución local queda completamente determinada cuando se han nombrado de manera explícita o implícita, todas las posibles configuraciones de las vecindades y su asignación dentro del conjunto de estados K, como se muestra en

la tabla 1. Dado que los autómatas celulares lineales (2,1) son solamente 255 en la notación decimal, en [68] se ha establecido una manera de distinguir todas las posibles reglas de evolución con un número entero entre 0 y 254. Esta manera de señalar las reglas de evolución es particular para esta clase de autómata celular, para otros tipos se han ideado otras maneras de clasificar las reglas de evolución, porque no hay una manera estándar de nombrar todas las reglas de evolución de todos los autómatas celulares.

El problema de nombrar las reglas de evolución, radica en el número exponencial de posibles reglas, así para los acl(2,1) hay $2^3 = 8$ posibles configuraciones para las vecindades y $2^8 = 256$ posibles reglas de evolución. Pero este es uno de los casos más sencillos, cuando aumentamos el número de estados, en un acl(3,1) habrá entonces $3^3 = 27$ configuraciones diferentes para las vecindades, y $3^{27} = 7.625597485E + 12$ posibles reglas de evolución. Y si en lugar de aumentar el número de estados, aumentamos el tamaño de la vecindad, entonces el número de posibles reglas es $2^5 = 32$ posibilidades para las vecindades y $2^{32} = 4,294,967,296$ reglas diferentes.

Cuando el número de estados es grande, o el tamaño de la vecindad lo es, se ha optado por catalogar la regla de evolución, nombrando el estado que resulta de cada una de las vecindades en un orden establecido.

Además, se requiere que el espacio celular esté definido con los valores de las células en el paso de tiempo inicial. En general pueden haber dos maneras de dar estos valores:

Arbitraria: En este tipo de configuración global, el valor de cada célula es puesto deliberadamente; un ejemplo común, es poner un valor igual para todas las células excepto una de ellas, normalmente la que está ubicada al centro del espacio celular.

Aleatoria: En este otro tipo de configuración global se requiere el uso de un generador de números aleatorios, que es usado de manera auxiliar para determinar el valor del estado de cada célula. La distribución de los números aleatorios usualmente es uniforme, sin embargo es posible usar cualquier otra.

El manejo del tiempo en la evolución de los autómatas celulares es muy simple, cada paso en el tiempo es contado progresivamente con números naturales, reservando el cero para la configuración global inicial, como se aprecia en la figura 5.

La función de evolución global: El estado global del autómata celular está determinado por el estado de cada una de las células que lo componen. De manera que es posible dar una descripción de un estado global mediante una tupla de n números enteros, en donde n es el tamaño del espacio celular, y cada número es el índice de

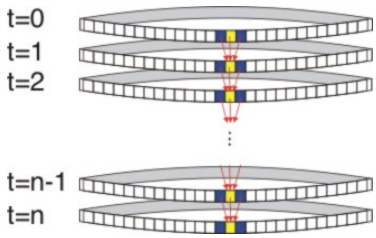

Figura 5. La función global de un autómata celular lineal determina la configuración global en el siguiente paso de tiempo en la evolución del autómata celular.

un elemento del conjunto de estados, que arbitrariamente ha sido ordenado desde 0 hasta $k-1$, donde $k = |K|$ y K es el conjunto de estados.

Con espacios celulares pequeños es más fácil visualizar una transformación del espacio de tuplas al espacio de números enteros, así por ejemplo, en el caso del autómata celular lineal de $k = 2$ estados y una vecindad de radio $r = 1$ que defina un espacio celular de n células, habá 2^n posibles tuplas de n entradas. Claramente, cuando el espacio celular crece se vuelve más difícil ver esa correspondencia 1-1 con los números enteros.

En la teoría de autómatas celulares frecuentemente se define la función de evolución global como una transformación del espacio celular en sí mismo. Si G es el conjunto de configuraciones globales, X es la configuración de un espacio celular y $\overrightarrow{x_c}$ es la vecindad de una célula central x_c, la función de evolución global F se aplica al evolucionar localmente las vecindades del espacio celular:

$$\begin{aligned} F &: G \to G \\ F(X) &= f(\overrightarrow{x_c}) \quad \forall\, x_c \in G \end{aligned} \tag{1}$$

4. Comportamiento emergente en la regla 110

De acuerdo con [6], comportamiento emergente[1] de un sistema, es un comportamiento inesperado de los elementos que lo constituyen, y que interactúan localmente entre sí a lo largo de la evolución. En ocasiones éste es un comportamiento inesperado que sólo se puede observar en cantidades macroscópicas.

Hay diferentes tipos de comportamientos emergentes que han sido objeto de estudio en autómatas celulares, como ejemplos, la computación universal, y densidades de células de un

[1]También llamado comportamiento colectivo no trivial

mismo estado que varían cíclicamente en 3 etapas, este comportamiento es conocido como "ciclo 3".

La teoría del campo promedio [20] es un modelo matemático con el que obtenemos propiedades estadísticas de los espacios celulares, en cualquier momento de tiempo en su evolución. La intensión de hacer este estudio, es saber cuál es la tendencia de valores de cada estado celular, para que de alguna manera se pueda predecir el comportamiento global de la cantidad de celulas vivas[2].

Esto nos ayuda a tener una idea de cuál será el porcentaje de células de un estado a medida que el tiempo transcurre. Sin embargo, la teoría del campo promedio se basa en el supuesto que los elementos del espacio celular son independientes entre sí, pero esta condición no se cumple por la definición de vecindad, de manera que el modelo del campo promedio proporciona solamente una idea general sin que llegue a ser precisa [33].

5. Modelos de computación universal con autómatas celulares

Los tres trabajos antes mencionados no son los únicos que relacionan autómatas celulares con computación universal. Como un panorama general, mencionaremos algunos otros trabajos que se hicieron a partir de la década de los 80's.

De los trabajos más importantessobre computabilidad y universalidad con autómatas celulares están los estudios de Stephen Wolfram[3]; en 1998 Moshe Sipper [56] estudió casos de autómatas celulares no uniformes[4] para hacer computaciones.

En 1989 el japonés Kenichi Morita junto con su equipo de trabajo [38] desarrollaron un modelo basado en autómatas celulares de 1D con 3 estados[5] e introdujeron el concepto de autómata celular particionado [38]. En ese mismo año, Mats G. Nordahl publicó un artículo que muestra la relación entre autómatas celulares y lenguajes formales [39].

Posteriormente en la década de 1990, Stephen Wolfram hizo un profundo estudio de las evoluciones de los autómatas celulares, principalmente en el caso unidimensional, y propuso las cuatro clases de evoluciones que un autómata podría mostrar.

[2]Como "célula" es un término prestado de las ciencias biológicas, también hemos usado (tal vez inadecuadamente) los estados "vivo" y "no-vivo" para las células que tienen estados complementarios.

[3]Publicado en Communications in Mathematical Physics, 96 (November 1984) 15-57. Compilado en [68]

[4]Con reglas de evolución heterogéneas, vecindades no uniformes, etc.

[5]Aunque al final de cuentas es de 5 estados, por la inclusión de estados auxiliares que significan un corrimiento, una dirección y un estado "blanco".

Wolfram propuso que los autómatas celulares que muestran evoluciones complejas [70, 68], podrían ser capaces de hacer computación universal. Esta sugerencia fue presentada en su libro *A New Kind of Science* [69][6]; la demostración fue hecha por Matthew Cook en 1994 [9], utilizando un significativo e ingenioso sistema *gliders* y de choques de gliders. Los gliders son patrones definidos en el espacio celular que aparentemente se desplazan por el espacio al transcurrir cierto intervalo de tiempo [13].

5.1. Modelo de John von Neumann (principios de 1950's). A principios de la década de 1950, von Neumann estudió los mecanismos que debe tener una máquina para que pueda tener la capacidad de construir otra máquina igual a la máquina creadora; de manera que diseñó un autómata celular con esas propiedades. A partir de la fecha de su muerte en 1957, dejó el proyecto sin terminar y desde entonces se han hecho varios intentos de completarlo; uno de los trabajos más destacados de reproducir y completar el trabajo de von Neumann se dio en el año 2000 por un grupo de especialistas en diseño de hardware en el Laboratorio de Sistemas Lógicos del Instituto de Tecnología Federal Suiza [3].

El *constructor universal* es uno de los conceptos que von Neumann definió y que es una parte fundamental en el diseño de esa máquina[7] constructor universal, que es capaz de construir cualquier otra máquina constructor universal a partir de su descripción. Este proceso requiere que la descripción del constructor universal incluya su propia descripción, idea que fue tomada de modelos celulares vivos que contienen información de cómo construir otras células del mismo tipo:

- La descripción muestra las características básicas de la máquina, al estilo de un genoma, que es interpretado para construir una copia del constructor universal.
- La descripción es literalmente copiada, una vez que ha sido detectada y leída.

Stanislaw Ulam hizo la sugerencia (a von Neumann) de implementar sus ideas en un espacio bidimensional discreto. De manera que el universo creado por von Neumann lo define una matriz bidimensional infinita, cuyas entradas, llamadas células, son máquinas de estados finitos. Después de haber estudiado el modelo con varias opciones, llegó a definir 29 estados y una regla de transición [66].

En la figura 6 se muestra el esquema de la máquina autoreproductora diseñada por John von Neumann. El constructor universal está dividido en el control de la cinta (*Tape control*) y

[6]Se dedican muchas páginas a este tema, pero en la página 707 se describe cómo la regla 110 puede ser universal.

[7]En la percepción original, esta máquina es teórica, pero en el documento citado [3] se muestra un desarrollo en *hardware*

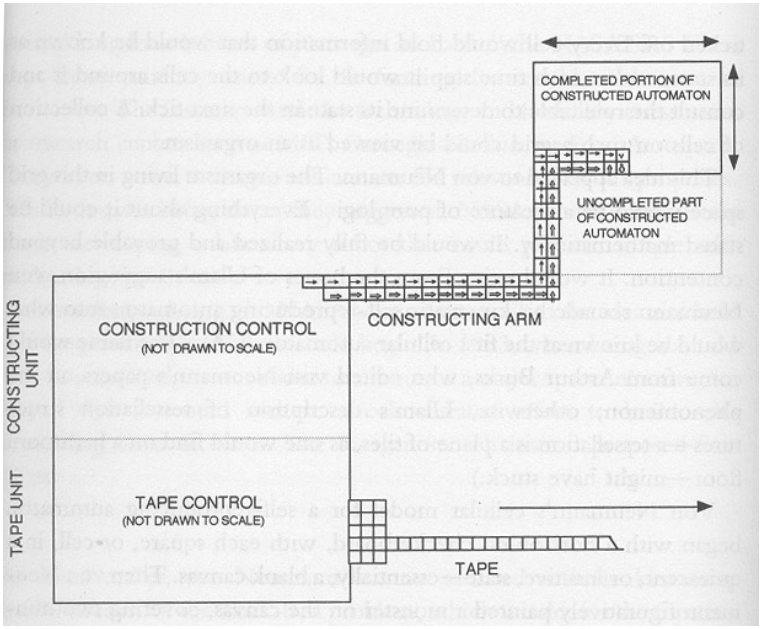

Figura 6. Esquema de la máquina universal de John von Neumann [66]

el control de construcción(*Construction control*); el control de la cinta obtiene la información de la máquina que se va a construir; y el control de construcción interpreta la descripción obtenida y construye el nuevo autómata por medio de un brazo constructor (*Constructing arm*). Podemos mencionar algunas características que tiene este constructor universal:

- Universalidad construccional, significa que es capaz de construir cualquier autómata, si cuenta con su descripción.
- Autoreproducción del constructor universal.
- Autoreproducción de la máquina universal, ya que el constructor universal está definido con una máquina universal de Turing, y la cinta contiene la información completa.

Cada célula del autómata puede tener uno de 29 posibles estados. Esos estados no son catalogados de manera numérica como usualmente se propone en las máquinas finitas de estados; la manera en que los diferentes estados son identificados tiene mucho que ver con su funcionalidad. En la tabla 2 se describen los conjuntos de estados posibles para cada célula. Enseguida una descripción de esos conjuntos de estados.

Una célula en el estado recesivo U no influye en otras células. Este estado se aplica a células que no se usan, ni en la descripción de la máquina, ni en el proceso de construcción.

Estado	Símbolo
Estado recesivo	U
Estados sensibles al ambiente	$S_\Theta, S_0, S_1, S_{00}, S_{01}, S_{10}, S_{11}, S_{000}$
Estados de transición ordinarios desactivados	$\uparrow, \downarrow, \leftarrow, \rightarrow$
Estados de transición ordinarios activados	$\Uparrow, \Downarrow, \Leftarrow, \Rightarrow$
Estados de transición especial desactivados	$\uparrow \cdot, \downarrow \cdot, \dot{\leftarrow}, \dot{\rightarrow}$
Estados de transición especial activados	$\Uparrow \cdot, \Downarrow \cdot, \dot{\Leftarrow}, \dot{\Rightarrow}$
Estado confluente desactivado	C_{00}
Estados confluentes activados	C_{01}, C_{10}, C_{11}

Cuadro 2. Espacio de estados de la máquina autoreproductora de von Neumann [3]

Las células tienen 16 estados que permiten la transmisión de información entre células que no están en estado recesivo. Cada uno de estos estados de transmisión tienen una de cuatro direcciones: norte, sur, este y oeste; hay una distinción entre estados activos o inactivos. Además, se pueden distinguir estados de transmisión ordinarios y de propósito especial que propagan diferentes tipos de activación.

Los estados de transmisión ordinarios propagan activaciones ordinarias en su dirección de salida, por su parte, los estados de transmisión especial, también propagan activaciones especiales en su dirección de salida. Las propagaciones ordinarias introducen un retardo de un paso de tiempo en la propagación de la activación y actúan como una compuerta OR. Un estado cambia a activo si recibe una señal de activación ordinaria o especial desde uno de sus tres lados confluentes.

Los 4 estados confluentes se usan para transmitir activaciones, funcionan como compuertas AND, generan un retardo de dos tiempos en la transmisión de una activación y pueden dividir el flujo de transmisión, comportándose como un distribuidor. Un estado confluente puede ser C_{10} o C_{11} cuando todas las células vecinas estén en estado de transmisión ordinaria dirigidas hacia ellas, si cualquiera está en estado activado.

En la figura 7 se pueden apreciar en 6 pasos de tiempo, las evoluciones del autómata celular de von Neumann en diferentes situaciones:

- en la columna **a** se muestra cómo se propagan las señales de activación en actividades ordinarias,
- en las columnas **b** y **c** se muestran las actividades de los estados confluentes, comportándose como compuertas AND (columna **b**) y OR (columna **c**).

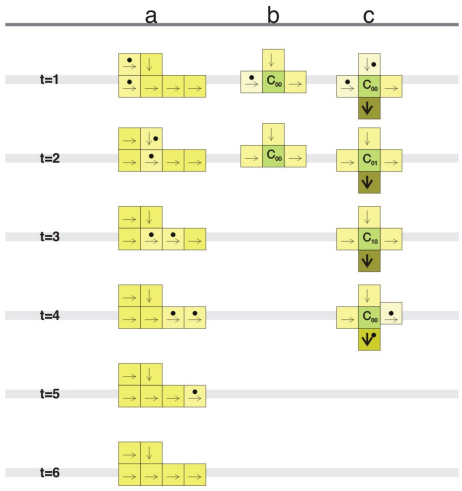

Figura 7. Evolución de los estados de transmisión y confluentes de la máquina de von Neumann [3]

Los 8 estados restantes se usan para construir los procesos, que también se llaman **procesos dirigidos**, que como excepción, sí pueden crear estados recesivos o cambiar de un estado recesivo a estados de transmisión no activados o a estados confluentes. Estos estados se llaman **estados sensibles** aunque cambian de estado en una unidad de tiempo.

Una célula que se encuentre en estado recesivo U puede cambiar su estado al estado sensible S_Θ si recibe una señal de activación ordinaria o especial por alguno de sus lados, luego puede cambiar al estado sensible S_0 si no recibe ninguna señal de activación, o a S_1 en caso de que sí reciba.

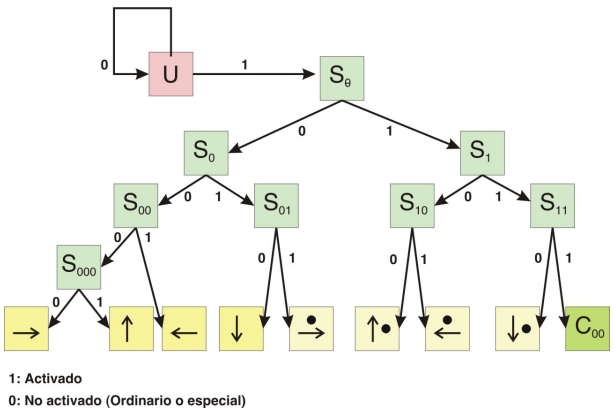

Figura 8. Proceso de construcción de la máquina de von Neumann [3]

La figura 8 muestra el ciclo constructivo en que pueden encontrarse las células en el autómata, mientras se encuentren en uno de esos 9 estados desactivados. Una célula que se encuentre en un estado sensible no ejerce influencia en sus vecinos.

Finalmente se describe una manera de cambiar el estado de transmisión o confluente a un estado recesivo, y se puede hacer de dos maneras: enviando una activación especial por uno de los 4 lados de un estado de transmisión ordinaria o de un estado confluente; y la otra manera es enviar una activación ordinaria por uno de los cuatro lados de un estado de transmisión especial.

5.2. Modelo de Alvi Ray Smith III(1968). Este modelo fue presentado en 1968 y posteriormente revisado en 1971 [49, 50]. Muchos de los trabajos anteriores al de Ray Smith III, se hicieron con base en el modelo original de John von Neumann. También se dieron a conocer algunos estudios del modelo de von Neumann por parte de J. W. Thatcher en 1964 [8], y un poco más tarde se mencionarían fundamentos de autómatas celulares, manteniendo la idea original de von Neumann por M. A. Arbib [2] y en 1968 por E. F. Codd [7].

Sin embargo en 1971 Alvi Ray Smith III dio a conocer su modelo que está basado en la idea de usar funciones parciales recursivas [53], y no en simular una cinta y la cabeza lectora de la cinta, como en el modelo de von Neumann.

El trabajo de Ray Smith III inicia con el teorema de la "Máquina universal de Turing" en términos de funciones parciales recursivas, y entonces hace analogías entre los números naturales y las configuraciones globales de un autómata celular, mencionando la idea de *computación* en espacios celulares. Dado un espacio celular Z, una función de transición global F, una configuración global inicial c y una función parcial recursiva $g : \mathbb{N} \to \mathbb{N}$, se dice que c **calcula** g si:

(1) Existe una secuencia de configuraciones globales (d_n), cada una diferente de la configuración global inicial c, que es una enumeración efectiva de las configuraciones globales (no necesariamente de todas).
(2) Existe una función parcial recursiva $h : (\chi_i) \to (\chi_i)$ tal que, si $g(n)$ está definida, entonces hay un momento t_0 tal que

$$h(F^{t_0}(c \cup d_n)) = d_{g(n)}$$

[8] "Universality in the von Neumann cellular model", Tech. Rep. 03105-30-T, ORA, U.Mich, Ann Arbor, 1964.

Donde (χ_i) es una secuencia de configuraciones globales que representan evoluciones del autómata. Se puede ver esta función h como una función decodificadora, y a la configuración global $d_{g(n)}$ como el resultado del cálculo.

(3) Hay un número $m \geq 1$ y una función recursiva $\pi : (\chi_i)^m \to \{0, 1\}$ para determinar, a partir de una secuencia finita de m configuraciones globales que t_0 ha ocurrido, es decir:

$$\pi(c_t, c_{t+1}, \cdot, c_{t+m-1}) = 1 \; ssi \; t = t_0$$

Este procedimiento π sirve para detectar el final de un cálculo, y sucede cuando se ha encontrado una configuración en particular.

Ahora, Z es un espacio celular computación-universal si existe un conjunto de configuraciones globales $U \in Z$ tal que para cualquier función parcial recursiva (enumeración de configuraciones globales) g, efectivamente se puede encontrar una configuración global $c \in U$ tal que c **calcula** g.

Se requieren de los espacios celulares para simular otros dispositivos de cómputo tales como las máquinas de Turing. En general un sistema lógico de manipulación de cadenas es uno que toma como dato de entrada una cadena S, y obtiene a su salida a lo más otra cadena S' (posiblemente $S = S'$) que puede ser obtenida de S en un solo paso. De manera que si se ven las configuraciones globales como cadenas de símbolos, la función de transición global es un sistema lógico de manipulación de cadenas.

Una generalización directa al espacio de d-dimensiones del concepto de *cadena*, es un *patrón*. Así un patrón en 1-D es una cadena, y una configuración con soporte[9] finito en un espacio celular d-D es un patrón d-D. Entonces un espacio celular es un sistema *monogénico*, es decir, de manipulación de patrones, como también lo son las máquinas de Turing o el sitema *tag* de Post [**42**, **36**].

Un sistema de manipulación de patrones T es un par ordenado (P, v), donde P es un conjunto de patrones finitos indizados con números enteros, y $v : \mathbb{N} \to \mathbb{N}$ es una transformación de patrones. Ahora, consideremos un espacio celular Z y un sistema de manipulación de patrones T. Sean además, k_1 y k_2 números enteros positivos, i el índice de los patrones en P y $\mathbb{C}$ el conjunto de todas las configuraciones en Z con soporte finito, es decir, configuraciones globales definidas en espacio celular finito. entonces Z *simula* T *en* $\frac{k_2}{k_1}$ *evoluciones* si y sólo si existe una función inyectiva y computable $\gamma : \mathbb{N} \to \mathbb{C}$ y existe también una función δ de funciones recursivas sobre funciones recursivas tal que:

$$F^{k_2}(\gamma(i)) = \gamma(v^{k_1}(i)); \text{ donde } F = \delta(v)$$

[9] espacio celular cuyas células tienen estado diferente al estado nulo

De manera que una máquina de Turing se puede describir en función de configuraciones globales de autómatas celulares, dando un conjunto P de "descripciones instantáneas" y una función v que está determinada por la función "siguiente estado" del control de la cabeza lectora de la máquina de Turing. Se ha supuesto que cuando la máquina de Turing termina, la secuencia asociada de patrones continúa pero es pasiva ($v(i) = i$).

Alvi Ray Smith III introduce el concepto de $(m, n)-$*Turing machine*, que es una máquina de Turing que especifica el par (m, n), y se refiere a la tabla de símbolos-estados como lo muestra la tabla 3

	q_0	q_1	$\cdots$	q_{n-1}
s_0				
s_1				
$\vdots$		$\frac{x_i X}{q_j}$		
s_{m-1}				

Cuadro 3. Tabla de símbolos y estados. $X \in \{R, L\}$, $0 \leq i \leq (m-1)$, $0 \leq j \leq (n-1)$

Alvi Ray Smith III demostró que *cualquier máquina de Turning (m,n) T, se puede simular con un autómata celular $2D$ de tamaño* $\max(m+1, n+1)$*, una vecindad de 7 células* [50]; con lo que establece una relación entre las evoluciones de los autómatas celulares que menciona en ese teorema y los cálculos en las máquinas de Turing.

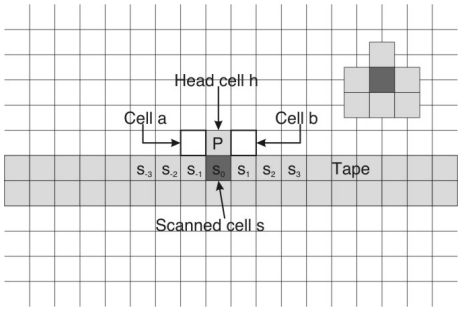

Figura 9. Modelo de vecindad creado para hacer computación universal [50]

La geometría del espacio de vecindades se usa para poder distinguir una célula en estado $Q_1 \in A = \{1, \cdots, m\}$ que corresponden a un símbolo de la máquina de Turing, de una célula en estado $Q_2 \in B = \{1, \cdots, n\}$ que corresponden a un estado de la máquina de Turing.

La manera en que este autómata celular simula una máquina de Turing T, es que sus evoluciones asemejan el comportamiento de la máquina T. El autómata celular diseñado por

Ray Smith III es entonces definido en un espacio bi-dimensional, dedicando un renglón de células para simular la "cinta" de la máquina de Turing, una célula del espacio celular corresponde a un cuadro en la cinta, y un renglón del espacio celular se dedica a los movimientos de la cabeza lectora de la cinta. Las células del otro renglón están en estado nulo Q_0.

Es interesante la comparación entre las pruebas de computabilidad universal hechas por von Neumann [66], Codd [7], Arbib [2] que se basan en el modelo original de von Neumann y por otro lado el modelo que Ray Smith propone [50]; es interesante porque la diferencia radica en que, en el modelo de Ray Smith el diseño del autómata celular depende de la máquina de Turing a ser simulada; y desde el punto de vista de los modelos basados en la idea de von Neumann, cualquier máquina de Turing se puede simular una vez que el diseño del autómata celular ha sido propuesto.

Sin embargo, la diferencia entre los autómatas celulares (dependientes de la máquina de Turing e independientes de la máquina de Turing) ya no es importante cuando la máquina de Turing que se va a simular es precisamente la máquina universal.

Se puede traducir el autómata celular de dos dimensiones definido originalmente por Ray Smith, en un autómata celular de una dimensión, también definido por él mismo. La configuración de vecindad en este nuevo autómata celuar 1D se muestra en la figura 10-a. En la figura 10-b se muestra cómo están distribuidas las células del antiguo espacio celular bidimencional en el nuevo espacio celular lineal.

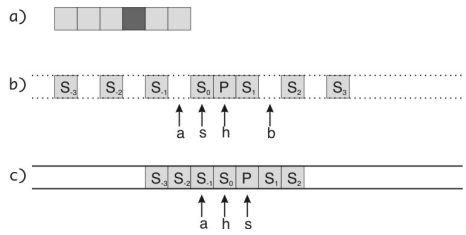

Figura 10. a) Modelo de vecindad creado para hacer computación universal en 1D. b) configuración de la máquina de Turing relacionada. c) Segundo ejemplo de una máquina de Turing relacionada a un espacio celular 1D. [50]

El trabajo de Ray Smith incluye el siguiente enunciado: *Para cualquier máquina de Turing* T (m, n)*, existe un espacio celular* Z_T *de 3 vecinos y un espacio de estados de* $(m+2n)$ *que simula computación universal* (figura 10-c). El razonamiento para validar la declaración anterior es dar un espacio celular Z_T con la configuración de vecindad de adecuada y un diseño de máquina de Turing como se muestra en la figura 10-c. La función de transición f no cambia el estado de las células, excepto en los casos a, h y s. Se pueden completar de manera más o menos sencilla los detalles de la función de transición para que las configuraciones de

la cinta $\cdots x_0x_1qx_2x_3\cdots$ se muevan a la derecha (L) y tener el nuevo estado q' y cambiar el símbolo x_2 a x_2':

$$\cdots x_0x_1qx_2x_3\cdots$$
$$\cdots x_0x_1x_2'q'x_3\cdots$$

Similarmente el movimiento a la izquierda tendría el siguiente aspecto:

$$\cdots x_0x_1qx_2x_3\cdots$$
$$\cdots x_0x_1q_L'x_2'x_3\cdots$$
$$\cdots x_0q'x_1x_2'x_3\cdots$$

Los estados q y q_L se necesitan para representar cada estado de la máquina de Turing. En este caso, el símbolo nulo es simulado por el estado nulo.

Finalmente, en el trabajo [50] se muestran las condiciones necesarias para que un autómata celular desarrolle computación universal. En particular, se muestra que es suficiente un autómata celular en 1D para hacer computación universal en el dominio de las funciones parciales recursivas. Un autómata celular capaz de hacer computación universal se llama *espacio celular computación universal*. El trabajo de Ray Smith III ha servido para obtener algunos resultados que se aplican al usar autómatas celulares como generadores de lenguajes.

5.3. Modelo de Conway: *Life* (1978). Desde que se dio a conocer en la revista *Scientific American* en octubre de 1970 por Martin Gardner, y luego al mostrar capacidades de autoreproducción [15, 16], el juego de "*life*" creado por John H. Conway, ha cautivado la atención de no pocas personas, intentando hacer patrones que generen comportamientos **complicados**[10] a tal grado de crear elementos muy complicados e ingeniosos, que han servido para construir una máquina de Turing, o compuertas lógicas como lo destaca Buckingham en [5].

El juego de "*life*" es un autómata celular definido en un espacio bidimensional infinito, dividido en células biestables, que son actualizadas todas a un mismo tiempo progresivo en pasos discretos. Una célula está *muerta* o vacía si tiene estado 0, o bien, la célula está *viva* si

[10]Como lo haría notar von Neumann en 1949 en su manuscrito acerca de la "Teoría y organización de autómatas complicados" [66], diciendo que el término "complejidad" relacionado con su trabajo, era más bien *vago, no científico e imperfecto.*

su estado es 1. Cada célula esta relacionada con sus 8 células vecinas que distan de la célula central en una unidad. La regla de evolución que es aplicada a cada célula es la siguiente:

- Si una célula está viva (estado 1), y tiene 2 ó 3 células vecinas vivas también en estado 1, entonces la célula permanece viva para la siguiente generación; de otro modo muere (estado 0).
- Si una célula está muerta y tiene exactamente 3 células vivas como vecinos, entonces cambia su estado y renace (estado 1).

En un documento que se puede obtener en internet [51], y en el libro [1] se pueden revisar los detalles de cómo se ha construido una máquina universal de Turing con los patrones del juego de "*life*", en los siguientes párrafos se hará una descripción general del procedimiento utilizado para crear esta máquina universal.

Hay tres patrones de *life* que son básicos para la construcción de la máquina de Turing: un **sumador**, una **celda de memoria** y un **bloque de memoria**.

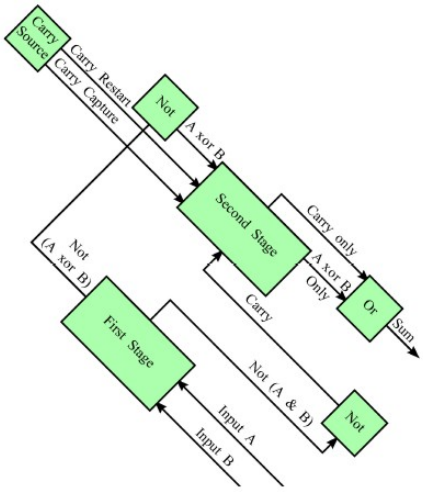

Figura 11. Esquema de un sumador construido con patrones *life* [51]

El primer patrón, el sumador, implementa la suma binaria de dos cadenas de *gliders* emitiendo una nueva cadena de *gliders* como resultado final de la suma. La figura 11 muestra un esquema de la construcción de un sumador binario en *life*. En este esquema, los datos (números) son codificados en secuencias binarias, en donde un *glider* representa un "1" y la ausencia de un *glider* representa un "0", dos 1's son representados por secuencias separadas por un espacio de 60 generaciones.

El sumador binario trabaja en dos etapas, la primera recibe los números en dos entradas. El resultado de la suma pasa como entrada a la segunda etapa, incluyendo su exceso (acarreo). Este exceso se agrega en la segunda etapa al resultado final, pudiendo generar de nuevo otro acarreo, que es devuelto a la segunda etapa para agregarse al siguiente bit y así sucesivamente.

El segundo patrón, una celda de memoria, tiene el objetivo de almacenar su contenido en una matriz junto con otras celdas de memoria. El contenido de la celda de memoria (que es un 1 o un 0) es el resultado de un choque de *gliders* que interactúan en otro patrón llamado *pentadecathlon* que produce un glider generando un ciclo debido al choque del glider con otro patrón que refleja el glider a 90 grados, que finalmente entra de nuevo y permanece en el ciclo indefinidamente.

El tercer patrón, el bloque de memoria [23, 51], es una implementación de Dean Hickerson [23] basado en el trabajo de Marvin Minsky [37], quien empleó el concepto de un registro contador que tiene la capacidad de almacenar un número positivo. Las únicas operaciones que se requieren implementar son sumas, restas y verificaciones de valor 0. En la figura 12 se muestra esquemáticamente el diseño del bloque de memoria. Este diseño genera 4 *gliders* cada 120 generaciones que se bloquean por dos patrones bloqueadores que, si uno de ellos falta, se borra uno de los 4 *gliders* y los tres restantes hacen las operaciones de incremento. Si faltara el otro patrón bloqueador, entonces se borra un *glider* más, y los 2 restantes hacen la operación decremento.

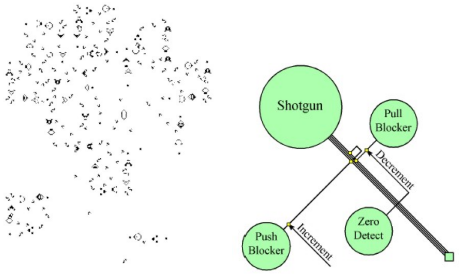

Figura 12. Izq.:Bloque de memoria construida con patrones *life* [23]. Der.: Esquema [51]

La construcción de la máquina universal de Turing, utiliza los patrones descritos anteriormente y otros más que se describen con detalle y con ejemplos de sus evoluciones en [51]. Las celdas de memoria se ordenan en arreglos. El sumador proporciona la capacidad de incrementar un número binario que indica la dirección de la siguiente instrucción a ser leída, y la obtiene de la unidad de memoria, y el bloque de memoria es un registro contador que es capaz de almacenar un valor de cualquier tamaño.

La máquina de estados contiene la unidad de memoria que ha sido construida en base a celdas de memoria que mantienen los estados activados (1's) como cadenas de *gliders* moviendose en ciclos (ver figura 13), las direcciones de las localidades de memoria son obtenidas de las pilas, que simulan la cinta de la máquina de Turing. La actividad conjunta de las pilas simulan los movimientos de la cabeza lectora, intercambiando los datos representados como flujos de *gliders*.

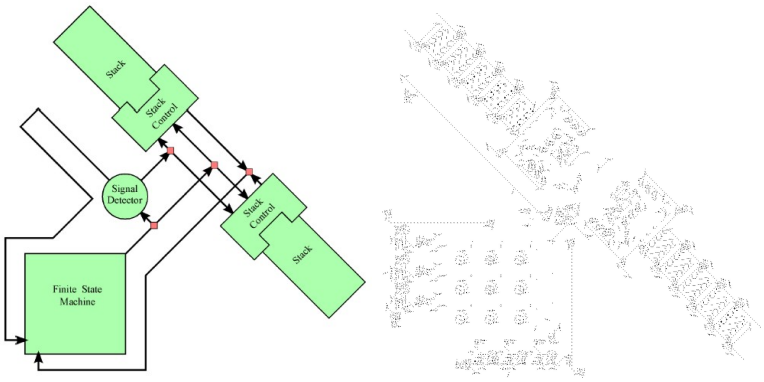

Figura 13. Izq.:Esquema de la máquina de Turing *life* [51]. Der.: Patrón *life* de la máquina de Turing [51]

El control de la pila es activado por una "señal presente" representada por un flujo de *gliders* para usarse como salida de la máquina de estados para determinar cuando hacer un *pop* y cuando hacer un *push* y también el símbolo que será agregado a la pila. El dato que resulta de la operación *pop* se envía a la máquina de estados para formar parte del siguiente ciclo en el arreglo de la memoria.

La máquina de estados es un arreglo de celdas de memoria, tiene dos entradas, una que representa el siguiente estado y luego se usa para seleccionar un renglón de ese arreglo y la otra entrada la usa una de las pilas que manda un flujo de *gliders* que representan el símbolo leído y luego se usa para seleccionar la columna del arreglo de celdas de memoria. Rendell concluye [51] que es muy fácil modificar el dato inicial en la pila para mostrar que la máquina de estados trabaja correctamente, solamente hay que poner una célula en un lugar adecuado, o modificar la secuencia de *gliders* que representan los datos. Para aplicaciones reales la máquina de Turing *life* se vuelve prácticamente inútil por el tiempo que requiere hacer las operaciones, pero ha servido para mostrar la extraordinaria capacidad del juego de "*life*" para construir máquinas de comportamiento *complicado* a partir de elementos simples.

Capítulo 2

El espacio geométrico de los triángulos de la regla 110

1. Resumen

Los triángulos que se forman en las evoluciones del $acl(2,1)_{110}$ muestran regularidades en forma, aunque el rango de sus tamaños varía ampliamente. En un espacio celular finito, el rango de tamaños de los triángulos, también es finito. Esta diversidad de tamaño es una de las razones que permiten la aparente interacción entre estos patrones, para formar estructuras más complejas [33, 34, 35].

En este capítulo se establecen las bases algebraicas para manipular estos triángulos, definiendo tanto los elementos que lo constituyen, como los operadores que permiten construir nuevos patrones gráficos, más complejos y con características particulares que determinarán la diferencia entre los patrones que son válidos y los que no lo son.

2. Conjunto de reglas con comportamiento global similar a la regla 110

En el caso que nos ocupa, estudiaremos la regla 110 en autómatas celulares lineales de 2 estados y radio de vecindad 1, que al evolucionar generan patrones gráficos. El interés de este estudio estriba en que los cubrimientos generados con estos patrones gráficos, pudieran significar algún tipo de computación. Si ocurre, se buscará mostrar sus capacidades. Sin embargo, la regla 110 no es la única regla que genera este tipo de patrones, como se puede apreciar en la figura 14.

2.1. Formación de triángulos y su relación con la regla de evolución local. El estudio de los patrones emergentes de un sistema se presenta en dos direcciones. La primera es encontrar cómo un patrón se forma a partir de un gran número de componentes; y la segunda es encontrar la manera en que se forman patrones debido a la interacción de los mismos [54].

		110	124	137	193
0	000 →	0	0	1	1
1	001 →	1	0	0	0
2	010 →	1	1	0	0
3	011 →	1	1	1	0
4	100 →	0	1	0	0
5	101 →	1	1	0	0
6	110 →	1	1	0	1
7	111 →	0	0	1	1

Cuadro 4. Grupo de simetrías de la regla 110

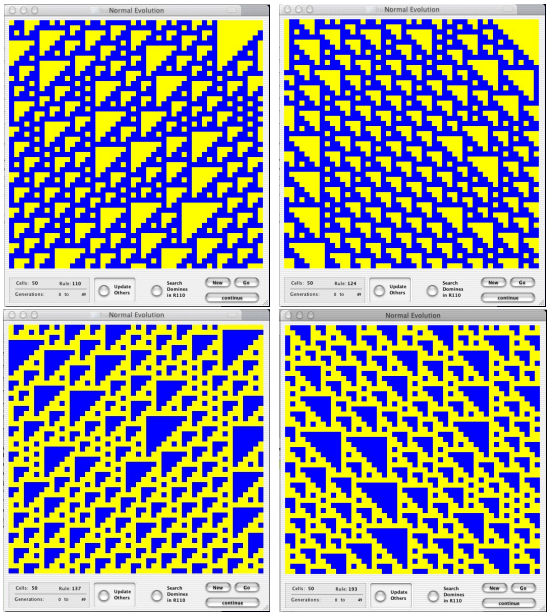

Figura 14. Grupo de simetría de la regla 110, que incluye la regla 110, 124, 137 y 193. el color amarillo (claro en impresiones sin color) representa el estado 0 y el color azul (obscuro en impresiones sin color) representa el estado 1.

2.1.1. *Descripción informal de un triángulo* R110. Tanto la regla 110 como las reglas 124, 137 y 193, tienen en común la generación de patrones gráficos similares, en las cuatro reglas surgen patrones como triángulos que difieren en la dirección y en el estado celular. En la tabla 4 vemos las cuatro reglas antes citadas en su expansión binaria, para observar las semejanzas y diferencias de todas ellas de una sola vez. De la figura 14 observamos que

0	$000 \to 0$	Forma el interior del triángulo.
1	$001 \to 1$	Forma la diagonal de arriba hacia abajo y de derecha a izquierda.
2	$010 \to 1$	Forma la frontera izquierda.
3	$011 \to 0$	Forma la frontera izquierda en el vértice superior izquierdo.
4	$100 \to 1$	Forma parte del interior del triángulo, precisamente la parte que está junto a la frontera izquierda.
5	$101 \to 1$	Cierra la frontera de 1's en el vértice inferior.
6	$110 \to 1$	Forma la frontera izquierda.
7	$111 \to 0$	Forma parte del interior del triángulo, precisamente la parte que está junto a la frontera superior.

Cuadro 5. La regla 110 y la formacion de los patrones triangulares

las cuatro reglas generan patrones con forma triangular, aunque como están en un espacio discreto, la diagonal tiene el efecto escalera.

Las diferencias entre los triángulos de las cuatro reglas se aprecian casi de forma inmediata, cambian en la orientación de los triángulos y cambian en los colores. Se puede apreciar también que no hay triángulos rotados, o superpuestos. Estas dos últimas características serán importantes para delimitar el espacio de patrones cosiderados y tener esas características como reglas para determinar la pertenencia de un triángulo al conjunto de triángulos permitidos. Los patrones triangulares, junto con su frontera se forman de la manera que se muestra en la tabla 5.

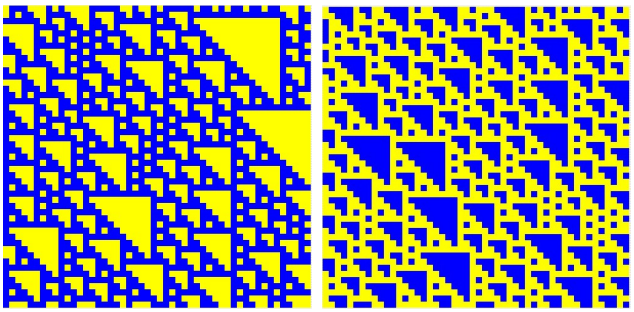

Figura 15. Detalle de las reglas 124 y 193 de izquierda a derecha, iniciando con la misma configuración inicial.

Es decir, si se quisieran tener formas triangulares orientadas a la derecha (ver figura 15 izquierda), partiendo de la regla 110, se tendrían que hacer los siguientes cambios:

(1) Poner la regla $100 \to 1$

(2) Poner la regla $001 \to 0$

Si ahora se desea intercambiar los colores de los triángulos (figura 15 derecha), manteniendo la orientacion de los triángulos a la derecha, se tienen que hacer los ajustes que se muestran en el cuadro 6:

1	000	$\to$	1	5	100	$\to$	0
2	001	$\to$	0	6	101	$\to$	0
3	010	$\to$	0	7	110	$\to$	1
4	011	$\to$	0	8	111	$\to$	1

Cuadro 6. Cambios para intercambiar los colores

Otro cambio interesante, es ahora tener diagonales hacia la derecha y hacia la izquierda, lo que producirá patrones con forma de triángulos isóceles dirigidos hacia abajo. Las modificaciones que hay que hacer son mínimas, de hecho, a partir de la regla 193 solamente se requiere poner la regla $110 \to 0$, produciendo formas como la que se muestra en la figura 16.

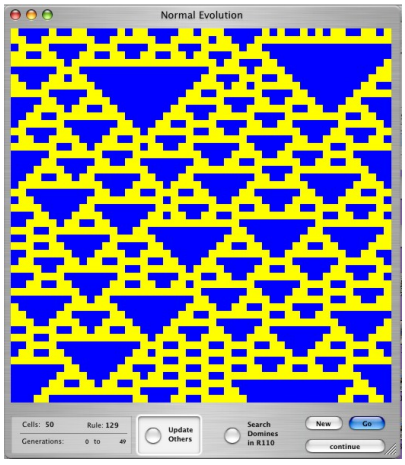

Figura 16. Regla 129, muy similar a la regla 193 haciendo diferencia únicamente en la vecindad $110 \to x$. Generando patrones gráficos en forma de triángulo isóceles orientados hacia abajo.

Aunque las definiciones, y en general todo el estudio de esta tesis, se ha hecho en base a los patrones gráficos generados por la regla 110, los resultados pueden ser usados en las otras reglas del mismo grupo (ver figura 14), casi sin modificaciones. Para otro tipo de reglas que generen patrones gráficos similares, los resultados de esta tesis sirven como referencia para generar las propias reglas.

Como se mencionó antes[1], la evolución local aplicada a todas las vecindades define la evolución global del espacio celular; de manera que cuando se dice que el autómata celular "evoluciona", significa que se ha obtenido una configuración global a partir de la configuración global previa.

Convencionalmente se ha decidido agrupar las evoluciones del autómata celular unidimensional en renglones sucesivos de tiempo, con el fin de observar los cambios en el tiempo del espacio celular lineal. Frecuentemente, se colocan de arriba hacia abajo de acuerdo al incremento del tiempo, aunque también se puede hacer a la inversa. Los patrones generados solamente son diferentes en su orientación, para hacer una conversión se requiere reflejar horizontalmente la figura. En el resto de la tesis tomaremos la convención de tomar las evoluciones de arriba hacia abajo.

La "evolución" del autómata celular, es decir, el espacio celular extendido dimensionalmente en un intervalo de tiempo, también se conoce como **diagrama espacio-temporal**, y es un subespacio de $\mathbb{Z}^2$. Cada punto del espacio celular en cualquier momento de su evolución, es una célula, de manera que cada punto gráfico del subespacio de $\mathbb{Z}^2$ que ha sido considerado, tiene un estado asociado dentro del espacio de estados definido para el autómata celular. En esta tesis el espacio de estados es $\{0, 1\}$, y para efectos visuales se puede relacionar un color con cada estado, así la figura 14 tiene 4 evoluciones de diferentes autómatas celulares.

Tomando cualquier evolución de cualquier regla que pertenezcan al mismo grupo que la regla 110[2], se observan patrones triangulares de diferentes tamaños pero conservando dirección y forma. Además de las zonas triangulares, aparece otra zona que aparenta ser el fondo que rodea los triangulos.

Al observar con más detalle se nota que esa frontera ocurre en los lados rectos del triángulo y no ocurre en la diagonal, como se puede apreciar en la figura 17. Aunque ésta es una apreciación completamente arbitraria. Un *triángulo* en la regla 110 de tamaño n denotado como t_n, es una subregión del diagrama espacio-temporal del autómata celular regla 110 (ver figura 18), que se extiende $n+1$ pasos en el tiempo y abarca $n+1$ puntos en el espacio, que se distribuyen de la siguiente manera:

- Una célula de estado 0 en el vértice superior izquierdo
- El cuerpo del triángulo esta compuesto por n renglones, contados de abajo hacia arriba:

[1] Se menciona en la página 17 y luego en la figura 5

[2] Se puede tomar cualquier regla del grupo de reglas al que pertenece la 110, ver página 35, pero en esta tesis se toma como representante a la regla 110.

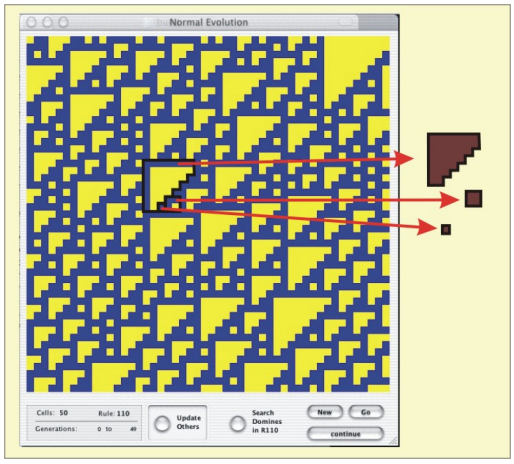

Figura 17. Patrones gráficos emergentes en la evolución de la regla 110

  - El renglón cero tiene 1 célula de estado 0
  - El renglón uno tiene 2 células de estado 0
  - ...
  - El renglón $n-1$ tiene n células de estado 0
- La frontera superior del triángulo se compone de n puntos de estado 1, que se extienden de izquierda a derecha.
- La frontera izquierda del triángulo se compone de n puntos de estado 1, que se extienden de abajo hacia arriba.

2.2. Computación universal en la regla 110. La regla 110 ofrece en sus evoluciones intrincadas intereacciones aparentes entre los patrones gráficos que produce. Este autómata celular fue definido en base a la regla local; posteriormente se observó un comportamiento sorprendente; de manera que se le decidió ubicar en la clase de comportamientos complejos.

2.2.1. *La conjetura de Wolfram.* En 1984 se publicó un artículo de Sthepen Wolfram[3] donde hace un profundo estudio acerca de las evoluciones de los autómatas celulares en 1D con una vecindad de 3 células y un espacio de 2 estados, en el cual se hace una distinción característica de éstas.

Clasificó las evoluciones en 4 clases, semejantes a las clases de sistemas dinámicos. En el artículo, Wolfram sugirió que los autómatas que generaban evoluciones de "clase 4" (complejos) eran capaces de hacer computación universal, aunque mencionaba que los autómatas de

[3] *Universality and Complexity in Cellular Automata*, in *Physica D*, volumen 10, 1984, pág. 1-35; compilado y editado en [68].

2 estados y 3 vecinos eran aparentemente demasiado "simples" para ser de clase 4; la sugerencia fue basada en la "compleja" apariencia de las configuraciones globales, relacionándola con los procesos en la máquina universal de Turing.

Un año más tarde, en 1985, Wolfram publicó otro artículo[4], allí se menciona que algunas evoluciones de los autómatas celulares, aun cuando su definición es muy simple, generan patrones complejos, de manera que decidió incluir estas reglas en la clase de comportamientos complejos. Así que ya habian dos declaraciones que se sospechaban eran ciertas: "Los autómatas complejos son capaces de hacer computación universal", y "algunos autómatas celulares elementales (2 estados, radio de vecindad 1) son complejos". En 1986, Wolfram propuso[5]que: "El autómata celular con regla de evolución 110 tiene un comportamiento suficientemente sofisticado como para hacer computación universal". En [69], describe cómo se puede hacer la demostración.

2.2.2. *El modelo de Mathew Cook.* Para simular una máquina de Turing con las evoluciones del autómata celular regla 110, Mathew Cook [9] se basó en la implementación de un sistema *tag* cíclico [42, 36], usando un sistema de gliders que ingeniosamente se puede establecer en la configuración global inicial del espacio celular. Un sistema *tag* es un conjunto de reglas que especifican un número fijo de elementos a ser removidos del inicio de una secuencia, En dependencia de la subsecuencia de elementos quitados se establece un conjunto de elementos a ser agregados al final. Consecuentemente, un sistema *tag* cíclico es un sistema *tag* que tiene una lista de n reglas *tag*, que todas son aplicadas en orden secuencial y entonces se aplican de nuevo empezando con la primera regla, formando así un ciclo.

Cook explica en su manuscrito porqué el modelo de *tag* cíclico hace computación universal; una vez que se comprende que un sistema *tag* cíclico es equivalente a una máquina de Turing, se describe cómo un sistema de *gliders* es equivalente a un sistema *tag* cíclico. Luego, se puede crear un sistema de *gliders* que haga computación universal; porque un adecuado sistema de *gliders* es equivalente a un sistema *tag* cíclico, y un adecuado sistema *tag* cíclico es equivalente a la máquina universal de Turing.

Un sistema *tag* cíclico funciona con una cinta finita en donde se leen y retiran símbolos del frente de la lista, y se escriben y agregan al final de la lista. Los símbolos que se agregan al final, se leen de una lista de símbolos agregables en su orden de aparición, cuando se termina la lista se regresa al inicio de ella para leer el primer símbolo agregable una vez mas. La decisión de agregar o no el símbolo depende del símbolo leído al frente de la cinta.

[4] *Undecidability and Intractibility in theoretical Physics* (*Physical review letters*, vol. 54, 1985, pp 735-738); que también se encuentra compilado en [68]

[5] *Theory and applications of Cellular Automata*, *Word Scientific 1986*; incluído en [68]

Se puede mostrar que un sistema *tag* cíclico hace computación universal mostrando que emula un sistema *tag* [42, 36]. Para mostrar cómo el sistema cíclico emula el sistema *tag* original:

(1) Se inicia la lectura de una secuencia de k caracteres que corresponden a un símbolo del sistema *tag* original, ocasionalmente se estará al inicio de la secuencia de caracteres, o bien a la mitad.
(2) Si se está al inicio de la lista cuando se empieza a leer la secuencia de k caracteres, entonces el caracter Y causa que se agreguen exactamente un agregable.
(3) Si se encuentra a la mitad de la lista cuando se inicia la lectura de los k caracteres, entonces no se agregan símbolos, puesto que la segunda mitad de la lista de agregables son símbolos de longitud 0 y la secuencia de k caracteres permanece sin cambio.

Ahora, para emular un sistema *tag* cíclico con un sistema de *gliders*, se puede considerar un sistema en 1D. De manera muy informal y descriptiva, cada punto que se mueve se llama *glider*. Un sistema de *gliders* debe tener un número finito de *gliders*.

El modelo de Cook tiene la cinta representada por *gliders* estacionarios, este tipo de *gliders* es útil para representar los datos de la cinta. La cinta está orientada con la parte frontal a la derecha y el final de la cinta al lado izquierdo. La lista de datos agregables, también son *gliders* que representan los símbolos Y y N. Estos símbolos se moverán desde la derecha y van a chocar con el primer elemento de la cinta. Si el *glider* estacionario representa un N, entonces el dato de la tabla de agregables se destruye. Sin embargo, cuando el *glider* choca con un *glider* estacionario que representa un Y, entonces se cruzan y el *glider* continúa su viaje a la izquierda.

Para limpiar el espacio celular, se mandan otro tipo de *gliders* que se llaman *gliders líderes*, que cuando chocan con los datos N se convierten en *rechazadores* y se destruyen, por otro lado, cuando chocan con datos Y se convierten en *aceptadores* conviertiéndose en datos que viajan hasta que eventualmente choquen con otro *lider* y sean absorbidos. Cada símbolo que se mueve, cruzará la cinta repitiendo el proceso. Con las ideas esbozadas anteriormente se puede simular una máquina de Turing con un sistema de *gliders*, si esta máquina de Turing simulada es universal, entonces se puede codificar en la cinta la descripción de una máquina de Turing y los datos de operación.

3. Descripción formal de un triángulo R110

Sea $\mathbb{U} = \mathbb{Z}_n \times \mathbb{N}$ el diagrama espacio-temporal de las evoluciones del $acl(2,1)_{110}$, en un espacio de tamaño n. Las células están determinadas por el par ordenado (i,j) con $0 \leq i \leq n-1$, y $j \geq 0$.

DEFINICIÓN 1. *Llamaremos* **patrón triangular**, *denotado por* $\mathcal{T}_{k,(i,j)}$, *al conjunto de puntos* $(a,b) \in \mathbb{U}$ *definido del siguiente modo:*

$$\mathcal{T}_{k,(i,j)} \leftrightharpoons \{(a,b) | 0 < (a-i) + (b-j) \leq k+1, i < a \leq (a+k), j < b \leq (b+k)\}$$

Donde $k \in \mathbb{Z}^+$, $(i,j) \in \mathbb{U}$ *es la* **célula origen** *del patrón triangular* $\mathcal{T}_{k,(i,j)}$.

DEFINICIÓN 2. *Sea el conjunto* $U_{k,(i,j)} \leftrightharpoons \{(i+1,j), \ldots, (i+k,j)\}$ *la* **frontera superior** *relacionada con la célula origen* (i,j)

DEFINICIÓN 3. *Sea el conjunto* $L_{k,(i,j)} \leftrightharpoons \{(i,j+1), \ldots, (i,j+k)\}$ *la* **frontera izquierda** *relacionada con la célula origen* (i,j)

Ahora podemos definir un triángulo R110, como:

DEFINICIÓN 4. *Llamaremos triángulo* $t_{k,(i,j)} \leftrightharpoons \{(i,j)\} \cup \mathcal{T}_{k,(i,j)} \cup U_{k,(i,j)} \cup L_{k,(i,j)}$

El estado de la célula origen (i,j) siempre debe ser 1. Luego, todas las células que pertenecen a la frontera superior $U_{k,(i,j)}$, o bien a la frontera izquierda $L_{k,(i,j)}$ son células de estado 1. Las células que pertenecen al patrón triangular $\mathcal{T}_{k,(i,j)}$ son células de estado 0.

En el resto de la tesis denotaremos por t_k a un triángulo $R110$ de tamaño $k \geq 0$, omitiendo la segunda parte del sub-índice: (i,j) cuando no sea necesaria, suponiendo que el par (i,j) ocurre en $\mathbb{U}$. Denotaremos con $\mathbb{T}$ al conjunto $\{t_k | k \geq 0\}$, y $T \subset \mathbb{T}$.

Para el desarrollo de los temas en los siguientes capítulos, será útil identificar las células que pertenecen a las fronteras superior, izquierda y diagonal. Un criterio de identificación arbitrario se muestra en la figura 18 izquierda.

En el resto de la tesis nos referiremos con el término **triángulo** a un triángulo R110; es decir, aquellos que se forman por la unión de los puntos que definen los patrones gráficos, que emergen debido a las evoluciones del autómata celular lineal regla 110. Esta notación será usada a menos que se indique otra.

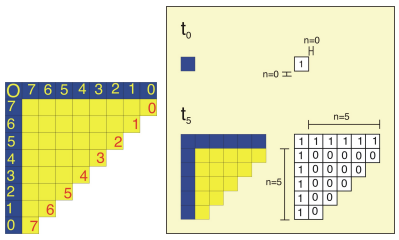

Figura 18. Izquierda: Identificación enumerada de las células que pertenecen a la frontera de un t_8. Derecha-arriba: Triángulo de tamaño 0 en la evolución del autómata celular regla 110. Derecha-abajo: Triángulo de tamaño 5.

4. Cubrimientos completos y Subcubrimientos

Las evoluciones de los espacios celulares gobernados con la regla de evolución 110, cubren un semiplano bidimensional discreto con triángulos de diferentes tamaños. Pero cuando consideramos los triángulos como elementos constructivos, también podemos cubrir regiones del plano; estos cubrimientos definen un subgrupo aditivo, lo que nos permite agregar nuevos triángulos y seguir teniendo un cubrimiento válido, como se muestra a lo largo de esta sección. Un problema clásico que está relacionado con este asunto es el problema del dominó.

4.1. El problema del dominó dentro del ámbito de los triángulos. A principios de la década de 1960, se dio a conocer un problema que llamaron "El problema del dominó" o la "Conjetura de Wang"[6], en donde se propuso la posibilidad de construir un conjunto finito de mosaicos que puedan cubrir el plano, esto significa que copias de esos mosaicos se pueden acomodar de manera que se pueda cubrir el plano infinito, con la condición de que las aristas contiguas de dos mosaicos sean del mismo color. Además, los mosaicos no pueden ser rotados ni reflejados.

En 1966 se publicó un documento [4] que muestra un conjunto de mosaicos con las condiciones que Wang propuso, pero que cubre el plano de manera aperiodica, lo que demuestra que la conjetura de Wang es equivocada y este problema es irresoluble.

El problema del dominó se puede describir en términos de los triángulos. Se tiene un plano infinito, que es cuadriculado en regiones del tamaño de una célula y un conjunto de triángulos; supondremos que se tiene un número ilimitado de copias de cada triángulo. Debemos buscar la manera de ensamblar las copias de los triángulos observando las reglas de los triángulos:

[6]Esta conjetura fue enunciada en 'An unsolvable problem on dominoes', Report BL-30, The Computation Laboratory, Harvard University, 1-5.

(1) Ningún triángulo se puede rotar o reflejar
(2) No puede haber un par de triángulos alineados por la parte superior

La primera condición es clara debido a que todos los triángulos ocurren con la misma orientación. La segunda condición tiene que ver con las condiciones locales de la regla 110, pero en particular una de ellas: $000 \rightarrow 1$ que se estudiará con detalle en la sección de cubrimientos prohibidos, en la página 48, en este mismo capítulo.

4.2. Definiciones relacionadas con los cubrimientos. Para estar de acuerdo en el significado de algunos términos, diremos que un par de puntos gráficos (células) $p_1, p_2 \in \mathbb{U}$ son **adyacentes** si son vecinos ortogonales. Si $p_1 = (x_1, y_1)$ y $p_2 = (x_2, y_2)$, decimos que p_1 es un **vecino ortogonal** [55] de p_2 si $|x_2 - x_1| + |y_2 - y_1| = 1$ (véase la figura 19).

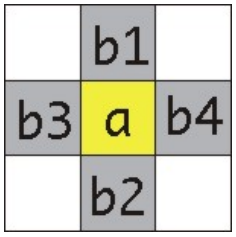

Figura 19. Descripción gráfica de una célula **a** y sus cuatro vecinos ortogonales **b1, b2, b3** y **b4**.

Definición 5. *Dados dos triángulos* $t_{k_1,(x_1,y_1)}, t_{k_2,(x_2,y_2)} \in \mathbb{T}$, *decimos que* $t_{k_1,(x_1,y_1)}$ *es* **adyacente** *a* $t_{k_2,(x_2,y_2)}$, *denotándolo como* $t_{k_1,(x_1,y_1)} \parallel t_{k_2,(x_2,y_2)}$ *si se cumple alguno de los siguientes casos:*

Adyacencia por la diagonal: $(x_1 \leq x_2),\ (y_1 \leq y_2)\ y\ (x_2 - x_1) + (y_2 - y_1) = k_1 + 2.$
Adyacencia por la frontera superior: $x_1 \leq x_2 \leq (x_1 + k_1)\ y\ (y_2 = y_1 - k_2 - 1).$
Adyacencia por la frontera izquierda: $(x_2 + k_2 = x_1 - 1)\ y\ (y_1 < y_2 \leq y_1 + k_1)$

4.2.1. *Cubrimientos completos del plano y subcubrimientos.*

Definición 6. *Llamamos* $T-$**cubrimiento** *a una región finita* $\Phi_T^{(m)}$ *en el plano discreto, que está compuesta de un conjunto de* m *triángulos definido como:* $\Phi_T^{(m)} \leftrightharpoons \{t_{k_1}, \ldots, t_{k'_m} | t_{\kappa_j} \in T, \kappa \in \mathbb{Z}^0,$ *si* $m > 1, \exists\, i \neq j : t_{k_i} \parallel t_{k'_j}\}$

En la definición anterior, el símbolo $\leftrightharpoons$, significa "definido cómo" o "igual por definición". Cuando es perfectamente claro el conjunto T usado, se puede omitir este símbolo y escribir simplemente **cubrimiento**.

Estos cubrimientos como han sido definidos, no necesariamente cumplen las reglas establecidas en la evolución del $acl(2,1)_{110}$, es decir, en la definición no se mencionan condiciones que hacen no válido un cubrimiento dentro del contexto de la regla 110. Posteriormente introduciremos algunas restricciones para que nuestras construcciones sean válidas.

Si $\mathbb{T}$ es el conjunto de todos los triángulos que ocurren en las evoluciones de la regla 110, entonces $\Phi_{\mathbb{T}}^{(\infty)}$ es un **$\mathbb{T}$-cubrimiento completo**, y es el espacio completamente cubierto por triángulos de $\mathbb{T}$, que no se translapan ni dejan huecos. El **T-cubrimiento completo** $\Phi_T^{(\infty)}$ es un espacio completamente cubierto por triángulos de $T \subseteq \mathbb{T}$.

Si es claro cuál es el conjunto de triángulos usado, no son necesarios los símbolos $\mathbb{T}$ ó T en las expresiones $\Phi_{\mathbb{T}}^{(\infty)}$ ó $\Phi_T^{(\infty)}$ respectivamente. En adelante, usaremos un conjunto T como el conjunto de triángulos, considerando que $T \subseteq \mathbb{T}$. Cuando se cubre un subespacio de área arbitrariamente finita lo llamamos simplemente "cubrimiento" (no cubrimiento completo).

4.3. El orden de un cubrimiento. El orden de un cubrimiento $\Phi^{(n)}$ compuesto por n triángulos, es denotado por $\|\Phi^{(n)}\| = n$ y es un elemento de la clase de cubrimientos que constan de n triángulos, que está definida como:

Definición 7. *Sea* $\mathcal{X}^{(1)} \rightleftharpoons \{\Phi_T^{(1)} | t \subset \mathbb{T}\}$ *la clase de cubrimientos de 1 triángulo, y progresivamente, si* $m > 1$, $\mathcal{X}^{(m)} \rightleftharpoons \{\Phi_T^{(1)} \cup \Phi_{T'}^{(2)} \cup \cdots \cup \Phi_{T''}^{(m)} | T, T', \ldots, T'' \subset \mathbb{T}\}$ *es la clase de cubrimientos de* m *triángulos.*

En particular $\mathcal{X}^{(0)}$ es la clase de cubrimientos que no tienen triángulos, el único elemento que contiene esta clase lo llamaremos p^λ el **elemento nulo**; $p^\lambda \in \mathcal{X}^\lambda$. Progresivamente, $\mathcal{X}^{(*)} \rightleftharpoons \mathcal{X}^\lambda \cup \mathcal{X}^{(1)} \cup \mathcal{X}^{(2)} \cup \cdots \cup \mathcal{X}^{(n)}$, para alguna $0 < n < \infty$ arbitraria. $\mathcal{X}^{(*)}$ es la clase de todos los cubrimientos de cualquier orden.

Desde el punto de vista del orden de los cubrimientos, dos cubrimientos p y q son equivalentes si son del mismo orden, es decir, si pertenecen a la misma clase, de manera que p y q son equivalentes siempre que tengan el mismo número de triángulos. La relación R "es del mismo orden que" es una *relación de equivalencia*:

(1) Un cubrimiento $p \in \mathcal{X}^{(n)}$ tiene n triángulos, luego pRp.
(2) Si $p, q \in \mathcal{X}^{(*)}$, ocurre que pRq,entonces $\|p\| = \|q\|$, entonces qRp.
(3) Si para $p, q, r \in \mathcal{X}^{(*)}$, se cumple que pRq y qRr, entonces p tiene el mismo número de triángulos que q, y q tiene el mismo número de triángulos que r, de manera que pRr.

5. La operación de concatenación gráfica define un monoide

Podemos crear una manera de unir diferentes cubrimientos de manera que formen nuevos cubrimientos con un mayor número de triángulos. La visión intuitiva de esta concatenación gráfica es **colocar adyacentemente dos células de las fronteras respectivas de los cubrimientos**.

La función de concatenación gráfica es definida en el dominio de los cubrimientos de cualquier número natural de triángulos como $\bowtie$: $\mathcal{X}_T^{(n)} \times \mathcal{X}_{T'}^{(m)} \to \mathcal{X}_{T\cup T'}^{(n+m)} \cup \{\nparallel\}$, donde el símbolo $\nparallel$ significa "indefinido".

Definición 8. *Sean $\Phi_T^{(n)} \in \mathcal{X}^{(n)}, \Phi_{T'}^{(m)} \in \mathcal{X}^{(m)}$ dos cubrimientos. Decimos que $\Phi_T^{(n)}$ está concatenado gráficamente con $\Phi_{T'}^{(m)}$, y lo escribimos como $\Phi_T^{(n)} \bowtie \Phi_{T'}^{(m)}$, cuando se cumple:*

$$\Phi_T^{(n)} \bowtie \Phi_{T'}^{(m)} = \begin{cases} \Phi_{T\cup T'}^{(n+m)} = \Phi_T^{(n)} \cup \Phi_{T'}^{(m)} & Si\ \exists\ t_{\kappa,(a,b)} \in \Phi_T^{(n)},\ t_{\kappa',(c,d)} \in \Phi_{T'}^{(m)} : \\ & t_{\kappa,(a,b)} \parallel t_{\kappa',(c,d)}. \\ \nparallel & En\ otro\ caso \end{cases}$$

El resultado de $\Phi_T^{(n)} \bowtie \Phi_{T'}^{(m)}$ es otro cubrimiento que tiene triángulos del conjunto $T \cup T'$ y el nuevo cubrimiento pertenece a la clase $\mathcal{X}^{(m+n)}$.

Esta función de concatenación gráfica tiene las siguientes propiedades:

(1) **Cerradura**: Para $x, y, z \in \mathcal{X}^{(*)}$, si $x \in \mathcal{X}^{(i)}$ y $y \in \mathcal{X}^{(j)}$, y se tiene que $x \bowtie y = z$, entonces z tiene orden $i + j \in \mathbb{N}$.
(2) **Asociativo**: Para $x, y, z \in \mathcal{X}^{(*)}$, si $(x \bowtie y) \bowtie z = x \bowtie (y \bowtie z) = x \bowtie y \bowtie z$.
(3) **Elemento idéntidad**: Existe un elemento, $p^\lambda \in \mathcal{X}^\lambda$ tal que si $x \in \mathcal{X}^{(*)}$, $x \bowtie p^\lambda = p^\lambda \bowtie x = x$.

Entonces el conjunto $\mathcal{X}^{(*)}$, junto con la operación de contatenación gráfica, definen el monoide $(\mathcal{X}^{(*)}; \bowtie)$ de cubrimientos en los triángulos generados en las evoluciones del $acl(2,1)_{110}$. La función de la operación $\bowtie$ se muestra en la tabla 7.

Podemos definir la operación **retirar gráficamente**, que es lo contrario de la concatenación gráfica porque ahora se trata de retirar cubrimientos de otro cubrimiento. La función "retirar gr]'aficamente" está definida por un par de cubrimientos y el resultado es un nuevo cubrimiento: $\mathcal{X}^{(n)} \times \mathcal{X}^{(m)} \to \mathcal{X}^{(n-m)}$, cuando $n \geq m$.

$\bowtie$	$\mathcal{X}^{\lambda}$	$\mathcal{X}^{(1)}$	$\mathcal{X}^{(2)}$	$\mathcal{X}^{(3)}$	$\ldots$	$\mathcal{X}^{(m)}$
$\mathcal{X}^{\lambda}$	$\mathcal{X}^{\lambda}$	$\mathcal{X}^{(1)}$	$\mathcal{X}^{(2)}$	$\mathcal{X}^{(3)}$	$\ldots$	$\mathcal{X}^{(0+m)}$
$\mathcal{X}^{(1)}$	$\mathcal{X}^{(1)}$	$\mathcal{X}^{(2)}$	$\mathcal{X}^{(3)}$	$\mathcal{X}^{(4)}$	$\ldots$	$\mathcal{X}^{(1+m)}$
$\mathcal{X}^{(2)}$	$\mathcal{X}^{(2)}$	$\mathcal{X}^{(3)}$	$\mathcal{X}^{(4)}$	$\mathcal{X}^{(5)}$	$\ldots$	$\mathcal{X}^{(2+m)}$
$\mathcal{X}^{(3)}$	$\mathcal{X}^{(3)}$	$\mathcal{X}^{(4)}$	$\mathcal{X}^{(5)}$	$\mathcal{X}^{(6)}$	$\ldots$	$\mathcal{X}^{(3+m)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ldots$	$\vdots$
$\mathcal{X}^{(n)}$	$\mathcal{X}^{(n)}$	$\mathcal{X}^{(n+1)}$	$\mathcal{X}^{(n+2)}$	$\mathcal{X}^{(n+3)}$	$\ldots$	$\mathcal{X}^{(n+m)}$

Cuadro 7. Operación de concatenación gráfica del monoide de cubrimientos en $acl(2,1)_{110}$ definido en $(\mathcal{X}^{(*)};\bowtie)$

DEFINICIÓN 9. *Sean $\Phi_T^{(n)} \in \mathcal{X}^{(n)}, \Phi_{T'}^{(m)} \in \mathcal{X}^{(m)}$ dos cubrimientos. Decimos que $\Phi_{T'}^{(m)}$ es retirado gráficamente de $\Phi_T^{(n)}$, y lo escribimos como $\Phi_T^{(n)} \ltimes \Phi_{T'}^{(m)}$, cuando se cumple:*

$$\Phi_T^{(n)} \ltimes \Phi_{T'}^{(m)} = \begin{cases} \Phi_T^{(n-m)} = \Phi_T^{(n)} \diagdown \Phi_{T'}^{(m)} & \textit{Si } \Phi_{T'}^{(m)} \subseteq \Phi_T^{(n)} \\ \nparallel & \textit{En otro caso} \end{cases}$$

Es fácil observar que la función "retirar gráficamente" no es conmutativa. Aún más, no es asociativa, si $\Phi_T^{(n_1)}, \Phi_T^{(n_2)}, \Phi_T^{(n_3)}$ son cubrimientos tales que $\Phi(n_1)_T \bowtie \Phi_T^{(n_2)} \bowtie \Phi_T^{(n_3)} \in \mathcal{X}^{(n_1+n_2+n_3)}$ y además, $n_1 \leq n_2 \leq n_3$, hacer $(\Phi_T^{(n_1)} \ltimes \Phi_T^{(n_2)}) \ltimes \Phi_T^{(n_3)}$ no siempre es lo mismo que $\Phi_T^{(n_1)} \ltimes (\Phi_T^{(n_2)} \ltimes \Phi_T^{(n_3)})$ puesto que puede ocurrir que $\Phi_T^{(n_1)} \nparallel (\Phi_T^{(n_2)} \ltimes \Phi_T^{(n_3)})$.

Monoide bajo la concatenación gráfica de cubrimientos La operación de concatenación gráfica definida en el conjunto de todos los triángulos, incluyendo el cubrimiento nulo p^{λ}, define un monoide, puesto que es un subgrupo con elemento neutro.

- $p \bowtie \lambda \eqsim \lambda \bowtie p \eqsim p$. El símbolo $\eqsim$ significa "gráficamente igual a".
- $p \bowtie q \eqsim r$, donde el orden de r es la suma de los ordenes de p y de q.

6. El conjunto de cubrimientos prohibidos

El resultado de concatenar gráficamente cubrimientos puede ser un nuevo cubrimiento que jamás ocurra en las evoluciones de la regla 110, por esto es necesario considerar un conjunto de cubrimientos prohibidos. Un cubrimiento es elemento de este conjunto si no cumple alguna de las condiciones de vecindad del autómata celular. Para asegurar que el comportamiento de la regla 110 se conserve dentro del cubrimiento, después de concatenar gráficamente un par de cubrimientos, se deben verificar las siguientes condiciones:

6.1. Los triángulos no deben alinearse por su frontera superior. Alinear los triángulos por el lado superior de cada uno de ellos, significa no cumplir una de las reglas de evolución del autómata celular regla 110, la regla $< 111 > \rightarrow 0$ del conjunto de reglas (vea la página 17). Esta regla dice, "si ocurren tres células juntas de estado 1, la célula central en el renglón siguiente (en el paso de tiempo siguiente) debe de tener estado 0".

La condición anterior ocurre en la frontera superior de cada triángulo. Es allí cuando hay posibilidad de que ocurran secuencias de células $< 111 >$, precisamente cuando una secuencia $< ..11 >$ de la frontera superior de un triángulo es adyacente a la secuencia $< 1.. >$ de la frontera superior otro triángulo. Sin embargo la frontera izquierda del segundo triángulo hace que la célula central también tenga estado 1, completando de esa manera la relación $< 111 > \rightarrow 1$ que jamás ocurre en el conjunto de reglas de evolución local del autómata celular lineal regla 110.

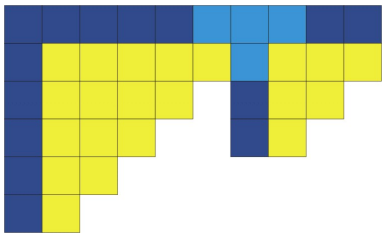

Figura 20. Alinear dos triángulos por sus fronteras superiores incumple la regla $< 111 > \rightarrow 0$ de la regla de evolución local del $acl(2,1)_{110}$

6.1.1. *El $i-$ésimo triángulo en la diagonal.* Una situación especial ocurre al concatenar gráficamente los triángulos, por la diagonal de uno de ellos. Cuando se intenta concatenar gráficamente un triángulo de tamaño i en la $i-$ésima posición de la diagonal de otro triángulo[7].

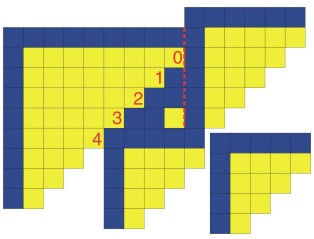

Figura 21. Al concatenar gráficamente un triángulo t_i en la $i-$ésima posición de un triángulo t_n; con $n \geq i$, se generará un cubrimiento prohibido.

[7]Una enumeración de las posiciones válidas de la diagonal se ha dado en la página 44

Un cubrimiento prohibido puede ocurrir cuando después de haber concatenado gráficamente algunos otros triángulos, aun cuando se halla hecho de manera legal. El cubrimiento ilegal ocurre cuando tenemos la necesidad de concatenar gráficamente un triángulo que inevitablemente viola la regla de no-alineación por la parte superior, como se muestra en la figura 21. El cubrimiento prohibido ocurre porque un triángulo de tamaño n tiene $n+1$ células en su frontera superior, pero solamente n células en la diagonal, y al concatenarlo gráficamente en la $n-$ésima posición de la diagonal de otro triángulo, ocurre un exceso de 1 célula. Ese exceso de 1 célula es determinante pues los triángulos tienen 2 células en su vértice inferior. De manera que una célula no es suficiente para colocar cualquier otro triángulo.

Al intentar colocar otro triángulo, cualquiera que sea, solamente se tiene una posibilidad y esa es ilegal, como se aprecia en la figura 21, lo que ocasiona el no cumplimiento de la regla de no-alineación, ocaisonando un cubrimiento prohibido.

6.1.2. *La unicidad del espacio ocupado.* Otra característica que podemos observar al impedir la alineación por la frontera superior, es que el espacio que ocupa algún triángulo t_n no se puede ocupar por la concatenación gráfica de otros triángulos menores.

Esto fácilmente se puede apreciar, si observamos que la frontera superior del cubrimiento generado por la concatenación gráfica de dos triángulos, no forma una línea recta, debido precisamente a la restricción de la no-alineación por la frontera superior.

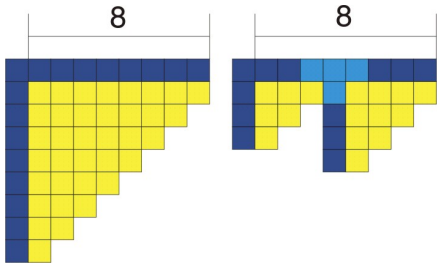

Figura 22. El espacio ocupado por un triángulo no se puede ocupar por la concatenación gráfica de otros triángulos menores.

6.2. Necesidad de verificación de las condiciones locales de la regla 110. Para determinar la legalidad de un cubrimiento, de acuerdo a las condiciones locales de la regla 110, se deben verificar las células que rodean el triángulo; las células que componen el cubrimiento se deben poder generar a partir de las evoluciones locales de la regla 110.

Esta característica se debe verificar en todo el cubrimiento. En los cubrimientos de orden 1, es decir en los triángulos, no es problema, porque ya se ha visto la relación que existe entre la regla de evolución local y los triángulos (vea la página 35). Sin embargo algun error puede

presentarse en las zonas de concatenación gráfica. De manera que cuando el cubrimiento es de orden mayor que uno, es necesario verificar las fronteras.

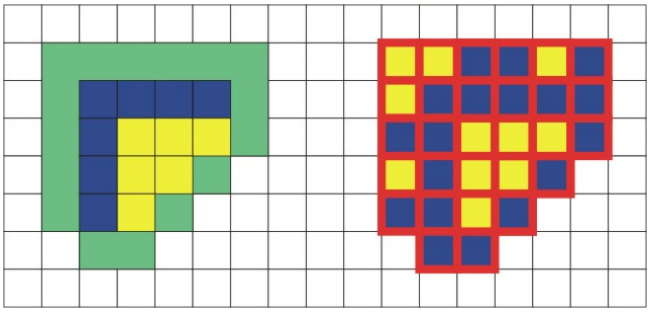

Figura 23. Las células en las fronteras del cubrimiento son comprobadas bajo las reglas de vecindades en la regla de evolución local.

Una necesidad de verificación más detallada es necesaria particularmente en la parte superior del cubrimiento, donde las células forman secuencias $< 111\ldots1 >$, porque hay más de una configuración que puede producirlas, lo que implica que se necesita elegir una de ellas.

Cuando es posible tener diferentes secuencias antecesoras que generen la misma secuencia que será la frontera superior del cubrimiento, el número de cubrimientos diferentes que se pueden lograr, es el mismo que el número de opciones de crear una subsecuencia de células a partir de secuencias antecesoras.

En la diagonal y en la frontera izquierda la situación es diferente. Se tienen 3 de las 4 células que definen una regla de evolución (tres de la vecindad y una del resultado). Las 3 células que se tienen son 2 de la vecindad y el resultado, de manera que es muy sencillo averiguar el estado de la tercera célula de la vecindad.

Una vez que podemos determinar la legalidad de un cubrimiento, podemos comparar cubrimientos y determinar qué tan diferentes pueden ser. Establecer una relación de similaridad es muy útil si tenemos un cubrimiento modelo y queremos construir otros cubrimientos, comparando el recién hecho con el modelo, determinando de este modo cuándo se ha obtenido una construcción satisfatoria respecto del modelo elegido.

7. Los triángulos y su relación con el problema del máximo número de besos

Para observar las regularidades que muestran los triángulos al rodear otro triángulo, estudiaremos este antiguo problema de geometría, en el entorno de los triángulos.

El problema del máximo número de besos fue descrito originalmente por Isaac Newton y David Gregory en 1694 [8] en la Universidad de Cambridge, en donde el problema es determinar el máximo número de hiperesferas del mismo tamaño que se "besan"[8] mutuamente. Este problema se puede entender en términos de triángulos de la siguiente manera:

"Dado un triángulo llamado el *triángulo central*, ¿Cuál es el máximo número de triángulos que besan (son adyacentes) al mismo tiempo y de manera legal al triángulo central?" Para dar respuesta al problema del máximo número de besos en los triángulos, consideraremos por separado los conjuntos de triángulos que tocan al triángulo central en cada una de sus fronteras y en los vértices. Estos conjuntos son los triángulos que besan al triángulo central por cada una de sus tres fronteras.

1. **Número de besos en la frontera diagonal:** El tamaño de los triángulos está determinado por el número de puntos gráficos equivalentes a las células de estado 0, alineados horizontalmente, sin incluir algún otro punto gráfico.

   La longitud de la frontera diagonal de un triángulo de tamaño n es también de n células. Cada uno de los puntos gráficos de la diagonal de un triángulo puede besar solamente a un triángulo, de manera que en una diagonal de n células[9] hay exactamente n triángulos. Ocurre también que la primera posición de la diagonal, es ocupada por un triángulo que no es adyacente por su vértice principal (el superior izquierdo), sino que es adyacente por su frontera izquierda; aun así cuenta como un triángulo adyacente, en la figura 24 se describen ejemplos de besos en la frontera diagonal de un triángulo que se muestra en tonos más ténues que el resto de los triángulos.

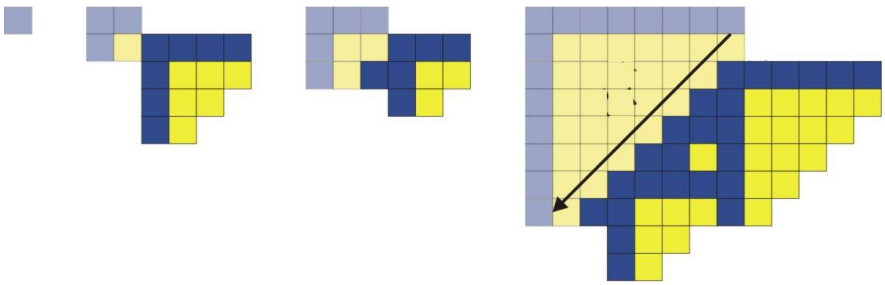

Figura 24. Ejemplos de triángulos que tocan a otro triángulo por la frontera diagonal.

[8]Aquí el término "besar" se ha tomado del lenguaje usado en el juego de billar y es equivalente a "tocar".
[9]Los términos "celula" y punto gráfico se usan de manera indistinta.

2. **Número de besos en la frontera superior:** Para el caso de la frontera superior, el triángulo central es adyacente con la parte inferior de cada triángulo, y la longitud de la superficie de toque es de 2 células, excepto cuando se trata del triángulo t_0 que solo tiene una célula.

   Debido a que la frontera superior de cada triángulo es constituída por células de estado 1, se pueden construir secuencias de células a partir de las vecindades que al aplicar la regla de evolución 110, generen como resultado una célula de estado 1. Las secuencias obtenidas son las partes inferiores de los triángulos que tocan al triángulo central y que se extienden espacialmente hacia arriba; podemos notar que:
   - Las subsecuencias obtenidas $< 1, 0 >$ corresponden a la parte inferior de algún triángulo de tamaño mayor o igual que 1. Porque el primer 1 de la secuencia corresponde a la frontera izquierda y el siguiente 0 es parte del cuerpo del triángulo.
   - Las subsecuencias $< 1, 1, 0 >$ significan que a un triángulo t_0 le sigue algún otro de tamaño mayor que 0.
   - El tercer caso son subsecuencias de la forma $< 1, 0, 1 >$ un triángulo mayor o igual que el t_1 es seguido por otro triángulo t_0.
   - Las subsecuencias $< 1, 1, 1 >$ no son permitidas, puesto que la regla de evolución local para esta vecindad produce un 0, que no es parte de la frontera superior.

   La dirección adecuada de contar estas secuencias es de derecha a izquierda, de manera que las secuencias $< 0, 1 >$ corresponden a triángulos mayores que el t_0, para efectos del conteo suman 1 al total de triángulos adyacentes por el lado superior. Se puede estimar como $\lceil \frac{n}{2} \rceil$.

3. **Número de besos en la frontera izquierda:** En la frontera izquierda sucede algo similar que en la frontera superior; basados en ejemplos representativos del comportamiento de los triángulos dentro de las evoluciones de la regla 110 se tienen las siguientes observaciones:
   - Un triángulo t_0 ocurre frecuentemente en la posición 1 de la diagonal de un triángulo, esto significa que se puede encontrar una secuencia $< 1, 0, 1 >$ leída de abajo hacia arriba.
   - El lado izquierdo de un triángulo no tiene tantas restricciones como en el lado superior y se tiene oportunidad de colocar un triángulo más.
   - De manera general, por cada 3 triángulos mayores o iguales que el t_1, debe haber un t_0, esto nos da un parámetro de 7 células.

   Considerando las observaciones anteriores, el número máximo de triángulos que se pueden colocar de manera adyacente a un triángulo de tamaño n es posible estimarlo con la relación $\lceil \frac{4n}{7} \rceil$

4. **Número de besos en los vértices:** En los vértices principal e inferior es posible colocar un sólo triángulo, pero en el vértice derecho es posible colocar 1 o hasta 2 triángulos, de manera que debemos agregar 3 o 4 triángulos más.

5. **La cuenta final:** Considerando el número de besos en cada uno de los tres lados, más los besos en los vértices, se puede dar una respuesta al problema del máximo número de besos, con la expresión siguiente donde n es el tamaño del triángulo central y K_n denota el número de "besos" para un triángulo de tamaño n. Debemos agregar un error de ± 1 debido al punto 4:

$$K_n = n + \lceil \frac{n}{2} \rceil + \lceil \frac{4n}{7} \rceil + 3 \tag{2}$$

8. Espacios métricos de cubrimientos

El espacio de cubrimientos son espacios métricos. Para mostrar la existencia de un espacio métrico se necesita un conjunto de elementos y una métrica con dominio en los pares de elementos del conjunto definido [28]. Vamos a considerar dos métricas que nos serán útiles en diferentes situaciones.

8.1. Métrica básica. Para establecer la comparación de dos cubrimientos nos basaremos en sus diferencias, consideraremos un mismo punto de referencia en los elementos comparados, al que llamaremos "centro". Las comparaciones se deberán hacer con un conjunto de vectores que indicarán los triángulos a comparar. Estos vectores tienen su origen en el centro y señalan triángulos en su célula origen[10], si los triángulos señalados son iguales en ambos cubrimientos, la suma de sus diferencias no es afectada, pero si los triángulos son distintos, aumenta el valor.

Las diferencias deben ser más significativas en el centro de los cubrimientos y cada vez menos importantes a medida que las diferencias se localicen más alejadas del centro. Lo que nos dará la oportunidad de comparar cubrimientos de tamaños arbitrariamente grandes.

Definición 10. *Llamaremos* **métrica básica** *a la triada* $(\mathcal{X}^{(*)} \times \mathcal{X}^{(*)}, \delta_b, \mathcal{V})$*, donde:* $\mathcal{X}^{(*)} \times \mathcal{X}^{(*)}$ *es un par ordenado de cubrimientos,* δ_b *es una función de comparación basada en diferencias, y* $\mathcal{V}$ *es un conjunto de vectores con entradas en* $\mathbb{U}$*, considerando como origen la célula elegida como centro.*

[10]Marcada con una **O** en el vértice recto. Vea la página 44.

La distancia δ_b con estructura $\delta_b : \mathcal{X}^{(*)} \times \mathcal{X}^{(*)} \to [0, \infty)$ está definida como:

$$\delta_b(x, y) = \sum_{v \in \mathcal{V}} \frac{\delta'(p, q)}{2^{|v|}} \tag{3}$$

Donde $|v|$ es la norma del vector v, y la función auxiliar δ' cuenta las diferencias y está definida como:

$$\delta'(p, q) = \begin{cases} 1 & si\ p \neq q \\ 0 & si\ p = q \end{cases} \tag{4}$$

Donde $p, q \in \mathcal{X}^{(1)}$ son dos triángulos comparados.

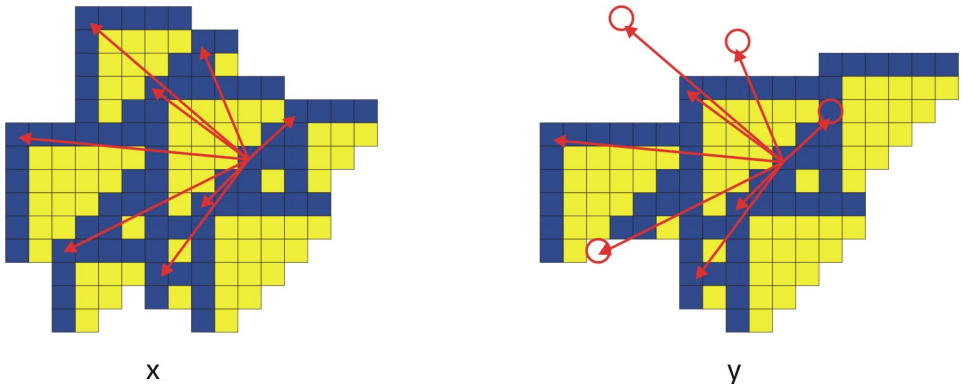

Figura 25. Uso de la métrica básica en dos cubrimientos similares.

El origen para el conjunto de vectores $\mathcal{V}$ se elige en la célula origen del triángulo que inició el cubrimiento.

Como ejemplo de cómo utilizar esta métrica sean x y y los dos cubrimientos que se muestran en la figura 25, con el conjunto de vectores:

$$\begin{aligned} \mathcal{V} \ \leftrightharpoons\ & \{(0,0),(2,2),(-5,-4), \\ & (-2,-2),(-8,-4),(-10,1),(-4,3),(-7,6),(-2,5)\} \end{aligned}$$

El centro está ubicado en un t_1 como también lo muestra la figura. En la figura 25 derecha, se muestra en un círculo las diferencias que ocurren en los cubrimientos x y y. El orden de análisis de cada vector será en el sentido de las manecillas del reloj.

$$\begin{aligned}\delta_b(x,y) = & \frac{1}{2^{|(2,2)|}} + \frac{1}{2^{|(-8,-4)|}} + \frac{1}{2^{|(-7,6)|}} + \frac{1}{2^{|(-2,5)|}} \\ = & \frac{1}{2^{2.828}} + \frac{1}{2^{8.944}} + \frac{1}{2^{9.219}} + \frac{1}{2^{5.3851}} \\ = & \ 0.1684\end{aligned}$$

$\mu-$semejanza gráfica El parámetro $\mu \in \mathbb{Z}^+$ es el *umbral de diferencia* que indica el grado máximo de diferencia permitido para que dos cubrimientos sean semejantes. Aún cuando dos cubrimientos p y q sean difentes en algunos vectores $v \in \mathcal{V}$, si la diferencia es menor o igual que el umbral μ, entonces deberán ser considerados en la misma clase de semejanza. Así entonces:

$$p \approx_\mu q \text{ si } \delta_b(p,q) \leq \mu \tag{5}$$

La relación de $\mu-$semejanza induce una relación de equivalencia cuando $\mu < 1$, puesto que para cualesquiera cubrimientos $p, q, r \in \mathcal{X}^{(*)}$:

- $p \approx_\mu p$ para cualquier $\mu \geq 0$, puesto que $\delta_b(p,p) = 0$
- Si $p \approx_\mu q$, entonces $q \approx_\mu p$, porque $\delta_b(p,q) = \delta_b(q,p)$
- Si $p \approx_\mu q$ y $q \approx_\mu r$, entonces $p \approx_\mu r$ porque el valor de $\mu < 1$ cuando los centros de los cubrimientos son los mismos.

Cuando es perfectamente claro y sobreentendido cuál es el valor del umbral de diferencias utilizado, puede omitirse en la escritura.

$\mu-$igualdad gráfica. Para dos cubrimientos $p, q \in \mathcal{X}^{(*)}$, p es gráficamente igual que q si el umbral de diferencias es $\mu = 0$. Cuando $\mu = 0$ se quiere verificar la igualdad gráfica para todos los triángulos señalados por los vectores del conjunto $\mathcal{V}$ y se ha encontrado que en todos son iguales. Podemos notar que para el cubrimiento x mostrado en la figura 25 $x \eqcirc x$ es verdadero, y para cualquier cubrimiento $p \in \mathcal{X}^{(*)}$, también $p \eqcirc p$ es verdadero.

8.1.1. *La métrica básica es un métrica válida.* Se mostrarán las propiedades de reflexibidad, transitividad y desigualdad del triángulo para la métrica básica δ_b con los cubrimientos $x, y, z \in \mathcal{X}^{(*)}$:

$\delta_b(x,x) = 0$ **:** Debido a que el conjunto de vectores $\mathcal{V}$ es el mismo en los cubrimientos comparados; para cada vector $v \in \mathcal{V}$ se tiene que $p_v \eqcirc p_v$ y $\delta'(p_v, p_v) = 0$ de manera

que la suma de las diferencias es 0.

$\delta_b(x,y) = \delta_b(y,x)$ **:** En primer lugar debemos notar que las diferencias se muestran en aquellos triángulos que son diferentes, es decir aquellos sitios en los que para algún vector $v \in \mathcal{V}$ se tiene que $a_v \eqsim b_v$ es falso. Luego, por transitividad $a_b \eqsim b_v = b_v \eqsim a_v$ y así se tiene que $b_v \eqsim a_v$ es falso en los sitios en que $a_v \eqsim b_v$ lo sea.

$\delta_b(x,z) \leq \delta_b(x,y) + \delta_b(y,z)$ **:** Para mostrar este tercer punto, consideraremos el caso en que los tres cubrimientos sean diferentes en todos los vectores de $\mathcal{V}$:

$\delta_b(x,z) = \frac{1}{1} + \frac{k_1}{2^{|v_1|}} + \cdots + \frac{k_{n-1}}{2^{|v_{n-1}|}}$, y de manera similar

$\delta_b(x,y) = \frac{1}{1} + \frac{k'_1}{2^{|v_1|}} + \cdots + \frac{k'_{n-1}}{2^{|v_{n-1}|}}$, y

$\delta_b(y,z) = \frac{1}{1} + \frac{k''_1}{2^{|v_1|}} + \cdots + \frac{k''_{n-1}}{2^{|v_{n-1}|}}$

Con todas las $k_i, k'_i, k''_i \geq 1$

Como los cubrimientos son diferentes en todos los vectores v_i, se tiene que:

$\delta_b(x,y) + \delta_b(y,z) = \frac{1+1}{2^{|v_0|}} + \frac{k'_1+k''_1}{2^{|v_1|}} + \cdots + \frac{k'_{n-1}+k''_{n-1}}{2^{|v_{n-1}|}}$

De manera que:
$\frac{1}{1} + \frac{k_1}{2^{|v_1|}} + \cdots + \frac{k_{n-1}}{2^{|v_{n-1}|}} < \frac{1+1}{2^{|v_0|}} + \frac{k'_1+k''_1}{2^{|v_1|}} + \cdots + \frac{k'_{n-1}+k''_{n-1}}{2^{|v_{n-1}|}}$

porque $k_i = k'_i = k''_i$ $i = 0 \ldots n-1$ y así.

$\delta_b(x,z) \leq \delta_b(x,y) + \delta_b(y,z)$

8.2. Métrica estricta. Una extensión de la métrica básica, es cuando consideramos el conjunto de vectores estricto $\mathbb{V}$ que es definido como el conjunto de vectores que señala a todos y a cada uno de los triángulos de un cubrimiento. De manera que la métrica estricta es definida como la métrica básica, pero considerando ahora el conjunto $\mathbb{V}$:

$$\delta(x,y) = \sum_{v \in \mathbb{V}} \frac{\delta'(p,q)}{2^{|v|}} \tag{6}$$

Con $\delta'(p,q)$ definida como antes.

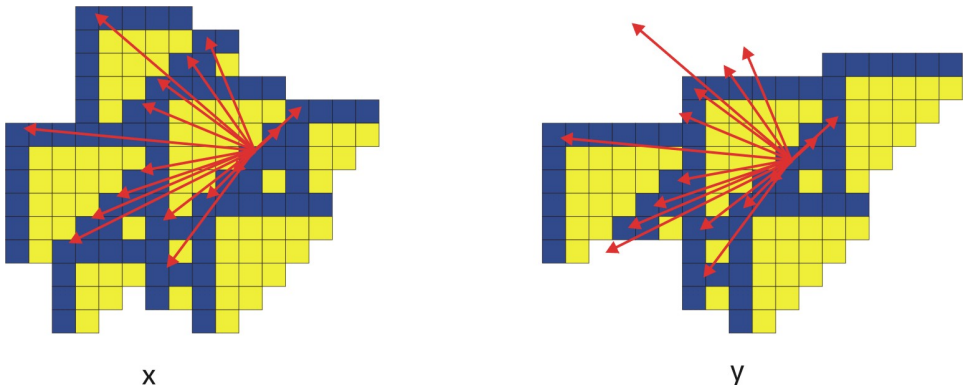

Figura 26. métrica estricta aplicada a dos cubrimientos considerando las diferencias señaladas por los vectores de $\mathbb{V}$.

Para el ejemplo de la figura 26, el conjunto de vectores es

$$\begin{aligned}\mathbb{V} \leftrightharpoons \; & \{(1,1),(2,2),(-1,-1),(-4,-5),(-4,-3),(-4,-3),\\ & (-8,-4),(-7,-3),(-6,-2),(-5,-1),(-10,1),\\ & (-5,2),(-4,3),(-7,6),(-3,4),(-2,5)\}\end{aligned}$$

Los vectores que señalan una diferencia (figura 27) son:

$$\begin{aligned}&\{(1,1),(2,2),(-1,-1),(-4,-3),(-8,-4),(-7,-3),\\ &(-6,-2),(-5,-1),(-5,-2),(-7,6),(-3,4),(-2,5)\}\end{aligned}$$

La distancia entre los cubrimientos x y y ahora es: $\delta(x,y) = 0.37521+0.14078+0.37521+0.03125+0.00203+0.00509+0.01247+0.02917+0.02392+0.00167+0.03125+0.02392 = 1.05197$

La diferencia entre los resultados obtenidos con las dos métricas: $\delta_b(x,y) = 0.1684$ y $\delta(x,y) = 1.05197$, se debe a que se han considerado todas las posibles diferencias. Esas diferencias se establecen en el conjunto de vectores considerado en cada caso.

La mínima y máxima diferencia entre cubrimientos comparados, se puede calcular considerando el caso cuando hay cero diferencias y el caso cuando hay todas las diferencias. El primer caso la mínima diferencia es cero. El otro caso se puede estimar suponiendo un conjunto estricto de vectores que tienen como origen el centro del cubrimiento.

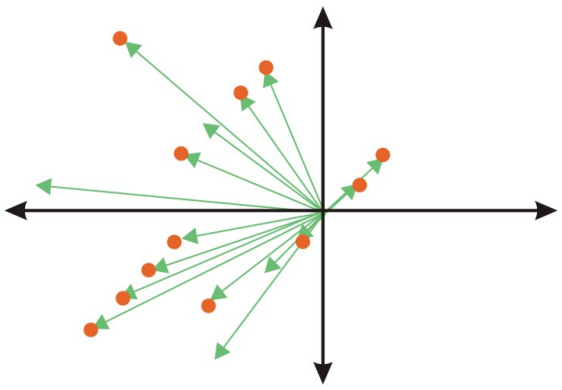

Figura 27. Conjunto estricto de vectores $\mathbb{V}$.

9. Conclusiones

En este capítulo se ha discutido lo siguiente:

- La capacidad de un autómata celular para hacer computación universal, es un campo de estudio de los comportamientos emergentes en las evoluciones de los autómatas celulares.

  Este estudio empezó desde el primer modelo de autómata celular, creado por John von Neumman, en donde reproducía el comportaiento de la máquina de Turing usando su "constructor universal".

- La computación universal se relaciona con autómatas celulares desde dos puntos de vista:

  (1) Definir un autómata celular con el propósito de que haga computación universal.

  Ejemplos de modelos de autómatas celulares que usan este punto de vista son el mismo modelo de von Neumann y el modelos de Alvy Ray Smith.

  (2) Que la capacidad de un autómata celular de hacer computación universal, es el resultado de su comportamiento emergente.

  El ejemplo más significativo es el modelo "*life*" de Conway, en donde se puede dar una configuración inicial global adecuada para que el autómata celular se

comporte como una máquina de Turing; o bien, se pueden simular las compuertas lógicas básicas.

- La regla 110 ofrece un comportamiento emergente complicado, en donde se observan patrones gráficos con forma de triángulos; y donde aparentemente esos triángulos interactúan entre sí. Estos comportamientos dieron la pauta para establecer la conjetura de que era posible hacer computación universal.

- Se han definido 2 funciones para agregar y quitar triángulos a un cubrimiento: la concatenación gráfica y la concatenación gráfica inversa.

  Con los triángulos como elementos de un conjunto, y la función de concatenación gráfica se define un semigrupo.

- Los cubrimientos con la función de concatenación gráfica definen un monoide.

  Agregando el cubrimiento nulo, se puede definir un monoide, sin embargo, el solo hecho de ser un semigrupo es sufieciente para considerar el problema de la palaba en semigrupos.

- Se ha mostrado que el espacio ocupado por un triángulo no puede ser llenado por otros triángulos de diferente tamaño.

- Se ha definido cómo comparar cubrimientos en base a métricas, para determinar su pertenencia a las clases de cubrimientos y su igualdad.

  En dependencia de un umbral de suficiencia, es posible considerar dos cubrimientos dentro del mismo conjunto de soluciones. A medida que el umbral de suficiencia se acerque más al valor cero, se pretende que los cubrimientos comparados sean cada vez más parecidos.

Capítulo 3

Mosaicos y el 18^{vo} problema de Hilbert

1. Resumen

En este capítulo se establecen de manera formal los conceptos de "mosaico" y "mosaico propio", verificando primero que sean figuras que no tienen huecos y que las fronteras de esas figuras estén bien determinadas; luego, para que sean mosaicos propios, se verifica que los mosaicos pueden cubrir el plano.

Se define la operación de traslado de mosaicos propios. Esta función nos va a permitir crear espacios vectoriales y crear un reticulado.

Se hace una relación con el 18^{vo} problema de Hilbert, los números de Heesch y los criterios de Conway para cubrir el plano con copias del mismo mosaico.

2. Concepto formal de mosaico

Una manera de distinguir los cubrimientos creados a partir de los patrones gráficos generados por el autómata celular regla110, es agruparlos en una clase especial, que en adelante llamaremos la clase de los **mosaicos**, lo cual concuerda con las definiciones preestablecidas en documentos relacionados con la teoría de mosaicos [19, 52, 58], y con la teoría general de topología algebraica [28, 29]. Dentro de esta clase de cubrimientos definirermos la clase de los mosaicos propios.

PROPOSICIÓN 1. *Un patrón gráfico,* $\Phi_{\mathbb{T}}^{(m)} \subset \mathcal{X}^{(m)}$; $m < \infty$ *es un mosaico si cumple con las siguientes dos condiciones:*

(1) $\Phi_{\mathbb{T}}^{(m)}$ *es compacto* [52].
(2) $\Phi_{\mathbb{T}}^{(m)}$ *es igual a la cerradura de su interior* [52].

Para mostrar esto, primero mostraremos que cada triángulo define un espacio topológico y entonces que cada triángulo es un mosaico, luego por extensión, mostraremos que los cubrimientos también son mosaicos, al unir estos espacios.

2.1. Espacios topológicos relacionados. Sea $(t_{k,(i,j)}; \mathbb{P}(t_{k,(i,j)}))$ el espacio topológico formado por un triángulo $t_{k,(i,j)}$ de tamaño k en alguna posición $(i,j) \in \mathbb{U}$, junto con el conjunto potencia de $t_{k,(i,j)}$. Recordemos que cada triángulo está definido como un conjunto de puntos gráficos en $\mathbb{U}$ (página 43).

Por tratarse de la topología discreta de $t_{k,(i,j)}$, es fácil ver que cumple con las tres condiciones de un espacio topológico, es decir:

(1) $\emptyset \in \mathbb{P}(t_{k,(i,j)}), \quad t_{k,(i,j)} \in \mathbb{P}(t_{k,(i,j)})$.
Porque $\emptyset \cap t_{k,(i,j)} = \emptyset \in \mathbb{P}(t_{k,(i,j)})$, y también $t_{k,(i,j)} \cap \mathbb{P}(t_{k,(i,j)}) = t_{k,(i,j)} \in \mathbb{P}(t_{k,(i,j)})$.

(2) Si $u_1, u_2 \in \mathbb{P}(t_{k,(i,j)})$, entonces $u_1 \cap u_2 \in \mathbb{P}(t_{k,(i,j)})$.
Como $u_1 \in \mathbb{P}(t_{k,(i,j)})$ y $u_2 \in \mathbb{P}(t_{k,(i,j)})$, entonces $t_{k,(i,j)} - u_1 \in \mathbb{P}(t_{k,(i,j)})$ y también $t_{k,(i,j)} - u_2 \in \mathbb{P}(t_{k,(i,j)})$ y son conjuntos finitos porque $t_{k,(i,j)}$ es finito. Luego $(t_{k,(i,j)}) - u_1) \cup (t_{k,(i,j)}) - u_2) = t_{k,(i,j)} - (u_1 \cap u_2)$ también es finito, y pertenece a $\mathbb{P}(t_{k,(i,j)})$.

(3) Si $u_1, u_2, \ldots, u_l \in \mathbb{P}(t_{k,(i,j)})$, entonces $u_1 \cup u_2 \cup \cdots \cup u_l \in \mathbb{P}(t_{k,(i,j)})$, con $l \in J$ un conjunto numerable.
Aquí usamos el hecho de que

$$t_{k,(i,j)} - (\bigcup_{l \in J} u_l) = \bigcap_{l \in J} (t_{k,(i,j)} - u_l)$$

Ahora que tenemos el espacio topológico $t_{k,(i,j)}$, veremos que es un conjunto compacto. Entonces debemos encontrar una subcobertura finita en cada cobertura abierta de $t_{k,(i,j)}$.

Resulta que la única cobertura del conjunto $t_{k,(i,j)}$ es precisamente él mismo, porque es el subconjunto abierto más grande que lo contiene, por cierto, también es una subcobertura que es finita, puesto que el número de subconjuntos es finito, y está dado por el número de células que componen el triángulo. Así el número de subconjuntos de puntos en $\mathbb{P}(t_{k,(i,j)}) = 2^{(\frac{k^2+k}{2}+2k+1)}$.

Lo que sigue es mostrar que el triángulo $t_{k,(i,j)}$ es igual a la cerradura de su interior:

$$t_{k,(i,j)} = \overline{t_{k,(i,j)}^{\circ}}$$

El interior de $t_{k,(i,j)}$ es el conjunto abierto más grande contenido en $t_{k,(i,j)}$. Llamemos $A = \{(x,y|(x,y) \in t_{k,(i,j)})\}$ que es el conjunto buscado, ahora debemos determinar la cerradura de A.

La cerradura de A es el conjunto cerrado más pequeño que contiene a A. Supongamos que A' es un conjunto cerrado tal que $A' \subseteq A$. Si A' es un subconjunto propio de A, entonces $A \cap A' \neq \emptyset$, entonces no contiene al conjunto A. De manera que la única posibilidad de que

A' contenga al conjunto A es que $A' = A$. Pero $A = t_{k,(i,j)}$ por su propia definición. Aún más, esto nos dice que $t_{k,(i,j)}$ es un conjunto conectado, porque el único conjunto cerrado que contiene a $t_{k,(i,j)}$ es él mismo. Concluímos entonces que $t_{k,(i,j)}$ es un mosaico.

2.2. Los cubrimientos r110 son mosaicos. Después de mostrar que un triángulo es un mosaico, veremos que un cubrimiento también es un mosaico. Para ello, es suficiente saber que una condición necesaria para que exista un cubrimiento de más de un triángulo, es que existan un par de triángulos del cubrimiento, tales que sean adyacentes (página 45).

Llamemos entonces $A = t_{k,(i,j)} \cup t_{k',(i,j)}$. Entonces A es el conjunto de todos los puntos que pertenecen a $t_{k,(i,j)}$ o bien pertenecen a $t_{k',(i,j)}$. Para mostrar que A es un conjunto conectado, es suficiente determinar que $t_{k,(i,j)} || t_{k',(i,j)}$, porque eso significa que hay un camino de longitud 1 desde $t_{k,(i,j)}$ hasta $t_{k',(i,j)}$ de puntos exclusivamente de $t_{k,(i,j)} \cup t_{k',(i,j)}$.

Supongamos que $t_{k,(i,j)}, t_{k',(i,j)} \subset \Phi^{(2)}$ esto significa que $t_{k,(i,j)} || t_{k',(i,j)}$ que es la condición que necesitamos para establecer la conexión entre los dos triángulos.

2.3. El conjunto fundamental de un mosaico. Podemos asociar a los mosaicos un conjunto de triángulos llamado *el conjunto fundamental*. Este conjunto es definido por todos los triángulos que lo conforman, cuando un triángulo es repetido, se escribe con un subíndice que indica el número de copia de ese triángulo.

$$C \leftrightharpoons \{t_{0_i}, t_{1_j}, \dots t_{k_m}\}; \text{con los subíndices } \ i, j, k \text{ y } m \in \mathbb{Z}^+ \tag{7}$$

De manera que si un mosaico está compuesto por un triángulo t_4, dos triángulos t_0 y un triángulo t_1, entonces su conjunto fundamental es $\{t_{0_1}, t_{0_2}, t_{1_1}, t_{4_1}\}$.

3. Operación de traslados de mosaicos

Ahora definiremos una manera de copiar los mosaicos y colocarlos en otro sitio del espacio bidimensional entero. Esta operación es el traslado. Sea $\mathcal{T} : \mathcal{V} \times \mathcal{X}^{(*)} \to \mathcal{X}^{(*)}$, una función donde $\mathcal{V}$ es un conjunto de vectores con origen en un punto arbitrario del mosaico, llamándolo *el centro del mosaico*, tal y como fue definido al final del capítulo anterior (página 57), y $\mathcal{X}^{(*)}$ el conjunto de mosaicos de cualquier número de triángulos. Si $v \in \mathbb{V}$ y $P \in \mathcal{X}^{(*)}$, entonces $\mathcal{T}(v, P)$ significa que todas las células que pertenecen al mosaico P, han sido afectadas por el traslado señalado por v. Esta acción también la podremos escribir como $\mathcal{T}^v(P)$.

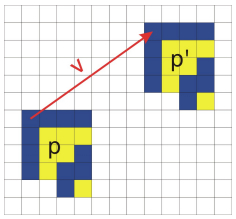

Figura 28. La operación traslado copia un mosaico en el lugar señalado por el vector v, dentro del espacio bidimensional entero. $p' = \mathcal{T}^v(p)$

Es muy posible que al hacer traslados por un vector arbitrariamente grande, se lleguen a cubrimientos prohibidos por varias razones:

- Si el vector v es demasiado corto ocurre un traslape de mosaicos, que es una de las condiciones para que exista un cubrimiento prohibido.
- Si el vector es demasiado largo, se corre un alto riesgo de crear un hueco imposible de cubrir de manera legal.
- La dirección del vector puede causar un cubrimiento prohibido debido a la alineación por la parte superior de dos triángulos.

La recomendación general es dar vectores lo suficientemente largos para que los mosaicos p y p' se besen[1].

Trasladar un mosaico a una posición tal que resulten un par de mosaicos que se besan es equivalente a concatenar gráficamente esos mosaicos. En cualquiera de los casos, el mosaico resultante no debe pertenecer a la clase de cubrimientos prohibidos.

3.1. Endomorfismo bajo el traslado y la concatenación gráfica. El sistema $(\mathcal{X}^{(*)}; \bowtie, \mathcal{T})$ define un endomorfismo[2]. Sea $(\mathcal{X}^{(*)}, \bowtie)$ el semigrupo de las concatenaciones gráficas, el homomorfismo es definido como:

$$\mathcal{T}^v(x \bowtie y) = \mathcal{T}^v(x) \bowtie \mathcal{T}^v(y);\ \forall x, y \in \mathcal{X}^{(*)},\ v \in \mathcal{V} \tag{8}$$

La interpretación de esta expresión simbólica debe ser que el traslado de la concatenación gráfica de un par de mosaicos, genera el mismo resultado que la concatenación gráfica del traslado de los mosaicos.

[1] Aquí el término besar se usa en el mismo contexto que en la página 51.

[2] Que también es un isomorfismo porque el resultado de la operación no cambia la forma del mosaico.

4. Cubrimientos del espacio con copias congruentes de un mosaico propio

Una clase particular de mosaicos son aquellos que pueden cubrir el plano, esta particularidad está estrechamente relacionada con el 18^{vo} problema de Hilbert, que describe el problema de cubrir el espacio con copias de un mismo poliedro. El problema propuesto por Hilbert queda descrito al establecer la pregunta: "¿Se puede cubrir el espacio Euclídeo de dimensión n, de manera simple y completa usando solamente copias congruentes del mismo poliedro? Ahora sabemos que la respuesta es sí [22].

Con los triángulos generados con las evoluciones del autómata celular regla 110, podemos construir mosaicos que cubren de manera simple y completa el plano Euclídeo entero, estos mosaicos los llamaremos *mosaicos propios*.

4.1. Concepto formal de mosaico propio.

Definición 11. *Un mosaico $p \in \mathcal{X}^{(*)}$ es un mosaico propio si cumple con las siguientes dos condiciones:*

(1) *p es un mosaico*
(2) *Existe un conjunto finito de vectores $\mathcal{V}$ tales que los traslados $\mathcal{T}^{kv}(p)$ cubran el plano. Con $v \in \mathcal{V}$ y $k \in \mathbb{Z}$.*

La segunda condición expresa la necesidad de cubrir el espacio de manera simple y completa usando copias del mismo mosaico. Llamaremos $\mathcal{M}^{(*)}$ al conjunto de todos los mosaicos propios de cualquier orden. Notemos que $\mathcal{M}^{(*)} \subset \mathcal{X}^{(*)}$.

En otros documentos [48, 19] se llama a los mosaicos propios "área fundamental del grupo de transformación 'cobertura' del espacio R^n", ya que un área fundamental junto con sus imágenes bajo el grupo, forman una cobertura del tipo que se requiere.

Los mosaicos obtenidos de las evoluciones del autómata celular regla 110 no se pueden rotar, de manera que pueden cubrir el plano únicamente como grupos de traslados, como se verá más adelante en la página 68. El problema inmediato a resolver es determinar el menor orden del mosaico; en otras palabras, determinar el menor número de triángulos necesarios para formar un mosaico propio.

4.2. Pertenencia de un mosaico a un mosaico propio. Como hemos visto casi desde el inicio de este capítulo, los mosaicos estan construidos en base a elementos que hemos llamado triángulos. Ahora es util definir las condiciones para que un triángulo sea considerado parte de un mosaico y luego para delimitar la extensión espacial del mosaico.

Los triángulos que forman los mosaicos se pueden agrupar en conjuntos, identificando cada triángulo que pertenezca al mosaico. Debido a que estamos particularmente interesados en los mosaicos propios, usaremos de manera indistinta los términos "mosaico" y "mosaico propio", cuando sea necesario hacer distinción entre esos objetos, se hará la debida aclaración.

DEFINICIÓN 12. *Decimos que un mosaico $p \in \mathcal{X}^{(*)}$ pertenece a un mosaico $m \in \mathcal{M}^{(*)}$ denotándolo como $p \in m$, si al hacer $r = p \ltimes m$, $r \notin \mathcal{M}^{(*)}$, es decir r no es un mosaico.*

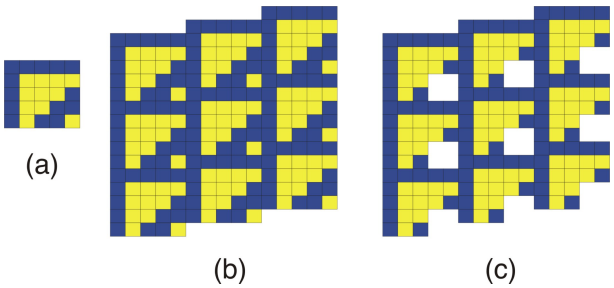

Figura 29. Pertenencia de un triángulo a un mosaico: (a) El mosaico propio a, cuyo conjunto fundamental es $M = \{t_4, t_{0_1}, t_{0_2}, t_1\}$. (b) El mosaico. (c) Un mosaico (prohibido) con un mosaico definido como $a' = a \ltimes t_1$

Las propiedades de un mosaico exigen que se pueda cubrir el plano con copias de él, sin dejar huecos y sin translapes. En la figura 29 se aprecia un caso en el cual se retira un triángulo t_1 del mosaico propuesto en (a) y se construye un cubrimiento que deja huecos (c), de manera que el triángulo t_1 que había sido retirado pertenece al mosaico (a).

Cuando $r = p \ltimes m$, $r \in \mathcal{M}^{(*)}$, podemos decir que m es un *supermosaico propio* de r y en consecuencia r es un *submosaico propio* para m. en la figura 29 b, es posible retirar submosaicos a de diferentes maneras, una de ellas es retirando una de las esquinas, lo que genera un submosaico propio.

4.3. Clasificación de mosaicos propios. Algunos problemas surgen al notar la capacidad que tienen los triángulos para formar mosaicos propios. Uno de esos problemas es determinar el número de mosaicos propios de orden 1, los de orden 2 y así en adelante. Otro problema es, dado un mosaico propio, determinar el mínimo submosaico propio que puede ser obtenido por aplicaciones sucesivas de la función $\ltimes$. Los mosaicos propios podemos clasificarlos en unitarios o compuestos.

4.3.1. *Mosaicos propios unitarios.* Los mosaicos propios unitarios son mosaicos de orden igual a 1, es decir triángulos que pertenecen al conjunto de los mosaicos propios.

Los únicos elementos que cumplen esta condición son los triángulos t_1 y t_2. El t_0 no es un mosaico propio porque al concatenar gráficamente mosaicos t_0, se viola la regla de evolución local $000 \rightarrow 1$, generando cubrimientos prohibidos.

Con los triángulos mayores al t_2 es imposible cubrir el plano sin dejar huecos, o generar cubrimientos prohibidos. Los mosaicos hechos con mosaicos propios unitarios t_3 o mayores, requieren al menos del mosaico t_0. Así que todos los demás triángulos a partir del t_3 no son mosaicos propios unitarios, porque requieren de al menos un t_0 más.

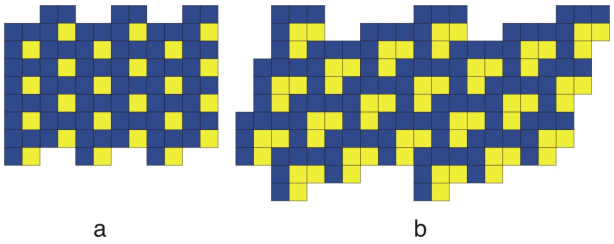

Figura 30. Mosaicos propios unitarios: (a) Cubrimiento con t_1 (b) Cubrimiento con t_2

4.3.2. *Mosaicos propios compuestos.* Los mosaicos propios compuestos son figuras o patrones gráficos tales que para cualquier mosaico $p \in \mathcal{M}^{(*)}$ el orden $||p|| > 1$. De esta nueva clasificación existe una infinidad de mosaicos propios.

Existe una estrecha relación entre los mosaicos propios y los *gliders* [35]. En el caso de los *gliders* son definidos por repeticiones de patrones celulares en un periodo de tiempo, característica que comparten con los mosaicos propios. Sin embargo en el caso de los *gliders* nada se menciona acerca de los límites espaciales, de manera que la extensión espacial de los gliders no está bien determinada en su definición; ésta es la diferencia, ya que en los mosaicos propios sí se sabe qué triángulos pertenecen a cada mosaico.

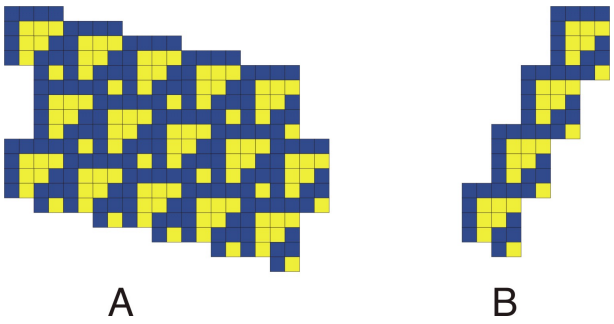

Figura 31. Mosaicos propios compuestos: (A) Cubrimiento con un mosaico propio de conjunto fundamental $\{t_3, t_1, t_0\}$ (B) Glider

4.4. Crecimiento de mosaicos. El proceso de cubrir el plano con mosaicos, puede crecer de manera cristalográfica[3] en dos maneras, haciendo copias simples del mosaico propio o haciendo copias de los supermosaicos propios encontrados. Ambas maneras muestran crecimiento hexagonal. En las siguientes secciones se describe a detalle cada una de estas formas de crecimiento.

4.4.1. *Crecimiento hexagonal simple.* Debido principalmente a la forma triangular de los mosaicos propios unitarios y al criterio de Conway [18] para los mosaicos, es posible rodear un mosaico propio por 6 copias congruentes de él mismo, lo que constituye la primera corona[4].

La segunda corona consiste en rodear el supermosaico obtenido con copias del mosaico original, como se muestra en la figura 32, y de esta manera crece el mosaico formando en cada corona un supermosaico propio.

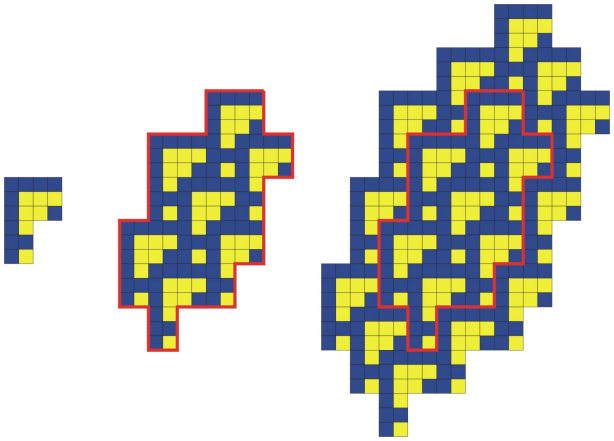

Figura 32. Crecimiento hexagonal simple de uno de los mosaicos propios de orden menor. Se muestran 2 coronas.

Se puede determinar el número de mosaicos en la $n-$ésima corona calculando $6 * n$ y el número de mosaicos en todo el cubrimiento calculando $3n^2 + 3n + 1$, como se muestra en la tabla 8.

4.4.2. *Crecimiento hexagonal en potencias.* En el crecimiento hexagonal en potencias se rodea un mosaico propio con copias de él mismo, también se requieren 6 copias para rodearlo. El mosaico que resulta vuelve a ser un mosaico propio, lo que da la oportunidad de hacer copias del supermosaico propio obtenido y rodearlo de nuevo con copias del mismo supermosaico propio, como se muestra en la figura 33. El proceso continúa hasta cubrir el plano.

[3]Entenderemos crecimiento cristalográfico como la agrupación de estructuras que son autosimilares

[4]Capa de mosaicos que rodea a otro mosaico congruente

Corona	En la $n-$ésima corona	En el subcubrimiento
0	1	1
1	1*6	1+6
2	2*6	(1+6)+6
3	3*6	((1+6)+6)+6
⋮	⋮	⋮
n	n*6	$1+6\sum_{i=1}^{n} i$ $=3n^2+3n+1$

Cuadro 8. Número de mosaicos en cada corona

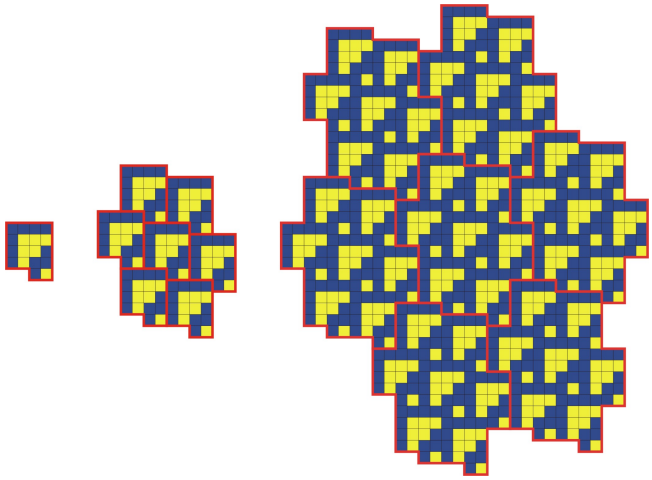

Figura 33. Crecimiento hexagonal en potencias de uno de los mosaicos propios de orden menor. Se muestra hasta la potencia 2.

Se puede determinar el número de mosaicos en la $n-$ésima potencia y el número de mosaicos en todo el cubrimiento. En la $n-$ésima corona (es el mismo número que la potencia) siempre habrá 6 mosaicos. Y el número total de copias del mosaico original está dado por 7^n, como se muestra en la tabla 9.

4.4.3. *Resultados de los tipos de crecimientos de los mosaicos propios.*

- El mosaico obtenido del crecimiento hexagonal simple de un mosaico propio es un supermosaico propio compuesto.
- El mosaico obtenido del crecimiento hexagonal en potencias de un mosaico es un supermosaico propio compuesto.

Potencia	En el subcubrimiento
0	1
1	$1 + 1\,(6^1)$
2	$1 + 2(6^1) + 1\,(6^2)$
3	$1 + 3(6^1) + 3(6^2) + 1\,(6^3)$
4	$1 + 4(6^1) + 6(6^2) + 4(6^3) + 1\,(6^4)$
5	$1 + 5(6^1) + 10(6^2) + 10(6^3) + 5(6^4) + 1\,(6^5)$
$\vdots$	$\vdots$
n	$(1 + 6)^n = 7^n$

Cuadro 9. Número de mosaicos en el cubrimiento con crecimiento en potencias.

5. Retículas inducidas por los mosaicos propios

Los puntos de referencia de cada copia del mosaico propio, genera una retícula en el plano. En la figura 34 izquierda se muestra un cubrimiento con el mosaico *ether*[5].

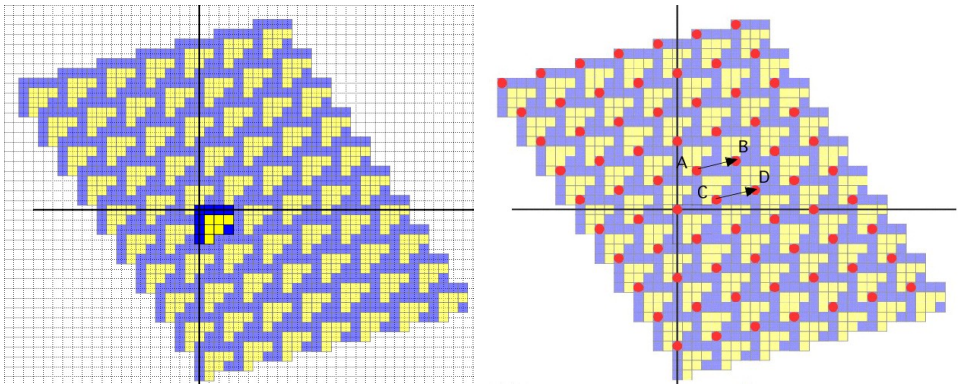

Figura 34. (Izq) Cubrimiento creado a partir de un mosaico compuesto por un t_3 y un t_0. Este cubrimiento genera una malla. (Der) Malla creada a partir del mosaico

5.1. Las bases del reticulado. El punto de referencia en cada copia del mosaico propio produce un conjunto de puntos $\mathcal{L} \in \mathbb{Z}^2$ separados de manera homogénea (figura 34derecha). Cualquier punto de ese conjunto puede ser el origen del espacio de puntos de referencia.

Podemos decir que el conjunto de puntos $\mathcal{L}$ forma un grupo bajo la suma de vectores y cada uno de esos puntos es el centro de una bola topológica en el plano, que no contiene

[5]Se llama así porque en las evoluciones de la regla 110 aparece con frecuencia y ocupa grandes porciones del espacio celular, haciendo parecer que el fondo de la imagen no está vacío [35].

algún otro punto de $\mathcal{L}$; que es una de las propiedades necesarias para que el conjunto de puntos usado defina un reticulado.

Si $A, B, C, D \in \mathcal{L}$, (figura 34 derecha) y si existe un vector que parte de A y llega a B sin incluir algún otro punto de $\mathcal{L}$, entonces para algún otro punto $C \in \mathcal{L}$ debe existir un vector que llegue al punto $C + (B - D)$. Entonces este reticulado es cerrado bajo la operación $C + (B - D)$, pero si hacemos el punto C el origen, el reticulado es cerrado bajo la operación $B - D$, que ahora dice que $\mathcal{L}$ es un subgrupo aditivo del espacio vectorial $\mathbb{R}^2$, en donde todas las entradas de las coordenadas de todos los puntos de $\mathcal{L}$ son números enteros; que es la otra propiedad que necesitamos para que los puntos definan un reticulado.

El siguiente paso es hacer una partición del espacio uniendo con líneas rectas algunos de esos puntos, lo que produce el reticulado del mosaico propio. Si $m \in \mathcal{M}^{(*)}$ es el mosaico propio, entonces $\mathcal{L}_m$ es el reticulado del espacio respecto del mosaico propio m (figura 35).

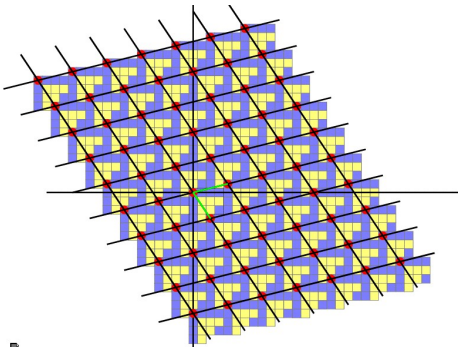

Figura 35. Paralelepípedos formados a partir de una de las bases de la malla de la figura 34.

Si $\mathcal{L}$ es un reticulado en el $n-$espacio, existen k vectores linealmente independientes $v_1, v_2, \ldots, v_k; k \leq n$, tales que $\mathcal{L}$ consiste de todos los puntos de la forma $m_1v_1 + m_2v_2 + \cdots + m_kv_k$, donde $m \in \mathbb{Z}$ y $1 \leq i \leq k$. Tal conjunto de vectores se llama la *base* del reticulado y k es la *dimensión* del reticulado [59].

5.2. El determinante del reticulado. Llamemos v_1, v_2 la base de nuestro reticulado $\mathcal{L}$ en el espacio de dimensión 2. El paralelogramo cuyas aristas son v_1, v_2 es el conjunto de puntos $\{x_1v_1 + x_2v_2 : x_i \in \mathbb{Z}, i = 1, 2\}$, que se conoce como el "paralelogramo fundamental"

de $\mathcal{L}$. El área de este paralelogramo es independiente de la base elegida, y se conoce como el *determinante* de $\mathcal{L}$ y se denota como $\mathcal{D}(\mathcal{L})$:

$$\mathcal{D}(\mathcal{L}) = \left| \begin{pmatrix} x_{v_1} & y_{v_1} \\ x_{v_2} & y_{v_2} \end{pmatrix} \right| \tag{9}$$

El resultado del determinante de la ecuación 9 es un número entero, y es igual al **volumen del mosaico**. El volumen del mosaico es un concepto diferente que el volumen del reticulado, pues cuenta todas las células que pertenecen a cada triángulo que conforma el mosaico propio; y se calcula mediante:

$$vol(t_n) = 1 + 2n + \sum_{i=1}^{n} i = 1 + \frac{n}{2}(n+5) \tag{10}$$

Y el volumen del mosaico (y en general del cubrimiento) $m \in \mathcal{X}^{(*)}$ es:

$$Vol(m) = \sum_{i=1;t_{n_i} \in m}^{\|m\|} vol(t_{n_i}) \tag{11}$$

La ecuación 11 suma el número de células de cada triángulo que pertenece al mosaico, desde 1 hasta el orden del mosaico $\|m\|$.

PROPOSICIÓN 2. *Si para un mosaico $p \in \mathcal{X}^{(*)}$ se tiene que $\mathcal{H}(p) = \infty$, entonces $p \in \mathcal{M}^{(*)}$.*

Para verificar la proposición anterior es necesario probar la existencia de un conjunto de vectores $\mathcal{V}$ que forman una base para crear un reticulado que sirva para cubrir el plano con los traslados $\mathcal{T}^{kv}$, con $v \in \mathcal{V}$ y $k \in \mathbb{Z}$.

En 1988, Martin Gardner propuso familias de mosaicos hexagonales que cubren el plano [17]; anteriormente John H. Conway había enunciado un criterio para determinar si una figura bidimensional puede ser considerada como un mosaico, el ahora llamado "criterio de Conway" [18].

Las figuras bidimensionales en las que estamos interesados son los mosaicos regla110. De acuerdo con el criterio de Conway, si tenemos un mosaico, y podemos dividir su contorno en 6 partes (figura 36), de tal manera que el lado a es un traslado del lado d, el lado b es un traslado del lado e, y el lado c es un traslado del lado f, entonces se puede cubrir el plano con copias del mosaico propuesto.

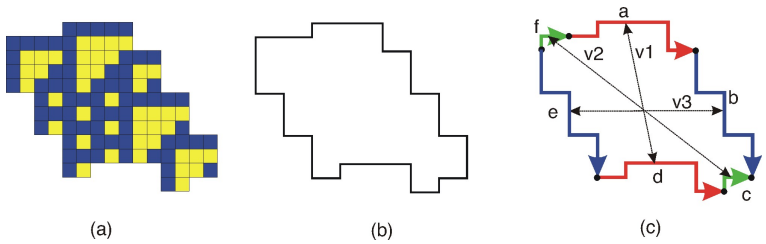

Figura 36. (a) Mosaico p, conocido como *glider* D (b) Contorno del mosaico. (c) División del contorno en 6 segmentos basados en traslados de lados opuestos.

El conjunto de vectores buscado $\mathcal{V} \leftrightharpoons \{v_1, v_2, v_3\}$ donde cada v_i es uno de los 3 traslados (figuras 36 y 37). Un par de vectores de este conjunto es suficiente para formar la base del reticulado con que se cubre el espacio con esos mosaicos.

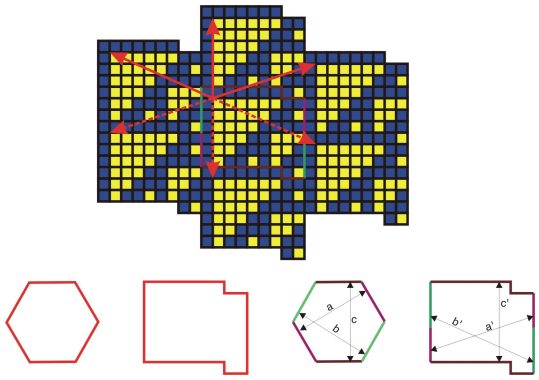

Figura 37. *Arriba:* Cualquier parte de un mosaico propio se puede localizar mediante traslados *Abajo:* Criterio de Conway para asegurar que este patrón es un mosaico propio.

De cualquier manera, el criterio de Conway ofrece una condición necesaria para cubrir el plano con los mosaicos propios, pero no es suficiente. Además de ese criterio se deben respetar las reglas de evolución local en cada triada de puntos gráficos. De otro modo resultaría un cubrimiento prohibido del plano.

6. Familias de mosaicos

Consideremos ahora un reticulado $\mathcal{L}$ producido por algún mosaico propio de orden k, $m \in \mathcal{M}^{(k)}$, donde la base del reticulado es el conjunto de vectores $\{v_1, v_2\}$ y el paralelogramo fundamental tiene área A. La relación que existe entre el área A del paralelogramo y el volumen del mosaico $\|m\|$ es de igualdad.

Como se puede apreciar en la figura 38 cualquier parte de un mosaico propio se puede localizar mediante traslados por sus vectores base del reticulado.

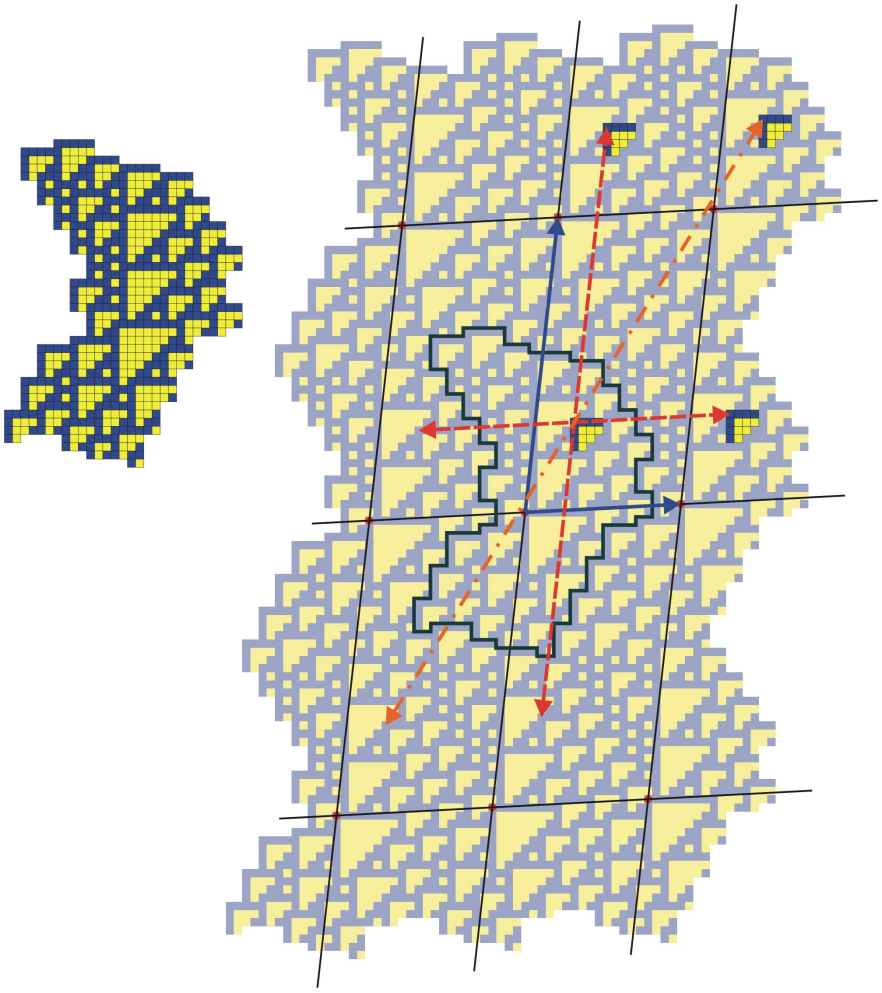

Figura 38. Mosaico formado por 102 triángulos. Cualquier subcubrimiento se puede localizar mediante traslados de sus vectores o por combinaciones de ellos.

De manera que el mosaico que resulta de mover un submosaico **de la frontera** de un mosaico propio a una posición establecida por los vectores de una de sus bases, vuleve a ser un mosaico propio. Se ha marcado la frase "de la frontera" porque mover un submosaico del interior resultaría en un nuevo cubrimiento con huecos, que de inmediato es condición suficiente para hacerlo miembro del conjunto de cubrimientos prohibidos; aún cuando al generar y concatenar gráficamente las copias de este mosaico, generara un cubrimiento legal.

DEFINICIÓN 13. *Sean $p = q \bowtie r \in \mathcal{M}^{(*)}$, $\mathcal{V}$ la base de su reticulado, y $p' = \mathcal{T}^{kv}(q) \bowtie r$, con $v \in \mathcal{V}; k \in \mathbb{Z}$; tal que $p' \in \mathcal{X}^{(*)}$, decimos que p y p' pertenecen a la misma* **familia de mosaicos propios** *si $p' \in \mathcal{M}^{(*)}$ y $\mathcal{V}$ es la base del reticulado creado con p'.*

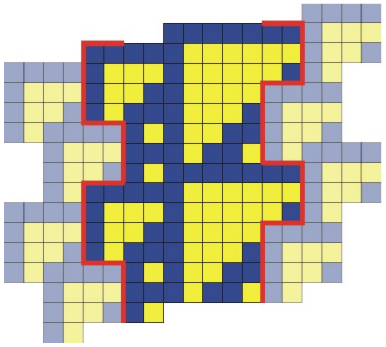

Figura 39. Las fronteras de un mosaico propio C tienen las misma características que los mosaicos propios *ether*, por eso pueden coexistir.

Para que dos mosaicos propios de diferentes familias puedan coexistir en el mismo espacio, se requiere que sus fronteras más cercanas sean "compatibles" con el mosaico propio llamado *ether*, como se muestra en la figura 39. La coexistencia de los distintos mosaicos propios, puede darse con espacio de *ether* o sin él, siempre que su frontera sea capaz de acomodar este tipo de mosaico propio.

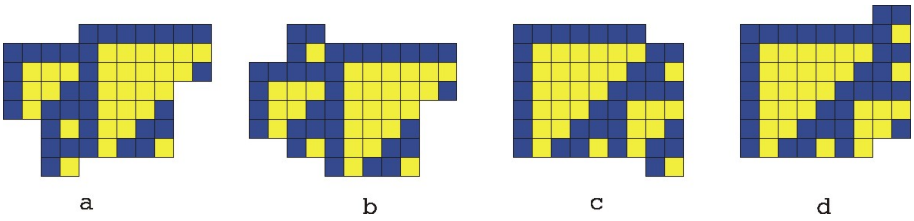

Figura 40. Mosaicos propios diferentes que pertenecen a la misma familia de mosaicos. Los mosaicos propios (a) y (b) ocurren frecuentemente en las evoluciones normales del autómata celular regla 110, debido a sus fronteras compatibles con el *ether*; los mosaicos propios (c) y (d) ocurren con poca frecuencia por sus fronteras no compatibles.

6.1. Coexistencia de más de una familia de mosaicos propios. Dos o más familias de mosaicos pueden coexistir en el mismo cubrimiento, siempre que tengan una frontera en común. Por la naturaleza hexagonal de los mosaicos propios de cada famila, las fronteras compartidas nunca son en sentido horizontal, esto ocasionaría un cubrimiento prohibido.

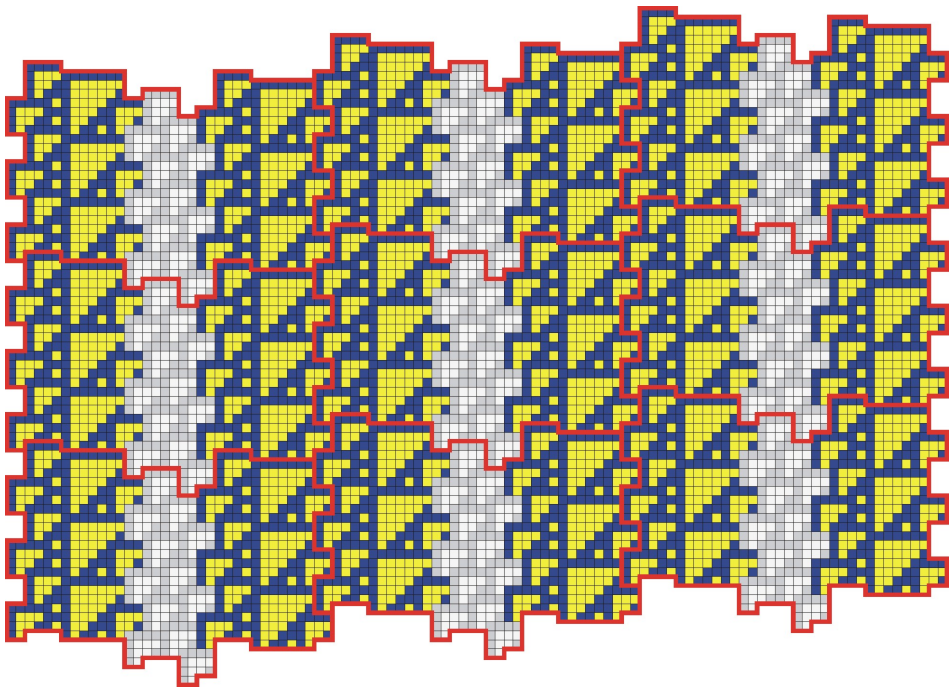

Figura 41. Diferentes familias de mosaicos propios se pueden agrupar para formar un nuevo mosaico propio compuesto por esas familias.

El cubrimiento del espacio entre dos mosaicos propios es ocupado en su mayoría por *ether*, como se muestra en la figura 41, sin embargo frecuentemente es necesario utilizar triángulos mayores. En el ejemplo mostrado en la figura 41 las familias de mosaicos generan un nuevo mosaico propio. Esto se debe a que tienen en común uno de los vectores de sus bases. Cuando ninguno de los vectores es común en sus bases, aún es posible unirlos pero no formarán un mosaico propio.

Cuando se crean mosaicos propios como alguno de la familia C y alguno de la familia D, pueden coexistir en un espacio que llamaremos "zona cordial" (parte (a) de la figura 42), pero llegará el momento en que esa zona cordial desaparezca, dando paso a la "zona crítica" en la parte (b) de la misma figura, en donde ya no hay espacio para ambos mosaicos propios.

Las siguientes zonas muestran los conflictos y consecuencias del choque entre estos dos mosaicos propios. La zona (c), que es la "zona de conflicto", muestra un mosaico t_3 que es compartido por ambos mosaicos propios, en esta zona uno de los dos mosaicos propios termina su existencia, pero las consecuencias hacen que el mosaico propio C (derecha) termine también su existencia en la continuación de la zona de conflicto. Hacia la parte (e) ya no existen ninguno de los dos mosaicos propios.

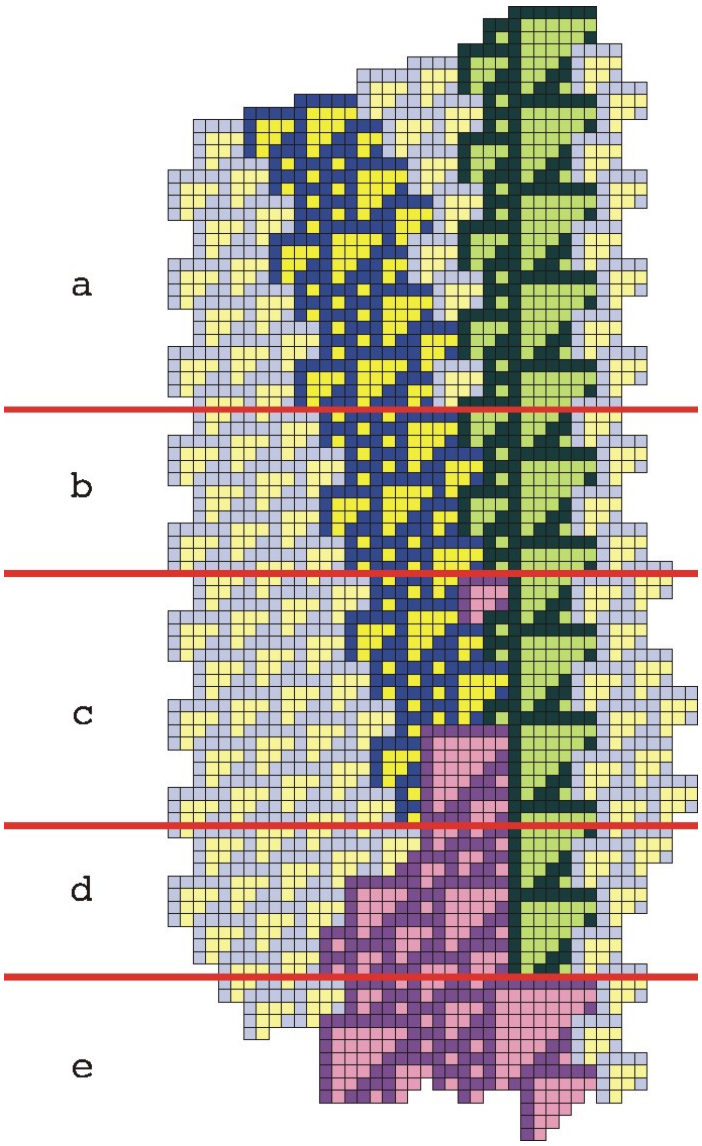

Figura 42. Uno de los posibles choques entre mosaicos propios que corresponden a los gliders D con velocidad $\frac{c}{5}$y C con velocidad 0.

La referencia [35], es un amplio, profundo y metódico estudio de los choques de *gliders* que son generados en las evoluciones del autómata celular regla 110. Muchos de esos estudios son basados en mosaicos, aunque los *gliders* no corresponden perfectamente con los mosaicos propios, debido a la ambigüedad de la definición de las fronteras espaciales de los gliders.

7. Conclusiones

En este capítulo se ha llegado a las siguientes conclusiones:

- El sistema formado por el conjunto de todos los mosaicos, junto con la concatenación gráfica y el traslado $(\mathcal{X}^{(*)}; \bowtie, \mathcal{T})$ define un endomorfismo.

- Un criterio para determinar la pertenencia de un mosaico a otro está basado en la ocurrencia de huecos.

- Con respecto al crecimiento hexagonal simple de un mosaico, el número de mosaicos en la $n-$ésima corona es $6 * n$ y el número de mosaicos en todo el cubrimiento es $3n^2 + 3n + 1$.

- Considerando el crecimiento hexagonal en potencia, el número total de copias del mosaico original está dado por 7^n.

- Se dan criterios en dos direcciones para determinar si es posible cubrir el plano con mosaicos propios:
  (1) La creación de un reticulado inducido por los traslados de mosaicos.
  (2) El criterio de Conway y las reglas de evolución local.

En el siguiente capítulo se estudiarán otro tipo de cubrimientos, que son generados mediante reglas de agregación. Estas reglas nos permitirá tener control sobre la forma y tamaño del mosaico creado.

Capítulo 4

Construcciones basadas en reglas y evoluciones de Post

1. Resumen

Las regularidades mostradas en las agregaciones de los triángulos hacen posible que se puedan definir reglas para establecer criterios de agregación por cada uno de los lados de cada triángulo. Estas reglas son llamadas "reglas de agregación". Hay reglas de agregación asociadas por el lado superior, por el lado izquierdo y por el lado diagonal de los triángulos.

Las tres palabras asociadas a cada triángulo, junto con el triángolo asociado, son los sistemas de palabras. Un conjunto de sistemas de palabras, hacen un esquema.

Las evoluciones de Post establecen la dinámica del proceso constructivo al aplicar reglas de evolución a partir de un cubrimiento inicial.

2. Proceso constructivo

Otra manera de cubrir con triángulos una región del plano, es siguiendo un conjunto de reglas que permiten tener el control de la forma y tamaño del mosaico. En este capítulo se estudia un proceso constructivo basado en reglas de agregación. Se estudian los símbolos permitidos y las maneras de concatenar estos símbolos para formar palabras y la relación de su apariencia gráfica con cadenas de símbolos. Se estudian las evoluciones de Post, que nos permiten observar el comportamiento dinámico de un mosaico al ser modificado con reglas de agregación.

Llamamos "proceso constructivo basado en mosaicos" a la forma establecida de concatenar gráficamente los triángulos, manipulando cadenas de símbolos; de manera que el resultado de esas concatenaciones sea un nuevo mosaico.

La concatenación de los elementos de construcción, es decir, los triángulos o los mosaicos de orden 1, debe ser una acción ordenada; la cantidad de elementos y el orden de éstos, es determinado por un triángulo, que es tomado como base de la construcción, y que también funciona como centro del mosaico para usar cualquiera de los criterios de comparación ya

definidos. Estas ordenaciones de triángulos en secuencias finitas las describiremos e identificaremos por secuencias de símbolos que llamaremos **palabras**. El proceso constructivo se organizará en dos partes, la primera involucra la manera de construir las palabras y la segunda describe cómo se pueden obtener palabras a partir de otras palabras usando las reglas de agregación.

3. El alfabeto de símbolos permitidos

El proceso constructivo inicia definiendo el conjunto de los símbolos permitidos.

$$\mathbb{T} \leftrightharpoons \{t_n | t_n \in \mathcal{X}^{(1)}; n \in \mathbb{Z}^{\geq 0}\} \tag{12}$$

El conjunto $\mathbb{T}$ contiene todos los mosaicos de orden 1, es decir, todos los triángulos, el triángulo nulo también esta en este conjunto.

Determinar el mayor tamaño para un triángulo representa un problema abierto [33]; sin embargo para el desarrollo de este trabajo, la indeterminación del máximo tamaño para un triángulo no implica dificultades, porque todos los triángulos de cualquier tamaño, son miembros del conjunto $\mathbb{T}$.

3.1. El alfabeto restringido de triángulos. Para el desarrollo de las siguientes secciones, trabajaremos con un subconjunto de triángulos, el tamaño de este subconjunto es arbitrario, pero existen algunas restricciones para la elección de los triángulos que pertenecerán a este conjunto. En adelante, usaremos el conjunto $T \subseteq \mathbb{T}$.

Los triángulos que ocurren con mayor frecuencia son los de menor tamaño; un subconjunto que no contenga al menos el triángulo t_0 estará muy limitado; aquellos conjuntos que contengan al menos los primeros 10 triángulos son capaces de hacer muchas combinaciones y fabricar muchos tipos de mosaicos. Los triángulos mayores ocurren con muy poca frecuencia. En [33, 34] se han coleccionado configuraciones iniciales que generan triángulos de diferentes tamaños, donde el resultado importante es que no puede haber uno mayor que 42 [34, 35].

Los símbolos permitidos estarán denotados por t_n, por ejemplo t_1, t_{15}, t_{22} y por supuesto t_0 que indican respectivamente triángulos de tamaños 1, 15, 22 y 0. Ocasionalmente cuando la tipografía lo permita, podremos escribir directamente los números a un lado del símbolo "t", como en t1, t15, t22 y t0.

4. Palabras basadas en el alfabeto T

4.1. Mecánica de construcción de las palabras. Con los símbolos de los triángulos como elementos de construcción, podemos crear cadenas de símbolos que representan mosaicos. Estas cadenas las nombraremos "palabras" y se pueden obtener de manera recursiva bajo las mismas reglas que definen las palabras en la teoría de lenguajes formales, como lo muestran [10, 11, 25]. Las reglas de creación de las palabras son las siguientes:

(1) La palabra vacía t_λ (o simplemente λ) es una palabra en el alfabeto T.
(2) Si una palabra P es una palabra en el mismo alfabeto T; Pt_n también es una palabra en el mismo alfabeto T si $t_n \in T$.

Pt_n es una cadena de símbolos que indica el objeto que resulta de concatenar gráficamente el triángulo t_n a un mosaico simbolizado por la palabra P. En un triángulo se pueden concatenar gráficamente otros triángulos en cada uno de sus tres lados.

4.1.1. *Concatenación de símbolos.* Concatenar símbolos del alfabeto T es equivalente a la concatenación gráfica. Si $T \leftrightharpoons \{t_k | 0 < k \leq n\} \subset \mathbb{T}$ y $t_i,\ t_j \in T$, entonces $t_i \cdot t_j \leftrightharpoons t_i \bowtie t_j$.

Podemos clasificar las palabras de acuerdo a la cantidad de símbolos que contienen. La clase de 0 símbolos la denotaremos por λ; la clase de todas las palabras compuestas por 1 símbolo será denotada por T^1; y así el conjunto de palabras de n símbolos por T^n. En general el conjunto de palabras de cualquier cantidad de símbolos será denotada por T^*.

La linearidad de la escritura simbólica debe tener una estrecha relación con la apariencia gráfica. Una manera de especificar el orden de la concatenación de símbolos en la escritura, es concatenar por la derecha. En el dibujo hay tres posibilidades, por la diagonal, por el lado superior o por el lado izquierdo de un triángulo.

Enseguida se describen los términos que serán usados para describir una palabra en términos de sus componentes.

Para cualquier palabra $P \in T^*$, existen palabras $Q, R \in T^*$ tal que $P = QR$, luego:

- Q es el segmento inicial de la palabra P
- R es el segmento final de la palabra P
- Si $R = \lambda$ y $P \neq \lambda$, entonces $Q = P$ y Q es un segmento inicial propio de P
- Si $Q = \lambda$ y $P \neq \lambda$, entonces $R = P$ y R es un segmento final propio de P.

4.1.2. *Longitud de las palabras.* La longitud de una palabra es un número natural (incluyendo al 0 para la palabra vacía) que expresa la cantidad de símbolos con los cuales está formada la palabra. La manera de denotar la longitud de una palabra será la misma que en [31] para mantener la misma idea del concepto.

La longitud $[P^{\,\delta}$ de una palabra P se puede obtener de manera inductiva mediante:

$$\begin{aligned} [\lambda^{\delta} &= 0 \\ [Pt_n{}^{\delta} &= 1 + [P^{\delta} \end{aligned} \tag{13}$$

De la descripción de la longitud de una palabra podemos decir que:

- **a.-** La longitud de 2 palabras concatenadas es la suma de las longitudes de cada una de ellas: $[XY^{\delta}=[X^{\delta} + [Y^{\delta}$.
- **b.-** Si X es el segmento inicial de Y, entonces $[X^{\delta} < [Y^{\delta}$
- **c.-** Si X es el segmento inicial propio de Y, entonces $[X^{\delta} \leq [Y^{\delta}$
- **d.-** Si $X \neq \lambda$, entonces $[X^{\delta} \neq 0$.

Sean entonces $\mathbb{T}$ el conjunto de todos los símbolos que representan los triángulos regla 110; y T un subconjunto de $\mathbb{T}$, tal como fue descrito en la página 80. Tenemos ahora la siguiente proposición que es verdadera en el caso de los lenguajes regulares [26].

PROPOSICIÓN 3. *Para todo alfabeto basado en mosaicos $T \subset \mathbb{T}$, T^* es infinito numerable*

Prueba. Asignaremos un número natural para cada uno de los símbolos del alfabeto, reservando el 0 para la palabra vacía λ. El número asignado corresponde al tamaño del triángulo representado más uno; así al símbolo t_0 le corresponde el número 1, al símbolo t_1 le corresponde el número 2 y en general, al símbolo t_k le corresponde el número $k+1$. Llamemos $\mathfrak{n}(t_k) = k+1$ el número natural que le corresponde al símbolo t_k.

El número que le corresponde a una cadena de símbolos de longitud m $w = s_1 s_2 \dots s_m$, donde cada $s_i \in T$ y los índices i van desde $i = 1, \dots, |T|$, es el número base $|T|+1$, que resulta de concatenar los números $\mathfrak{n}(s_1)\mathfrak{n}(s_2)\dots\mathfrak{n}(s_m)$.□

Sin embargo, debemos hacer notar que no todas las palabras que se pueden construir en base a los símbolos de un alfabeto $T \subset \mathbb{T}$ son palabras válidas, a partir de la evolución de alguna configuración inicial del autómata celular lineal regla 110. Aquellas palabras que no es posible representarlas gráficamente considerando las reglas de concatenación gráfica de los triángulos regla 110, deben pertenecer al conjunto de mosaicos prohibidos.

4.1.3. *Comparación de palabras.* Las palabras creadas con los elementos constructivos generados con el autómata celular regla 110 pueden ser comparados de acuerdo al mismo alfabeto T. Podemos expresar la igualdad o diferencia de dos palabras mediante el símbolo $=_T$ para la igualdad o bien $\neq_T$ para la desigualdad. Cuando es sobreentendido el alfabeto, podemos omitir el subíndice. Podemos comparar las palabras usando cualquiera de los siguientes métodos:

Comparación lineal: Es la comparación usual de las palabras. Decimos que dos palabras $P, Q \in T^*$ son iguales si contienen los mismos símbolos en el mismo orden.
- $\lambda =_T \lambda$
- $Pt_n =_T Qt_n$ es verdadero siempre que $P =_T Q$
- $P \neq_T Q$ es verdadero siempre que $P =_T Q$ sea falso

Métrica completa: Como cada palabra representa simbólicamente un mosaico, podemos elegir el primer símbolo de la palabra y tomarlo como centro del mosaico, luego considerar un conjunto de vectores $\mathcal{V}$ que contenga elementos que señalen a cada uno de los triángulos del mosaico. Si la diferencia es 0, serán iguales bajo el criterio de $\mu-$semejanza (ver página 56).

4.2. Palabras asociadas a las fronteras de los triángulos. Los triángulos tienen otros triángulos concatenados gráficamente en cada uno de sus tres lados, cada lado será considerado por separado para estudiar las propiedades de esas palabras.

4.2.1. *Asociación de palabras por la frontera diagonal.* Llamaremos $[t_n{}^{\swarrow}$ a la palabra asociada por la frontera diagonal de un triángulo t_n en el alfabeto T. Aún cuando la palabra es leída de izquierda a derecha, gráficamente se muestran de derecha a izquierda y de arriba hacia abajo. Esto es para hacer corresponder la enumeración de los espacios en la diagonal de un triángulo, con el índice de la posición de los símbolos en su palabra asociada; como lo muestra la figura 43.

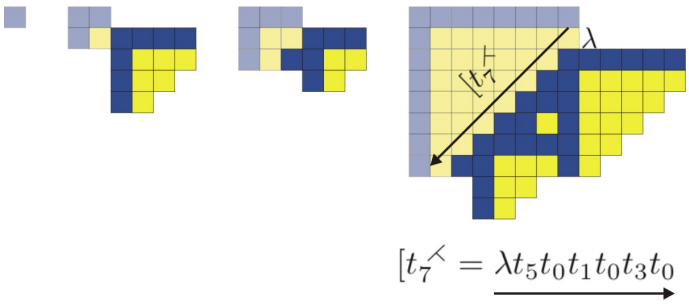

Figura 43. La longitud de la palabra asociada por la frontera diagonal de un triángulo es el tamaño del triángulo asociado. $[[t_n{}^{\swarrow\delta} = n$

En la posición con índice 0 de la diagonal, no puede haber un triángulo t_0, por las restricciones descritas en la sección de los mosaicos prohibidos, en la página 48; sin embargo, el lugar queda vacante, pues allí se concatenará otro triángulo por el lado izquierdo y no por el vértice superior izquierdo, como lo hacen el resto de los triángulos de la palabra diagonal.

En la figura 43, en la palabra asociada por la diagonal hay un t_0 en cada posición impar, también es posible intercalar triángulos t_0 en cada posición par. Poner dos triángulos t_0 en posiciones seguidas generaría una configuración ilegal en la siguiente concatenación gráfica.

La forma general para las palabras asociadas por la diagonal de un triángulo de tamaño $n > 0$ es:

$$[t_n{}^{\nwarrow} = \nwarrow \tau_0 t_{\kappa_1} t_{\kappa_2} \dots t_{\kappa_{n-1}} \nwarrow$$

La palabra es leída de izquierda a derecha, pero graficada sobre la diagonal del triángulo t_n de derecha a izquierda. En la palabra se debe cumplir que $\tau_0 \neq t_0$; de hecho, $t_{\kappa_i} \neq t_i$, lo que significa que en el $i-$ésimo lugar de la palabra, no debe de ir un triángulo de tamaño i.

En el caso particular de asociar alguna palabra a un t_0 es simple, porque no tiene diagonal (ni lados izquierdo o superior) de manera que la única palabra que es posible asociar a un t_0 es la palabra vacía: $[t_0{}^{\nwarrow} = \lambda$.

Para el caso del t_1, hay un único sitio posible para derivar, sin embargo coincide con la primera posición de la diagonal, etiquetada con 0, de manera que por las reglas de los cubrimientos prohibidos, no podemos concatenar gráficamente un t_0 en ese sitio, ubicado en la posición 0 de la diagonal de un t_1. Así que $[t_1{}^{\nwarrow} = t_n$, con $n > 0$.

PROPOSICIÓN 4. *Es posible dar un procedimiento para encontrar una configuración del autómata celular lineal regla 110, que genera una evolución que contiene los mosaicos representados por las palabras asociadas por la diagonal de un triángulo regla 110.*

Prueba. La prueba debe hacerse para cada regla con el procedimiento que se describe a continuación. Vamos a mostrar el procedimiento con un ejemplo. Sea $[t_5{}^{\nwarrow} = \nwarrow \sqcup\ t_0 t_1 t_0 t_3 \nwarrow$. El estado de las células conocidas son las que se muestran en la figura 44

La figura muestra tres secciones etiquetadas con las letras a, b, c, el estado de las células en la zona c es fácil de determinar, pues $x10 \to 1$, con $x \in \{0, 1\}$ y deduciendo $111 \to 0$ a partir de las reglas de evolución local.

La zona etiquetada con a no representa ninguna complicación, pues el estado de las células contiguas a la frontera del triángulo, puede ser o bien 0 ó 1 y seguirá dando el mismo

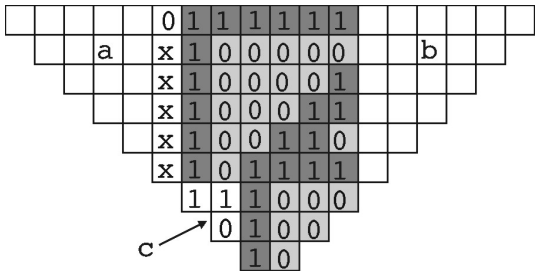

Figura 44. Estado conocido de las células que intervienen para formar la palabra$[t_5{}^{\nearrow} = \nearrow \sqcup\ t_0t_1t_0t_3 \nearrow$.

resultado; excepto por la célula marcada con 1 en la parte superior del dibujo. Así que dedicaremos nuestra atención a la zona b de la figura.

Utilizaremos el diagrama de Bruijn[32] para la regla 110 como herramienta para determinar las configuraciones anteriores que dan origen a las que se muestran en la figura, ayudándonos de las pistas que ya hemos establecido. El diagrama de Bruijn se muestra en la figura 45.

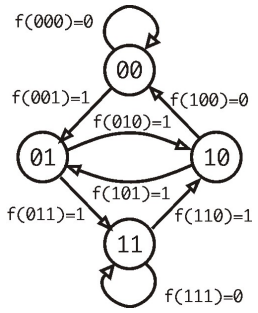

Figura 45. Las secuencias que son preimágenes de una cadena s, se determinan al seguir en las aristas, los valores de estados celulares de s. La secuencia s^{-1} se determina al unir los valores de los nodos, respetando el traslape de una célula.

La primera secuencia que trataremos es $s = 111000$, ubicada casi en la parte baja de la figura 44. Siguiendo los nodos: $01 - 10 - 01 - 11 - 11 - 11 - 11$ se obtiene la secuencia $s^{-1} = 01011111$. El único estado valioso es el 1 del extremo derecho. Este estado 1 nos hace deducir que el triángulo t_3 es adyacente a otro, pero no por su célula origen.

Ahora, tomando la cadena $s^{-1} = 01011111$ se obtiene la cadena $s^{-2} = 110011010$, siguiendo los nodos $11 - 10 - 00 - 01 - 11 - 10 - 01 - 10$. El primer estado 1, el de más a la izquierda de s^{-2}, es uno de los dígitos x, de manera que la siguiente palabra a considerar es $s^{-2} = 10011010$.

Con un procedimiento similar, se obtiene la secuencia de estados $s^{-3} = x100011100$, visitando los nodos $11 - 10 - 00 - 00 - 01 - 11 - 11 - 10 - 00$. Las anteriores secuencias son $s^{-4} = x1000011000$, $s^{-5} = x10000010000$ y $s^{-6} = 0111111100000$.

De manera que la secuencia de estados buscada es $xxxxx0111111100000$.

Hay aún otra manera de demostrar que una cadena $s = s_1 \dots s_k$ con cada $s_i \in \{0, 1\}$, $1 \leq i \leq k$, ha sido generada por una secuencia $s^{-1} = s_1 \dots s_{k+2}$. Esta manera es crear el diagrama de Bruijn con la profundidad deseada y luego seguir las aristas etiquetadas con los símbolos de s, formando una cadena con los símbolos de las etiquetas de los nodos, considerando un traslape de 3 células se forman nodos de 4 dígitos, véase la figura 46, sin embargo, buscar cadenas s^{-10} ya es casi imposible.

Para buscar en 3 generaciones anteriores se requiere determinar cadenas de longitud 7, lo que significa nodos de 6 símbolos, es decir $2^6 = 64$ nodos y $2^7 = 128$ ligas. Cadenas de 4 generaciones anteriores tienen cadenas de longitud 9, es decir, una gráfica con $2^8 = 256$ nodos y $2^9 = 512$ aristas. Y en general, para justificar el estado de una célula en g generaciones anteriores, se necesita un diagrama de Bruijn con 2^{2g} nodos y 2^{2g+1} ligas.□

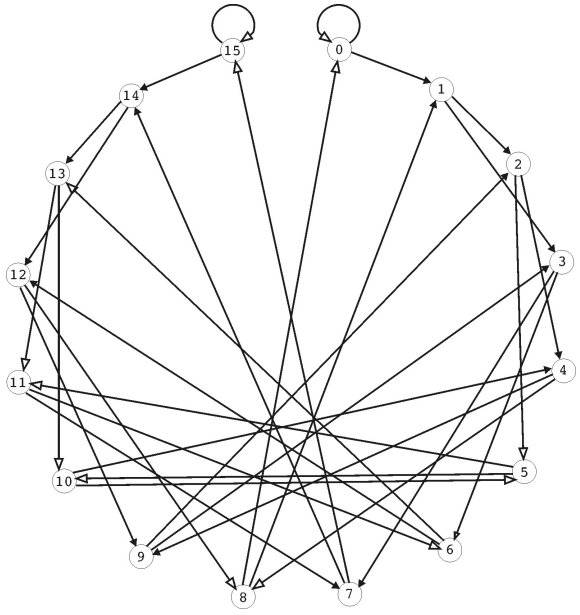

Figura 46. Cada arista es etiquetada con 1 (flecha negra) o con un 0 (flecha blanca), en dependencia del resultado de $\phi(\phi(xy_1y_2y_3z))$, donde $xy_1y_2y_3$ es la etiqueta de un nodo, que está unido a un nodo con etiqueta $y_1y_2y_3z$ por una secuencia de aristas de longitud 1.

4.2.2. *Asociación de palabras por la frontera superior.* Denotaremos por $[t_n{}^{\perp}$ a la palabra $\perp\ t_{\kappa_0}t_{\kappa_1}t_{\kappa_2}\ldots t_{\kappa_{n-1}}\ \perp$ asociada a un triángulo de tamaño n por su frontera superior. Los triángulos que forman la palabra $[t_n{}^{\perp}$ tienen una longitud acotada por el tamaño del triángulo al que está asociada esa palabra.

Cada triángulo t_n, con $n \geq 1$ termina en su parte inferior con una secuencia de células $< 0, 1 >$ vistas de derecha a izquierda; en el caso del t_0 termina con una secuencia de una célula, $< 0 >$. Para determinar la secuencia de células de la frontera superior hay que escoger subsecuencias que, de acuerdo con la regla de evolución local, generen un 1, es decir, aquellas que $c_{i_1}, c_{i_2}, c_{i_3} \to 1$.

La palabra $[t_n{}^{\perp}$ se lee de izquierda a derecha, pero se dibuja de derecha a izquierda, empezando con la casilla 0[1] Se puede iniciar cubriendo 1 lugar o 2 lugares de la frontera superior.

La regla es seguir una de las siguientes secuencias en ciclos $< 1, 2_a, 2_b >$, $< 2_a, 2_b, 1 >$ o $< 2_b, 1, 2_a >$ que son leídas de derecha a izquierda. Cada uno de esos números significa la cantidad de sitios que se deben cubrir con un triángulo, como se aprecia en la figura 47. Los subíndices son para distinguir entre los tipos de triángulos que pueden ocupar esas plazas. El 2_b debe ser un t_1 que siempre debe ir seguido de un t_0 que ocupa 1 lugar. De manera que las secuencias de triángulos deben ser $< t_0, t_{n>1}, t_1 >$, $< t_{n>1}, t_1, t_0 >$ o $< t_1, t_0, t_{n>1} >$.

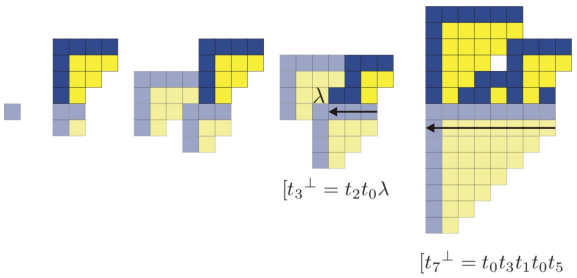

Figura 47. Palabra asociada por la frontera superior de un triángulo. La longitud de la palabra asociada por la frontera superior es $[[t_n{}^{\perp\delta} = \lceil \frac{3}{5}n \rceil$

Hay 3 triángulos por cada 5 lugares en la frontera superior. el procedimiento es colocar triángulos que cubran los espacios 2_b, 2_a y 1 en ese orden hasta que se cubra toda la frontera superior, o bien, toda la frontera superior menos un sitio, en cuyo caso se escribe $\sqcup$ para saber que no se cubrió completamente la frontera superior (ver la figura 47).

[1]En el primer capítulo, en la página 44, se establecieron las etiquetas para las casillas de las fronteras de los triángulos.

Es posible tener huecos en el cubrimiento generado por la concatenación gráfica de un triángulo con su palabra asocuada por la frontera superior, como se muestra en la figura de la extrema derecha en la ilustración 47. El cubrimiento que resulta entonces, pertenece al conjunto de cubrimientos prohibidos. Esto sugiere condiciones para terminar el proceso constructivo. El grafo de la figura 48 se puede observar el tamaño del los triángulos en el orden adecuado que se deben seguir para no tener palabras ilegales.

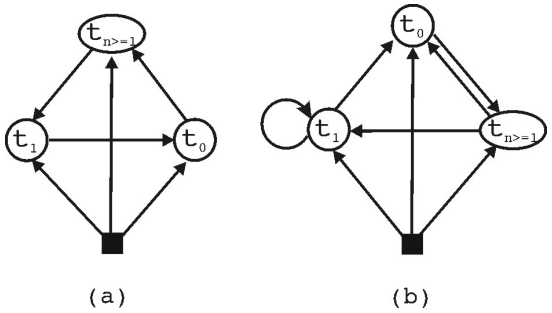

Figura 48. Grafos que ilustran el tamaño y el orden de los triángulos asociados a triángulo de tamaño n. (a) De una palabra asociada por el lado superior y (b) de una palabra asociada por el lado izquierdo

PROPOSICIÓN 5. *Es posible dar un procedimiento para encontrar una configuración del autómata celular lineal regla 110, que genera una evolución que contiene los mosaicos representados por las palabras asociadas por el lado superior de un triángulo regla 110.*

Prueba. En este caso se debe hacer el procedimiento descrito en la página 84, pero iniciando con una secuencia de estados 1, con la longitud deseada. Sin embargo se ha demostrado en [35] que no puede haber triángulos de tamaño 43 después de la décima generación, lo que significa que en una palabra asociada por la parte superior, no debe haber triángulos mayores que el t_{10}. □

4.2.3. *Asociación de palabras por la frontera izquierda.* Llamaremos $[t_n{}^{\dashv}$ a la palabra $\dashv\ t_{\kappa_0}t_{\kappa_1}t_{\kappa_2}\ldots t_{\kappa_{n-1}}\ \dashv$ asociada por el lado izquierdo de un triángulo t_n. Los triángulos que conforman la palabra asociada por la frontera izquierda, tocan al triángulo en 2 sitios, excepto si se trata de un t_0, que toca en 1 sitio. La longitud de la palabra también es menor que el tamaño del triángulo. También puede ser menor que la palabra asociada por el lado superior, ya que es obligatorio no ocupar el vértice superior izquierdo del triángulo, como se puede apreciar en los ejemplos de la figura 49.

Las palabras están escritas en secuencias $< (2_a)^*, 1, 2_b >$, $< 2_b(2_a)^*, 1 >$, o $< 1, 2_b, (2_a)^* >$, de nuevo, los números representan la cantidad de casillas que ocupan en la frontera izquierda del triángulo. Hemos distinguido dos tipos de triángulos que ocupan 2 espacios, los t_1 ocupan

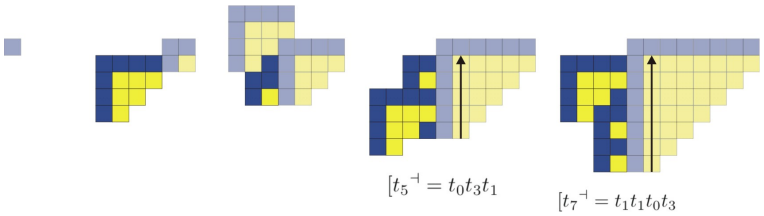

Figura 49. Palabra asociada por la frontera izquierda de un triángulo. $[[t_n{}^{\dashv\delta} \leq \lceil \frac{3}{5}n \rceil$

2_a casillas y pueden haber más de uno seguidos. Un $t_{n>1}$ ocupa 2_b espacios y siempre va seguido de un t_0 que ocupa 1 espacio.

El sentido de la escritura y lectura de las palabras del tipo $[t_n{}^{\dashv}$ es de izquierda a derecha, como la manera usual; sin embargo su correspondencia en el dibujo es de abajo hacia arriba, iniciando con la casilla etiquetada con 0 en el lado izquierdo (ver el orden de la enumeración en la página 44.

Proposición 6. *Es posible dar un procedimiento para encontrar una configuración del autómata celular lineal regla 110, que genera una evolución que contiene los mosaicos representados por las palabras asociadas por el lado izquierdo de un triángulo regla 110.*

Prueba. El procedimiento es el mismo que el descrito en la página 84.□

Proposición 7. *La longitud de las palabras sociadas por cualquiera de los lados de un triángulo es finita.*

Prueba. Es fácil desde que para algún $T \subset \mathbb{T}$, $|T| < \infty$, y la cantidad de símbolos que deben de ir concatenados a un tri'angulo depende del tamaño de éste triángulo. De manera que si la longitud de las cadenas está acotada por el tamaño del triángulo y existe un número limitado de símbolos, se sigue que la longitud de las palabras es finita.□

4.3. Ocurrencias. Decimos que la palabra P ocurre en la palabra Q si existe un par de palabras R y S tales que $Q = RPS$, también es válido decir que la palabra Q contiene a la palabra P. Con respecto a las palabras R y S decimos que son los segmentos inicial y final respectivamente (ver página 81 y siguientes). Cuando una palabra Q ocurre en una palabra P, lo denotamos como $(P \triangleleft Q)$ y cuando la palabra Q no ocurre en la palabra P se expresa como $(P \not\triangleleft Q)$.

Gráficamente también podemos hablar de ocurrencia, en el sentido de averiguar si cierta secuencia de triángulos ocurrren dentro de alguna palabra asociada a otro triángulo. Un

mosaico $Q \in \mathcal{X}^{(*)}$ ocurre en el mosaico $P \in \mathcal{X}^{(*)}$, denotado también como $(P \triangleleft Q)$ si existe un mosaico $R \in \mathcal{X}^{(*)}$ tal que $P = Q \bowtie R$ (ver pág 65). Podemos entonces enunciar la siguiente proposición con respecto a la ocurrencia de mosaicos dentro de otros.

PROPOSICIÓN 8. *Para cualquier palabra* $[t_n{}^{\nwarrow}, [t_n{}^{\perp}$, *o* $[t_n{}^{\dashv}$, *determinar si una palabra ocurre dentro de ellas, es un problema resoluble.*

Para probar la proposición anterior es suficiente saber que la longitud de las palabras es finita y luego hacer un proceso de búsqueda dentro de la palabra. Como $T \subset \mathbb{T}$ es finito, el tamaño de los triángulos también es finito. La longitud de las palabras $[t_n{}^{\nwarrow}, [t_n{}^{\perp}, [t_n{}^{\dashv}$ depende del tamaño del triángulo t_n, $[[t_n{}^{\nwarrow\delta} \ll \infty, [[t_n{}^{\perp\delta} \ll \infty, [[t_n{}^{\dashv\delta} \ll \infty$. □

El proceso de buscar una palabra en el alfabeto T dentro de otra palabra en el mismo alfabeto, es igual al estudiado en la teoría de lenguajes formales [10, 11, 25].

Un triángulo t_n pertenece al **interior de un mosaico**, cuando existen palabras $P, Q, R \neq \lambda$ tales que $P = [t_n{}^{\nwarrow}$, $Q = [t_n{}^{\dashv}$ y $R = [t_n{}^{\perp}$ y $P, Q, R \in T^*$. Es decir, que el triángulo t_n está rodeado de triángulos. Por el contrario, t_n pertenece a la **frontera del mosaico** si $(P \triangleleft \sqcup)$ o $(Q \triangleleft \sqcup)$ o $(R \triangleleft \sqcup)$ siendo $P = [t_n{}^{\nwarrow}$, $Q = [t_n{}^{\perp}$ y $R = [t_n{}^{\dashv}$. Si $P \in \mathcal{X}^{(*)}$, el interior de P es denotado por $\overset{\circ}{P}$, y la frontera de P es denotada por $\hat{P}$.

5. Sistemas de palabras

Vamos ahora a describir una clase de palabras que son usadas para expresar tanto los elementos de la palabra como la posición de la palabra en el gráfico. Primero definiremos un nuevo alfabeto, el alfabeto $T\omega$ y luego construiremos palabras que nos indiquen la posición de las palabras al rededor de un triángulo.

5.1. El alfabeto Tω y las $\omega-$palabras. Sea ahora $\omega \leftrightharpoons \{\perp, \dashv, \nwarrow\}$ y $T\omega \leftrightharpoons T \cup \omega$, en donde $T \cap \omega = \emptyset$. Las $\omega-$palabras son de forma:

$$w_{i_0} t_{n_1} \dots t_{n_{k-2}} w_{i_{k-1}} \text{ con } w_{i_0} = w_{i_{k-1}} \text{ y } \; w_i \in \omega; t_n \in T \tag{14}$$

De este modo, una $\omega-$palabra asociada por la diagonal a un triángulo $t_n \in T$ es de la forma $\nwarrow P \nwarrow$; de manera similar las $\omega-$palabras asociadas por la frontera superior y por la frontera izquierda de un triángulo t_n son respectivamente $\perp P \perp$ y $\dashv P \dashv$; donde P es una palabra compuesta por símbolos de T.

Los siguientes enunciados son verdaderos para las $\omega-$palabras, y su comprobación es inmediata de su definición:

(1) Si P es una $\omega-$palabra, cualquier segmento inicial de P que termine con un elemento de ω, es una $\omega-$palabra.
(2) Si P es una $\omega-$palabra, cualquier segmento terminal de P que inicie con un elemento de ω, es una $\omega-$palabra.
(3) Si $P, Q \in (T\omega)^*$, entonces $PQ \in (T\omega)^*$

La validez de las $\omega-$palabras es fácil comprobar, pues los símbolos ω que delimitan la palabra en T^* deben ser iguales. Cualquier palabra en T^* que sea de la forma:

$$w_{i_0} t_{n_1} \dots t_{n_{k-2}} w_{j_{k-1}}\ t_n ... \in T; w_i \neq w_j$$

es una palabra no válida porque no son delimitadas por símbolos ω.

Los siguientes son ejemplos de $\omega-$palabras válidas:

$\nearrow \sqcup t_0 t_3 t_0 t_1 t_0 t_5 t_5 t_1 t_0 \nearrow$**:** Una secuencia de triángulos concatenados gráficamente por la diagonal de un triángulo t_{10}, que se muestra al lado izquierdo de la figura 50

$\perp t_1 t_0 t_4 \sqcup \perp$**:** Una secuencia de triángulos concatenados gráficamente por la frontera superior de un triángulo t_5, que se muestra al lado derecho de la figura 50

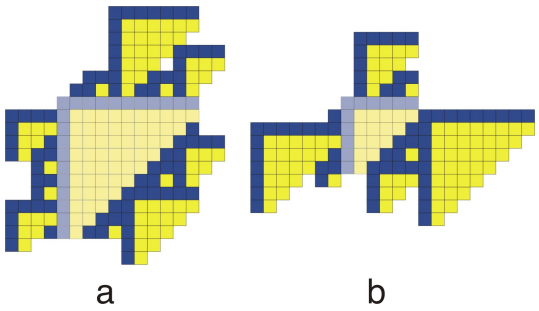

Figura 50. Palabra asociada por las fronteras de un a) t_{10} y asociadas a un b) t_5

6. Reglas de agregación

Para describir las maneras de concatenar gráficamente triángulos a un triángulo inicial, hemos incluído este sistema de reglas llamado "reglas de agregación", que a su vez fueron ideadas en base a las reglas de sustitución [11, 25], y a los esquemas [30, 31]. La diferencia fundamental entre reglas de agregación y de sustitución, es que en la reglas de agregación el mosaico resultante siempre va creciendo, y en las reglas de sustitución es posible que el tamaño del resultado sea menor.

El alfabeto $T\omega\gamma$
Sea $\gamma \leftrightharpoons \{\leftharpoonup, \multimap\}$ el alfabeto que contiene los símbolos que representan los tipos de agregaciones, donde $\leftharpoonup$ representa las agregaciones normales y $\multimap$ representa las agregaciones terminales. Llamaremos entonces $T\omega\gamma$ el alfabeto extendido de T, cuyos símbolos están en $(T \cup \omega \cup \gamma)$, considerando que $(\gamma \cap T\omega) = \emptyset$.

Reglas de agregación normal y terminal
Las palabras en $(T\omega\gamma)^*$ son concatenaciones de símbolos que pertenecen al alfabeto $T\omega\gamma$, sin embargo el lenguaje $T\omega\gamma$ involucra únicamente las palabras de la forma:

$$t_{n_0}\gamma_{i_1}\omega_{i_2}t_{n_3}\dots t_{n_{l-2}}\omega_{i_{l-1}} \tag{15}$$

En la expresión 15 se deben distinguir tres partes:

t_{n_0}**:** Llamada "el lado izquierdo" de la regla de agregación. Es un elemento de T. El sub-subíndice 0 indica la posición del símbolo en la palabra.
γ_{i_1}**:** Es un elemento del conjunto γ e indica el tipo de agregación.
$\omega_{i_2}t_{n_3}\dots t_{n_{l-2}}\omega_{i_{l-1}}$**:** es una $\omega-$palabra.

Las palabras de la forma $t_n \leftharpoonup Q$, donde $Q \in (T\omega)^*$ se llaman "reglas de agregación normal" y las palabras de la forma $t_n \multimap Q$ se llaman "reglas de agregación terminal". Cualquier regla de agregación es normal o terminal, ninguna regla de agregación normal puede ser terminal. Al tener un triángulo t_n y concatenar por alguno de sus lados una palabra, decimos que "se agrega la palabra Q al triángulo t_n" cuando Q ocurre en el lado derecho de alguna regla de agregación y t_n es el lado izquierdo. El resultado de esa acción es un mosaico de orden $[Q^\delta + 1$.

6.1. Esquemas. Si $\mathcal{L}(T\omega\gamma)$ es el conjunto de todas las palabras válidas en el alfabeto $T\omega\gamma$, consideremos un subconjunto $Z_T \subset \mathcal{L}(T\omega\gamma)$ de palabras con un significado válido. Llamaremos a Z_T un esquema en el alfabeto $T\omega\gamma$. No es difícil ver que $\ll Z_T$ *es un esquema*$\gg$

es decidible, puesto que todos los elementos de Z_T deben ser reglas de agregación y $\ll Q$ *es una regla de agregación* $\gg$ también es decidible, verificando cada una de las tres partes de las reglas de agregación.

Decimos que el esquema Z_T actúa en el triángulo t_n si $(R \triangleleft t_n)$ para alguna regla de agregación $R \in (T\omega\gamma)^*$ y ocurre precisamente en el lado izquierdo de la regla R. Si $z = t_n\gamma_i Q$ con $z \in Z_T$, actúa en algún triángulo t_n, podemos decir que el esquema Z_T transforma el mosaico t_n en el nuevo mosaico $t_n \bowtie Q$:

$$Z_T : t_n \Vdash Q \tag{16}$$

Si se cumple que $Z_T : t_n \Vdash [{t_n}^{\diagup}$, $Z_T : t_n \Vdash [{t_n}^{\perp}$ y $Z_T : t_n \Vdash [{t_n}^{\dashv}$, entonces el esquema Z_T es capaz de generar el mosaico $t_n \bowtie [{t_n}^{\diagup} \bowtie [{t_n}^{\perp} \bowtie [{t_n}^{\dashv}$. Las reglas de agregación se aplican a los triángulos que pertenecen a la frontera de un mosaico $P \in \mathcal{X}^{(*)}$. Al terminar el proceso de agregación, decimos el esquema Z_T finalmente genera el mosaico $Q \in \mathcal{X}^{(*)}$:

$$Z_T : P \Vdash \cdot\, Q \tag{17}$$

Si para algún esquema z_T se tiene que $Z_T : P \Vdash \cdot\, Q$, entonces existe un número entero positivo $i > 1$, tal que $Z_T : P_0 \Vdash P_1$; $Z_T : P_1 \Vdash P_2$; $\ldots$; $Z_T : P_{i-1} \Vdash P_i$, donde $P_0 = t_n$ y $P_i = Q$. Esto significa que si se puede aplicar un esquema a un mosaico P entonces $Z_T : P \Vdash \cdot\, Q$, pero no es cierto que $Z_T : P \Vdash Q$ porque $i > 1$. Notemos que las aplicaciones de las reglas de agregación se aplican a mosaicos de orden 1. La imposibilidad de aplicar un esquema a un mosaico la denotamos como:

$$Z_T : P \nVdash \tag{18}$$

Las palabras asociadas a un triángulo no son únicas; pueden existir más de una regla de agregación para el mismo lado izquierdo de la regla. Para simplificar la notación podemos escribir como $t_n \multimap \cdot$ aquellas reglas de agregación terminal aplicadas a un triángulo $t_n \in T$, que tengan en el lado derecho una expresión $\omega_1\lambda\omega_1$, donde $\omega_1 \in \omega$.

Ejemplo del uso de los esquemas Sea $T \rightleftharpoons \{t_0, t_1, t_2, t_3, t_4, t_5\}$, y el esquema Z_T definido con las reglas de agregación que se muestran en la tabla 10. El proceso de agregación se inicia con cualquier triángulo, digamos por ejemplo el t_5. Este proceso se debe hacer al mismo tiempo que se van colocando los triángulos en un 2-espacio discreto, como se ilustra en la figura 51.

Regla n^o			
1.-	t_5	$\longmapsto$	$\perp\, t_1 t_0 t_4 \sqcup\, \perp$
2.-	t_5	$\longmapsto$	$\nearrow \sqcup\, t_0 t_1 t_0 t_3 \nearrow$
3.-	t_5	$\longmapsto$	$\dashv\, t_1 t_0 t_4 t_0\, \dashv$
4.-	t_4	$\longmapsto$	$\nearrow \sqcup\, t_0 t_1 t_0 \nearrow$
5.-	t_3	$\multimap$	$\perp\, t_1 t_0 \sqcup\, \perp$
6.-	t_3	$\longmapsto$	$\nearrow \sqcup\, t_0 t_3 \nearrow$
7.-	t_1	$\multimap$	$\perp\, t_0\, \perp$
8.-	t_1	$\longmapsto$	$\nearrow t_1 \nearrow$
9.-	t_1	$\multimap$	$\dashv\, t_0\, \dashv$
10.-	$\cdot$	$\multimap$	$\cdot$

Cuadro 10. Esquema Z_T

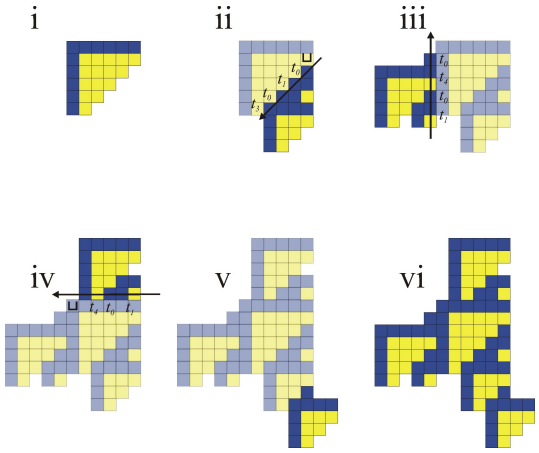

Figura 51. Ejemplo del uso de un esquema Z iniciando con un mosaico t_5, y usando las reglas 2,3,1,7 y 8

7. Evoluciones de Post

Como las reglas de agregación se pueden aplicar una a cada momento, eso nos abre la posibilidad de introducir el concepto del tiempo en las aplicaciones de las reglas de agregación. El tiempo que usaremos está segmentado en pasos discretos, y en cada momento de tiempo se aplica una sola regla. Así, llamamos *Evoluciones de Post* bajo el esquema Z, denotándolo por $\mathcal{P}_Z$, a las transformaciones de un mosaico en el transcurso del tiempo, debido a las aplicaciones de las reglas de agregación definidas en un esquema. El nombre de "evoluciones de Post" ha sido escogido al recordar la mecánica de construcción de una solución al problema

de correspondencia de Post [45], en donde se colocan las piezas del dominó de acuerdo a las reglas establecidas.

Al inicio del proceso constructivo se elige uno de los triángulos del alfabeto T que llamaremos *palabra inicial*, a este triángulo podemos distinguir como el mosaico inicial $p^{(0)}$. Si $Z : p^{(0)} \Vdash Q$, la palabra (o el mosaico) Q corresponde a la primera aplicación del esquema Z sobre la palabra inicial $p^{(0)}$. De manera que la dinámica generada por las evoluciones de Post se puede describir como: $\mathcal{P}_Z(p^{(n-1)}) = p^{(n)};\ n = 1, 2, \ldots$.

Cuando es completamente claro cuál es el esquema que se usa, se puede eliminar de la notación el subíndice Z. En forma de composición de funciones, podemos expresar las evoluciones de Post aplicando el esquema Z al mosaico inicial p como:

$$\begin{aligned} \mathcal{P}(p) = \mathcal{P}^{(1)}(p) = p^{(1)} \\ \mathcal{P}(\mathcal{P}(p)) = \mathcal{P}^{(2)}(p) = p^{(2)} \ldots \end{aligned} \tag{19}$$

Evoluciones de Post acordes al tiempo de evolución Es claro que el autómata celular lineal regla 110 evoluciona en pasos de tiempo que avanzan, el hecho de concatenar gráficamente palabras por la parte superior de un triángulo, significa dibujar células que existieron en anteriores pasos de tiempo. Siempre que se haga una debida justificación de tales reglas, es posible colocarlas. Una posible justificación es tener una evolución normal y verificar la existencia de esas reglas de agregación por ellado superior.

Normalmente usaremos las reglas de agregación por los lados izquierdos y diagonales, lo que significa ir avanzando en el tiempo de acuerdo con la evolución normal del autómata celular lineal regla 110.

El estudio de la dinámica del crecimiento de los mosaicos debido a la aplicación de los esquemas, ofrece actividades interesantes, como descubrir el mosaico que dio origen a determinado subcubrimiento, y también predecir el mosaico que se producirá después de suficientes aplicaciones de un esquema sobre un triángulo inicial.

Descubrir el triángulo inicial es un problema semidecidible, puesto que en las aplicaciones del esquema se conserva el triángulo inicial debido a que las reglas son de agregación y no de substitución. Se vuelve indecidible cuando el cubrimiento es un cubrimiento completo y cuando hay más de una regla de agregación con lados derechos diferentes; si los lados derechos son iguales, entonces los lados izquierdos deben ser diferentes.

Para encontrar la historia de las evoluciones de Post, se pueden trazar líneas de derivación [34, 35] en cada triángulo, estas líneas permiten ver el flujo de las evoluciones en cada uno

de los tres lados. Esta técnica es bastante útil para determinar el origen de las evoluciones de Post mediante evoluciones inversas.

Regla n^o		
1.-	t_8	$\longleftarrow \quad \bot\, t_1t_0t_5t_1t_0 \sqcup \bot$
2.-	t_8	$\longleftarrow \quad \nwarrow \sqcup\, t_5t_0t_1t_0t_3t_0t_1 \nwarrow$
3.-	t_8	$\longleftarrow \quad \dashv\, t_1t_1t_0t_3t_0\, \dashv$
4.-	t_6	$\longleftarrow \quad \bot\, t_5t_0t_1t_2\, \bot$
5.-	t_6	$\longleftarrow \quad \nwarrow \sqcup\, t_0t_1t_0t_5t_0 \nwarrow \mid \nwarrow \sqcup\, t_0t_1t_0t_3t_0 \nwarrow$
7.-	t_6	$\longleftarrow \quad \dashv\, t_0t_4t_1t_0\, \dashv$
8.-	t_5	$\longleftarrow \quad \bot\, t_5t_1t_0 \sqcup \bot \mid \bot\, t_3t_1t_0 \sqcup \bot \mid \bot\, t_4t_1t_0 \sqcup \bot \mid \bot\, t_1t_0t_2\, \bot$
12.-	t_5	$\longleftarrow \quad \nwarrow t_3t_0t_1t_0t_5 \nwarrow \mid \nwarrow \sqcup\, t_0t_3t_0t_1 \nwarrow$
14.-	t_5	$\longleftarrow \quad \dashv\, t_1t_0t_4t_0\, \dashv \mid \dashv\, t_3t_1t_1\, \dashv \mid \dashv\, t_1t_0t_6\, \dashv \mid \dashv\, t_1t_1t_0\, \dashv$
18.-	t_4	$\multimap \quad \bot\, t_0t_6t_0 \sqcup \bot \mid \bot\, t_1t_0t_4\, \bot$
20.-	t_4	$\longleftarrow \quad \nwarrow \sqcup\, t_0t_4t_0 \nwarrow \mid \nwarrow \sqcup\, t_0t_1t_0 \nwarrow \mid \nwarrow t_6t_0t_1t_0 \nwarrow$
23.-	t_4	$\longleftarrow \quad \dashv\, t_1t_1t_0\, \dashv \mid \dashv\, t_3t_1t_0\, \dashv \mid \dashv\, t_0t_2t_0\, \dashv$
26.-	t_3	$\multimap \quad \bot\, t_1t_0 \sqcup \bot \mid \bot\, t_3t_0 \sqcup \bot$
28.-	t_3	$\longleftarrow \quad \nwarrow t_4t_0t_1 \nwarrow \mid \nwarrow \sqcup\, t_2t_0 \nwarrow \mid \nwarrow \sqcup\, t_0t_3 \nwarrow$
31.-	t_3	$\longleftarrow \quad \dashv\, t_2t_0\, \dashv \mid \dashv\, t_1t_0\, \dashv \mid \dashv\, t_0t_3\, \dashv$
34.-	t_2	$\multimap \quad \bot\, t_0t_4\, \bot \mid \bot\, t_3\, \bot$
36.-	t_2	$\multimap \quad \nwarrow \sqcup\, t_0 \nwarrow \mid \nwarrow t_1, t_0 \nwarrow$
28.-	t_2	$\longleftarrow \quad \dashv\, t_5\, \dashv \mid \dashv\, t_2\, \dashv \mid \dashv\, t_3\, \dashv \mid \dashv\, t_1t_0\, \dashv \mid \dashv\, t_4t_0\, \dashv$
43.-	t_1	$\multimap \quad \bot\, t_1\, \bot \mid \bot\, t_0\, \bot \mid \bot\, t_5\, \bot$
46.-	t_1	$\longleftarrow \quad \nwarrow t_1 \nwarrow \mid \nwarrow t_2 \nwarrow \mid \nwarrow \sqcup \nwarrow$
49.-	t_1	$\multimap \quad \dashv\, t_0\, \dashv \mid \dashv\, t_8\, \dashv \mid \dashv\, t_1\, \dashv \mid \dashv\, t_3\, \dashv \mid \dashv\, t_6\, \dashv$
54.-		$\multimap \quad \cdot$

Cuadro 11. Esquema Z de las derivaciones en 3 direcciones. La enumeración se conserva para cada regla, así, la regla 7 contiene a la regla 8 también.

7.1. Identificación única de cubrimientos. A cada paso del tiempo en el desarrollo de las agregaciones de triángulos, se va modificando el cubrimiento. Es posible identificar de manera única a cada cubrimiento del siguiente modo:

(1) Localizar la célula ubicada más arriba y más a la izquierda. Tal célula será llamada el "origen del recorrido del cubrimiento".
(2) Recorrer cada célula de la frontera del cubrimiento en el sentido de las manecillas del reloj y escribir el siguiente código:
   1: si la dirección es $\rightarrow$

2: si la dirección es ↓
3: si la dirección es ←
4: si la dirección es ↑

Esta clase de identificación es muy útil al usar herramientas como los mapas de retorno al llevar un rastro de la dinámica del cubrimiento.

7.2. Evoluciones inversas. Se aplican las evoluciones inversas para encontrar un mosaico unitario dentro de un mosaico que ha sido creado a partir de aplicaciones sucesivas de un esquema. El procedimiento es como sigue:

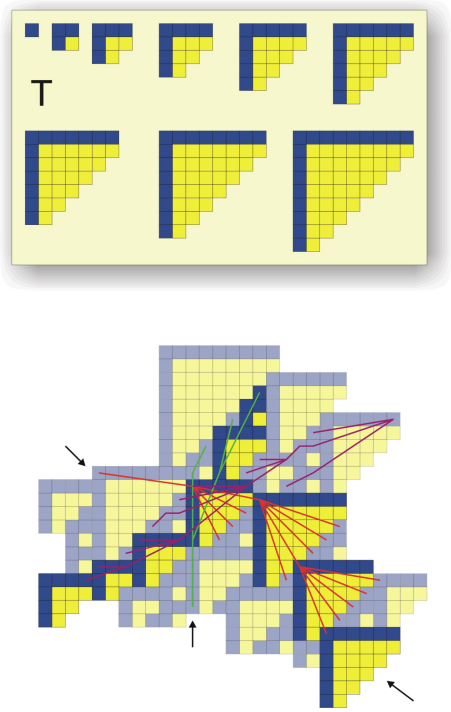

Figura 52. Árboles de derivaciones en las 3 direcciones.

(1) Elegir un triángulo de la frontera diagonal, llamemos t_n a este triángulo. Los triángulos que pertenecen a esta frontera se pueden hallar en la parte inferior derecha de cualquier mosaico.
(2) Seleccionar del esquema Z un lado derecho asociado a la diagonal del triángulo t_n elegido, llamemos P a esa palabra. P debe ser tal que $(P \triangleleft t_n)$ en cualquier posición.
(3) Elegir el triángulo asociado a P, es decir, el lado izquierdo de la regla de agregación $t_n\gamma_1 P$, en donde $\gamma_1 \in \gamma$; y volver al paso anterior.

(4) Repetir los pasos 2 y 3 hasta que ocurra que no se puede hallar un lado derecho adecuado, o bien que se encuentre la frontera del mosaico, lo que significa haber alcanzado la frontera superior.

Un ejemplo de este procedimiento (y los otros dos similares) se ilustra el la figura 52, usando el alfabeto $T \leftrightharpoons \{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, \}$, y el esquema Z descrito en la tabla 11.

El triángulo buscado será aquel que comparta los tres flujos de derivaciones, marcados con líneas en el dibujo. El procedimiento sobre las palabras diagonales que se muestra en el dibujo, corresponde a la secuencia inversa de las reglas de agregación : $< 11, 7, 26 >$, lo que significa que iniciando el proceso computacional se agregó al t_4 inicial la palabra $\nwarrow t_6 t_0 t_1 t_0 \nwarrow$, luego se eligió derivar el t_6 agregando la palabra $\nwarrow \sqcup\ t_0 t_1 t_0 t_5 t_0 \nwarrow$, en donde se eligió agregar al t_5 la palabra $\nwarrow t_5 t_1 t_0 \sqcup \nwarrow$. Un procedimiento similar ocurre en las otras dos fronteras.

8. Conclusiones

- Los triángulos definen un alfabeto de símbolos con los que se pueden formar palabras; estas palabras tienen un significado gráfico en el diagrama espacio-temporal de las evoluciones de la regla 110.

- A cada triángulo se le puede asociar un sistema de 3 palabras, que se pueden concatenar gráficamente en cada uno de sus lados.

- Es posible definir un conjunto de reglas de agregación llamado esquema, que significan formas de concatenar triángulos de manera legal.

- Las evoluciones de Post nos muestran los cambios que experimenta un mosaico a cada aplicación de una regla de agregación.

- Se puede suponer una historia de agregaciones, al trabajar con las evoluciones inversas de Post.

Capítulo 5

Computación basada en los mosaicos R110

1. Resumen

Los algoritmos tienen un significado muy particular en esta parte del trabajo, antes de describir formalmente un algoritmo basado en mosaicos r110, un algoritmo es una secuencia ordenada de aplicaciones de las reglas de agregación, para obtener un mosaico en particular.

La frase "secuencia ordenada... para obtener" implica la intención de llevar a cabo un proceso con un método y con un objetivo. Para describir esta clase de algoritmos es necesario tener un lenguaje, que llamaremos el *lenguaje del algoritmo*, que está compuesto por todas las palabras válidas en el alfabeto definido $T \subseteq \mathbb{T}$.

Un algoritmo es una prescripción que determina el curso de cierto proceso constructivo [30, 31]. Tomaremos este punto de vista en nuestra tarea de generar mosaicos mediante la concatenación gráfica de triángulos. Un algoritmo en el alfabeto T consiste en generar palabras del mismo alfabeto de manera sucesiva, siguiendo una secuencia de reglas de agregación establecidas.

La aplicación de un algoritmo se describe con detalle en este capítulo, de manera general, el proceso empieza con una *palabra inicial*, una palabra en el alfabeto definido. Esa palabra inicial es un mosaico de orden 1 y es seleccionada de manera arbitraria. Luego, se aplican las reglas de agregación a los mosaicos de la frontera. La finalización del algoritmo puede ocurrir de dos maneras:

- Mediante la aplicación de una regla de agregación terminal
- Mediante la semejanza gráfica del mosaico obtenido con otro mosaico, que debe ser miembro de un conjunto meta llamado *conjunto de mosaicos objetivo*. Dependiendo de la métrica usada, se debe dar un parámetro de suficiencia para establecer la semejanza del mosaico obtenido con un miembro del conjunto de mosaicos objetivo.

El mosaico obtenido al finalizar el proceso algorítmico es conocido como el *resultado de la aplicación* del algoritmo sobre la palabra inicial. En términos de las evoluciones de Post, decimos que el algoritmo transforma el mosaico inicial en el resultado de la aplicación

del algoritmo. Por eso, decimos que si el procedimiento consecutivo de aplicar reglas de agregación sobre un mosaico inicial termina, entonces el algoritmo es aplicable a ese mosaico inicial.

2. Algoritmos basados en mosaicos

De manera formal, un algoritmo es una tupla $\mathfrak{A} \leftharpoons (T, \mathbf{T}, Z')$ en donde $T \subseteq \mathbb{T}$ es un conjunto finito de triángulos r110; $\mathbf{T} = T\omega\gamma$ es el alfabeto extendido de T, como fue definido en la página 92, y $Z' \subseteq Z$ es subconjunto de un esquema definido en el alfabeto $T\omega\gamma$. Los símbolos que pertenecen al alfabeto $\omega \cup \gamma$ sirven para delimitar las palabras construidas.

Un algoritmo $\mathfrak{A}$ queda completamente determinado al dar la lista de reglas de agregación, es decir, su esquema con un orden de aplicación. Esto nos permite saber que T es el alfabeto usado, y que $\mathfrak{A}$ se ha presentado con la norma establecida; podremos decir entonces que $\mathfrak{A}$ es un algoritmo normal.

Usaremos el vocabulario de la tabla 12 para describir las actividades de un algoritmo $\mathfrak{A} \leftharpoons (T, \mathbf{T}, Z)$. Este vocabulario ha sido retomado de [30, 31] para respetar la notación creada:

Símbolo	Significado
$\mathfrak{A} : P \vDash Q$	El algoritmo $\mathfrak{A}$ transforma el mosaico P en el mosaico Q
$\mathfrak{A} : P \vDash \cdot\, Q$	Después de sucesivas aplicaciones, el algoritmo $\mathfrak{A}$ transforma el mosaico P en el mosaico Q.
$!\mathfrak{A}(P)$	El algoritmo $\mathfrak{A}$ se aplica al mosaico P.
$\mathfrak{A} : P\rceil$	El algoritmo $\mathfrak{A}$ no se aplica al mosaico P
$(\mathfrak{A} : P)$	Es el número de aplicaciones de las reglas de agregación del algoritmo $\mathfrak{A}$ iniciando en el mosaico P.
$\mathfrak{A}(P)$	La aplicación del algoritmo $\mathfrak{A}$ sobre el mosaico P
$\mathfrak{A} : P \Rightarrow Q$	El algoritmo $\mathfrak{A}$ transforma en una o en varias aplicaciones el mosaico P en el mosaico Q

Cuadro 12. Vocabulario relacionado con algoritmos basados en mosaicos r110.

El ciclo operativo del algoritmo

Un algoritmo basado en mosaicos r110 tiene el siguiente ciclo de operación:

- Se toma un triángulo $P \in \mathcal{X}^{(1)}$ que pertenece a la frontera de un mosaico.

- Se busca en la lista de reglas de agregación, una que tenga en su lado izquierdo el mosaico P, y en el lado derecho de la regla, una palabra del alfabeto $Q = T\omega$; que será la palabra asociada por alguno de los tres lados del triángulo P.
- Se genera el mosaico que resulta de concatenar gráficamente P con Q, es decir, el resultado de la primera aplicación del algoritmo $\mathfrak{A}$ sobre P es $P \bowtie Q$; usando nuestro nuevo vocabulario se expresa como: $\mathfrak{A} : P \vDash P \bowtie Q$.
- Se verifican las condiciones de la detención del proceso constructivo.
- Si es posible, se sigue el proceso constructivo tomando un triángulo de la frontera de $P \bowtie Q$,
- y se repite el proceso descrito mientras exista una regla de agregación aplicable a los triángulos de la frontera del mosaico.
- Pero si $\mathfrak{A} : P\rceil$ o sucede que $\mathfrak{A} : Q \vDash \cdot\ S$, el proceso debe terminar y el resultado de la aplicación del algoritmo $\mathfrak{A}$ sobre el mosaico P, es $\mathfrak{A}(P) = P$ en el primer caso y S en el segundo caso.

La única manera que un algoritmo puede terminar es que $!\mathfrak{A}(P)$; si esto es cierto, entonces $\mathfrak{A} : P \Rightarrow \mathfrak{A}(P)$; si $\mathfrak{A} : P\rceil$ entonces simplemente $\mathfrak{A} : P \Rightarrow P$.

Una manera de escribir un algoritmo normal, es dar su esquema Z en forma de lista, escribiendo primero las reglas de agregación que afectan a los triángulos en el lado izquierdo de la regla, desde el triángulo mayor hasta el menor. Esto es una regla de orden únicamente.

Otra forma de escribir un algoritmo normal es de forma lineal. Para describirlo de este modo se requiere hacer las sustituciones que se marcan en la tabla 13. El alfabeto $\mathbf{T}' \leftrightharpoons T \cup \{\text{d,s,l,b,f,n,g,c}\}$.

Símbolo en $\mathbf{T}$	Símbolo en $\mathbf{T}'$	Significado
$\diagup$	d	Palabra asociada por la diagonal
$\perp$	s	Palabra asociada por el lado superior
$\dashv$	l	Palabra asociada por el lado izquierdo
$\sqcup$	b	Lugar en blanco
$\multimap$	f	Regla de agregación terminal
$\hookleftarrow$	n	Regla de agregación normal
/	c	reglas de cerradura (se describe adelante)
	g	inicio/fin de una regla (se describe adelante)

Cuadro 13. Sustituciones para presentar un algoritmo normal en forma lineal.

Los arreglos tipográficos son:

- El símbolo de agregación normal por 'n'.
- El símbolo de agregación terminal por 'f'.
- Los símbolos de inicio/fin de palabra asociada, por sus respectivos símbolos, de acuerdo a la tabla 13.
- En cada regla de agregación se agrega el símbolo g, al inicio y al final de la cadena de símbolos de la regla.

Como ejemplo, consideremos el algoritmo normal $\mathfrak{A}$, definido por el esquema de la tabla 14. $\mathfrak{A}$ produce un mosaico como el que se muestra en la figura 53.

Usando el alfabeto $\mathbf{T}'$, $\mathfrak{A}$ se presenta de forma lineal como:

gt3ndbt0t3dgt3nst3bsgt3nlbt0t3lgt3fgt0fg

Regla n^o		
1.-	t_3	$\longleftarrow \quad \nearrow \sqcup\ t_0 t_3 \nearrow$
2.-	t_3	$\longleftarrow \quad \perp\ t_3 t_0\ \perp$
3.-	t_3	$\longleftarrow \quad \dashv\ t_0 t_3\ \dashv$
4.-	t_3	$\multimap \quad \cdot$
5.-	t_0	$\multimap \quad \cdot$

Cuadro 14. Esquema $Z_{\mathfrak{A}}$ del algoritmo $\mathfrak{A} \leftrightharpoons \{\{t_3, t_0\}, \{t_3, t_0\} \cup \omega \cup \gamma, Z_{\mathfrak{A}}\}$. Con estas reglas es posible hacer el *ether*

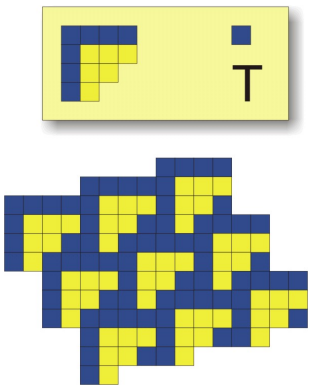

Figura 53. mosaico generado por aplicaciones sucesivas del algoritmo $\mathfrak{A}$ definido en la tabla 14.

Esta notación lineal nos ofrece ventajas, al usar un mecanismo de lectura lineal con movimientos hacia adelante y hacia atras, como la cabeza lectora de un mecanismo sobre

una cinta. Sin embargo seguiremos presentando los algoritmos en forma de una lista de reglas de sustituciones porque es más claro a la lectura.

2.1. Operaciones con algoritmos. Con los conjuntos de reglas de agregación de los algoritmos basados en mosaicos podemos hacer las operaciones usuales de conjuntos, que también son útiles al trabajar con algoritmos.

2.1.1. *Contención de algoritmos.* Un algoritmo $\mathfrak{A}$ contiene a otro algoritmo $\mathfrak{B}$, si todas las reglas de agregación de $\mathfrak{B}$ pertenecen al conjunto de reglas de agregación de $\mathfrak{A}$.

Esta operación de contención la representamos de la misma manera que en la teoría de conjuntos, donde $\mathfrak{B} \subset \mathfrak{A}$ significa que el conjunto de reglas de agregación de $\mathfrak{A}$, contiene al conjunto de reglas de $\mathfrak{B}$. Podemos decir que $\mathfrak{B}$ es un subalgoritmo de $\mathfrak{A}$.

Por otro lado, si $\mathfrak{B} \subset \mathfrak{A}$, entonces $T_{\mathfrak{B}} \subseteq T_{\mathfrak{A}}$.

2.1.2. *Unión de algoritmos.* La unión de algoritmos está definida en pares de algoritmos, y produce como resultado un nuevo algoritmo que contiene todas las reglas de agregación de ambos algoritmos. Si $\mathfrak{A} y \mathfrak{B}$ son algoritmos normales y $Z_{\mathfrak{A}}, Z_{\mathfrak{B}}$ son sus esquemas correspondientes, la unión de algoritmos está definida como:

$$\mathfrak{A} \cup \mathfrak{B} \leftrightharpoons (T_{\mathfrak{A}} \cup T_{\mathfrak{B}}, \mathbf{T}_{\mathfrak{A}} \cup \mathbf{T}_{\mathfrak{B}}, Z_{\mathfrak{A}} \cup Z_{\mathfrak{B}}) \tag{20}$$

Regla n^o			
1.-	t_3	$\longmapsto$	$\prec t_1 t_0 t_3 \prec$
2.-	t_3	$\longmapsto$	$\perp\, t_1 t_0 \sqcup \perp$
3.-	t_3	$\longmapsto$	$\dashv\, t_3 t_1\, \dashv$
4.-	t_1	$\longmapsto$	$\prec \sqcup \prec$
5.-	t_1	$\longmapsto$	$\perp\, t_3\, \perp$
6.-	t_1	$\longmapsto$	$\dashv\, t_0\, \dashv$
7.-		$\multimap$	$\cdot$

Cuadro 15. Algoritmo $\mathfrak{B}$

Entonces, si $!\mathfrak{A}(P)$ o bien $!\mathfrak{B}(P)$, entonces con toda seguridad $!\mathfrak{A} \cup \mathfrak{B}(P)$, porque en el nuevo conjunto de reglas de agregación existe al menos una regla, ya sea del esquema de $\mathfrak{A}$ o del esquema de $\mathfrak{B}$, que se pueda aplicar a la palabra P. Sean por ejemplo, el algoritmo $\mathfrak{A}$ como en la tabla 14, y $\mathfrak{B}$ el algoritmo definido por el esquema de la tabla 15. La unión de los algoritmos $\mathfrak{A}$ y $\mathfrak{B}$ se muestra en la tabla 16.

Regla n^o			
1.-	t_3	$\longmapsto$	$\nwarrow \sqcup\ t_0 t_3 \nwarrow$
2.-	t_3	$\longmapsto$	$\nwarrow t_1 t_0 t_3 \nwarrow$
3.-	t_3	$\longmapsto$	$\perp\ t_3 t_0\ \perp$
4.-	t_3	$\longmapsto$	$\perp\ t_1 t_0 \sqcup\ \perp$
5.-	t_3	$\longmapsto$	$\dashv t_0 t_3\ \dashv$
6.-	t_3	$\longmapsto$	$\dashv t_3 t_1\ \dashv$
7.-	t_1	$\longmapsto$	$\nwarrow \sqcup \nwarrow$
8.-	t_1	$\longmapsto$	$\perp\ t3\ \perp$
9.-	t_1	$\longmapsto$	$\dashv t_0\ \dashv$
10.-		$\multimap$	$\cdot$

Cuadro 16. Algoritmo $\mathfrak{C} = \mathfrak{A} \cup \mathfrak{B}$

Note que en la tabla 16 aparecen las reglas de agregación que ocurren en el conjunto de reglas de $\mathfrak{A}$ o en el conjunto de reglas de $\mathfrak{B}$ y se han omitido las repeticiones.

La unión de algoritmos tiene mucha relación con la coexistencia de familias de mosaicos, ya vista con anterioridad en el capítulo referente a los cubrimientos del plano con mosaicos propios, en la página 75. Enseguida vamos a enunciar algunas implicaciones que surgen al aplicar nuestra aproximación de las reglas de agregación. Un ejemplo se puede observar en la figura 54

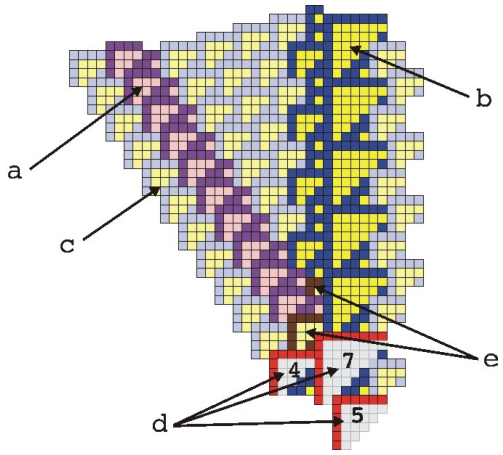

Figura 54. La unión de los algoritmos que generan el mosaico propio tipo A (a), el mosaico propio tipo C (b) y el *ether* (c), generan espacios que deben ser ocupados por símbolos que no estan en el alfabeto que resulta de esa unión.

t_3	$\longmapsto$	$\prec \sqcup t_0 t_3 \prec$
t_3	$\longmapsto$	$\dashv t_0 t_3 \dashv$
	$\multimap$	

Cuadro 17. Esquema del algoritmo $\mathfrak{A}_1$ que genera mosaicos tipo *ether*

t_3	$\longmapsto$	$\prec t_1 t_0 t_3 \prec$
	$\multimap$	

Cuadro 18. Esquema del algoritmo $\mathfrak{A}_2$ que genera mosaicos tipo A

t_1	$\longmapsto$	$\prec t_6 \prec$
t_6	$\longmapsto$	$\prec \sqcup t_0 t_3 t_0 t_1 t_0 \prec$
t_6	$\longmapsto$	$\dashv t_1 t_0 t_3 t_1 \dashv$
	$\multimap$	

Cuadro 19. Esquema del algoritmo $\mathfrak{A}_3$ que genera mosaicos tipo C

Sean entonces los algoritmos $\mathfrak{A}_1$ como se muestra en la tabla 17, $\mathfrak{A}_2$ como se muestra en la tabla 18 y $\mathfrak{A}_3$ como se muestra en el cuadro 19. Al hacer la unión de estos algoritmos, tenemos el nuevo algoritmo $\mathfrak{A}_4 = \mathfrak{A}_1 \cup \mathfrak{A}_2 \cup \mathfrak{A}_3 = (\mathfrak{A}_1 \cup \mathfrak{A}_2) \cup \mathfrak{A}_3 = \mathfrak{A}_1 \cup (\mathfrak{A}_2 \cup \mathfrak{A}_3)$.

Este algoritmo nuevo $\mathfrak{A}_4$ tiene como alfabeto $T_{\mathfrak{A}_4} = \{t_0, t_1, t_3, t_6\}$ con el esquema que se muesrta en el cuadro 20

t_1	$\longmapsto$	$\prec t_6 \prec$
t_3	$\longmapsto$	$\prec t_1 t_0 t_3 \prec$
t_3	$\longmapsto$	$\prec \sqcup t_0 t_3 \prec$
t_3	$\longmapsto$	$\dashv t_0 t_3 \dashv$
t_6	$\longmapsto$	$\prec \sqcup t_0 t_3 t_0 t_1 t_0 \prec$
t_6	$\longmapsto$	$\dashv t_1 t_0 t_3 t_1 \dashv$
	$\multimap$	

Cuadro 20. Esquema del algoritmo $\mathfrak{A}_4 = \mathfrak{A}_1 \cup \mathfrak{A}_2 \cup \mathfrak{A}_3$ que genera mosaicos de los algoritmos $\mathfrak{A}_1$,$\mathfrak{A}_2$ y $\mathfrak{A}_3$

.

Después de aplicar el algoritmo $\mathfrak{A}_4$ descubrimos dos cosas:

(1) En la figura 54 se observa que no hay reglas de agregación que nos ayuden a agregar los triángulos que se señalan con la flecha etiquetada con “e”

(2) En la misma figura 54 se observa que no hay símbolos en el alfabeto que sirvan para considerar los triángulos de tamaños 4,5 y 7, señalados por la flecha "d" de manera que de la aplicación del algoritmo resulta el mosaico que se muestra en la figura 55

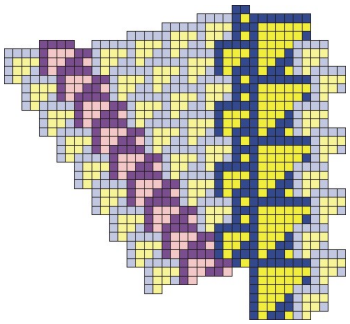

Figura 55. Aplicación del algoritmo $\mathfrak{A}_4$

La decisión de aceptar el resltado del algoritmo $\mathfrak{A}_4$ (figura 55) dependerá enteramente de la métrica usada y de los criterios de diferencia al comparar este resultado con los modelos de solución proporcionados (figura 54).

2.1.3. *Intersección de algoritmos.* La intersección de algoritmos es una operación definida de manera similar que la unión de los algoritmos. Está definida por la intersección de los esquemas de cada algoritmo involucrado en la operación.

$$\mathfrak{A} \cap \mathfrak{B} \leftrightharpoons (T_{\mathfrak{A}} \cap T_{\mathfrak{B}}, \mathbf{T}_{\mathfrak{A}} \cap \mathbf{T}_{\mathfrak{B}}, \mathcal{Z}_{\mathfrak{A}} \cap \mathcal{Z}_{\mathfrak{B}}) \tag{21}$$

El resultado de esta operación posiblemente causará que $\mathfrak{A} \cap \mathfrak{B} : P\rceil$, es decir, que no exista regla alguna que se pueda aplicar al mosaico P. Lo que ocasiona que el resultado del algoritmo $\mathfrak{A} \cap \mathfrak{B}(P) = P$.

2.1.4. *Cerradura de un algoritmo.* La cerradura de un algoritmo $\mathfrak{A}$ será denotada por $\bar{\mathfrak{A}}$ y es el conjunto de reglas de agregación que se pueden aplicar a cualquier triángulo. La estructura de las reglas que pertenecen a $\bar{\mathfrak{A}}$ establece que el lado izquierdo sea cualquier triángulo, seguido de un indicador para determinar la clase de regla, terminal o normal; y en el lado derecho la palabra agregada.

$$\lambda\gamma_{i_1}\omega_{i_2}t_{n_3}\ldots t_{n_{l-2}}\omega_{i_{l-1}} \tag{22}$$

En la expresión 22 el símbolo λ (nulo) indica que puede ser aplicada esta regla en cualquier triángulo, en los siguientes ejemplos, el símbolo λ será sustituído por un lugar vacío. El

símbolo ω_{i_2} indica el tipo de regla (terminal o normal) y el resto de la expresión es la palabra a ser agregada; terminando con el mismo símbolo $\omega_{i_{l-1}}$ con que se delilmitó la palabra asociada.

Un caso particular son las reglas de cerradura dedicadas a cubrir los huecos de la diagonal de cualquier triángulo de tamaño n. Son entonces dos reglas en este tipo: la regla que cubre los huecos t_0 en posiciones pares de la diagonal y la regla que cubre esos huecos en las posiciones impares.

Como convención diremos que $\leftharpoondown\ /$ representa la regla de cubrir los espacios t_0 de la diagonal, y será puesta en todos los algoritmos para indicar que se puede aplicar esa regla con el fin de cubrir los espacios de tamaño 0 en la diagonal. Con lo anterior, el algoritmo $\mathfrak{C}$ de la tabla 16 puede ser reescrito como se muestra en la tabla 21.

El símbolo de agregación normal indica que se puede aplicar esta regla más de una ocasión sin hacer que el proceso constructivo termine. La finalidad es asegurarse que el mosaico generado no contenga huecos y así pertenezca al conjunto de mosaicos prohibidos.

Hemos de notar que cuando el triángulo asociado (a la izquierda de las reglas) es el mismo t_0, la única posibilidad en las reglas de cerradura es $\leftharpoondown\ \cdot$ y esto es precisamente porque el triángulo t_0 no deriva palabra alguna. Finalmente podemos decir que la cerradura de un algoritmo basado en mosaicos es una medida que apoya la legalidad del mosaico construido.

Regla n^o		
1.-	t_3	$\leftharpoondown\ \measuredangle \sqcup\ t_0 t_3 \measuredangle \mid \measuredangle t_1 t_0 t_3 \measuredangle$
3.-	t_3	$\leftharpoondown\ \perp t_3 t_0 \perp \mid \perp t_1 t_0 \sqcup \perp$
5.-	t_3	$\leftharpoondown\ \dashv t_0 t_3 \dashv \mid \dashv t_3 t_1 \dashv$
7.-	t_1	$\leftharpoondown\ \measuredangle \sqcup \measuredangle$
8.-	t_1	$\leftharpoondown\ \perp t_3 \perp$
9.-	t_1	$\leftharpoondown\ \dashv t_0 \dashv$
10.-		$\multimap\ /$
11.-		$\multimap\ \cdot$

Cuadro 21. Algoritmo $\mathfrak{C} = \mathfrak{A} \cup \mathfrak{B}$ con cerradura

3. Máquina celular de computación basada en mosaicos R110

Una máquina celular de computación basada en mosaicos [del tipo r110], es una dispositivo teórico que describe el proceso constructivo de crear un mosaico a partir de un triángulo.

El criterio de semejanza elegido (ver páginas 54 y siguientes) nos permite decidir el margen de diferencias permitido, entre el mosaico obtenido y el mosaico deseado. El término "celular" evoca el origen de esta clase de mosaicos.

DEFINICIÓN 14. *Una máquina celular de computación basada en mosaicos está definida mediante una tupla* $\mathfrak{M} \leftrightharpoons (\mathfrak{A}, p, \mathcal{O}, \mu)$*; donde* $\mathfrak{A}$ *es un algoritmo,* $p \in \mathcal{X}^{(1)}$ *es un triángulo r110;* $\mathcal{O}$ *un conjunto de mosaicos denominado "mosaicos objetivo" y* μ *una métrica. El conjunto de todas las máquinas de esta clase también es denotado por* $\mathfrak{M}$*.*

Los datos de entrada de la máquina son los triángulos del alfabeto definido; los mosaicos que resulten después de sucesivas aplicaciones de las reglas de agregación, los conoceremos como el producto o el resultado de la aplicación de la máquina $\mathfrak{M}$ sobre el mosaico p.

Los elementos del conjunto de mosaicos objetivo $\mathcal{O}$, deberán tener como característica de identidad que sea el mismo centro, para al menos garantizar una semejanza importante con el mosaico inicial. Esto no es una restricción importante, pues al medirlos respecto a la métrica resultarán suficientemente distintos como para no considerarlos como solución. Podemos dar una muestra de algunas máquinas de computación basada en mosaicos que reproducen algunos *gliders* que han sido catalogados y estudiados en [34, 35]:

Glider D1: Es compuesto principalmente por triángulos t_4, teniendo como conjunto fundamental $\{t_{4_1}, t_{4_2}, t_{3_1}, t_{3_2}, t_{3_1}, t_{2_1}, t_{1_{1\ldots7}}, t_{0_{1\ldots6}}\}$, de donde obtenemos el alfabeto T usado, $T \leftrightharpoons \{t_4, t_3, t_2, t_1, t_0\}$, estos elementos se ilustran a la izquierda en la figura 56. El conjunto de mosaicos objetivo, también se muestra en esa figura, arriba de cada mosaico.

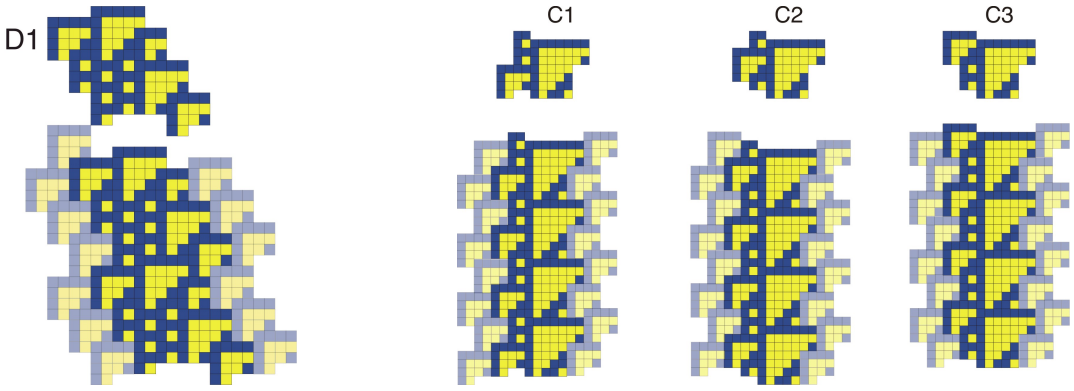

Figura 56. Izquierda: *glider* D1, rodeado de *ether*. Derecha: Familia de *gliders* Cx, rodeados de *ether*

Regla n^o			
1.-	t_4	$\longmapsto$	$\nwarrow \sqcup\ t_2t_0t_1 \nwarrow \mid \nwarrow \sqcup t_0t_3t_0 \nwarrow$
3.-	t_4	$\longmapsto$	$\perp\ t_0t_4t_1 \perp \mid \perp\ t_3t_0t_2 \perp$
5.-	t_4	$\longmapsto$	$\dashv t_1t_0t_3 \dashv \mid \dashv t_1t_1 \dashv$
7.-	t_3	$\longmapsto$	$\nwarrow \sqcup\ t_0 \sqcup \nwarrow \mid \nwarrow t_1t_0t_3 \nwarrow \mid \nwarrow \sqcup t_0t_3 \nwarrow$
10.-	t_3	$\longmapsto$	$\perp\ t_3t_0 \sqcup \perp \mid \perp\ t_1t_0 \sqcup \perp$
12.-	t_3	$\longmapsto$	$\dashv t_0t_3 \dashv \mid \dashv t_4t_0 \dashv$
14.-	t_2	$\longmapsto$	$\nwarrow \sqcup t_0 \nwarrow$
15.-	t_2	$\longmapsto$	$\perp\ t_3 \sqcup \perp$
16.-	t_2	$\longmapsto$	$\dashv t_1t_0 \dashv$
17.-	t_1	$\longmapsto$	$\nwarrow t_1 \nwarrow \mid \nwarrow t_4 \nwarrow \mid \nwarrow \sqcup \nwarrow$
20.-	t_1	$\longmapsto$	$\perp\ t_1\ \perp \mid \perp\ t_0 \sqcup \perp$
22.-	t_1	$\longmapsto$	$\dashv t_1 \dashv \mid \dashv t_3 \dashv$
24.-		$\longmapsto$	$\diagup$
25.-		$\multimap$	$\cdot$

Cuadro 22. Máquina celular que calcula un mosaico gráficamente igual a un *glider* D

El esquema de esta máquina, que es capaz de generar el *glider* $D1$ se muestra en la tabla 22. Este *glider* avanza a la derecha 2 células en 10 pasos de tiempo, por eso se dice que su velocidad es $\frac{2}{10}c$. En [34] la configuración del *glider* $D1$ excluye el t_3 de la izquierda, pero aquí es incluido debido a la necesidad de tener un mosaico propio. Aún hay otro glider denominado $D2$ que difiere del $D1$ en 2 triángulos t_1 concatenados gráficamente debajo del t_4 superior en el mosaico mostrado aparte, en la misma figura 56 a la izquierda.

Este ejemplo muestra que la retícula inducida por este mosaico propio tiene una base que coincide con la retícula inducida por el mosaico *ether* compuesto por t_3 y t_0.

Glider C: El *glider* C tiene un conjunto fundamental $\{t_6, t_3, t_{1_1}, t_{1_2}, t_{1_3}, t_{0_1}, t_{0_2}, t_{0_3}, t_{0_4}\}$. El alfabeto usado por esta máquina es $T \rightleftharpoons \{t_6, t_3, t_1, t_0\}$. Y el conjunto de mosaicos objetivo se muestra en la figura 56 en el lado derecho. Este *glider* tiene velocidad 0. Estos tres miembros de la familia C se caracterizan por tener en el lado izquierdo un triángulo t_3 que funciona muy bien para acoplarse al *ether*. Por el lado derecho, una combinación de un t_1 con espacio justo para un t_3 es la combinación necesaria para acoplarse con el *ether* por ese lado, como se muestra en la parte derecha de la figura 56. El esquema de la máquina celular basada en mosaicos que genera este mosaico se muestra en la tabla 23.

Regla n^o			
1.-	t_6	$\longmapsto$	$\nwarrow \sqcup\ t_0t_3t_0t_1t_0 \nwarrow$
2.-	t_6	$\longmapsto$	$\perp\ t_3t_1t_0t_6\ \perp$
3.-	t_6	$\longmapsto$	$\dashv t_1t_1t_0t_3 \dashv \mid \dashv t_1t_0t_3t_1 \dashv \mid \dashv t_3t_1t_1t_0 \dashv$
6.-	t_3	$\longmapsto$	$\nwarrow \sqcup\ t_0t_1 \nwarrow \mid \nwarrow \sqcup t_0t_3 \nwarrow \mid \nwarrow t_6t_0t_1 \nwarrow$
9.-	t_3	$\longmapsto$	$\perp\ t_1t_0 \sqcup \perp \mid \perp\ t_3t_0 \sqcup \perp \mid$
11.-	t_3	$\longmapsto$	$\dashv t_0t_3 \dashv \mid \dashv t_1t_0 \dashv \mid \dashv t_0t_6 \dashv$
14.-		$\longmapsto$	$\diagup$
15.-		$\multimap$	$\cdot$

Cuadro 23. Máquina celular que calcula los mosaicos de la familia de *gliders* C

4. Problemas indecidibles

4.1. El problema de la palabra basado en mosaicos. Si $\mathfrak{A}$ es un algoritmo normal basado en mosaicos, y $\alpha, \beta \in \mathcal{X}^{(*)}$ dos mosaicos en el alfabeto de $\mathfrak{A}$; el problema de la palabra expresado en términos de mosaicos (*TWP Tile Word Problem*), se puede expresar como un problema de decisión de la siguiente manera:

$$TWP(\mathfrak{A}, \alpha, \beta) = \begin{cases} 1 & \text{si} \quad \mathfrak{A} : \alpha \Rightarrow \beta \\ 0 & \quad\ \text{e.o.c.} \end{cases} \tag{23}$$

Hay condiciones **necesarias** que deben cumplirse para que $TWP(A, \alpha, \beta) = 1$, aun cuando cada una de ellas no es suficiente:

Condición de posibilidad: $\|\alpha\| \leq \|\beta\|$. El primer caso es cuando $\|\alpha\| = \|\beta\|$, esto significa que $\mathfrak{A} : \alpha\rceil$ (página 101); lo que a su vez implica que $\delta_{\mathfrak{A}}(\alpha, \beta) = 0$ (página 54). El segundo caso es cuando $\|\alpha\| < \|\beta\|$. Esto significa que $(\mathfrak{A} : \alpha) \neq 0$. Si $\|\alpha\| > \|\beta\|$, entonces $TWP(\mathfrak{A}, \alpha, \beta) = 0$, puesto que no hay regla de agregación que elimine mosaicos, de manera que se reduzca el orden de β.

Condición de frontera: Si $s = pqr \in Z_{\mathfrak{A}}$ es una regla del esquema, entonces $(\hat{\beta} \triangleleft r)$, siendo r una $T\omega-$palabra; esto es que en la frontera del mosaico objetivo, debe ocurrir la parte derecha de alguna regla de agregación del esquema del algoritmo $\mathfrak{A}$.

4.2. El problema de la detención basado en mosaicos. Uno de los problemas clásicos que se deben mencionar al usar transformaciones basadas en reglas, es el problema

de la detención, en el que hay que determinar si existe una solución para una determinada máquina.

Enumeración de las máquinas celulares de computación basadas en mosaicos. Podemos enlistar todas las máquinas celulares de computación basadas en mosaicos, en algún orden arbitrario, si denotamos por $\mathfrak{M}_i$ una tupla que defina una máquina de este tipo, $\mathfrak{M}_1, \mathfrak{M}_2, \mathfrak{M}_3, \ldots, \mathfrak{M}_n$ es una lista de las primeras n máquinas que hemos decidido considerar. Se pueden ordenar estas máquinas, en base al número obtenido a partir de la traducción de $\mathfrak{M}$ al alfabeto de dos símbolos $\{0, 1\}$, que es el tema siguiente.

Si $[\mathfrak{M}^{\#}$ es el número que le corresponde a la máquina $\mathfrak{M}$, entonces podemos hacer que *halt* sea una función que está definida en el dominio del producto cartesiano de los números naturales y el conjunto de los mosaicos de orden 1; $halt : \mathbb{Z} \times \mathcal{X}^{(1)} \to \{0, 1\}$. Si $x = [\mathfrak{M}_{\mathfrak{i}}{}^{\#}$ y $y \in \mathcal{X}^{(1)}$, la función **detener** está definida como:

$$halt(x, y) = \begin{cases} 1 & \text{si } !\mathfrak{M}_x(y) \\ 0 & e.o.c. \end{cases} \tag{24}$$

Es decir, diremos que la máquina eventualmente se podría detener si es posible aplicar la máquina $\mathfrak{M}_i$ al mosaico y; siempre que se verifique la existencia del conjunto cerradura $\bar{\mathfrak{M}}$, en particular aquellas en donde $(\multimap \triangleleft Q)$, y Q es una $\mathbf{T}$-palabra.

Además de aplicar el algoritmo a un mosaico, debemos saber si se ha llegado a una solución. Para eso debemos suponer que se conoce al menos un miembro del conjunto de soluciones. Una solución para una máquina es un mosaico, y considerando un parámetro de semejanza, podemos dar un conjunto de mosaicos que permisiblemente sean diferentes con un criterio μ.

Si el umbral de semejanza es $\mu < 1$, podemos garantizar que el mosaico que resulte del proceso constructivo y un mosaico solución, al menos coinciden en el centro, puesto que si difieren en el centro del mosaico, se tiene que $\delta_b(x, y) = \frac{1}{2^0} = 1$, para cualesquiera mosaicos $x, y \in \mathcal{X}^{(1)}$. Si $\mu < 1$ y a medida que μ se acerque a ser 0, se va restringiendo más el espacio de soluciones permitido para la máquina celular de computación basada en mosaicos. De manera que cuando $\mu = 0$ se permitirá solamente una solución.

Se hace funcionar esta máquina por un intervalo de tiempo que estimamos suficiente, como para obtener un mosaico que se pueda considerar como una posible solución. Se pueden dar dos puntos de vista al respecto. Uno es determinar el volumen del mosaico solución y considerando esa característica, aproximar el resultado del algoritmo hasta tener un mosaico de tamaño similar. El otro punto de vista tiene que ver con el parecido o el

grado de diferencia entre el resultado obtenido y una solución. La estimación se da cuando suponemos el número de evoluciones de Post que se pueden hacer antes de rebasar el umbral de diferencias.

Acerca del tamaño del mosaico. Si sabemos el rango de volúmenes de uno los mosaicos del conjunto objetivo $\mathcal{O}$, podemos calcular el máximo número de evoluciones de Post necesarias antes de entrar en ese rango, al sumar el volumen de los mosaicos de menor orden que son agregados. Por otro lado, también podemos calcular el mínimo número de evoluciones de Post necesarias, al sumar el orden de los mosaicos con mayor número de triángulos.

Acerca del grado de diferencias. El otro punto de vista para estimar el número de evoluciones de Post que es necesario esperar antes de encontrar una solución, tiene que ver con las diferencias encontradas. Para estimar este número procederemos de manera inversa que en el caso anterior. Supondremos que cada evolución genera las menores diferencias, de manera que podamos obtener un límite máximo en el número de evoluciones.

Sin embargo, estas mínimas diferencias ocurren en los vectores que son más largos, de manera que hay que determinar en el conjunto de vectores propuesto, cuál es el tamaño del vector más largo, porque ese vector proporcionará las menores diferencias. Lo que nos indica de nuevo un radio de mosaico que se puede usar para estimar el número de evoluciones necesarias.

4.3. Traducción de un algoritmo basado en mosaicos a un algoritmo de 2 símbolos. Para generalizar la descripción de una máquina de computación basada en mosaicos, de manera que se pueda usar esa descripción por otra máquina, daremos un procedimiento que traduce cualquier algoritmo basado en mosaicos a una secuencia de unos y ceros. Consideraremos el alfabeto $\mathbf{T}' \cap \delta$ ordenado en una secuencia:

símbolo	g	n	d	s	l	b	f	c	δ	t_0	t_1	t_2	$\ldots$	t_n
índice	1	2	3	4	5	6	7	8	9	10	11	12	$\ldots$	$n+10$

(25)

Ahora vamos a definir la operación *traduce* denotándola con $[P^{\tau}$, donde P debe ser una palabra en el alfabeto $\mathbf{T}'$, definido con los elementos de la secuencia 25, denotada por el símbolo Γ; el resultado de esa operación será una nueva palabra en el alfabeto $\{0, 1\}$. La definición de la operación de traducción es inductiva en las palabras $P \in \mathbf{T}'$. El orden en los elementos de la secuencia ha sido arbitrario, pero considerando la frecuencia de aparición de los símbolos en el algoritmo.

$$[\Lambda^{\tau} \leftrightharpoons \Lambda$$
$$[Pa_i{}^{\tau} \leftrightharpoons [P^{\tau}10^i1$$

El símbolo a_i es el $i-$ésimo elemento de la secuencia 25, que será sustituído por i ceros entre un par de unos. De esta manera corresponde un número natural para cada máquina de computación basada en mosaicos, lo que se puede aprovechar para identificar esa máquina.

Como ejemplo, tomemos una de las máquinas más simples que generan un cubrimiento legal, el cubrimiento utilizando el mosaico propio t_1, cuya descripción de la máquina aparece en la tabla 24.

Regla n^o			
1.-	t_1	$\leftarrowtail$	$\measuredangle t_1 \measuredangle$
2.-	t_1	$\leftarrowtail$	$\perp\, t_1\, \perp$
3.-	t_1	$\leftarrowtail$	$\dashv t_1 \dashv$
4.-		$\multimap$	$\diagup$
5.-		$\multimap$	

Cuadro 24. Máquina celular basada en mosaicos que calcula el cubrimiento con el conjunto $\{t_1\}$

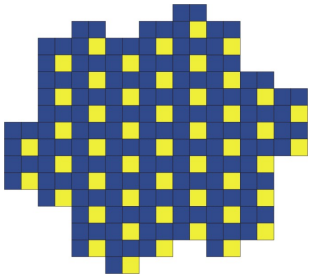

Figura 57. Cubrimiento generado con las reglas de agregación de la tabla 24

En primer lugar debemos obtener la palabra $P \in \mathbf{T}'$ que corresponde a la máquina mostrada en la tabla 24. Esta palabra es la que se muestra enseguida.

g t1 n d t1 d g t1 n s t1 s g t1 n l t1 l g f c g f g (26)

Que después de traducirla al alfabeto $\{0,1\}$ produce la siguiente palabra.

$$\begin{aligned}\mathfrak{A}^{Rp} = \; & 10110000000000011001100011000000000000110001101 \\ & 10000000000011001100001100000000000001100001101 \\ & 100000000000110011000011000000000000011000001101 \\ & 10000000110000000011011000000001101\end{aligned} \tag{27}$$

Cada secuencia de n ceros seguidos, corresponde al $n-$ésimo elemento de la secuencia 25. Esta nueva secuencia de ceros y unos genera un único número binario que por supuesto, tiene un correspondiente número en el sistema decimal, que nos puede servir para identificar cada máquina, lo que se relaciona directamente con los números de Gödel [**10**, **53**]. Como se puede apreciar, la máquina celular basada en mosaicos más sencilla genera un número extremadamente grande, lo que hace pensar en las máquinas que generan cubrimientos más complejos, como la máquina que se establece en la tabla 11.

5. Universalidad de los algoritmos basados en mosaicos

Uno de los problemas fundamentales de un algoritmo universal, es determinar una representación que abarque cualquier otro algoritmo, $\mathfrak{A}^{Rp}\delta P$ será la representación del algoritmo $\mathfrak{A}$ aplicado al mosaico P. Donde el algoritmo universal $\mathfrak{U}$ debe tomar como palabra inicial $\mathfrak{A}^{Rp}\delta P$ y dar el mismo resultado que el obtenido de aplicar el algoritmo $\mathfrak{A}$ con el dato inicial P. Ahora describiremos la construcción de un algoritmo $\mathfrak{U}$ sobre $\mathbf{T}'\delta$ tal que se cumpla $\mathfrak{U}(\mathfrak{A}^{Rp}\delta P) \simeq \mathfrak{A}(P)$.

A la descripción 27 debemos agregar los símbolos δt_1, que indican una separación y el triángulo que servirá de dato inicial. Con lo que la representación de la máquina basada en mosaicos que cubre el plano con copias del mosaico propio t_1 es la siguiente, donde aparecen en negrillas la parte δt_1.

$$\begin{aligned}\mathfrak{A}^{Rp}\delta t_1 = \; & 10110000000000011001100011000000000000110001101 \\ & 1000000000001100110000110000000000001100001101100000000000001 \\ & 1001100000110000000000011000001101100000001100000000001 \\ & 10110000000110\mathbf{1000000000110000000000001}\end{aligned} \tag{28}$$

$\mathfrak{U}(\mathfrak{A}^{Rp}\delta t_1)$ es el algoritmo universal $\mathfrak{U}$ que recibe como dato de entrada la descripción de la máquina $\mathfrak{A}$ y el dato inicial de $\mathfrak{A}$, con lo que se puede obtener el mismo comportamiento que $\mathfrak{A}(t_1)$. El funcionamiento de este algoritmo universal es:

Fase de recuperación y especificación de datos: Se define una función llamada "función de recuperación" que recibe como único parámetro una palabra en el alfabeto de dos símbolos y genera una palabra en el alfabeto completo $\mathbf{T}'\delta$.

Para cualquier máquina universal con una palabra $\Pi = \mathfrak{A}^{Rp}\delta P$ como dato de entrada $\mathfrak{U}(\Pi)$, Π es una palabra en el alfabeto de dos símbolos $\{0, 1\}$, como se muestra en el ejemplo 28. La fase de recuperación consiste en tomar una palabra en este alfabeto (el de dos símbolos) y obtener una palabra en el alfabeto $\mathbf{T}'\delta$, que contiene la descripción de la máquina de computación basada en mosaicos y el dato inicial.

El procedimiento de esta función de recuperación es leer los símbolos cero que sean consecutivos entre dos marcas de unos, y colocar el número correspondiente, luego, separando con una coma el siguiente número, que representa la longitud de la siguiente secuencia de ceros; y así sucesivamente hasta terminar la palabra. Tomando el ejemplo 27, el resultado de la fase de recuperación es la secuencia de números decimales:

$$1, 11, 2, 3, 11, 3, 1, 11, 2, 4, 11, 4, 1, 11, 2, 5, 11, 5, 1, 7, 8, 1, 7, 1, 9, 11 \tag{29}$$

Posteriormente, sustituir el símbolo cuyo índice es el dígito en la secuencia obtenida, para de esa forma obtener una palabra en el alfabeto $\mathbf{T}'\delta$. Siguiendo la reconstrucción del ejemplo, tenemos la $\mathbf{T}'\delta$-palabra:

$$\text{g t1 n d t1 d g t1 n s t1 s g t1 n l t1 l g f c g f g } \delta \text{ t1} \tag{30}$$

Ahora tomando la palabra 30 podemos convertira en una palabra en $\mathbf{T}$ haciendo las debidas transformaciones (ver tabla 13), resultando la palabra 31.

$$\mathrm{g}t_1 \leftarrowtail \nwarrow t_1 \nwarrow \mathrm{g}t_1 \leftarrowtail \perp\, t_1 \perp \mathrm{g}t_1 \leftarrowtail \dashv t_1 \dashv \mathrm{g} \multimap \diagup \mathrm{g} \multimap \mathrm{g}\delta t_1 \tag{31}$$

Los símbolos 'g' marcan el inicio y el final de cualquier regla de agregación, entonces podemos obtener nuevamente la tabla 24 al representar la descripción 31 en forma de lista como en la tabla 25.

t_1	$\leftarrowtail$	$\nwarrow t_1 \nwarrow$
t_1	$\leftarrowtail$	$\perp\, t_1 \perp$
t_1	$\leftarrowtail$	$\dashv t_1 \dashv$
	$\multimap$	$\diagup$
	$\multimap$	

Cuadro 25. Conjunto de reglas para para la descripción de la máquina $\mathfrak{A}$

Además de tener la palabra δt_1 que indica una separación δ y el símbolo inicial t_1, con que se inicia el proceso de agregación de la máquina $\mathfrak{A}$.

Fase de aplicación y resultado: Se aplica el algoritmo descrito al dato inicial dado y se observa el resultado obtenido en base a una métrica elegida.

Normalmente se aplica una métrica basada semejanzas con un parámetro $\mu \leq 1$ por las razones descritas en la página 8.

6. Conclusiones

- Las regularidad con se agregan los triángulos ofrece la posibilidad de definir y aplicar reglas de agregación. Dando así la idea de aplicar un algoritmo (página 100).

- Una máquina celular de computación basada en mosaicos, calcula mosaicos creados con triángulos originados en las evoluciones de la regla 110.

- El clásico problema de la palabra se puede aplicar aquí, porque se trata de semigrupos aditivos.

- Una máquina celular de computación universal con evoluciones de Post, toma como dato de entrada la representación de una máquina computacional basada en mosaicos, que debe incluir el dato de entrada de esa máquina.

Conclusiones y trabajos posteriores

Las principales conclusiones de este trabajo están descritas en función de algunas características de los algoritmos[53].

(1) *Un algoritmo tiene un conjunto de instrucciones de tamaño finito.* En todos los algoritmos que mostramos el número de instrucciones de agregación es finito, pero esto no es una restricción. No hay restricciones del alfabeto de triángulos usado, pues hemos definido $\mathbb{T}$ como el conjunto de todos los triángulos; a la fecha, aún permanece abierto el problema de determinar el máximo tamaño posible para un triángulo, que no ocurra como parte de la configuración inicial del autómata celular. Esto sugiere que cualquier triángulo puede ocurrir del lado izquierdo de una regla de agregación, lo que implica que el conjunto de instrucciones puede crecer hasta donde se requiera.

A pesar de lo anterior, se puede pensar en resolver algunos problemas relacionados, uno de ellos ya se ha mencionado, el de encontrar el tamaño máximo de un triángulo. Otro es determinar el número de reglas que se aplican, y cómo se va modificando este número después de aplicar las reglas de agregación.

(2) *Hay un agente que reacciona ante las instrucciones y considera la respuesta.* Usualmente este agente es una persona, al menos por ahora. La respuesta que da este tipo de máquinas es un mosaico, decidir si la respuesta es correcta o no, depende principalmente de la métrica utilizada. Las métricas propuestas en esta tesis, tienen la ventaja de ser aplicables a mosaicos de orden muy grande, tan grande como se quiera. Sin embargo se convierte en una desventaja si las diferencias ocurren cerca de las fronteras, porque el grado de diferencia decrece de manera exponencial a medida que la longitud del vector aumenta. Esto hace que los vectores moderadamente largos ya no tengan influencia significativa en las diferencias de los mosaicos.

Se puede idear otro tipo de métrica que considere las diferencias en las fronteras, pensando en que si alguna diferencia ocurre cerca del centro del mosaico, esa diferencia puede ser más significativa en las fronteras a medida que el mosaico crezca.

Esto parece ser cierto basados en que el espacio de un triángulo no puede ser ocupado por otros triángulos menores (ver página 50).

(3) *Hay maneras de crear, almacenar y recuperar información de un proceso algorítmico.* A pesar de que teóricamente no hay límites ni para el máximo número de símbolos usados, ni para el número de reglas; si hay límites de espacio físico, tanto de almacenamiento como de representación. Estas restricciones se deben considerar en el mismo sentido que se ha considerado el tamaño infinito del conjunto de símbolos y de instrucciones, es decir, también de manera teórica. Podemos dar descripciones teóricas del funcionamiento de un lector de instrucciones desde una cinta infinita, también de manera teórica.

Posteriormente se debe trabajar en crear un método de identificación para cada mosaico, de manera que pueda ser almacenado y reconstruido sin tener que seguir la historia de las evoluciones de Post que se usaron al crearlo. Una manera puede ser una secuencia de pares de números (n, v), que dicen que el triángulo t_n debe ser colocado en la posición que indica el vector v.

Otras conclusiones se han obtenido al hacer observaciones del comportamiento de las reglas de agregación. En el capítulo 2 se hizo mención acerca de las familia de la regla 110. Las siguientes son los cambios necesarios para considerar los triángulos que se originan en las otras reglas que pertenecen a la misma familia, de acuerdo a sus características observables:

(1) **Cambiar los estados 0 a 1 y 1 a 0, pero manteniendo la misma orientación**. Se requiere solamente hacer las correspondientes modificaciones en las definiciones 1, 2, 3 y 4 de la página 43.

La nota ahora debe decir: "El estado de la célula origen (i, j) siempre debe ser **0**. Luego, todas las células que pertenecen a la frontera superior $U_{k,(i,j)}$, o bien a la frontera izquierda $L_{k,(i,j)}$ son células de estado 1. Las células que pertenecen al patrón triangular $T_{k,(i,j)}$ son células de estado **1**".

(2) **Cambiar la orientación de los triángulos pero sin modificar su estado**. Los cambios también deben hacerse a las definiciones 1 y 3, en donde ahora se extienden a la izquierda para que la frontera quede a la derecha, en lugar de extenderse a la derecha y que la frontera quede a la izquierda.

Otros cambios se deben hacer en las definiciones de las palabras asociadas por la frontera izquierda, que, claramente ahora deben ser palabras asociadas por la frontera derecha. Sin embargo, la dirección de la lectura de las palabras asociadas por la frontera superior ahora debe ser de izquierda a derecha.

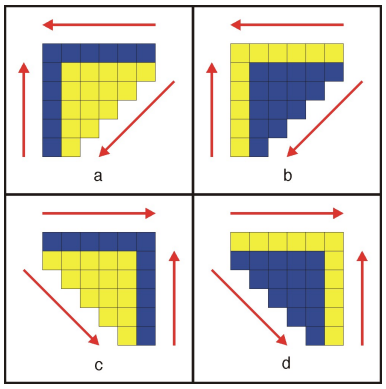

Figura 58. Dirección de lectura de los triángulos generados por la regla: (a) 110; (b) 137; (c)124; (d)193

(3) **Cambiar los estados y cambiar la dirección de los triángulos** Primero es necesario hacer los cambios de estado y posteriormente los cambios a los sistemas de palabras.

Con los cambios descritos, todas las demás definiciones y pruebas se deben hacer sin mayores complicaciones, de manera que es posible decir que tanto la regla 110, como las reglas 137, 124 y 193 hacen computación universal.

Cuando se aplican las reglas de agregación, se consideran únicamente los triángulos de la frontera del cubrimiento, de manera que los triángulos del interior no son considerados al determinar cuál debe ser la siguiente regla de agregación. Se tienen entonces las siguientes conclusiones:

(1) Tomando en cuenta solamente el contorno de los triángulos de la frontera de un cubrimiento, se puede proponer una manera única de describir cada cubrimiento, esto es posible gracias a la unicidad del espacio ocupado (página 50).

La manera de describir esta curva es describir la orientación de cada segmento de recta del tamaño de una célula. Así por ejemplo, un triángulo t_0 se podría describir con la secuencia $< e, s, o, n >$ en donde cada letra significa una dirección

{*este*, *sur*, *oeste*, *norte*}. Para que esta descripción pueda er viable, se debe determinar un punto de referencia que sirva de inicio de lectura de la cadena. Como sugerencia este punto de referencia es la coordenada $(0, y) \in \mathbb{Z}^2$, en donde y es el punto de la curva que pasa exactamente por $x = 0$.

(2) Al observar los contornos que se crean al hacer crecer los cubrimientos debidos a los mosaicos propios, se observan estructuras similares a otras curvas fractales conocidas. Tal es el caso de la estrella de Koch, esta similitud nos hace suponer que existe al menos un sistema de Lyndenmayer que genere esa curva.

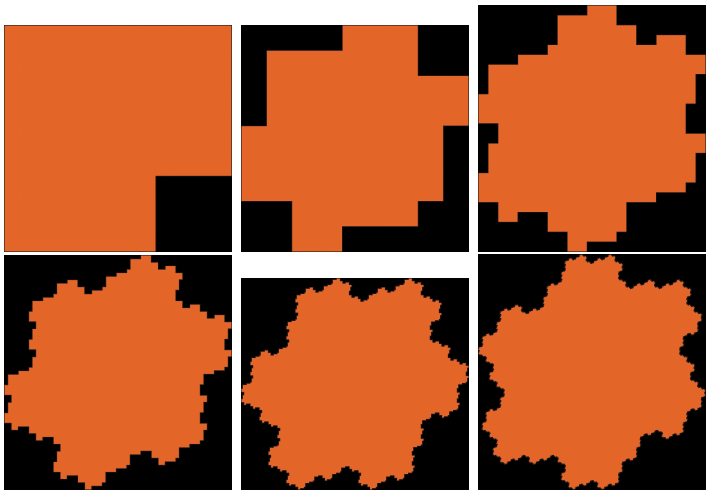

Figura 59. Curvas fractales generadas por los contornos de los crecimientos en potencia de los mosaicos propios

Bibliografía

[1] Adamatzky, A. *Collision-Based computing*. Springer-Verlag, 2002.

[2] Arbib, M. A. Simple self-reproducing universal automata. *Information and Control 9*, 2 (April 1966), 177–189.

[3] Beauchat, J.-L., and Haenni, J.-O. Von Neumann's 29-state cellular automaton: A hardware implementation. *IEEE Transactions on education 43*, 3 (August 2000), 300–308.

[4] Berger, R. *The undecibility of the domino problem*, vol. 66 of *Memoirs of the AMS*. American Mathematical Society, 1966.

[5] Buckingham, D. J. Some facts of life. *Byte 2* (December 1978), 54–67.

[6] Chopard, B., and Droz, M. *Cellular Automata Modeling of Physical Systems*. Cambridge University Press, 1998.

[7] Codd, E. F. *Cellular automata*. ACM Monograph Series. Academic Press NY, 1968.

[8] Conway, J. H., and Sloane, N. J. A. *Sphere Packings, Lattices and Groups*, 1 ed ed., vol. 290 of *Grundlehren der Mathematischen Wissenschaften*. Springer, New York, 1988.

[9] Cook, M. Introduction to the activity of rule 110-(1998 unpublished). *Document avalaible by e-mail request to acaceres@computacion.cs.cinvestav.mx* (On e-mail communication-2003. Finally published on Complex Systems vol 15, number 1 (2004) pp 1 - 40.).

[10] Davis, M., and Weyuker, E. J. *Computability, complexity, and languages*. Academic Press, 1983.

[11] Denning, P. J. *Machines, languages and computation*. Prentice-Hall, Inc., 1967.

[12] Dunne, P. E. *Computability theory: concepts and applications*. Ellis Horwood, 1991.

[13] Eppstein, D. Which "life"-like systems have gliders? *URL: http://www.ics.uci.edu/~eppstein/ca/* (Last visited October 2004).

[14] Eppstein, D., Sullivan, J. M., and Üngör, A. Tiling space and slabs with acute tetrahedra. *ACM Computing Research Repository*, cs.CG/0302027 (February 2003).

[15] Gardner, M. The fantastic combinations of John Conway's new solitaire game "life"'. *Sci. Amer. 223* (April 1970), 120–123.

[16] Gardner, M. Mathematical games: On cellular automata, self-reproduction, the garden of eden and the game "life". *Sci. Amer. 224* (February 1971), 112–117.

[17] Gardner, M. *"Tilings with Convex Polygons" in Time Travel and Other Mathematical Bewilderments*. New York: W. H. Freeman, 1988, ch. 13, pp. 162–176.

[18] Grünbaum, B., Shepard, G., and (Editor), D. A. K. *"Some problems on plane tiling" on The mathematical Gardner*. Wadsworth International, 1981.

[19] Grünbaum, B., and Shephard, G. C. *Tilings and Patterns*. W. H. Freeman, New York, 1987.

[20] Gutowitz, H. A., Victor, J. D., and Knight, B. W. Local structure theory for cellular automata. *Physica D 28D* (1987), 18–48.

[21] Hales, T., and et.al. Cannonballs and honeycomb: Hilbert. *Departament of Mathematics U. Pittsburgh* (2001), http://www.math.pitt.edu/articles/hilbert.html. Last visit Nov/2004.

[22] Heesch, H. Reguläres parkettierungsproblem. *Westdeutscher Verlag. Cited in Tiling and Patterns by B. Grunbaum and Geoffrey C. Shephard. 1986* (1968).

[23] Hickerson, D. Description of sliding block memory (website), 1990. http://www.radicaleye.com/lifepage/patterns/sbm/sbm.html -(Jul/2003).

[24] Hilbert, D. Aufbau des raumes aus congruenten polyedern. *Akad. Wiss. Göttingen* (1900), 235–286. English traslation URL http://aleph0.clarku.edu/~djoyce/hilbert/problems.html#prob18 Nov/2004.

[25] Hopcroft, J. E., and Ullman, J. D. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Company, Inc., 1993.

[26] Kelley, D. *Automata and formal languages: An introduction*. Prentice Hall, 1995.

[27] Kendall Jr., P., and Duff, M. J. B. *Modern Cellular Automata: Theory and Applications*. Plenum Press, N.Y., 1984.

[28] Kosniowski, C. *A first course in algebraic topology*. Cambridge University Press, 1980.

[29] Lefschetz, S. *Algebraic topology*. No. 27 in Colloquium Publications. American Mathematical Society, 1991.

[30] Markov Jr, A. A. The theory of algorithms. *American Mathematical Society Translations 15* (1960), 1–14.

[31] Markov Jr, A. A., and Nagorny, N. M. *The theory of algorithms*. Kluwer Acadenic Publishers, 1988.

[32] McIntosh, H. V. *Linear cellular automata via de Bruijn diagrams*, tech. report avalaible at http://delta.cs.cinvestav.mx/~mcintosh/oldweb/pautomata.html ed. México, Universidad Autónoma de Puebla, August 1991.

[33] McIntosh, H. V. *Rule 110*. Universidad Autónoma de Puebla, México, Tech. Report avalaible at http://delta.cs.cinvestav.mx/~mcintosh/oldweb/pautomata.html, 1999.

[34] McIntosh, H. V. *A concordance for rule 110*. Universidad Autónoma de Puebla, México, Tech. Report avalaible at http://delta.cs.cinvestav.mx/~mcintosh/oldweb/pautomata.html, 2000.

[35] McIntosh, H. V. *Rule 110 as it relates to the presence of gliders*. Universidad Autónoma de Puebla, México, Tech. Report avalaible at http://delta.cs.cinvestav.mx/~mcintosh/oldweb/pautomata.html, 2002.

[36] Minsky, M. L. Recursive unsolvability of Post's problem of "*TAG*" and others topics in theory of Turing machines. *Annals of Mathematics 74*, 3 (November 1961), 437–455.

[37] Minsky, M. L. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.

[38] Morita, K., and Harao, M. Computation universality of one-dimensional reversible (injective) cellular automata. *Trans IEICE E 72*, 6 (1989), 758–762.

[39] Nordahl, M. G. Formal languages and finite cellular automata. *Complex systems 3* (1989), 63–78.

[40] Penrose, R. Pentaplexity: a class of non-periodic tilings of the plane. *Math. Intelligencer 2* (1979), 32–37.

[41] Post, E. L. Finite combinatory processes - formulation 1. *The Journal of Symbolic Logic 1* (1936), 810–811.

[42] Post, E. L. Formal reductions of the general combinatorial decision problem. *Journal of Mathematics 65* (1943), 197–215.

[43] Post, E. L. Recursive enumerable sets of positive integers and their decision problems. *Bulletin of the American Mathematical Society 50* (1944), 284–316.

[44] Post, E. L. Recursive unsovability of a problem of *T*hue. *Bulletin of the American Mathematical Society 52* (1946), 1015–1016.

[45] Post, E. L. A variant of a recursively unsolvable problem, 1946.

[46] Post, E. L. Recursive unsovability of a problem of *T*hue. *The Journal of simbolic logic 12* (1947), 1–11.

[47] Radin, C. Space tilings and substitutions. *Geometriae dedicata 55* (1995), 661–702.

[48] Raedschelders, P. Heesch tiles based on regular polygons. *Geombinatorics 7* (1998), 101–106.

[49] Ray Smith III, A. Simple computation-universal cellular spaces and self-reproduction. *9th IEEE FOCS Conference Record 9* (October 1968), 269–277.

[50] Ray Smith III, A. Simple computation-universal cellular spaces. *Journal of the Association for Computing Machinery 18*, 3 (July 1971), 339–353.

[51] Rendell, P. A *T*uring machine in *C*onway's game life. http://rendell.server.org.uk/gol/tm.htm (WEBsite visited on jul/2003), August 2001.

[52] Robinson Jr., E. A. Symbolic dynamics and tilings of $\mathbb{R}^d$. Unpublished Lecture Notes for a Short Course in Symbolic Dynamics at the January 2002 Joint Mathematics meetings in San Diego, California., 2002.

[53] Rogers, H. J. *Theory of recursive functions and effective computability.* McGraw-Hill Co., 1967.

[54] Rohilla Shalizi, C., and Crutchfield, J. P. *Computational mechanics: Pattern and prediction, structure and simplicity.* Santa Fé Institute, 2001.

[55] Rosenfeld, A. *Picture languages.* Academic Press, Inc., USA, 1979.

[56] Sipper, M. Computing with cellular automata: three cases for nonuniformity. *Physical review E 57*, 3 (1998), 3589–3592.

[57] Sloane, N. J. A. The sphere packing problem. *Proceedings of the International Congress of Mathematicians Extra Vol. III* (Berlin 1998), 387–396.

[58] Solomyak, B. Dynamics of self-similar tilings. *Ergodic Theory Dynam. Systems 17*, 3 (1997), 695–738.

[59] Stein, S., and Szabs, S. *Algebra and tiling. Homomorphism in the service of geometry.* No. 25 in The Carus Mathematical Monographs. The Mathematical Association of America, 1994.

[60] Thompson, D. W. *On growth and form*, Canto edition 1992. Edited by John Tyler Bonner ed. Cambridge University Press, UK, 1961.

[61] Toffoli, T. Computation and construction universality of reversible cellular automata. *Journal of Computer and Systems Sciences 15*, 2 (October 1977), 213–231.

[62] Toffoli, T., and Margolus, N. *Cellular automata machine: A new enviroment for modeling*. The MIT Press, Cambridge, Massachusetts, 1989.

[63] Turing, A. M. On computable numbers with an application to the *E*ntscheidungs-problem. *Proc. London Math. Soc. 2*, 42 (1936), 230–265.

[64] Turing, A. M. Computing machinery and intelligence. *Mind. 59*, 236 (1950), 433–460.

[65] Ulam, S. On some mathematical problems connected with patterns of growth od figures. *Proceedings of Symposia in Applied Mathematics*, 14 (1962), 215–224 also in "Essays on cellular automata" edited by Arthur W. Burks, U. Illinois Press, 1970.

[66] von Neumann, J., and Burks(compiler & editor), A. W. *Theory of self-reproducing automata*. University of Illinois Press, 1966.

[67] Wang, H. Proving theorems by pattern recognition-*II*. *Bell Systems Technical Journal 40* (January 1961), 1–42.

[68] Wolfram, S. *Celullar Automata and complexity: Collected papers*. Addison-Wesley Publishing Company, Inc. USA,, 1994.

[69] Wolfram, S. *A new kind of science*. Wolfram Media, Inc., 2002.

[70] Wuensche, A. Classifying cellular automata automatically (1998). *Available at http://www.santafe.edu/~wuensch/ddlab.html* (On mail communication-2000).

Segunda Edición.
Esta edición se terminó de imprimir el 23 de Febrero de 2005
en las intalaciones del
Centro de Investigación y de Estudios Avanzados - IPN.
El tiraje consta de 10 ejemplares.

"Miré yo luego todas las obras que habían hecho mis manos, y el trabajo que tomé para hacerlas: y he aquí todo vanidad y aflicción de espíritu, y no hay provecho debajo del sol... Entonces dije yo en mi corazón: 'Como sucederá al necio sucederá también a mí: ¿Para qué pues he trabajado hasta ahora por hacerme más sabio?' Y dije también en mi corazón que también esto era vanidad'." Eccl 2:11 y 15