Genaro J. Martínez, Héctor Zenil,
Christopher R. Stephens (Eds.)

Sistemas Complejos Como Modelos de Computación

(Complex Systems as Computing Models)

Luniver Press

G. J. Martínez, H. Zenil, C. R. Stephens (Eds.)
Sistemas Complejos Como Modelos de Computación

Actualmente, el procesamiento de información y el estudio de sistemas complejos juegan un papel fundamental en el entendimiento de fenómenos no-lineales a cualquier escala. Por otro lado, la teoría de la computación juega un rol indispensable para describir, a través de un procedimiento efectivo, un fenómeno en particular. En la intersección, la manera de procesar dicha información y la complejidad derivada de ello es objeto de estudio, y también lo es ahora la forma en que hemos cambiado la manera de ver los sistemas complejos para considerarlos como modelos de computación por sí mismos convirtiéndonos de observadores a programadores de sistemas complejos, haciéndolos procesar información como otro dispositivo de computación.

Destacados investigadores, jóvenes y experimentados, contribuyen con artículos que presentan resultados en estas líneas de investigación. Los autores abarcan una variedad de temas relacionados con sistemas complejos, evaluación de complejidad de sistemas, cifrado de datos, computación cuántica, modelos de computación inspirados en sistemas biológicos y matemáticos, entre otros. El libro ofrece una excelente introducción a las distintas áreas de interés apuntando hacia las líneas de investigación del futuro. Está dirigido a estudiantes y académicos que deseen estudiar la complejidad de sistemas de computación y explorar sistemas complejos como modelos de computación.

Currently, information processing and the study of complex systems play a key role in the understanding of nonlinear phenomena at any scale, while the theory of computation plays an indispensable role in describing particular phenomena through the use of effective procedures. At their intersection new questions arise, questions about the complexity resulting from information processing, as well as about the way we've changed the way complex systems are viewed, that is, as computer models in themselves, which transforms us from observers to programmers of complex systems.

Leading researchers, both young and experienced, contribute articles presenting work along these lines. The authors cover a variety of topics related to complex systems, among them the evaluation of complexity, data encryption, quantum computing, and computational models inspired by biological and mathematical systems. The book provides an excellent introduction to a multi-faceted and burgeoning area of research. It is aimed at students and scholars who wish to study the complexity of computer systems and explore complex systems as models of computation.

ISBN 978-1-905986-35-4
90000
9 781905 986354

G.J. Martínez, H. Zenil, C.R. Stephens (Eds.)
Sistemas Complejos como Modelos de Computación

Sistemas Complejos como Modelos de Computación

Genaro Juárez Martínez, Héctor Zenil,
Christopher Rhodes Stephens Stevens
Editores

Sistemas Complejos como Modelos de Computación
(Complex Systems as Computing Models)

Luniver Press
2011

Published by Luniver Press
Frome BA11 3EY United Kingdom

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

Sistemas Complejos como Modelos de Computación
Complex Systems as Computing Models

ISBN-10: 1-905986-35-1
ISBN-13: 978-1-905986-35-4

Comité científico

Comité organizador

Adriana de la Paz Sánchez Moreno
Escuela Superior de Cómputo, IPN, México.

Alejandra Reyes Mancilla
Instituto de Ciencias Nucleares, UNAM, México.

Alejandro Frank Hoeflich
Instituto de Ciencias Nucleares, UNAM, México.

Kahorik González Flores
Centro de Ciencias de la Complejidad, UNAM, México.

Jeanett López García
Facultad de Acatlán, UNAM, México.

Liliana Jiménez Barrón
Instituto de Ciencias Nucleares, UNAM, México.

María Concepción García Aguirre
Centro de Ciencias de la Complejidad, UNAM, México.

Principales patrocinadores de WCSCM2011

Instituto de Ciencias Nucleares (ICN)
Universidad Nacional Autónoma de México, México.
`http://www.nucleares.unam.mx/`

Centro de Ciencias de la Complejidad (C3)
Universidad Nacional Autónoma de México, México.
`http://c3.fisica.unam.mx/`

International Center of Unconventional Computing (ICUC)
University of the West of England, Bristol, United Kingdom.
`http://uncomp.uwe.ac.uk/`

Laboratorio de Ciencias de la Computación (LCCOMP)
Universidad Nacional Autónoma de México, México.
`http://uncomp.uwe.ac.uk/LCCOMP/`

Laboratoire de Recherche Scientifique (LABORES)
Maison des Associations, Paris, France.
`http://labores.eu/`

Lista de participantes

Andrés Anzo Hernández
Instituto Potosino de Investigación Científica y Tecnológica, San Luis Potosí, México.

Andrew Wuensche
Discrete Dynamics Lab. International Center of Unconventional Computing, University of the West of England, Bristol, UK.

Carlos Adrián Jaramillo Hernández
Centro de Investigación Avanzada en Ingeniería Industrial, Universidad Autónoma del Estado de Hidalgo, Hidalgo, México.

Elena Villarreal Zapata
Universidad Politécnica de San Luis Potosí, San Luis Potosí, México

Emmanuel Garcés Medina
Laboratorio de Ciencias de la Computación, Laboratorio de Dinámica No Lineal, Facultad de Ciencias, UNAM. México.

Enrique Zeleny Vazquez
Wolfram Research, Inc., USA.

Francisco Cruz Ordaz Salazar
Universidad Politécnica de San Luis Potosí, San Luis Potosí, México

Héctor Zenil
Department of Computer Science, University of Sheffield, Sheffield, UK. Special Projects Department, Wolfram Research, IL, USA. Centro de Ciencias de la Complejidad, Laboratorio de Ciencias de la Computación, UNAM, México.

José Manuel Sausedo Solorio
Laboratorio de Física Avanzada, Universidad Autónoma del Estado de Hidalgo, Hidalgo, México.

Joselito Medina Marín
Centro de Investigación Avanzada en Ingeniería Industrial, Universidad Autónoma del Estado de Hidalgo, Hidalgo, México.

Juan Carlos Seck Tuoh Mora
Instituto de Cencias Básicas e Ingeniería, Universidad Autónoma del Estado de Hidalgo, Hidalgo, México.

Juan Gonzalo Barajas Ramírez
Instituto Potosino de Investigación Científica y Tecnológica, San Luis Potosí, México.

Kenichi Morita
Hiroshima University, Higashi-Hiroshima, Japan

Luis Alvarez-Icaza
Instituto de Ingeniería, Universidad Nacional Autónoma de México, México.

María Elena Lárraga Ramírez
Instituto de Ingeniería, Universidad Nacional Autónoma de México, México.

Paulina Anaid León Hernández
Centro de Investigación y de Estudios Avanzados, Instituto Politécnico Nacional, México

Rogelio Basurto Flores
Centro de Investigación y de Estudios Avanzados, Instituto Politécnico Nacional, México

Salvador Elías Venegas Andraca
Grupo de Procesamiento Cuántico de la Información, Tecnológico de Monterrey Campus Estado de México, México.

Todd Rowland
Wolfram Research, Inc., IL, USA.

Índice general

Prefacio

El estudio de sistemas complejos, y el manejo de información juegan actualmente un papel fundamental en el entendimiento de fenómenos no-lineales a cualquier escala. En esta dirección, indudablemente, la teoría de la computación juega un rol indispensable para describir, a través de un procedimiento efectivo, un fenómeno en particular. La manera de procesar dicha información y la complejidad derivada de ello son objetos de estudio, y también lo es ahora la forma en que hemos cambiado la manera de ver los sistemas complejos para considerarlos como modelos de computación por sí mismos.

En otras palabras, el estudio de la complejidad resulta bidireccional: Por un lado, el estudio de la complejidad de un modelo de computación como objeto de estudio y, por el otro, el estudio de un sistema complejo como modelo de computación. Nos hemos convertido de observadores a programadores de sistemas complejos, haciéndolos procesar información.

Destacados investigadores contribuyen en ambas direcciones con artículos que presentan resultados relevantes en varias líneas de investigación: desde aspectos teóricos, prácticos y aplicaciones, hasta temas especializados. De igual forma, se ofrecen excelentes introducciones para estudiantes y académicos que deseen estudiar la complejidad de sistemas de computación y explorar sistemas complejos como modelos de computación.

Editores.
Genaro J. Martínez
Héctor Zenil
Christopher R. Stephens

Reversible computing and cellular automata as complex systems

Kenichi Morita

Hiroshima University
Higashi-Hiroshima 739-8527, Japan
km@hiroshima-u.ac.jp

Resumen Reversible computing is a paradigm of computation where every computational configuration of the system has at most one predecessor. Hence, it can be regarded as a "backward deterministic"system. Though its definition is thus rather simple, it has a close connection to physically reversible systems such as classical mechanical systems without friction, quantum mechanical systems, and others. In this talk we discuss how reversible computing systems can be designed and constructed from simple reversible primitives. We shall see that such systems have good abilities of computing and information processing even when they are composed of very simple reversible primitives, and thus show complex behaviors. In particular, we consider reversible logic circuits and reversible cellular automata as models of reversible systems, and investigate how computation-universality, and life-like phenomena such as self-reproduction can emerge in these systems.

Referencias

[1] Morita, K. (2008). Reversible computing and cellular automata — A survey, *Theoret. Comput. Sci.*, 395, 101–131.

[2] Morita, K. (2010). Constructing a reversible Turing machine by a rotary element, a reversible logic element with memory, Hiroshima University Institutional Repository, http://ir.lib.hiroshima-u.ac.jp/00029224.

[3] Morita, K. (2011). Universal reversible cellular automata in which counter machines are concisely embedded, Hiroshima University Institutional Repository, http://ir.lib.hiroshima-u.ac.jp/00031367.

[4] Morita, K., Imai, K. (2011). Self-reproduction in two- and three-dimensional reversible cellular automata, Hiroshima University Institutional Repository, http://ir.lib.hiroshima-u.ac.jp/00031368.

[5] Morita, K. (2011). Simulating reversible Turing machines and cyclic tag systems by one-dimensional reversible cellular automata, *Theoret. Comput. Sci.*, 412, 3856–3865.

Sistemas dinámicos complejos y caóticos, conjuntos de atractores, memoria y redes discretas

Andrew Wuensche

Discrete Dynamics Lab, United Kingdom.
http://www.ddlab.org/[**]

Resumen El comportamiento emergente de estructuras y su interacción en autómata celular, se encuentran fuertemente relacionados con las nociones de orden, complejidad y caos, que dependen, en buena medida del grado en el que convergen sus atractores. Dicha información puede encontrarse encriptada y además oculta dentro de sus trayectorias caóticas. En el caso de las redes "aleatorias", la memoria de contenido direccionable se encuentra afectada aparentemente desde sus conjuntos de atractores y la estructuras de sus subárboles, el concepto de memoria y aprendizaje en su más básico nivel. El presente artículo es una revisión de estas ideas, resultados y aplicaciones, además ilustradas con varias imágenes creadas con el sistemas DDLab.

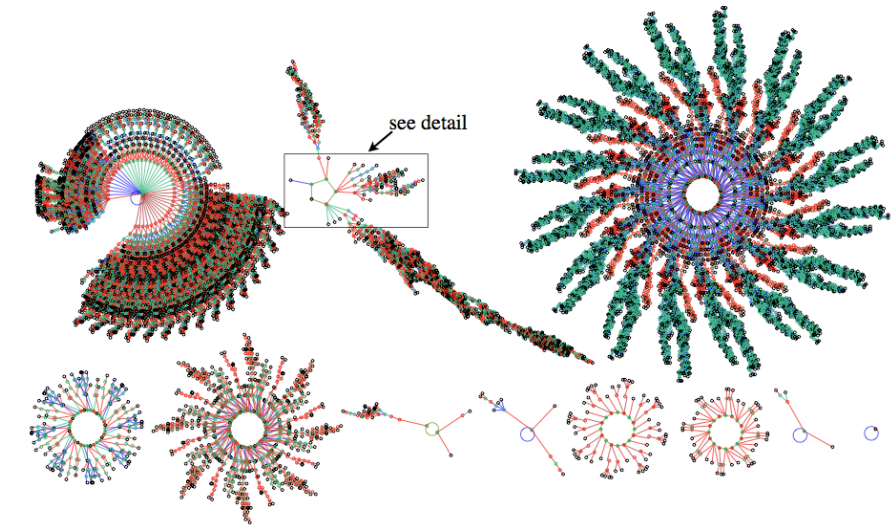
Palabras clave: conjuntos de atracción, atómata celular, redes booleanas aleatorias, caos, complejidad, ancestros, algoritmos reversibles, auto organización, memoria, aprendizaje, redes genéticas reguladas.

1. Introducción

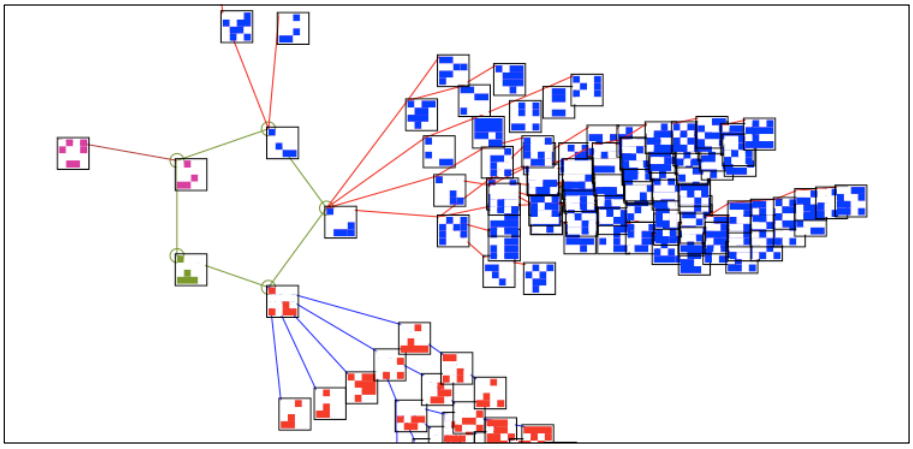
El libro "The Global Dynamics of Cellular Automata" (Dinámica Global en Autómata Celular) [11], publicado en 1992, presentó un algoritmo invertible para calcular ancestros (o pre-imágenes) de estados para cualquier autómata celular (CA) binario en una dimensión (1D) con propiedades a la frontera. Esto ayudó a comprender la topología de los "conjuntos de atracción" – gráficas de estados de transición – los estados se encuentran conectados en la raíz de los árboles como ciclos atractores y éstos pueden ser calculados automáticamente (ver Fig. 1).

Posteriormente, un nuevo algoritmo invertible fue inventado para calcular los ancestros, pero ahora, en redes booleanas aleatorias (RBN) y sus conjuntos de atracción pueden ser igualmente graficados (ver Fig. 15). De hecho, fueron

[**] Original publication, January 11, 2010. Artículo presentado en el *Summer Solstice 2009 International Conference on Discrete Models of Complex Systems*, Gdansk, Polonia, junio 22-24 de 2009. La versión original de este artículo fue publicada en inglés [21]. La presente versión en español (traducida por Genaro J. Martínez) aparece con el permiso de publicación del ACTA PYSICA POLONICA B. Cualquier comentario enviarlo por favor a andy@ddlab.org. Traducción al español, 6 de octubre de 2011.



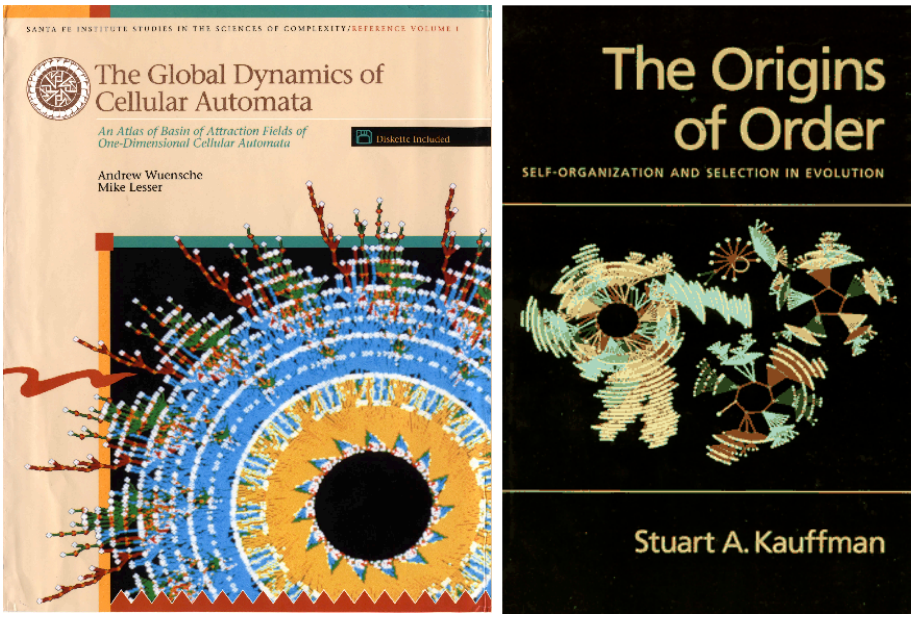
(a)



(b)

Figura 1. En (a) se ilustra el campo de conjuntos de atracción de un CA binario en 1D para $n = 16$ (n es la longitud del anillo, configuración inicial o tamaño del sistema). Los 2^{16} estados en el tiempo, se encuentran conectados dentro de 89 conjuntos de atracción, pero en este caso únicamente se ilustran los 11 conjuntos no equivalentes, con simetrías características del CA [11]. El flujo va hacia adentro del atractor en el sentido de las agujas del reloj (alrededor del ciclo atractor). (b) El segundo conjuto de atracción es amplicado de manera que podemos ver todos los detalles del atractor, aquí se ilustran los patrones de 4×4 bits.

graficados para la portada del libro de Stuart Kauffman in 1993 “The Origins of Order” (Los Orígenes del Orden) (ver Fig. 2). El algoritmo desarrollado para las RBN se encuentra ahora mejorado, más versátil y generalizado, ahora como las “redes dinámicas discretas” (DDN). Estos algoritmos calculan ancestros directamente, realizadon una muy eficiente búsqueda en el espacio y tiempo, y se encuentran implementados en el software DDLab [19].



(a) (b)

Figura 2. Ilustramos las portadas de los libros de: (a) Wuensche y Lesser (1992) “The Global Dynamics of Cellular Automata” [11] y (b) el de Kauffman (1993) “The Origins of Order” [5]. Podemos ver un conjunto de atracción de un CA y un campo de conjuntos de atracción en un RBN, ambos fueron calculados con el precursor de DDLab.

Un DDN es un conjunto finito de n elementos con estados o valores discretos. Los elementos están conectados a través de aristas dirigidas – el esquema de conexión. Entonces, cada elemento actualiza su propio valor de manera sincronizada, de acuerdo a una regla lógica que es aplicada en sus k entradas – el sistema se actualiza en tiempos y pasos discretos. Los CA son mucho más restringuidos (que los RBN y DDN) porque ellos tienen una regla universal y un arreglo regular con condiciones a la frontera, creado por la conexión homogénea de “células” determinando vecindades locales, mientras que los RBN y DDN no tienen estas restricciones. Langton [2] describió los CA como “un universo

artificial discretizado con su propia física local". Obviamente existen incontables variaciones y arquitecturas intermedias entre DDN y CA – RBN clásicos [4] tienen valores binarios {0,1} y k homogéneos, aunque todos estos sistemas reorganizan su espacio en conjuntos de atracción de la misma manera.

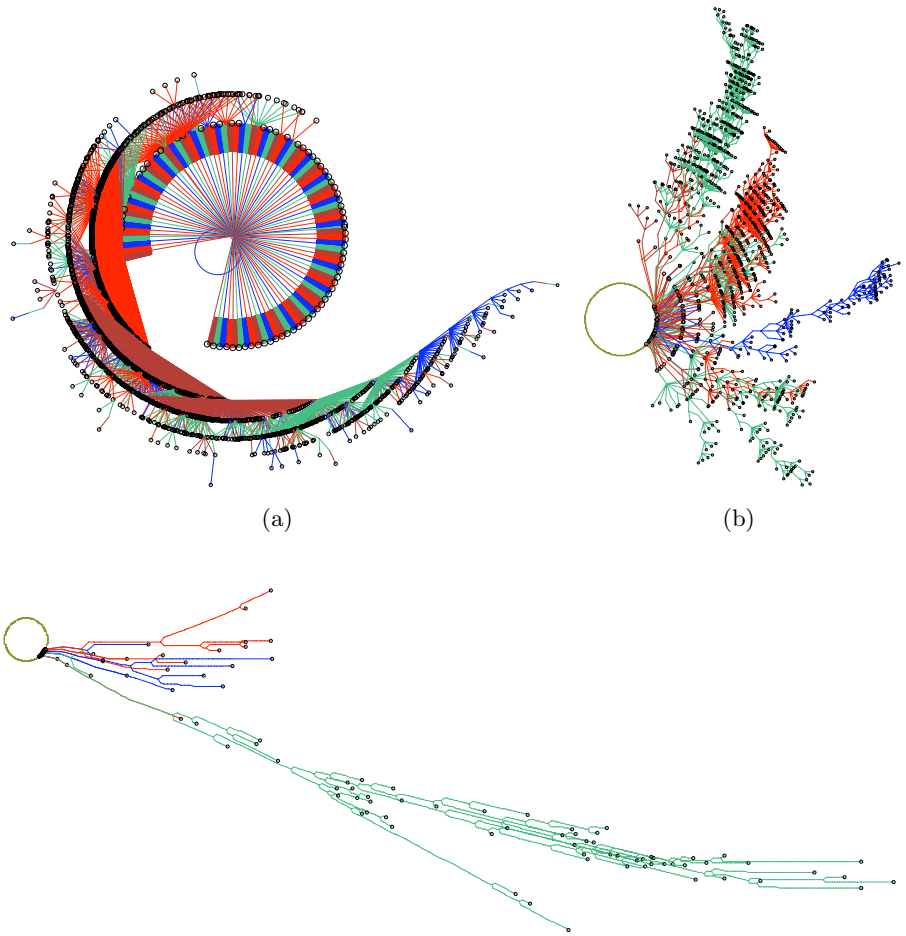
Ejecutando o simulando un CA, RBN o DDN hacia atrás en el tiempo y construyendo todas sus posibles ramificaciones calculando sus ancestros, abre una nueva perspectiva en el estudio de sus dinámicas. Una trayectoria empezando desde algún estado inicial puede estar situado en el contexto del flujo del espacio conduciendo a los atractores, análogo al conocido "estado fase" de Poincaré en dinámica continua, pero aplicados en sistemas donde el tiempo y espacio son discretos (como en la naturaleza). Estas implicaciones fueron discutidas por Langton en su prefacio [11]. Los sistemas dinámicos contínuos y discretos comparten conceptos análogos, como son: puntos fijos, cíclos límite, caos, sensibles a condiciones iniciales y atractores caóticos. La frontera entre sus conjuntos de atracción tienen algunas afinidades en los estados (hojas) inalcanzables (conocidos como Jardín del Edén, "Garden of Eden"). La extensión de una conexión local de transiciones medido por el exponente de Liapunov, tiene su analogía en el grado de convergencia dada la densidad de los subárboles – el grado de profundidad de un estado, establecido por el parámetro Z [11, 17].

En este punto podemos indentificar ciertas analogías y discrepancias (con merecimiento para el desarrollo de una tesis) en tres importantes comportamientos de ciertos fenómenos, revisando este artículo podríamos situar algunos aspectos en dinámicas discretas como opuestas en la dinámica continua: complejidad por la interacción de estructuras – en el entendimiento de la auto organización; información oculta en caos – con aplicaciones a la criptografía; y el concepto de memoria y aprendizaje en su más básico nivel – para modelar redes neuronales y genéticas.

2. Complejidad derivada por la interacción de estructuras

En algunos CA raros, la interacción dada por estructuras estáticas y móviles o partículas – gliders y glider guns, emergen y dominan la dinámica del sistema. Algunos ejemplos particulares son el Juego de la Vida, la regla 110 y la recientemente descubierta regla espiral [10] (Fig.7), donde ellos son estudiados a través del choque de sus partículas para modelar computación lógica o universal (Fig.8) [8, 7].

Desde otra perspectiva, éstos son sistemas extremadamente simples y completamente bien definidos, que también son capaces de auto organizarse y componerse en estructuras aún más complejas. El comportamiento emergente parece ser impredecible, de duración indefinida y limitada únicamente por el tamaño del espacio de evoluciones. Como sucede en la naturaleza estos CA complejos pueden ser descritos en un nivel ascendente – desde la "física" esencial, en la observación de las "leyes" de choques de partículas, en la descripción de interacciones complejas que resulta cada vez más importante – argumentado – en el



(a) (b)

(c)

Figura 3. Ilustramos tres conjutos de atracción en CA elemental (ECA) con topologías opuestas, para las reglas (a) 250, (b) 110 y (c) 30 con $n = 15$ y $k = 3$. Un conjunto completo de árboles equivalentes se presenta en cada caso, junto con sus estados (hojas o nodos) inalcanzables. La topología varía en los tres conjuntos con una alta densidad de ramificación – escasa ramificación, dado por el número de nodos concentrados con algunas medidas, como son la densidad de hojas, la longitud de las transiciones y la distribución de profundidad (ancestros a un estado) pronósticada por el parámetro Z.

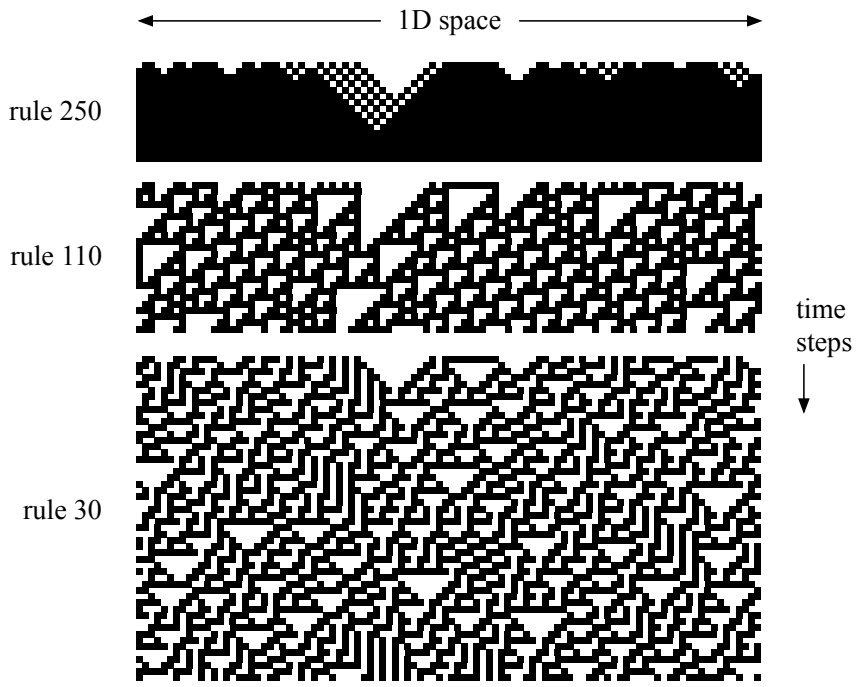
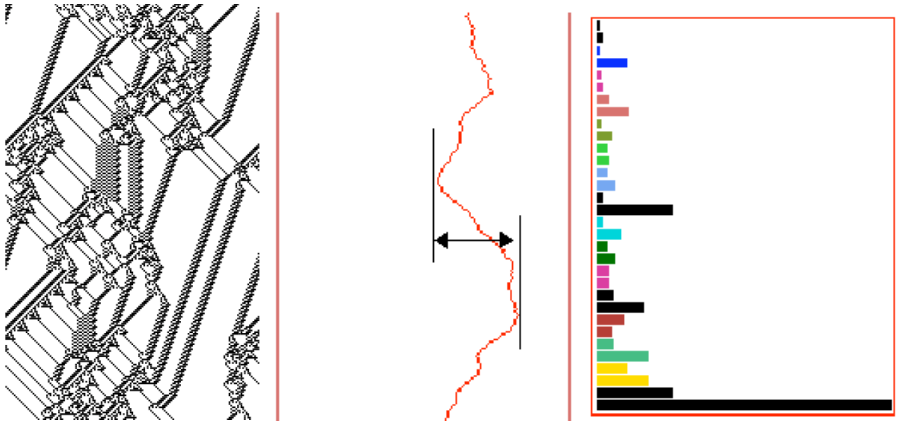


Figura 4. Aquí presentamos la producción de patrones espacio y tiempo en ECA de 1D para reglas de orden $k = 3$, desde la Fig. 3, se ilustran los comportamientos globales que presentan: orden, complejidad y caos, respectivamente. El tamaño del sistema es $n = 100$ con condiciones a la frontera. La misma condición inicial aleatoria es utilizada para las tres reglas. Un patrón espacio y tiempo es precisamente una ruta desde algún nodo seleccionado en el conjuto de atracción.

"posible contiguo" de Kauffman [6]. Desde esta perspectiva, la complejidad de un sistema es el número descriptivo de sus niveles existenciales [12].

De esta manera surge la pregunta: ¿qué es la auto organización? Es generalmente aceptado que las reglas complejas son raras y ocurren en reglas de transición que se encuentran entre el orden y el caos [2] (Fig.10) – aunque para encontrarlas no es tan claro y simple. Una amplia variedad de reglas complejas se encuentran probablemente fuertemente relacionadas al descubrimiento de los principios generales de la auto organización. Podemos encontrar innumerables ejemplos de reglas complejas, utilizando como entrada su entropía para clasificarlas automáticamente en reglas con orden, complejas y caóticas [17, 10]. Las Figs. 5 y 6 ilustran el método (implementado en DDLab) que señala la entropía de Shannon dada la frecuencia de reglas entrantes, generando patrones en el espacio de evoluciones y la entropía variable[1] produciendo los siguientes resultados,

[1] La variabilidad es establecida como la desviación estándar o alternativamente, como el intervalo máximo entre un mínimo seguido de un máximo de entropía.



(a) (b)

Figura 5. (a) Patrones espacio y tiempo en CA complejo de 1D con $n = 150$ evolucionando en 200 generaciones. (b) Ilustra una parte del histograma de frecuencias especificado en una ventana que representa 10 pasos de la evolución (gráfica izquierda). La gráfica central muestra el histograma calculando la entropía cambiante, su variabilidad otorga una posible medida para descriminar entre funciones con comportamientos globales ordenados, complejos y caóticos automáticamente. Una alta variabilidad relaciona fuertemente la presencia de dinámicas complejas.

	ordenado	complejo	caos
entropía media	bajo	medio	alto
entropy variable	bajo	alto	bajo

Las reglas que son únicamente complejas tienen una alta entropía variable y pueden ser separadas – la entropía media separa el orden y el caos (Fig.6). La alta variabilidad relaciona la interacción de estructuras a gran escala, frecuentemente producidas por el choque de partículas porque los choques crean caos local que nos da la entropía misma, donde las partículas vuelven a emerger regulando la entropía.

La dinámica de partículas puede ser visto desde la perspectiva de un conjunto de atracción. Los estados desordenados, antes de que emergan las partículas, están compuestos por las hojas con trayectorias cortas mientras que las trayectorias largas inducen la existencia de partículas que encuentran interactuando. Finalmente, las partículas finales o que sobreviven se encuentran en el atractor mismo.

3. Información oculta en sistemas caóticos

El estado y espacio, por definición, incluye cada posible pieza de la información codificada en el tamaño del arreglo celular de un CA – incluyendo sonetos

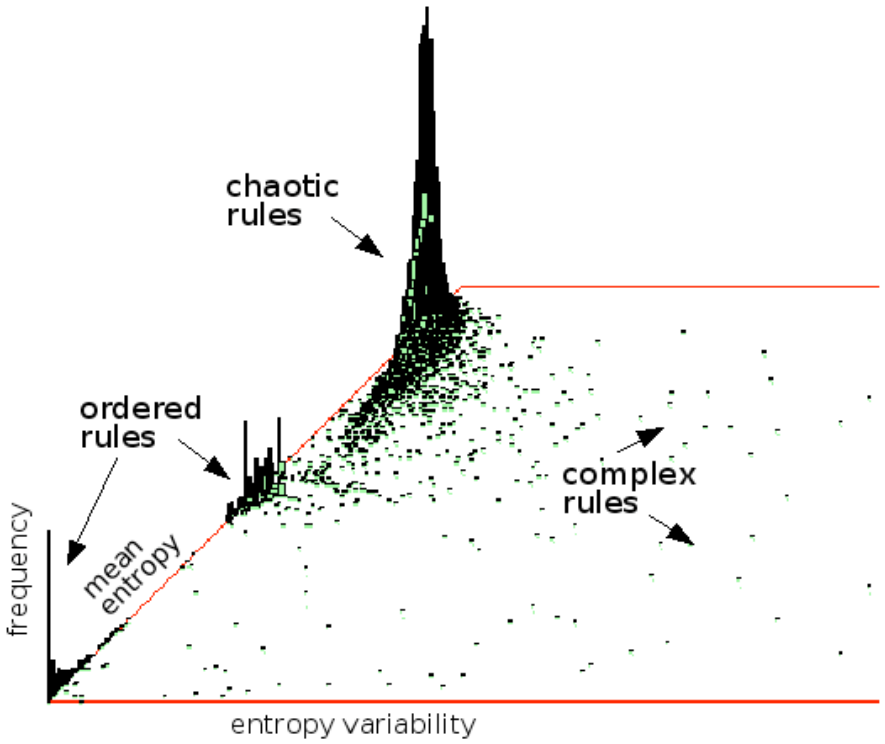


Figura 6. La presente gráfica de dispersión es calculada para 15,800 reglas de CA hexagonal, con parámetros $(v = 3, k = 6)$, graficando la entropía media contra su entropía variable, que nos da como resultado la clasificación de reglas con comportamiento ordenado, complejo y caótico. La ordenada vertical representa la frecuencia de las reglas – la mayoría son caóticas. La gráfica de dispersión clasifica las reglas de evolución automaticamente.

de Shakespeare, copias de La Mona Lisa, la copia del pulgar, pero principalmente desordenado. Un CA organiza los estados tiempo en conjuntos de atracción donde cada estado tiene su lugar específico y donde los estados en la misma trayectoria están conectados por los estados del tiempo hacia adelante, de esta manera, el enunciado "el estado tiempo $B = A + x$" es plenamente justificado. Aunque su inversa, $A = B - x$ generalmente no lo es porque sus trayectorias viajan de regreso y deberá por lo tanto seleccionar correctamente la ramificación que debe de ser. Un punto importante son los estados que representan las hojas del atractor (estados globales sin ancestros), ya que para estos estados "$-x$" el tiempo hacia atrás no existe.

En grados, la convergencia del flujo dinámico puede ser inferido desde la regla del mismo CA, a través del parámetro Z, la probabilidad de la siguiente célula desconocida, para un ancestro, es obtenida sin ambigüedad por su algoritmo CA

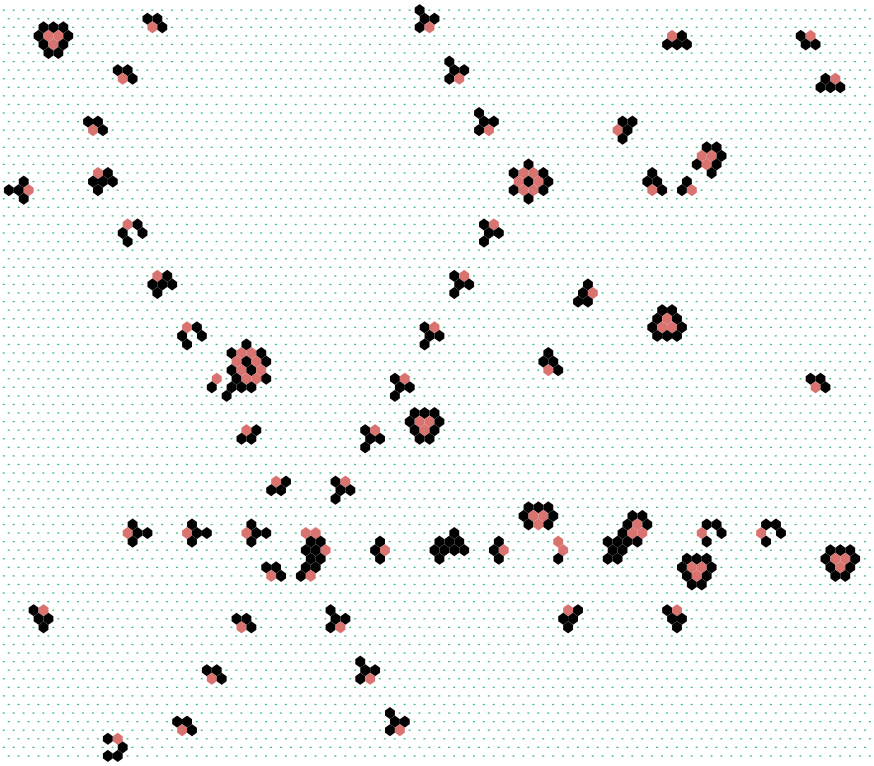


Figura 7. Presentamos un estado global del CA hexagonal 2D con tres estados, conocido como la regla espiral (spiral rule) [10], con parámetros $n = 88 \times 88, k = 7$. Esta regla presenta, particularmente, una interesante diversidad de estructuras complejas estáticas y con desplazamientos, emergiendo en su espacio de evoluciones, entre ellas tenemos: glider guns espirales, glider guns movibles, auto reproducción a través de choques de gliders. Los gliders se mueven orientados por la posición de la célula en color rojo.

inverso [11, 12, 17]. Esto es procesado en dos direcciones, con $Z_{izquierdo}$ y $Z_{derecho}$ para altos valores de Z. Ya que Z es llevada desde 0 hasta 1 con un cambio de dinámicas desde lo ordenado hasta los caóticos (Fig.10), con la densidad de sus hojas se obtiene una buena medida de su convergencia (decrementando) (Fig.3). Si el tamaño del sistema se incrementa entonces éste converge a reglas con orden y en un radio inferior converge a las reglas complejas, el restante es para las reglas caóticas que componen mucho del espacio de las reglas de evolución (Fig.11).

Sin embargo, existe una clase de máximo caos, reglas de "cadena", donde $Z_{izquierdo}$ XOR $Z_{derecho}$ es igual a 1. La convergencia y densidad de hojas decrece con el tamaño de n (Fig.11). Cuando n se incrementa en grados ≥ 2 llega a ser menos probable y además decrece la densidad de hojas – en el límite (de un tamaño muy grande) ambas medidas se aproximan a cero (Fig.9). Para valores

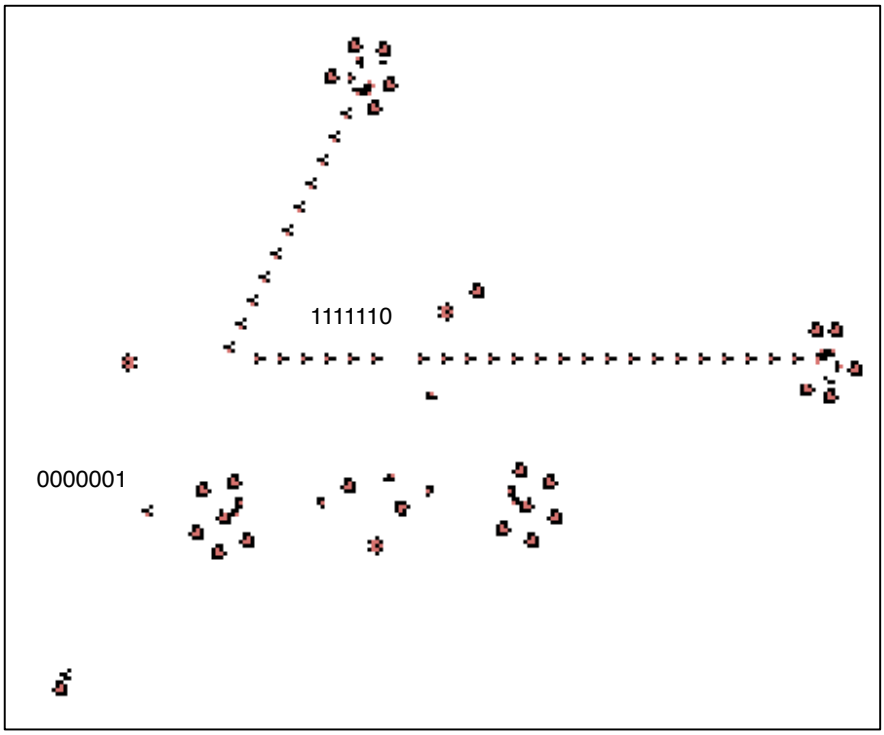


Figura 8. Una compuerta NOT es implementada en el 2D CA complejo, la regla espiral. Inspirado desde las construcciones en el Juego de la Vida, aquí se ilustra una condición inicial diseñada para producir una cadena binaria como entrada 1111110, ésta es tranformada a través de una negación y produce continuamente la cadena de bits 0000001. Todas estas operaciones son realizadas desde la producción de partículas en glider gun, choques de gliders y sincronización. La simulación fue implementada en DDLab y diseñada por Genaro J. Martínez (21 de abril de 2008) `http://www.youtube.com/watch?v=_bC5ucq_sKc`.

grandes de n y usos prácticos, las trayectorias están compuestas de largas cadenas de estados sin ramas (Fig.12), y esto es posible gracias a la unión de dos estados tanto hacia adelante como hacia atrás.

Supongamos que B es un estado con información y puede ser encriptado (Fig.12) iterando hacia atrás dada la función $A = B - x$ con el algoritmo inverso de un CA, que es especialmente eficiente para las reglas de cadena. Entonces A puede ser decodificado (Fig.13) ejecutando hacia adelante x pasos la regla adecuada, es decir, con la llave de encriptación. Acerca de la raíz cuadrada del espacio de reglas binarias, ésta es compuesta de reglas de cadena, que pueden ser contruidas aleatoriamente para demostrar un gran número de llaves de encriptación. La Fig.13 ilustra la información transformada, el estado inicial contiene la información sin transformar mientras que los demás estados, antes y después son

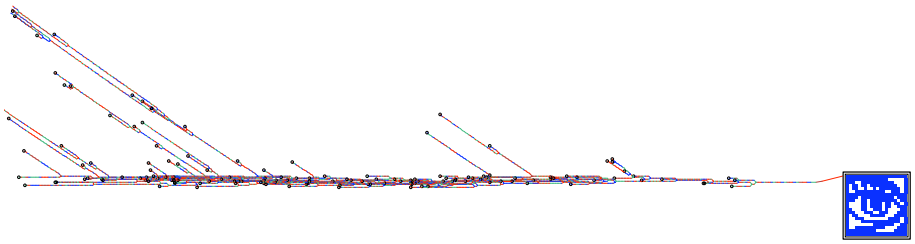


Figura 9. Mostramos el subárbol para un CA de 1D (regla de cadena) con $n = 400$. La raíz del árbol (el ojo) es ilustrado en 2D (20×20). Los iteraciones hacia atrás se detienen hasta los 500 pasos (como en la Fig. 12). El subárbol tiene 4,270 estados y la densidad tanto de las hojas como las ramificaciones es muy bajo (cerca de 0.03) – donde la máxima rama es igual a 2.

caóticos. Esto hace que el sistema salga y entre en caos, limitado únicamente por la "velocidad de la luz" de un 1D CA. Los métodos [20] están implementados en DDLab.

4. Memoria y aprendizaje

El campo de conjuntos de atracción (Fig.15) revela que el contendio de la memoria de contenido direccionable está presente en las redes dinámicas discretas y además muestra su composición exacta, donde la raíz de cada subárbol (también de cada atractor) clasifica todos los estados que fluyen dentro de él y si el estado raíz es un disparador a algún otro sistema, todos los estados en el subárbol podrían, en principio, ser reconocidos como corresponde a una entidad conceptual particular. Esta noción de memoria alejada del equilibrio [13, 14], es una extensión del trabajo de Hopfield [1] y otros conceptos clásicos de memoria en redes neuronales artificiales, que depende exclusivamente de los atractores.

Viendo la dinámica desde dentro de un atractor, ésta se encuentra precisamente de forma descendente y abre toda una jerarquía de subcategorías. Aprendiendo en este contexto, es un proceso de adaptación de las reglas y las conexiones en la red, para modificar subcategorías en el comportamiento requerido – modificando la fina estructura de subárboles y conjuntos de atracción.

Los CA clásicos no son sistemas ideales para implementar estos delicados cambios, porque se encuentran limitados a una regla universal y vecindad local, un *requisito* para el surgimiento de estructuras emergentes, y que limita seriamente su flexibilidad para establecer categorías. Por otra parte, la dinámica en CA tiene simetrías y jerarquías producto de sus condiciones a la frontera [11]. Sin embargo, los CA tienen un grado de estabilidad en su comportamiento cuando se mutan algunos bits en su regla – donde algunos bits son más sensibles que otros. La regla puede ser observada como el genotipo y su comportamiento (patrones

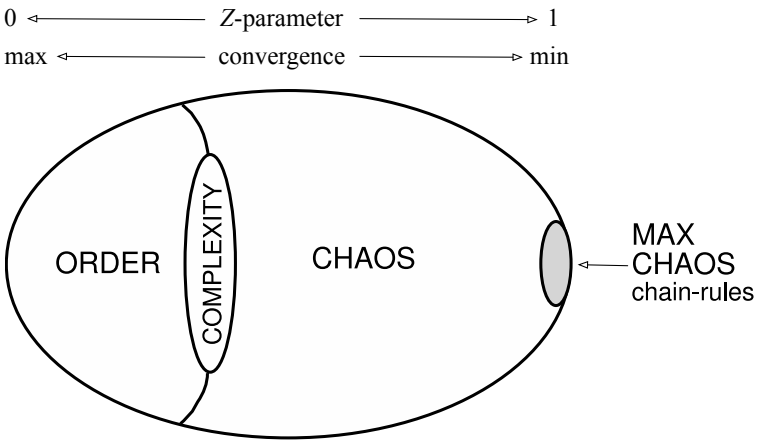


Figura 10. Una vista general del espacio de reglas (después de Langton[2], ya que Langton desconocía la existencia de las reglas de cadena). Ajustada al parámetro Z, de 0 a 1 cambia la dinámica desde la convergencia máxima hasta la mínima, desde orden hasta caos, atravesando la fase de transición donde se ubica la complejidad. Las reglas de cadenas, a la derecha, ilustran la región con máximo caos y muestran la mínima convergencia, decrementando con el tamaño del sistema y haciendo ellos ideales para la encriptación dinámica.

espacio tiempo o conjuntos de atracción) como el fenotipo [11]. La Fig.14 ilustra la mutación de algunos CA campos de atracción.

Con las RBN y DDN existe más libertad para modificar sus reglas y conexiones que con los CA. Los algoritmos para el aprendizaje y la perdida de aprendizaje fueron inventados e implementados en DDLab [13, 14, 15]. Los métodos designan ancestros a un estado objetivo para corregir una unión mal hecha entre el objetivo y el estado actual, cambiando bits en la reglas o cambiando conexiones. En los sitios afectados la generalización es evidente y los árboles de transiciones algunas veces son transplantados con la reasignación de ancestros.

4.1. Modelando redes neuronales

Ahora podemos establecer algunas conjeturas y especulaciones ¿Cuáles son las implicaciones de la memoria en el cerebro de los animales? La primera conjetura, quizás no tan contraversial, es que el cerebro es un sistema dinámico (no una computadora o una máquina de Turing), compuesto de la interacción de subredes. Segunda, el código neuronal es basado en patrones distribuidos y activados en subredes neuronales (no es la frecuencia de disparo de una neurona), donde cada disparo es sincronizado por muchos mecanismos posibles: fijación de la fase, inter neuronas, uniones gap, nanotubos de la membrana, interacciones efáticas.

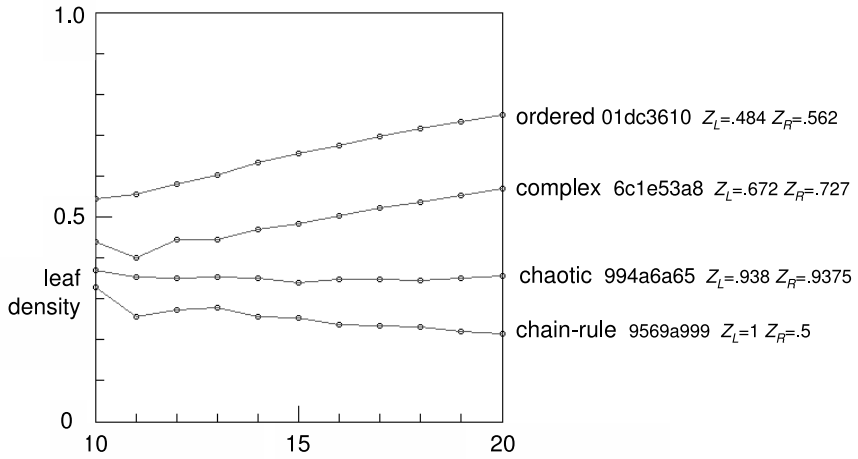


Figura 11. La densidad de las hojas (configuraciones Jardín del Edén) están graficadas con respecto al tamaño de n, para cuatro reglas típicas de CA, ilustrando la convergencia que es predicha por el parámetro Z. Únicamente, las reglas de cadena fuertemente caóticas muestran un decremento. Las medidas son realizadas desde los campos de conjuntos de atracción y consecuentemente para el espacio de evoluciones completo, parámetros $k = 5$, $n = 10$ hasta $n = 20$.

El comportamiento aprendido y la memoria trabajan por patrones de activación en subredes, siguido automáticamente con las subredes de los conjuntos de atracción. El reconocimiento es fácil porque el estado inicial es conocido. El hecho de recordar es lo realmente complicado, porque una asociación debe ser evocada para iniciar el flujo en el subárbol correcto.

En un nivel muy básico nos podemos preguntar. ¿Cómo podríamos hacer de un modelo DDN un remedio semiautónomo de neuronas en el cerebro cuya actividad es sincronizada? Esto sería un modelo basado en conexión de redes, donde un subconjunto de neuronas conectadas se conectaría entonces a una neurona dada. Entonces la regla lógica en una red elemental, que podría ser reemplazada por un circuito (como árbol) combinatorial equivalente, modela la lógica ejecutada por el micro circuito sináptico de un árbol de nueronas dendríticas, determinando si o no debería de disparar en el siguiente tiempo. Esto es más complejo que el umbral de la función en redes neuronales artificiales. El aprendizaje evoluciona cambios en el árbol dendrítico, o más radicamente, los axones pueden alcanzar a conectar (o desconectar) neuronas fuera del subconjunto actual.

4.2. Modelando redes regulatorias genéticas

Los diversos tipos de células en organismos multicelulares, como los músculos, cerebro, piel, hígado y demás (cerca de 210 organismos dentro de los humanos),

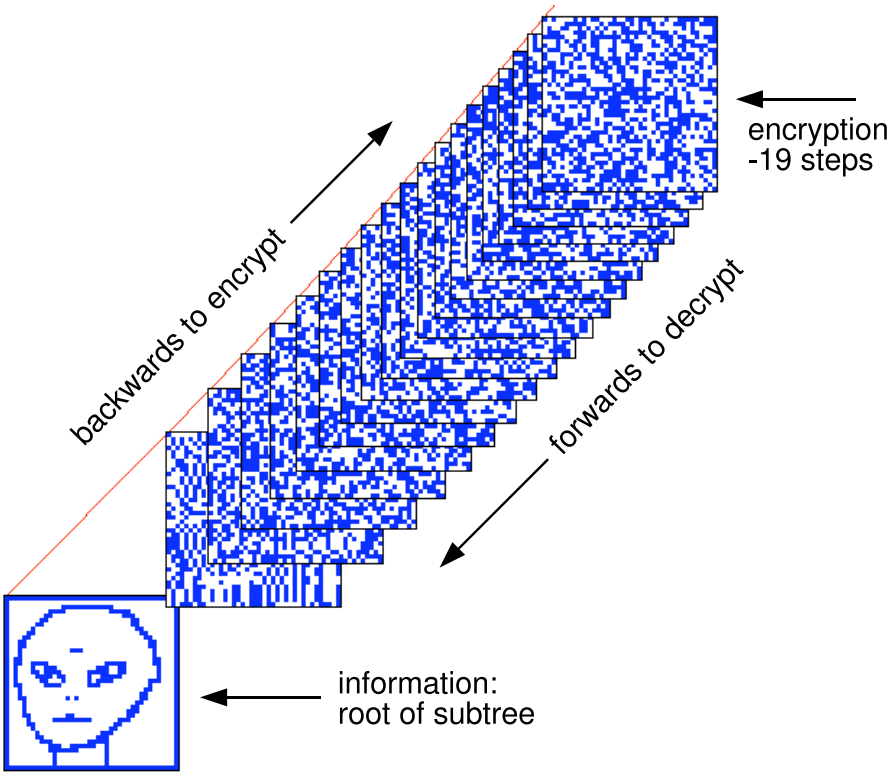


Figura 12. Mostramos un subárbol de un CA en 1D con $k = 7$ encriptando desde la raíz a un "alien," aqui se muestra la construcción del patrón en 1D a 2D ($n = 1600, 40 \times 40$) y que puede ser representado además en ASCII o algún otro formato de información. Las iteraciones hacia atrás se detienen hasta los 19 pasos.

tienen el mismo ADN y el mismo conjunto de genes. Los diferentes tipos se derivan desde diferentes patrones dada las expresiones de los genes. Por lo que surge la pregunta ¿Cómo hacer que los patrones conserven su identidad? ¿Cómo hacer que la célula recuerde qué es lo que contenía?

Es bien conocido en biología, que existe una red regulatoria genética donde los genes regulan cualquier otra actividad con proteínas regulatorias [9]. Un tipo de célula depende de su subconjunto particular de genes activos, donde cada patrón derivado de la expresión del gen necesita ser estable y además adaptable. Algunos biólogos celulares que no se encuentran familiarizados con los sistemas complejos, encuentran las ideas de Kauffman contraversiales [4, 5], donde una red regulatoria genética es un sistema dinámico donde los tipos de células son atractores, y que además él modeló con RBN. Aunque esta aproximación tiene

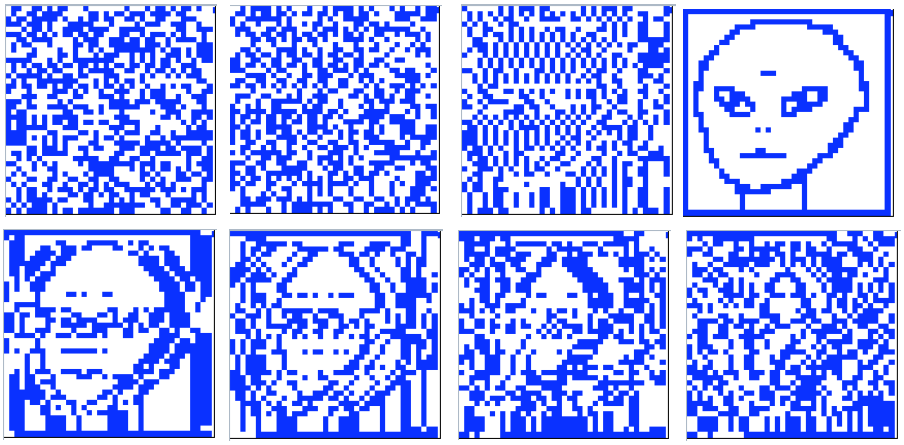


Figura 13. Para descifrarlo, empezamos desde el estado en que se quedó encriptado en la Fig. 12 y ejecutamos la misma regla hacia adelante 19 pasos. Esta figura ilustra las -3 hasta $+6$ iteraciones del espacio celular, donde podemos ver como la cara del "alien" se pierde antes y después del tiempo 0.

un tremendo poder explicativo y difícil de verlo como una alternativa plausible [16].

Un gen es regulado por proteínas desde otros genes, que además debe incluir al mismo gen. En un nivel molecular, una combinación de proteínas regulatorias vincula a una secuencia promotora, que cambia el estado del gen en ON (ENCENDIDO) y OFF (APAGADO). En un nivel macro, éste determina el radio en que el gen transcribe ARN para producir su proteína específica, que puede ser medida a través de un análisis de micro arreglos.

En el modelo de Kauffman basado en RBN, un estado del gen está ENCENDIDO o APAGADO y sus conexiones son los conjuntos de genes que conservan sus proteínas regulatorias. Su regla (función booleana) representa como las proteínas se combinan en el sitio de enlace para determinar el estado del gen. Kauffman trabajó su modelo desde numerosos estados iniciales para identificar los principales atractores – la longitud de la trayectoria y el volumen del conjunto pueden ser determinados estadísticamente, un método además implementado en DDLab y útil para largos sistemas ordenados.[2] Los resultados demostraron que el número de entradas, k, fueron una clave variable para el número de atractores [5]. Esta aproximación enfatiza el balance de las dinámicas orden/caos, que dependen de k o alternativamente canalizando entradas, en una diagonal para valores mayores que k inducen orden [3]. Las medidas para orden/caos incluyen: la gráfica de Derrida, extensión de daños, genes congelados y distribución de atractores. La localidad de conexiones aleatorias reduciendo la extensión de conexiones cuando

[2] Los atractores caóticos son difíciles de encontrar con este método porque las trayectorias y los atractores llegan a ser muy largos para ser identificados.

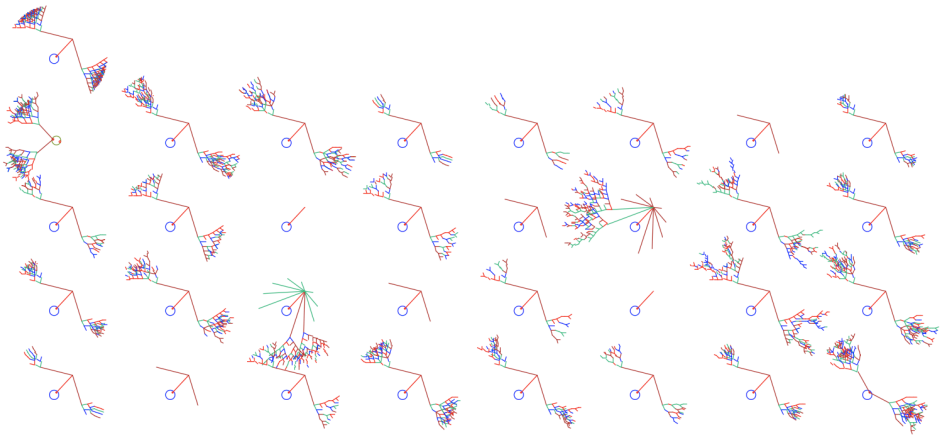
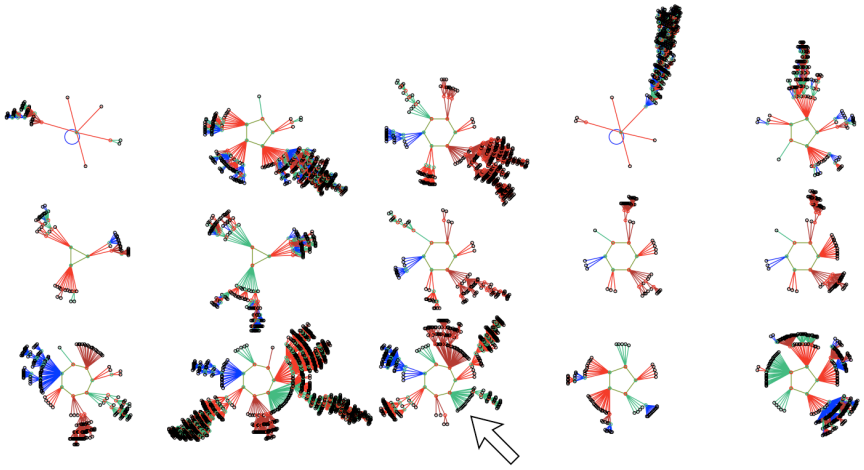


Figura 14. Mutaciones de conjuntos de atracción para $v = 2$, $k = 3$, ECA regla 60 ($n = 8$). En la partre superior izquierda se ilustra la regla original, donde todos los estados caen en un conjunto regular base. Entonces la regla es transformada, primero, por su regla equivalente $k = 5$ (f00ff00f en notación hexadecimal), con 32 bits en su tabla. Todas las 32 mutaciones son calculadas. Si la regla es el genotipo entonces el conjunto de atracción puede ser visto como el fenotipo.

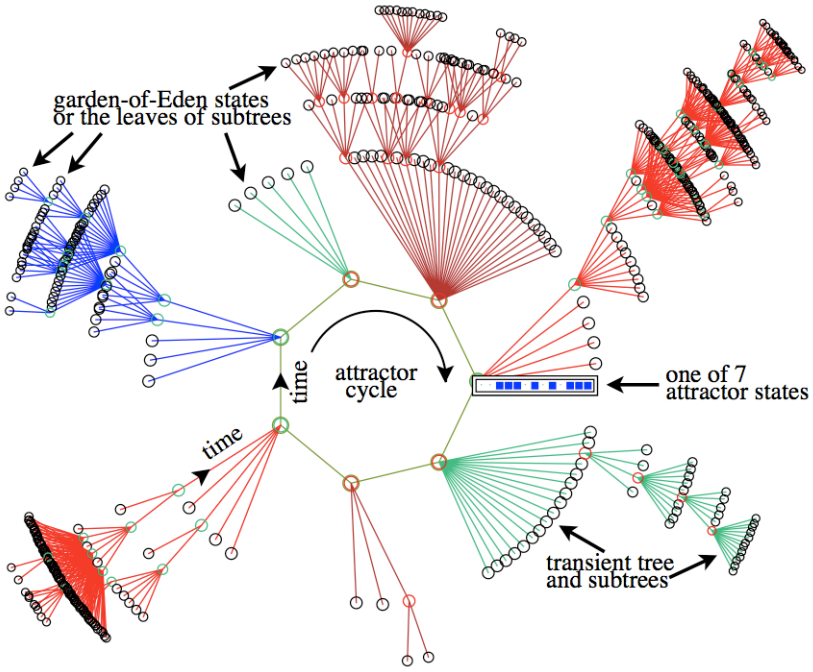
la red es presentada en un arreglo regular, además induce orden [15]. El nuevo método procesa conjuntos de atracción de RBN con todos los detalles [13], otorgando más profundidad al modelo [9, 3].

En un tipo de célula, el patrón derivado de la expresión del gen, pueden ser vistos como patrones (patrones en el espacio y tiempo), un gen en particular puede pasar por algunos largos intervalos de su tiempo APAGADO (congelado) o repentinamente alterado. Pero si varios genes están cambiando rápidamente (dinámica caótica) entonces la célula deberá estar inestable. Inversamente, si varios genes están congelados entonces la célula deberá estar demasiado estable para su comportamiento adaptativo. Las células necesitan constantemente adaptarse a sus patrones de expresión de genes, como una respuesta a factores de crecimiento/diferenciación, inter celulares y otras señales luego reviertan a su dinámica usual. Un tipo de célula es probablemente un conjunto de patrón de expresión de gen, estrechamente vínculado y no solo en los atractores, sino que cambia en su entorno con el conjunto de atracción, permitiendo de esta manera una medida esencial de flexibilidad en su comportamiento. Aunque también, demasiada flexibilidad debe permitir una perturbación para mover la dinámica a un conjunto de atracción diferente, desde una célula del hueso hasta una célula de la grasa o en alguna célula extraña – una célula cancerosa.

El modelo indica que la evolución a alcanzado a un delicado balance entre orden y caos – aunque con inclinación hacia una convergencia creciente y ordenada



(a)



(b)

Figura 15. En (a) se ilustra el campo del conjunto de atracción de una red booleana aleatoria (RBN), $k = 3$, $n = 13$. Los $2^{13} = 8,192$ estados están organizados en 15 conjuntos con periodos de atracción que van de 1 a 7 y con un volumen entre 68 y 2,724. En (b) se presenta en detalle un conjunto de atracción, como configuraciones de bits (flecha de arriba indicada en (a)) con 604 estados de los cuales 523 son hojas y el atractor es de periodo igual a 7. La dirección del tiempo es hacia dentro del atractor y con orientación al sentido de las manecillas del reloj.

[3]. La estabilidad de los atractores a perturbaciones pueden ser analizadas por la gráfica de salto (jump-graph) (Fig.16), que permite ilustrar la probabilidad de saltar entre conjuntos de bits perdidos (bit-flips) a los estados atractores [18]. Estos métodos están implementados in DDLab y generalizados para los DDN, donde los valores de v pueden ser mayores que 2 (binario) y un gen puede ser fracionado como un simple estado ON/OFF.

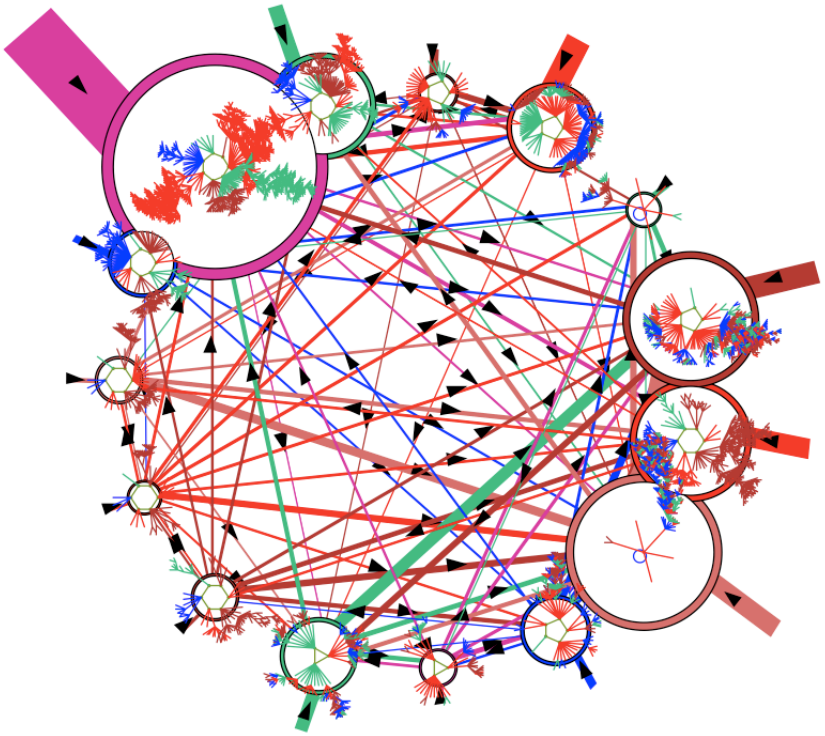


Figura 16. La gráfica de salto (jump-graph) (de la RBN de la Fig. 15) muestra la probabilidad de cambiar entre conjuntos de atracción, cambiando una célula en el estado desde 0 a 1 o de 1 a 0. Los nodos representan los conjuntos que están representados a escala de acuerdo al número de estados en el conjunto (volumen del conjunto). Las aristas están a escala de acuerdo a dos conjuntos: el volumen y la probabilidad de salto. De esta manera, las flechas indican la dirección del salto y las flechas cortas representan una auto conexión (self-jumps); los saltos que regresan al conjunto padre indican el grado de estabilidad, es decir, donde existe más estabilidad por casualidad. El conjunto de atracción relevante es dibujado dentro de cada nodo.

Un cambio reciente en el modelo del problema inverso, es la inferencia de una arquitectura basada en redes desde la información de patrones en el espacio

de evoluciones. Esto es aplicado a la inferencia de redes regulatorias genéticas reales, desde la dinámica observada en la expresión de genes [3].

5. Conclusiones

El artículo presenta una revisión acerca de una variedad de redes dinámicas discretas, donde el conocimiento desde sus conjuntos de atracción nos ofrece un novedoso entendimiento y algunas aplicaciones: en la dinámica de partículas de CA complejo y en la auto organización; en CA más caótico (reglas de cadena) donde la información puede encontrarse oculta y recuperada desde un umbral de caos; y en el caso de las redes booleanas aleatorias y redes multi-valor aleatorias, que son aplicadas a modelos de redes naturales y genéticas en biología. Sin embargo, varias líneas de investigación permanecen abiertas (en sistemas dinámicos discretos), debemos ver el mérito que es pensar acerca de ellos en la perspectiva de los conjuntos de atracción.

6. Manual y software para DDLab

Los resultados obtenidos y graficados en el presente artículo, así como las simulaciones y experimentos descritos, fueron realizados con el software (de código abierto) "Discrete Dynamics Laboratory" (DDLab)[19]. La reciente versión de DDLab y la reciente edición del manual "Exploring Discrete Dynamics" [22] se encuentran disponibles desde `http://www.ddlab.org/`.

Referencias

[Note] Referencias y más publicaciones de A. Wuensche, se encuentran disponibles en `http://www.cogs.susx.ac.uk/users/andywu/publications.html`.

[1] Hopield, J.J. (1982) Neural networks and physical systems with emergent collective abilities, *Proceeding of the National Academy of Sciences* **79** 2554–2558.

[2] Langton, C.G. (1990) Computation at the edge of chaos: Phase transitions and emergent computation, *Physica D* **42** 12–37.

[3] Harris, S.E., Sawhill, B.K., Wuensche, A., & Kauffman S.A. (2002) A Model of Transcriptional Regulatory Networks Based on Biases in the Observed Regulation Rules, *Complexity* **7(4)** 23–40.

[4] Kauffman, S.A. (1969) Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets, *Theoretical Biology* **22(3)** 439–467.

[5] Kauffman, S.A. (1993) *The Origins of Order*, Oxford University Press.

[6] Kauffman, S.A. (2000) *Investigations*, Oxford University Press.

[7] Martínez, G.J., Adamatzky, A., Seck-Tuoh-Mora, J.C., & Alonso-Sanz, R. (2010) How to make dull cellular automata complex by adding memory: Rule 126 case study, *Complexity* **15(6)** 34–49.

[8] Martínez, G.J., Adamatzky, A., Stephens, C.R., & Frank, A. (2011) Cellular automaton supercolliders, *International Journal of Modern Physics C* **22(4)** 419–439.

[9] Somogyi, R. & Sniegoski. C.A. (1996) Modeling the complexity of genetic networks: understanding multigene and pleiotropic regulation, *Complexity* **1** 45–63.

[10] Wuensche, A. & Adamatzky, A. (2006) On spiral glider-guns in hexagonal cellular automata: activator-inhibitor paradigm, *International Journal of Modern Physics C* **17(7)** 1009–1026.

[11] Wuensche, A. & Lesser, M.J. (1992) *The Global Dynamics of Cellular Automata; An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*, Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley, Reading, MA.

[12] Wuensche, A. (1994) Complexity in 1D cellular automata; Gliders, basins of attraction and the Z parameter, Santa Fe Institute *Working Paper* 94-04-025.

[13] Wuensche, A. (1994) The ghost in the machine: Basin of attraction fields of random Boolean networks. In: *Artificial Life III*, Langton, C.G. (ed.), Addison-Wesley, Reading, MA, 496–501.

[14] Wuensche, A. (1996) The Emergence of Memory: Categorisation Far From Equilibrium, In: *Towards a Science of Consciousness: The First Tuscon Discussions and Debates*, Hameroff, S.R., Kaszniak, A.W., & Scott, A.C. (eds.), MIT Press, Cambridge, MA, 383–392.

[15] Wuensche, A. (1997) "Attractor basins of discrete networks: Implications on self-organisation and memory," Cognitive Science Research Paper 461, *DPhil Thesis*, University of Sussex.

[16] Wuensche, A. (1998) Genomic Regulation Modeled as a Network with Basins of Attraction, *Proceedings of the 1998 Pacific Symposium on Biocomputing*, World Scientific, Singapore.

[17] Wuensche, A. (1999) Classifying cellular automata automatically; finding gliders, filtering, and relating space-time patterns, attractor basins, and the Z parameter, *Complexity* **4(3)** 47–66.

[18] Wuensche, A. (2004) Basins of Attraction in Network Dynamics: A Conceptual Framework for Biomolecular Networks, In *Modularity in Development and Evolution*, G. Schlosser & G.P. Wagner (eds.), Chicago University Press, chapter 13, 288–311.

[19] Wuensche, A., "Discrete Dynamics Lab" (DDLab), software for investigating discrete dynamical networks, `http://www.ddlab.org/`, 1993–2009.

[20] Wuensche, A. (2009) Cellular Automata Encryption: The Reverse Algorithm, Z-Parameter and Chain Rules, *Parallel Processing Letters* **19(2)** 283–297.

[21] Wuensche, A. (2010) Complex and Chaotic Dynamics, Basins of Attraction, and Memory in Discrete Networks", *Acta Pysica Polonica B* **3(2)** 463–478.

[22] Wuensche, A. (2011) *Exploring Discrete Dynamics; The DDLab Manual*, Luniver Press, UK.

Áreas de oportunidad en el estudio de autómatas celulares reversibles

Juan Carlos Seck Tuoh Mora

Centro de Investigación Avanzada en Ingeniería Industrial
Universidad Autónoma del Estado de Hidalgo
Carr. Pachuca Tulancingo Km 4.5, Pachuca 42184 Hidalgo, México
jseck@uaeh.edu.mx

Resumen Este manuscrito presenta una visión general de los resultados más relevantes en la investigación que se ha desarrollado en los últimos 40 años en autómatas celulares reversibles en una dimensión, así como la exposición de cuatro áreas de oportunidad que son factibles de desarrollar con los resultados actuales, para el análisis de dichos sistemas. Estas áreas incluyen: Caracterización de la máxima longitud de la mínima vecindad inversa usando herramientas de dinámica simbólica, conteo de autómatas celulares reversibles construyendo solamente especímenes válidos, obtención del periodo de una configuración finita sin evolucionar el autómata y la caracterización del comportamiento reversible en autómatas celulares con memoria.

1. Introducción

La investigación de los autómatas celulares ha tenido como puntos de interés primero explorar sus posibilidades para simular sistemas reales. En este sentido, la disponibilidad de equipos computacionales cada vez más rápidos ha permitido experimentar y aplicar los autómatas celulares para este fin. La segunda razón tiene sus orígenes tanto en los trabajos elaborados por Post y Turing entre otros, que establecen las condiciones y restricciones de un dispositivo para realizar procesos computables. Un ejemplo de esto es el mo- delo del sistema nervioso de McCulloch y Pitts , el cual está basado en interacciones locales de unidades elementales o neuronas, dicho modelo muestra que las interacciones locales de unidades simples son capaces de generar un comportamiento global complejo . Así, otra línea de investigación es entender como en un autómata celular , las interacciones locales de sus partes inducen un comportamiento global capaz de realizar comportamientos dinámicos *interesantes*.

Un tipo de autómata celular ampliamente estudiado es aquel cuyo comportamiento global es invertible , en otras palabras, cada uno de estos autómatas puede regresar a todos los estados globales que anteriormente había generado. Los autómatas celulares con estas características son llamados reversibles y su estudio matemático resulta relevante por la convergencia que tienen diversos

campos de investigación: como computación concurrente, procesos que preservan información, computación cuántica, modelado de sistemas granulares, codificación y cifrado de datos entre otros [47], [50], [21], [26].

Por mucho tiempo, los autómatas celulares reversibles parecían ser muy raros y lo que se sabía de ellos era fácilmente resumido. Sin embargo, esta falta de interés desaparece en 1962, cuando Moore [28] examina autómatas celulares con estados globales sin ancestros. Este trabajo es enriquecido por Myhill [31]. El tema de reversibilidad es directamente abordado hasta 1972 por los artículos de Richardson [36] y Amoroso y Patt [6], que presentan un procedimiento sistemático para decidir si un autómata celular unidimensional es reversible o no. Pero parte de los resultados de estos artículos ya habían sido anticipados en un contexto matemático más abstracto por Hedlund [17].

En 1977, Toffoli prueba la existencia de autómatas reversibles que también son constructores universales; independientemente, Fredkin analiza recurrencias invertibles por medio de primitivas booleanas [15]. Tomando como base el trabajo de Hedlund , Nasu define propiedades fundamentales de los autómatas reversibles usando herramientas gráficas [32], [33] y [34].

Después de finales de los 80's, aparecen trabajos muy interesantes acerca de los autómatas celulares unidimensionales reversibles. Usando la técnica de estados particionados, Morita prueba la existencia de autómatas reversibles que hacen computación universal y que son autorreproductivos [29] y [30]. Hillman [18] y Moraal [27] presentan nuevos algoritmos para detectar autómatas celulares unidimensionales reversibles; Boykett da un esquema algebraico para construirlos [9] y Seck et al. utilizan herramientas matriciales para el mismo fin, detectando autómatas reversibles con diferentes tamaños en su vecindad inversa [39].

Aunque los diagramas de de Bruijn habían sido utilizados por Nasu, estos se vuelven populares para analizar el comportamiento local de los autómatas celulares unidimensionales reversibles gracias a los artículos de Jen quien los utiliza para calcular los ancestros de una secuencia de estados [19]. Otras aplicaciones incluyen las desarrolladas por Voorhees para obtener propiedades de autómatas reversibles y sobreyectivos [48], por Sutner para definir una cota máxima del comportamiento inverso de un autómata reversible [45] y [46]; por Seck et al. para encontrar la regla inversa de una clase particular de dichos sistemas [40], [41] y por McIntosh , quien los aplica para establecer procedimietos para verificar si un autómata celular es reversible [26].

Un artículo fundamental que presenta una caracterización determinística basada en permutaciones en bloque y corrimientos del comportamiento local para autómatas reversibles es desarrollado por Kari [20], en donde se explica como se conserva la información inicial de un autómata durante su evolución.

Por otra parte, el estudio en dinámica simbólica está en relación estrecha con entender el comportamiento a largo plazo de los autómatas celulares. En dinámica simbólica, los autómatas reversibles son mapeos invertibles entre sistemas de corrimiento. En 1973, Williams establece resultados matriciales importantes para comparar sistemas de corrimiento, incluyendo mapeos que coinciden con autómatas celulares unidimensionales reversibles [49]. Esta investigación con

base en matrices continúa con el trabajo de Boyle [10]. Excelentes referencias sobre este tema se encuentran en los libros de Lind y Marcus [24] y Kitchens [23]. Estos trabajos han inspirado estudios más detallados que presentan resultados importantes sobre la dinámica topológica de autómatas reversibles [35], [7], [12], [13], [11], [38], [42]; y que analizan la decibilidad de problemas dinámicos fundamentales en estos sistemas [8], [22], [25].

Los párrafos anteriores son solo una pequeña muestra del rico desarrollo que han tenido los autómatas celulares tanto en su estudio como en su aplicación; sin embargo, esto no significa que su investigación ya esté completa o terminada. Así, este manuscrito presenta varias propuestas de estudio para desarrollarse con base en los resultados descritos anteriormente. Por supuesto, este listado no es único ni excluyente, pero corresponde con los trabajos que el autor ha desarrollado, así como con los avances que se han obtenido recientemente en el estudio de autómatas celulares. Estas propuestas incluyen:

- Caracterización de la máxima longitud de la mínima vecindad inversa usando herramientas de dinámica simbólica.
- Conteo de autómatas celulares reversibles construyendo solamente ejemplares válidos.
- Obtener el periodo de una configuración finita sin evolucionar el autómata.
- Caracterizar el comportemiento reversible en autómatas celulares con memoria.

2. Conceptos básicos

En este trabajo estamos interesados en autómatas celulares con un número finito de células y condiciones periódicas de frontera, por lo que las definiciones que se den, serán dentro de este contexto.

Un autómata celular unidimensional consiste de un conjunto S de estados, cuya cardinalidad es s; un arreglo finito de células $c = x_1 \dots x_m$ con $m \in \mathbb{Z}^+$, donde cada célula toma un valor del conjunto S. Cada arreglo con una asignación de estados es una configuración del autómata; por lo tanto, el conjunto de configuraciones se define como $C = S^m$.

Para cada configuración c, cada célula evoluciona a partir de su estado actual y el de sus r vecinos a cada lado, de esta forma la evolución depende de una vecindad determinada por cada célula y sus $2r$ vecinos. Así, el mapeo del conjunto S^{2r+1} al conjunto S es una regla de evolución $\varphi : S^{2r+1} \to S$.

A cada célula de x_i^t de una configuración c^t se le aplica la regla de evolución $\varphi(x_{i-r}^t \dots x_i^t \dots x_{i+r}^t) = x_i^{t+1}$, donde el superíndice significa tiempo, y esto se hace para cada $x_i \in c^t$.

El resultado es una nueva configuración c^{t+1} en donde los estados de cada célula son actualizados simultaneamente por la regla de evolución. De esta manera, el mapeo local de la regla de evolución induce un mapeo global $\phi : C \to C$ entre configuraciones. Si consideramos una conguración inicial c^1, aplicando este proceso obtendremos la configuración c^2, y esto continúa de forma consecutiva para tiempos subsecuentes.

Cada vecindad forma un estado aplicando la regla de evolución φ; así, en general, cualquier ancestro de $w \in S^n$ tiene $n + 2r$ estados. Un ancestro de una secuencia de 2r estados tendrá $4r$ estados. Entonces, cada ancestro puede particionarse en dos secuencias disjuntas de $2r$ estados cada una. Con esto, se puede definir un nuevo conjunto de estados con cardinalidad s^{2r}, en donde se representa cada secuencia de S^{2r} con un único estado . Sobre este nuevo conjunto de estados, se define una nueva regla de evolución que simula a la regla original φ. Esta regla mapea elementos de S^{4r} a S^{2r}. Esta transformación demuestra que todo autómata celular unidimensional puede simularse por otro con s^{2r} estados y radio de vecindad 1/2. Con esto, una propiedad particular en autómatas celulares con radio de vecindad 1/2, se cumple también para todo el conjunto de autómatas celulares. Por lo anterior, en lo que sigue de este trabajo, se tratará con autómatas celulares de s estados y radio de vecindad 1/2.

Un autómata celular es reversible si su mapeo global es invertible por la acción de una regla de evolución inversa a la original. El conjunto de estados del autómata permanece sin cambio, pero el tamaño de la vecindad en la regla inversa puede ser diferente al tamaño de la vecindad en la regla original.

La parte interesante de este comportamiento reversible es que cada regla de evolución mapea una vecindad de varias células a una sola; es decir, el comportamiento local no es reversible ya que el número de vecindades es mayor que el número de estados, sin embargo, este comportamiento local define un comportamiento global reversible . Así, primero se debe caracterizar el comportamiento local para entender la reversibilidad global. Relativo a lo anterior, Hedlund en [17] obtiene resultados fundamentales para la caracterización de los autómatas celulares unidimensionales reversibles, que se puede resumir con las siguientes propiedades para autómatas con tamaño de vecindad 1/2:

- Cada secuencia finita de estados tiene s ancestros.
- Para una longitud dada, los ancestros de cada secuencia tienen L estados izquierdos distintos, una secuencia central única y R estados derechos distintos, cumpliendo con $LR = s$.

Así, en un autómata celular unidimensional reversible , cada secuencia de estados tiene el mismo número de ancestros que todas las demás, sin importar su longitud o los estados que la conformen. Para una longitud dada, los ancestros de cada secuencia tienen una parte central común y sus diferencias aparecen en los extremos. De lo anterior se desprende que los ancestros definen una única forma en la cual una secuencia puede regresar en la evolución del autómata.

A continuación se proponen algunas áreas de investigación a desarrollar en el estudio de autómatas celulares reversibles.

3. Cota máxima para la mínima vecindad inversa

Éste es un problema clásico en el estudio de reversibilidad , y de manera simple se puede formular como cuál es la mínima información necesaria para

regresar en la evolución de un autómata; es decir, cuál es el mínimo número de células que se requieren para definir una regla de evolución inversa.

Este problema en una dimensión fue resuelto por Sutner [45] dando una cota cuadrática respecto al número de estados del autómata, y después este resultado fue mejorado por Czeisler y Kari [14] probando una cota lineal con base en la representación del problema con conceptos de algebra lineal.

En este sentido, se propone investigar el desarrollo de otra demostración basada en herramientas gráficas y de dinámica simbólica que sea más cercana al funcionamiento de un autómata reversible; en particular usando diagramas de de Bruijn , diagramas de Welch y amalgamaciones de estados .

Para autómatas con radio de vecindad 1/2; su representación por diagramas de de Bruijn es una gráfica completa de s nodos donde cada uno es un estado y cada arco dirigido es la evolución de los nodos adyacentes [43], [26]. De esta gráfica, se desprenden otras dos, los diagramas de Welch [32], [40]; cuyos nodos representan los diferentes conjuntos de Welch, cada uno formado por L estados si es el diagrama izquierdo, o cada uno formado por R estados si es el diagrama derecho. Los arcos dirigidos en los diagramas de Welch se obtienen agrupando las conexiones del diagrama de de Bruijn, tomando todas las evoluciones idénticas que surgen de un nodo del diagrama de Welch y que por las propiedades de estos sistemas, mapean a otro conjunto de Welch dentro del mismo diagrama.

En términos de dinámica simbólica, dado que un autómata reversible puede generar toda posible secuencia de símbolos, su diagrama de de Bruijn asociado es isomorfo al corrimiento completo, y por lo tanto se puede convertir en éste por medio de amalgamaciones. Esto ya se ha hecho para autómatas con un índice de Welch unitario [43], sin embargo, para los demás casos, la formalización de este proceso está abierta ya que los mapeos en el diagrama de de Bruijn no se comportan como función.

La propuesta consiste en usar diagramas de Welch los cuales son siempre determinísticos, para demostrar que después de $s-1$ amalgamaciones, todos los ancestros de una determinada secuencia empiezan en un nodo de Welch izquierdo , terminan en un nodo de Welch derecho y tienen un único estado o secuencia interna en común.

4. Conteo de autómatas celulares reversibles

Otra línea de investigación en autómatas reversibles es el conteo de los autómatas reversibles; en este aspecto los primeros trabajos fueron desarrollados por Amoroso y Patt [6] encontrando pocos especímenes para 2 estados y diversos tamaños de vecindad. De este trabajo original han surgido otros avances de mayor envergadura utilizando diversas herramientas tanto gráficas, matriciales y de teoría de grupos como los desarrollados por Hillman [18], Mooraal [27], Boykett [9] y Seck et al. [39]. En estos trabajos, una constante es la generación rápida de reglas candidatas a ser reversibles, las cuales luego son verificadas con algún proceso más *pesado* computacionalmente para revisar si realmente tienen esta característica.

En este sentido, un área de oportunidad se encuentra en la generación directa de reversibles; es decir, no generar reglas candidatas sino comprender cuáles son las propiedades distintivas que conforman una regla de evolución reversible y con base en dichas propiedades, generar solo reglas válidas.

Un paso más sencillo de abordar es tratar este problema para autómatas con un índice de Welch unitario , ya que en este caso la representación matricial de una regla está conformada por permutaciones de S. De esta manera, el proceso para contar solamente permutaciones válidas puede ofrecer una solución para este caso. Generalizar este proceso para autómatas reversibles con índices de Welch diferentes de 1 es otro problema que permanece abierto.

5. Periodo de órbitas

El entender y predecir el comportamiento dinámico de un autómata celular es un problema clásico, y el caso reversible no es la excepción. Resultados recientes de Kari y Lukkarila [22], [25] demuestran que el problema de encontrar el ciclo límite de un autómata reversible es indecidible para configuraciones infinitas.

Sin embargo, para el caso finito, un problema que puede tratarse es conocer el periodo de las órbitas de un autómata reversible; esto es, dada una configuración inicial, en cuantos pasos del autómata regresará a la misma condición inicial .

Una propuesta de solución para este problema es utilizar representaciones del autómata por medio de redes de Petri , y utilizar sus propiedades algebraicas para calcular el número de pasos requerido para que el sistema regrese a la condición inicial.

La relación entre redes de Petri y autómatas celulares ha sido poco explorada [44] [16], [37], y hasta el conocimiento del autor, no existen trabajos que traten el tema de reversibilidad usando ambos conceptos.

La propuesta consiste en construir una red de Petri que represente la dinámica de un autómata celular, una vez probando que los marcados de dicha red reflejen de manera exacta el comportamiento dinámico del autómata correspondiente; se propone aplicar la ecuación de estado asociada a dicha red para resolver un sistema de ecuaciones que represente tanto los disparos de transiciones de la red, como las evoluciones a nivel local que tienen lugar en el autómata. De esta manera, la solución del sistema de ecuaciones representará el número de pasos para llegar de una configuración inicial a otra deseada; si ambas son la misma, entonces la solución deberá mostrar el periodo de dicha configuración.

6. Reversibilidad y memoria

Una variante del modelo clásico de autómatas celulares es aquel en donde se hace uso de memoria para complementar la regla de evolución; es decir, cada célula toma en cuenta sus estados anteriores para formar una célula temporal, así se forma una nueva configuración temporal a la cual se le aplica regla de evolución para definir la nueva configuración del autómata.

Autómatas celulares con memoria se han estudiado para producir comportamientos periódicos y complejos a partir de autómatas clásicos con comportamiento caótico ; como se muestra en el gran número de resultados obtenidos por Ramon Alonso-Sanz [4] y Alonso-Sanz y Bull [3], [5].

El fenómeno de reversibilidad ha sido constantemente estudiado en autómatas celulares con memoria, sin embargo solo para tratar casos particulares y expander comportamientos reversibles desde autómatas clásicos [1], [2].

En este sentido, un área de oportunidad es generalizar las propiedades de los autómatas celulares unidimensionales reversibles en un ambiente con memoria. El objetivo es caracterizar el tipo de memoria que se requiere para conservar la reversibilidad desde autómatas celulares clásicos, o que tipo de memoria resulta adecuada para obtener comportamientos reversibles desde autómatas celulares clásicos que no lo son.

7. Observaciones finales

Si bien el tema de autómatas celulares reversibles ha sido ampliamente estudiado y se han obtenido un número importante de resultados relevantes, sobre todo en su caracterización local por medio de herramientas combinatorias, matriciales y de algebra lineal, así como de problemas indecidibles en su comportamiento global tomando configuraciones infinitas; todavía quedan muchos aspectos por investigar para el caso finito, sobre todo en la aplicación de herramientas combinatorias y de dinámica simbólica, en su relación con otro tipo de herramientas para sistemas discretos, y en el análisis y caracterización del comportamiento reversible para variantes del modelo clásico.

Los problemas anteriormente descritos conllevan la definición de algoritmos que funcionen de manera adecuada para configuraciones con un número *pequeño* de células (decenas o cientos), lo cual aún es conveniente para aplicaciones computacionales donde el número de bits no necesite ser demasiado grande; por ejemplo, para problemas de cifrado de datos, problemas de búsqueda, de optimización o de modelado de sistemas que por lo regular no son muy extensos, como lo son buena parte de los sistemas de ingeniería.

Referencias

[1] Alonso-Sanz, R. (2003). Reversible cellular automata with memory: Patterns starting with a single site seed. *Physica D*, 175(1-2), 1-30.

[2] Alonso-Sanz, R. (2007). Reversible structurally dynamic cellular automata with memory: A simple example. *Journal of Cellular Automata*, 2(3), 197-201.

[3] Alonso-Sanz, R., & Bull, L. (2008). Random number generation by cellular automata with memory. *International Journal of Modern Physics C*, 19(2), 351-375.

[4] Alonso-Sanz, R. (2009). *Cellular automata with memory.* Old City Publishing.

[5] Alonso-Sanz, R., & Bull, L. (2010). One-dimensional coupled cellular automata with memory: Initial investigations. *Journal of Cellular Automata*, 5(1-2), 29-49.

[6] Amoroso, S., & Patt, Y. (1972). Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures. *Journal of Computer and System Sciences*, 6, 448-464.

[7] Blanchard, F., Kurka, P., & Maass, A. (1997). Topological and measure theoretic properties of one-dimensional cellular automata. *Physica D*, 103, 86-99.

[8] Blanchard, F., & Tisseur, P. (2000). Some properties of cellular automata with equicontinuity points. *Annales de l'Institut Henri Poincare (B) Probability and Statistics*, 36(5), 569-582.

[9] Boykett, T. (1994). Combinatorial construction of one-dimensional reversible cellular automata. *Contributions to General Algebra*, 9, 81-90.

[10] Boyle Mike. (1993). Symbolic dynamics and matrices. *Combinatorial and GraphTheoretical Problems in Linear Algebra*, 50, 1-38.

[11] Boyle, M., & Maass, A. (2000). Expansive invertible onesided cellular automata. *Journal of the Mathematical Society of Japan*, 52(4), 725-740.

[12] Cattaneo, G., & Margara, L. (1998). Topological definitions of chaos applied to cellular automata dynamics. *Mathematical Foundations of Computer Science*, 1450, 816-824.

[13] Cattaneo, G., Formenti, E., Margara, L., & Mauri, G. (1999). On the dynamical behavior of chaotic cellular automata. *Theoretical Computer Science*, 217, 31-51.

[14] Czeizler, E., & Kari, J. (2005). A tight linear bound on the neighborhood of inverse cellular automata. In L. Caires, G. F. Italiano, L. Monteiro & M. Yung (Eds.), *Automata, Languages and Programming: Proceedings of the 32nd International Colloquium (ICALP 2005)* (pp. 410-420). Lecture Notes in Computer Science, 3580.

[15] Fredkin, E. (1991). Digital mechanics, an informational process based on reversible universal cellular automata. In H. A. Gutowitz (Ed.), *Cellular Automata, Theory and Experiment* (pp. 254-270). MIT/North-Holland.

[16] Gronewold, A., & Sonnenschein, M. (1998). Event-based modelling of ecological systems with asynchronous cellular automata. *Ecological Modelling*, 108(1-3), 37-52.

[17] Hedlund, G. A. (1969). Endomorphisms and automorphisms of the shift dynamical system. *Mathematical Systems Theory*, 3, 320-375.

[18] Hillman, D. (1991). The structure of reversible one-dimensional cellular automata. *Physica D*, 52, 277-292.

[19] Jen, E. (1989). Enumeration of preimages in cellular automata. *Complex Systems*, 3(5), 421-456.

[20] Kari, J. (1996). Representation of reversible cellular automata with block permutations. *Mathematical Systems Theory*, 29, 47-61.

[21] Kari, J. (2005). Theory of Cellular Automata: A survey. *Theoretical Computer Science*, 334, 3-33.

[22] Kari, J., & Lukkarila, V. (2009). Some undecidable dynamical properties for one-dimensional reversible cellular automata. *Algorithmic Bioprocesses*, Natural Computing Series, Part 9, 639-660.

[23] Kitchens, B. P. (1998). *Symbolic dynamics: One-sided, two-sided and countable Markov shifts*. Springer-Verlag.

[24] Lind, D., & Marcus, B. (1995). *An introduction to symbolic dynamics and coding*. Cambridge University Press.

[25] Lukkarila, V. (2010). Sensitivity and topological mixing are undecidable for reversible one-dimensional cellular automata. *Journal of Cellular Automata*, 5(3), 241-272.

[26] McIntosh, H. V. (2009). *One Dimensional Cellular Automata*. Luniver Press.
[27] Moraal, H. (2000). Graph-theoretical characterization of invertible cellular automata. *Physica D*, 141, 1-18.
[28] Moore, E. F. (1970). Machine models of self-reproduction. In A. W. Burks (Ed.), *Essays on Cellular Automata* (pp. 187-203). University of Illinois Press.
[29] Morita, K. (1992). Computation-universality of one-dimensional one-way reversible cellular automata. *Information Processing Letters*, 42(6), 325-329.
[30] Morita, K. (1995). Reversible simulation of one-dimensional irreversible cellular automata. *Theoretical Computer Science*, 148(1), 157-163.
[31] Myhill, J. (1963). The converse of Moore's Garden-of-Eden Theorem. *Proceedings of the American Mathematical Society*, 14, 685-686.
[32] Nasu, M. (1978). Local maps inducing surjective global maps of one dimensional tesellation automata. *Mathematical Systems Theory*, 11, 327-351.
[33] Nasu, M. (1979). Indecomposable local maps of tessellation automata. *Mathematical Systems Theory*, 13, 81-93.
[34] Nasu, M. (1980). An interconnection of local maps inducing onto global maps. *Discrete Applied Mathematics*, 2, 125-150.
[35] Nasu, M. (1995). Textile systems for endomorphisms and automorphisms of the shift. *Memoirs of the American Mathematical Society*, 546, American Mathematical Society.
[36] Richardson, D. (1972). Tessellations with local transformations. *Journal of Computer and System Sciences*, 6, 373-388.
[37] Schaller, M., & Svozil, K. (2009). Scale-invariant cellular automata and self-similar Petri nets. *The European Physical Journal B*, 69(2), 297-311.
[38] Seck-Tuoh-Mora, J. C., González-Hernández, M., & Pérez-Lechuga, G. (2005). An algorithm for analyzing the transitive behavior of reversible one-dimensional cellular automata with both Welch indices different. *International Journal of Unconventional Computing*, 1(2), 101-121.
[39] Seck-Tuoh-Mora, J. C., Chapa-Vergara, S. V., Martínez, G. J., & Mcintosh, H. V. (2005). Procedures for calculating reversible one-dimensional cellular automata. *Physica D*, 202, 134-141.
[40] Seck-Tuoh-Mora, J. C., Martínez, G. J., & Mcintosh, H. V. (2006). The inverse behavior of a reversible one-dimensional cellular automaton obtained by a single Welch diagram. *Journal of Cellular Automata*, 1(1), 25-39.
[41] Seck-Tuoh-Mora, J. C., González-Hernández, M., & Chapa-Vergara, S. V. (2008). Pair diagram and cyclic properties characterizing the inverse of reversible automata. *Journal of Cellular Automata*, 3(3), 205-218.
[42] Seck-Tuoh-Mora, J. C., González-Hernández, M., Martínez, G. J., Chapa-Vergara, S. V., & Mcintosh, H. V. (2005). Unconventional invertible behaviors in reversible one-dimensional cellular automata. *International Journal of Bifurcation and Chaos*, 18(12), 3625-3632.
[43] Seck-Tuoh-Mora, J. C., González-Hernández, M., Mcintosh, H. V., & Chapa-Vergara, S. V. (2009). Construction of reversible cellular automata by amalgamations and permutations of states. *Journal of Cellular Automata* 4(4), 311-322.
[44] Shen, H. C., Chau, H. L., & Wong, K. K. (1996). An extended cellular automaton model for flexible manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, 11(4), 258-266.
[45] Sutner, K. (1999). Linear cellular automata and de Bruijn automata. In M. Delorme & J. Mazayer (Eds.), *Cellular automata: A parallel Model* (pp. 303-320). Kluwer Academic Publishers.

[46] Sutner, K. (1999). The size of power automata. *Theoretical Computer Science*, 295(1-3), 371-386.
[47] Toffoli, T., & Margolus, N. (1987). *Cellular automata Machines: A New Environment For Modeling*. MIT Press.
[48] Voorhees, B. H. (1996). *Computational analysis of one-dimensional cellular automata*. World Scientific.
[49] Williams, R. F. (1973). Classification of subshifts of finite type. *Annals of Mathematics*, 98(2), 120-153. With errata ibid., (1974), 99, 380-381.
[50] Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media.

Introducción a la computación cuántica: definiciones, tendencias y caminatas cuánticas como caso de estudio

Salvador Elías Venegas Andraca

Grupo de Procesamiento Cuántico de la Información
Tecnológico de Monterrey Campus Estado de México
sva@mindsofmexico.org, salvador.venegas-andraca@keble.oxon.org

Resumen La computación cuántica, rama multidisciplinaria de la ciencia que nace de una combinación ecléctica de la ciencia computacional y la mecánica cuántica, tiene por objetivo utilizar las teorías de las que nace para incrementar sustancialmente la capacidad de los ordenadores para procesar información y resolver problemas. El cómputo cuántico no sólo adopta modelos matemáticos para la creación de algoritmos, también usa las propiedades de la materia con la que se procesa información. Este artículo presenta una introducción concisa a la computación cuántica mediante tres elementos: breve introducción a la historia de las ideas en eta área, presentación rigurosa de algunas ideas fundamentales del cómputo cuántico y un caso de estudio: caminatas cuánticas.

1. Historia de las ideas en computación cuántica

La ciencia computacional es una disciplina que ha permeado y transformado todos los aspectos de la vida moderna. Por este motivo y por la necesidad de tener computadoras más poderosas, en esta rama del conocimiento se hace investigación de frontera en el diseño de nuevos modelos teóricos de computación, el desarrollo de nuevos algoritmos para reducción de complejidad computacional y la creación de nuevas arquitecturas, entre muchos otros temas.

En su forma tradicional, la teoría de la computación tiene una propiedad que es virtud y, posiblemente, también defecto: es una estructura construida exclusivamente sobre la base de la matemática. Dicho de otra manera, las teorías de autómatas, de la computabilidad y de la complejidad no toman en cuenta las propiedades físicas de los dispositivos sobre los cuales, al final de cuentas, se ejecutan los algoritmos (matizando, es posible encontrar textos en los que se toque tangencialmente este tema, más como un guiño hacia la ingeniería electrónica que como componente sustancial de un curso en computación teórica).

La característica enunciada en el párrafo anterior es una virtud por el rigor que define al pensamiento matemático , el cual es, en muy buena medida, responsable del descomunal éxito de la computación. Sin embargo, no tomar en cuenta las propiedades físicas de los sistemas sobre los que se ejecuta un algoritmo podría convertirse en un daño autoinfligido, debido a que dichas propiedades

físicas podrían ser empleadas en la formulación de algoritmos más eficientes. Pongo un ejemplo para provocar la discusión: la simulación del plegado de proteínas es un problema NP-duro [27] mas el cuerpo humano es capaz de plegar proteínas en milisegundos. ¿Cómo explicar esta diferencia abismal en tiempo de ejecución? Si nos atreviésemos a mirar al cuerpo humano como una máquina de procesamiento de datos, ¿podríamos aprender a construir mejores computadoras? ¿Encontraríamos propiedades físico-químicas, minimalistas o emergentes [53], necesarias para aumentar la velocidad de procesamiento, propiedades que no tenemos en la máquina universal de Turing?

Lo anterior invita a preguntarnos si tiene sentido replantear la relación entre la teoría de la computación y otras disciplinas pues, posiblemente, la inclusión de paradigmas nuevos permita encontrar nuevas respuestas a problemas añejos y abiertos. Para apalancar la pertinencia de esta propuesta, observemos que la llegada de la computación a todas las ramas de la ciencia y la ingeniería, así como el empleo de ideas provenientes de la física, la química y la biología para la creación de nuevas computadoras y algoritmos, ha coadyuvado en la construcción de puentes interdisciplinarios, logrando con esto la fertilización de la ciencia computacional con nuevas ideas y aplicaciones (ver, por ejemplo, las ideas sobre modelos emergentes de computación publicados en [56]).

Como resultado de la polinización referida en el párrafo anterior, una de las fronteras de la investigación en ciencia computacional consiste en (re)construir las nociones de información y computación sobre principios emanados de la física. Entre las ramas de la física que se han empleado en este propósito, la mecánica cuántica ocupa un lugar de privilegio.

La mecánica cuántica es una teoría sobre el comportamiento de la masa y la luz, en particular a escala atómica y subatómica [26, 14]. La historia de la mecánica cuántica (1900 - *circa* 1930) incluye un conjunto de resultados experimentales que pusieron en tela de juicio las ideas que sobre la Naturaleza se tuvieron hasta el principio del siglo XX [22, 32, 64]. Gracias al trabajo de varias generaciones de científicos, la mecánica cuántica es hoy una teoría científica robusta, utilizada diariamente en la labor teórica y experimental.

Computación cuántica es el nombre del multidisciplinario campo de la ciencia que reformula la teoría de la computación y construye nuevo hardware empleando la mecánica cuántica. El propósito de la Computación Cuántica es utilizar las teorías de las que nace para incrementar sustancialmente la capacidad de los ordenadores para procesar información y resolver problemas. Esta nueva capacidad se traduce en aumentar la rapidez con la que se ejecuta un algoritmo o bien en añadir elementos de seguridad a transmisiones de datos. El cómputo cuántico no sólo adopta modelos matemáticos para la creación de algoritmos, también usa las propiedades de la materia con la que se procesa información.

El estudio formal de la computación cuántica comenzó con las preguntas que Richard Feynman planteó sobre dos temas: 1) la posibilidad de simular sistemas cuánticos, y 2) las leyes de la física que caracterizan al proceso de calcular [24, 25]. Tomando como punto de partida el trabajo de Feynman, me permito dividir la historia de la computación cuántica en dos etapas:

1. La primera consiste en el empleo de la estructura matemática de la mecánica cuántica para la creación de nuevos algoritmos, con el propósito de encontrar nuevas y/o más eficientes soluciones a problemas nacidos en la ciencia computacional. Entre los descubrimientos teóricos y conjeturas promisorias de esta época se encuentran: la definición formal de la estructura de una computadora cuántica [17], la definición de las propiedades que debe observar una computadora cuántica de propósito general [20] el algoritmo de Shor [69] (capaz de factorizar números enteros muy largos en tiempo polinomial utilizando una computadora cuántica), el algoritmo de Grover [29] (capaz de encontrar elementos en conjuntos desordenados de forma más eficiente que cualquier algoritmo posible ejecutado en computadoras convencionales) y el algoritmo de Childs *et al* [12], que permite cruzar un grafo creado a partir de árboles balanceados en tiempo probabilístico-polinomial .
   Más aún, en este periodo se construyó una teoría y práctica de la criptografía usando las propiedades de la física cuántica [7] que ha dado lugar a la creación de tecnología cuántica robusta, lista para ser comercializada (`http://www.idquantique.com/`).
   Como transición entre esta primera parte y la historia reciente de la computación cuántica, encontramos el deseo de usar el conocimiento ya generado en áreas de aplicación distintas de la matemática pura y de lo que en inglés se llama *theoretical computer science* . Ejemplos de estas aplicaciones se encuentran, por ejemplo, en la Inteligencia Artificial, el Reconocimiento de Patrones [5, 70, 71, 72, 18, 76, 75, 31, 47, 48, 8] y la Bioinformática [80, 78, 33, 15].
2. La segunda y más reciente parte de la historia de la computación cuántica, además de continuar en la búsqueda y descrubrimiento de nuevos y más eficientes algoritmos provenientes de diversas áreas de la ciencia computacional, comprende el incremento de actividades científicas en dos áreas:

   a) Simulación de procesos naturales en computadoras cuánticas. Los resultados en esta área incluyen la modelación del transporte de energía en procesos fotosintéticos (e.g. [54, 34, 10]), modelación de procesos biológicos varios (e.g. [63, 3, 28, 45]) y de diversos sistemas cuánticos (e.g. [57, 38]).

   b) Simulación de algoritmos cuánticos en plataformas computacionales clásicas (tanto *stand-alone* como distribuidas). La simulación de algoritmos cuánticos empleando plataformas clásicas *stand-alone* y masivas/distribuidas (e.g. supercómputo, grids, nubes, GPUs) es crucial, a efecto de desarrollar nuestra intuición respecto del comportamiento de los dispositivos físicos empleados en la construcción de sistemas de procesamiento de datos basados en la mecánica cuántica.
   Los primeros trabajos en esta área, presentados por Ömer en [61], Bettelli *et al* en [6], Viamontes *et al* en [77] y Bañuls *et al* en [4] entre otros, introdujeron la idea de implementar simuladores de algoritmos cuánticos combinando lenguajes computacionales clásicos (e.g. C++) con la estructura matemática de la mecánica cuántica. Como siguientes pasos, Nyman propuso emplear lenguajes simbólicos para computadoras clásicas a efecto de simular algo-

ritmos cuánticos [60], en tanto que Ömer presentó el uso de de estructuras semánticas abstractas para modelar algoritmos cuánticos [62] y Altenkirch *et al* construyeron un lenguaje de simulación de algoritmos cuánticos basado en programación funcional [1]. A la par de los esfuerzos ya reseñados, la comunidad científica ha desarrollado diversos paquetes de simulación en diferentes arquitecturas y plataformas (el repositorio `http://www.quantiki.org/` tiene una lista bastante completa de paquetes disponibles). Recientemente, la facilidad de acceso a sistemas distribuidos masivos como *grids*, nubes y *GPUs* ha llamado la atención de grupos de investigación interesados en explotar de manera óptima los vastos recursos computacionales paralelos hoy disponibles; algunos resultados en esta área han sido elaborados por De Raedt *et al* [65], Caraiman *et al* [9] y Díaz-Pier *et al* [21], entre otros.

Si el lector está interesado en obtener un panorama más detallado de las ideas fundamentales y avances más importantes en computación cuántico, las siguientes fuentes serán de utilidad: [59, 30, 7, 41, 35, 51, 46, 13].

El resto de este artículo se compone de la siguiente manera: en la sección 2 presento una introducción concisa a la estructura matemática de la mecánica cuántica, a efecto de tener los elementos necesarios para revisar el caso de estudio que presento en la sección 3: caminatas cuánticas.

2. Introducción concisa a la computación cuántica

En esta sección presentamos las definiciones y estructuras matemáticas fundamentales de la mecánica cuántica, listas para ser empleadas en la elaboración de algoritmos cuánticos. Comenzamos estas líneas con algunos preliminares matemáticos esenciales para el resto de este artículo.

Definition 2.1. Espacio de Hilbert (versión computación cuántica).
- Un espacio de Hilbert $\mathcal{H}$ es cualquier espacio vectorial completo con producto interno.
- Dos espacios de Hilbert $\mathcal{H}_1$, $\mathcal{H}_2$ son isomórficos si los espacios vectoriales asociados son isomórficos a su vez y dicho isomorfismo preserva el producto interno.
- En particular, cuando solamente se trata con espacios vectoriales complejos de dimensión finita, un espacio de Hilbert se define como un espacio vectorial con producto interno (el requerimiento de completitud es eliminado). Este es el tipo de espacios de Hilbert con los que generalmente se trabaja en computación cuántica.

Observación. De acuerdo a lo establecido en la Def. (2.1), el espacio de Hilbert con el que trabajaremos en computación cuántica es $\mathbb{C}^n(\mathbb{C})$, donde n es generalmente un múltiplo de 2.

Definition 2.2. Operador lineal. *Sean $\mathbb{V}$ y $\mathbb{W}$ espacios vectoriales . Un* **operador lineal**

$$\hat{T} : \mathbb{V} \to \mathbb{W}$$

es una función que asigna a cada vector $|\psi\rangle \in \mathbb{V}$ *un único vector* $\hat{T}|\psi\rangle \in \mathbb{W}$ *tal que* $\forall\ |\psi\rangle, |\phi\rangle \in \mathbb{V}$ *y* $\forall\ \alpha \in F$ *(el campo usado en la definición de* $\mathbb{V}$ *y* $\mathbb{W}$*), se cumple que:*

$$\hat{T}(|\psi\rangle + |\phi\rangle) = \hat{T}|\psi\rangle + \hat{T}|\phi\rangle$$
$$\hat{T}(\alpha|\psi\rangle) = \alpha\hat{T}|u\rangle$$

Para cada $|\psi\rangle \in \mathbb{V}$, $\hat{T}|\psi\rangle$ *se llama imagen de* $|\psi\rangle$*. La imagen del dominio* $\mathbb{V}$, $\hat{T}(\mathbb{V})$*, es el recorrido de* $\hat{T}$*.*

Definition 2.3. Notación de Dirac. *Sea* $\mathcal{H}$ *un espacio de Hilbert* $\Rightarrow$
- Un vector $\boldsymbol{\psi} \in \mathcal{H}$ *se denota con el símbolo* $|\psi\rangle$ *y recibe el nombre de* **ket**.
- El funcional correspondiente $f_{|\psi\rangle}$ *recibe el nombre de* **bra** *y se denota con el símbolo* $\langle\psi|$.
- $\langle\cdot|$ *puede ser visto como una función (operador) que mapea un estado arbitrario* $|\psi\rangle$ *en el funcional* $\langle\psi|$ *tal que* $f_{|\psi\rangle}(|\phi\rangle) = (|\psi\rangle, |\phi\rangle) = \langle\psi|\phi\rangle$.
- Luego, la notación $\langle\cdot|\cdot\rangle$ *será utilizada, en el resto de este artículo, para referirse al producto interno de dos vectores en un espacio de Hilbert. Más aún, dicha notación es un standard en la formulación y escritura de la mecánica cuántica.*
- Por último, definimos $|\psi\rangle^\dagger \equiv \langle\psi|$.

Definition 2.4. Representación de kets y bras en vectores columna y renglón. *Sea* $\mathcal{H}$ *un espacio de Hilbert,* $dim\mathcal{H} = n$ *y* B *una base de* $\mathcal{H}$ $\Rightarrow$
- Cualquier $|\psi\rangle \in \mathcal{H}$ *se puede representar, usando la base* B*, como un vector columna con* n *componentes, i.e.* $|\psi\rangle = (\psi_1, \psi_2, \ldots, \psi_n)^t$*, donde* $\psi_i \in \mathbb{C}$, $i \in \{1, 2, \ldots, n\}$.
-Más aún, el bra $\langle\psi| \in \mathcal{H}^*$ *se puede representar como un vector renglón también de* n *componentes, i.e.* $\langle\psi| = (\psi_1^*, \psi_2^*, \ldots, \psi_n^*)$*, donde los* ψ^* *son los conjugados de* ψ_i*,* $i \in \{1, 2, \ldots, n\}$
- En consecuencia, si deseamos calcular el producto interno $\langle\psi|\phi\rangle$*, donde* $|\psi\rangle, |\phi\rangle \in \mathcal{H}$*, y para ello queremos usar las representaciones* $\langle\psi| = (\psi_1^*, \psi_2^*, \ldots, \psi_n^*)$ *y* $|\phi\rangle = (\phi_1, \phi_2, \ldots, \phi_n)^t$ *obtenidas mediante el uso de una base* B *de* $\mathcal{H}$ $\Rightarrow$

$$\langle\psi|\phi\rangle = (\psi_1^*\phi_1 + \psi_2^*\phi_2 + \ldots + \psi_n^*\phi_n)^{1/2} = (\sum_{i=1}^{n} \psi_i^*\phi_i)^{1/2}$$

Suponga que $\hat{A} : \mathbb{V} \to \mathbb{V}$ es un operador lineal. Por una parte, sabemos que $\forall\ |\psi\rangle \in \mathbb{V} \Rightarrow \hat{A}|\psi\rangle \in \mathbb{V}$ y que, para cada ket $|\psi\rangle$ en $\mathbb{V}$, existe un único bra $\langle\psi|$ en el espacio dual $\mathbb{V}^*$. Por extensión, si $\hat{A}|\psi\rangle = |\psi'\rangle$ es un elemento de $\mathbb{V}$ entonces el bra $\langle\psi'|$ es un elemento de $\mathbb{V}^*$.

Dados estos elementos, procedemos a definir a $\hat{A}^\dagger$, el operador adjunto de $\hat{A}$, como el operador capaz de generar al bra $\langle\psi'|$ a partir del bra $\langle\psi|$, esto es,

$$\hat{A}|\psi'\rangle = |\psi'\rangle \Leftrightarrow \langle\psi'| = \langle\psi|\hat{A}^\dagger$$

Ahora bien, suponga que $\hat{A}$ es un operador lineal y A una de sus representaciones matriciales. Entonces, la representación matricial de $\hat{A}^\dagger$, basada en la matriz A, es la matriz $(A^*)^t$, esto es, la matriz que representa a $\hat{A}^\dagger$ es la transpuesta de la matriz conjugada de A.

Definition 2.5. Operador hermitiano. *Sea $\mathcal{H}$ un espacio de Hilbert de dimensión finita y $\hat{A} : \mathcal{H} \to \mathcal{H}$ un operador lineal. Si $\hat{A} = \hat{A}^\dagger$ entonces $\hat{A}$ es un* **operador hermitiano**. *La definición se extiende de forma natural a cualquier representación matricial de $\hat{A}$.*

Definition 2.6. Operador unitario. *Sea $\mathcal{H}$ un espacio de Hilbert y $\hat{U} : \mathcal{H} \to \mathcal{H}$ un operador lineal. $\hat{U}$ es un* **operador unitario** *si $\hat{U}\hat{U}^\dagger = \hat{I}$, donde $\hat{I}$ es el operador identidad. Como en el caso de los operadores hermitianos, la definición de matrices unitarias es una extensión natural de la definición de un operador unitario.*

Los operadores unitarios son muy importantes en mecánica cuántica porque preservan el valor del producto interno: sean $|\alpha\rangle = \hat{U}|a\rangle$ y $|\beta\rangle = \hat{U}|b\rangle \Rightarrow \langle\alpha|\beta\rangle = \langle a|\hat{U}^\dagger|\hat{U}|b\rangle = \langle a|\hat{I}|b\rangle = \langle a|b\rangle$.

El **operador de Hadamard** es un operador lineal ampliamente usado en computación cuántica:

$$\hat{H} = \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|) \tag{1}$$

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2}$$

Los conceptos matemáticos revisados hasta el momento serán utilizados para el análisis de sistemas cuánticos *individuales*. En este apartado presentamos al lector un formalismo matemático usado para representar sistemas cuánticos multipartitas .

Definition 2.7. Producto tensorial. *Sean $\mathbb{V}(\mathbf{F})$ y $\mathbb{W}(\mathbf{F})$ espacios vectoriales de dimension m y n respectivamente. Definimos a $\mathbb{X}$ como el producto tensorial de $\mathbb{V}$ y $\mathbb{W}$, i.e. $\mathbb{X} = \mathbb{V} \otimes \mathbb{W}$.*
- Los elementos de $\mathbb{X}$ son combinaciones lineales de vectores $|a\rangle \otimes |b\rangle$, donde $|a\rangle \in \mathbb{V}$ and $|b\rangle \in \mathbb{W}$.
- En particular, si $\{|i\rangle\}$ y $\{|j\rangle\}$ son bases ortonormales de $\mathbb{V}$ y $\mathbb{W}$ entonces $\{|i\rangle \otimes |j\rangle\}$ es una base [1] para $\mathbb{X}$.

Sean ahora $\hat{A}, \hat{B}$ operadores lineales en $\mathbb{V}$ y $\mathbb{W}$ respectivamente $\Rightarrow \forall\ |a\rangle_1, |a\rangle_2 \in \mathbb{V}$, $|b\rangle_1, |b\rangle_2 \in \mathbb{W}$, $\alpha \in F$ se cumple que

1) $\alpha(|a\rangle_1 \otimes |b\rangle_1) = (\alpha|a\rangle_1) \otimes |b\rangle_1 = |a\rangle_1 \otimes (\alpha|b\rangle_1)$
2) $(|a\rangle_1 + |a\rangle_2) \otimes |b\rangle_1) = |a\rangle_1 \otimes |b\rangle_1 + |a\rangle_2 \otimes |b\rangle_1$
3) $|a\rangle_1 \otimes (|b\rangle_1 + |b\rangle_2) = |a\rangle_1 \otimes |b\rangle_1 + |a\rangle_1 \otimes |b\rangle_2$

[1] Un ejemplo concreto: sea $\{|0\rangle, |1\rangle\}$ una base ortonormal de un espacio de Hilbert bidimensional $\mathcal{H}^2$. Entonces, una base para $\mathcal{H}^4 = \mathcal{H}^2 \otimes \mathcal{H}^2$ es $\{|0\rangle\otimes|0\rangle, |0\rangle\otimes|1\rangle, |1\rangle\otimes|0\rangle, |1\rangle\otimes|1\rangle\}$.

4) $\hat{A} \otimes \hat{B}(|a\rangle_1 \otimes |b\rangle_1) = \hat{A}|a\rangle_1 \otimes \hat{B}|b\rangle_1$
5) Sean $|a\rangle_i \in \mathbb{V}$, $|b\rangle_i \in \mathbb{W}$ *y* $\alpha_i \in F \Rightarrow \hat{A} \otimes \hat{B}(\sum_i \alpha_i |a\rangle_i \otimes |b\rangle_i) = \sum_i \alpha_i \hat{A}|a\rangle_i \otimes \hat{B}|b\rangle_i$

El producto tensorial $|a\rangle \otimes |b\rangle$también se puede escribir $|ab\rangle$ o $|a, b\rangle$. Por otra parte, el producto tensorial de $|a\rangle$ consigo mismo n veces $|a\rangle \otimes |a\rangle \otimes \ldots \otimes |a\rangle$ se puede escribir como $|a\rangle^{\otimes n}$.

El producto de Kronecker es una representación matricial del producto tensorial. Sean $A = (a_{ij}), B = (b_{ij})$ dos matrices de orden $m \times n$ y $p \times q$ respectivament. Entonces $A \otimes B$ se calcula de la siguiente manera:

$$A \otimes B = \begin{pmatrix} A_{11}B & A_{12}B & \ldots & A_{1n}B \\ A_{21}B & A_{22}B & \ldots & A_{2n}B \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}B & A_{m2}B & \ldots & A_{mn}B \end{pmatrix}.$$

$A \otimes B$ es de orden $mp \times nq$.

En el resto de esta sección nos concentraremos en el estudio de los postulados de la mecánica cuántica, siguiendo la formulación que de los mismos se hace en [59]. Además, revisaremos varios resultados presentados en las obras [7], [14], [19], [26], [30] y [59]. Entre muchas obras y artículos, se recomienda al lector interesado en profundizar en los temas aquí expuestos que consulte [66, 46].

2.1. Espacio de estados

Este primer postulado consiste en la descripción matemática de sistemas físicos aislados, esto es, aquellos que no están en contacto con ningún otro sistema físico.

Postulado 1. A cada sistema físico aislado asociaremos un espacio de Hilbert $\mathcal{H}$, el cual recibe el nombre de *espacio de estados*.

La descripción total y absoluta de las características que nos interesan del sistema físico en cuestión se encuentra contenida en su *vector de estado*, el cual es un vector unitario $|\psi\rangle \in \mathcal{H}$. La dimensión del espacio de estados $\mathcal{H}$ depende del número de grados de libertad de la propiedad física en consideración.

Este primer postulado tiene una implicación muy importante: una combinación lineal de vectores de estado es también un vector de estado [14]. Este es el **principio de superposición** y es una característica fundamental de la descripción cuántica de sistemas físicos. En particular, observe que cualquier vector de estado $|\psi\rangle$ se puede expresar como una superposición de vectores de estado pertenecientes a una base de $\mathcal{H}$, i.e. $|\psi\rangle = \sum_i c_i |e_i\rangle$, $c_i \in \mathbb{C}$.

El qubit En la teoría de la computación clásica la unidad fundamental de almacenamiento y manipulación de información es el *bit*, cuya estructura matemática es bastante simple: basta con definir dos valores tradicionalmente etiquetados

como $\{0, 1\}$ y con relacionar dichas etiquetas con dos resultados posibles generados a través de una medición clásica. Un ejemplo de este procedimiento es tomar un transistor TTL como el sistema físico a medir y hacer la siguiente asignación:
- Si la diferencia de potencial entre colector y emisor se encuentra en el conjunto $[0, 0{,}5]V$ entonces hemos leído un '0' lógico.
- Si la diferencia de potencial entre colector y emisor se encuentra en el conjunto $[4{,}5, 5]V$ entonces hemos leído un '1' lógico.

Así pues, es evidente que la descripción matemática de un bit clásico es un elemento de un espacio escalar.

En computación cuántica, la unidad básica de almacenamiento, manipulación y medición de información es el *qubit*. Un qubit es un sistema físico cuyo comportamiento se describe a través de las leyes de la mecánica cuántica y que se puede representar matemáticamente como un vector unitario en un espacio de Hilbert bidimensional, esto es

$$|\psi\rangle = \alpha|p\rangle + \beta|q\rangle \tag{3}$$

donde $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$ y $\{|p\rangle, |q\rangle\}$ es una base cualquiera de $\mathcal{H}^2$. Es común que la base de elección sea $\{|0\rangle, |1\rangle\}$, la llamada base computacional o canónica de $\mathcal{H}^2$.

Como se puede observar de la Ec. (3), un qubit $|\psi\rangle$ es una superposición de los estados base $|p\rangle$ y $|q\rangle$, la cual se puede preparar en un número infinito de formas sólo con modificar los valores de los coeficientes $\alpha, \beta \in \mathbb{C}$, siempre sujetos a la restricción de normalización.

La Ec. (3) también se puede escribir de la siguiente manera:

$$|\psi\rangle = e^{i\gamma}(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle) \tag{4}$$

donde $\gamma, \theta, \varphi \in \mathbb{R}$. Puesto que $e^{i\gamma}$ no tiene efectos experimentales [59], podemos prescindir de este factor. En consecuencia,

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle \tag{5}$$

Los números θ y φ definen un punto en la esfera unitaria conocida como la **esfera de Bloch** (Fig. (1)).

2.2. Evolución de un sistema cuántico aislado

En este apartado estudiaremos la evolución de un sistema cuántico, esto es, el formalismo matemático que describe el comportamiento temporal de un sistema físico aislado, de acuerdo a las leyes de la mecánica cuántica.

Postulado 2 (versión operador unitario).

Sea $|\Psi\rangle$ el vector de estado de un sistema cuántico aislado. La evolución de $|\Psi\rangle$ se describe mediante un operador unitario $\hat{U}$ (Def. (2.6)), también conocido como operador de evolución . Luego, el estado del sistema en el tiempo t_2 dado el vector de estado del mismo sistema en el tiempo t_1 es:

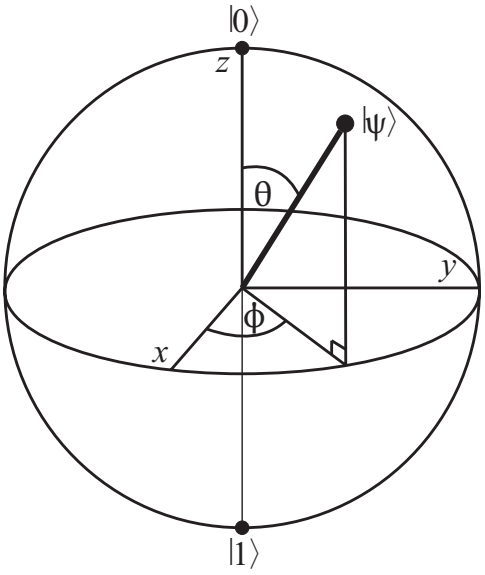


Figura 1. Un qubit como elemento de la esfera de Bloch $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$.

$$|\Psi(t_2)\rangle = \hat{U}|\Psi(t_1)\rangle. \tag{6}$$

El postulado 2 sólo describe las características matemáticas que debe cumplir un operador de evolución cualquiera. Las propiedades particulares que debe tener $\hat{U}$ a fin de reflejar la naturaleza de la evolución de cierto sistema físico dependen de este mismo sistema.

El postulado 2 también se puede escribir en una forma más tradicional, usando la famosa ecuación de Schrödinger .

Postulado 2 (versión operador hermitiano). La evolución de un sistema cuántico aislado está dada por la ecuación de Schrödinger:

$$i\hbar\frac{d|\psi\rangle}{dt} = \hat{\mathbf{H}}|\psi\rangle \tag{7}$$

donde $\hbar$ es la constante de Planck y $\hat{\mathbf{H}}$ es un operador hermitiano (Eq. (2.5)) conocido como el *hamiltoniano* del sistema.

Cada sistema cuántico tiene un hamiltoniano asociado, el cual debe ser calculado. En general, la construcción de hamiltonianos es una tarea difícil (en cierto sentido, tal y como es problemático construir algoritmos para resolver problemas difíciles).

Nota aclaratoria. Tenga en mente que, a pesar de lo similar de la notación, $\hat{\mathbf{H}}$ y $\hat{H}$ representan dos cosas distintas: el primero es el hamiltoniano al que se hace referencia en el postulado 2, en tanto que el último es el operador hadamard.

Veamos el efecto del operador Hadamard (Eq. (1)), usado como operador de evolución, sobre un qubit.

$$\hat{H}|0\rangle = \frac{1}{\sqrt{2}}[|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|]|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$\hat{H}|1\rangle = \frac{1}{\sqrt{2}}[|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|]|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

2.3. Medición de un sistema cuántico

En la teoría de la mecánica cuántica, la medición de propiedades de sistemas físicos es un proceso alejado de la intuición debido a las siguientes razones:
1. La medición en mecánica cuántica es intrínsicamente probabilística. Esto significa que, sin importar el detalle y control que tengamos sobre un experimento, la generación de los resultados obtenidos en la medición de una propiedad física obedece una función de distribución (o función de densidad, según sea el caso).
2. Además, al momento de llevar a cabo una medición, el estado del sistema físico en cuestión se altera, de forma inevitable, debido a la interacción que dicho sistema tiene con el aparato de medición. Esto significa que, en general, el estado cuántico que describe al sistema *antes* de la medición es distinto del estado que describe a este mismo sistema *después* de ser medido.

Las siguientes líneas contienen la versión del postulado de medición más frecuentemente usada en computación cuántica.

Postulado 3. Medición proyectiva. Una medición proyectiva es descrita por un *observable* $\hat{M}$, el cual es un operador hermitiano definido en el espacio de estados que se desea observar. El observable $\hat{M}$ se puede escribir, gracias al teorema de la descomposición espectral, de la siguiente manera:

$$\hat{M} = \sum_i r_i \hat{P}_{r_i}$$

donde $\hat{P}_{r_i}$ es el proyector al eigenespacio $E(r_i)$ definido por el eigenvalor r_i. Los resultados posibles de la medición corresponden a los eigenvalores r_i del observable.

Este postulado provee de medios para cuantificar la función de distribución que determina las frecuencias relativas correspondientes a las funciones de distribución de resultados.

Sea $|\psi\rangle$ el vector de estado de un sistema cuántico *inmediatamente antes de la medición*. Entonces, la probabilidad de obtener el resultado r_i se calcula usando la siguiente expresión:

$$p(r_i) = \langle\psi|\hat{P}_{r_i}|\psi\rangle \tag{8}$$

Y el estado de post-medición asociado al resultado r_i es:

$$|\psi\rangle_{pm} = \frac{\hat{P}_{r_i}|\psi\rangle}{\sqrt{p(r_i)}} \tag{9}$$

Veamos un ejemplo. Suponga que tiene en su poder un fotón polarizado con orientaciones de polarización vertical y horizontal. Simbolizamos a la polarización horizontal con el vector $|0\rangle$ y a la polarización vertical con $|1\rangle$. Luego, la polarización inicial de nuestro fotón se puede describir con la expresión

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

donde $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$ y $\{|0\rangle, |1\rangle\}$ conforman la base computacional de $\mathcal{H}^2$.

Ahora construyamos dos operadores de proyección $\hat{P}_{a_0} = |0\rangle\langle 0|$ y $\hat{P}_{a_1} = |1\rangle\langle 1|$, los cuales corresponden a los resultados a_0, a_1. Entonces, el *observable* utilizado en este experimento es

$$\hat{M} = a_0|0\rangle\langle 0| + a_1|1\rangle\langle 1|$$

Con esta información y de acuerdo al postulado 3, podemos decir lo siguiente:

1) Hay sólo dos resultados posibles en la medición de la polarización de nuestro fotón: a_0 y a_1.

2) La probabilidad de obtener el resultado a_0 es, de acuerdo a la Ec. (8):

$$p(a_0) = \langle\psi|\hat{P}_{a_0}|\psi\rangle = (\langle 1|\beta^* + \langle 0|\alpha^*)\hat{P}_{a_0}(\alpha|0\rangle + \beta|1\rangle) = |\alpha|^2$$

3) Si la medición efectivamente arrojase el resultado $a_0 \Rightarrow$ el estado de post-medición sería, de acuerdo a la Ec. (9):

$$\frac{\hat{P}_{a_0}|\psi\rangle}{\sqrt{p(a_0)}} = \frac{|0\rangle\langle 0|(\alpha|0\rangle + \beta|1\rangle)}{\sqrt{|\alpha|^2}} = |0\rangle$$

4) De forma análoga, la probabilidad de obtener el resultado a_1 es, de acuerdo a la Ec. (8):

$$p(a_1) = \langle\psi|\hat{P}_{a_1}|\psi\rangle = (\langle 1|\beta^* + \langle 0|\alpha^*)\hat{P}_{a_1}(\alpha|0\rangle + \beta|1\rangle) = |\beta|^2$$

5) Si la medición efectivamente arrojase el resultado $a_1 \Rightarrow$ el estado de post-medición sería dado por la Ec. (9), esto es:

$$\frac{\hat{P}_{a_1}|\psi\rangle}{\sqrt{p(a_1)}} = \frac{|1\rangle\langle 1|(\alpha|0\rangle + \beta|1\rangle)}{\sqrt{|\beta|^2}} = |1\rangle$$

2.4. Sistemas cuánticos multipartitas

Terminamos esta sección con la descripción matemática de un sistema cuántico multipartita , i.e. un sistema que está compuesto por varios sistemas cuánticos (e.g. un sistema cuántico de cinco fotones).

Postulado 4. El espacio de estados de un sistema cuántico compuesto es el producto tensorial de los espacios de estados constituyentes.

- Si tenemos n sisteams cuánticos expresados como vectores de estado $|\psi\rangle_1, |\psi\rangle_2, \ldots, |\psi\rangle_n$ entonces el estado del sistema total está dado por $|\psi\rangle_T = |\psi\rangle_1 \otimes |\psi\rangle_2 \otimes \ldots \otimes |\psi\rangle_n$.

Como un ejemplo de las operaciones que haremos en la siguiente sección, muestro los detalles de aplicar un operador de evolución a un sistema cuántico compuesto : sea $\hat{H}^{\otimes 2}$ el producto tensorial del operador Hadamard (Eq. (1)) consigo mismo y sea $|\psi\rangle = |00\rangle$. Luego,

$$\begin{aligned}
&\hat{H}^{\otimes 2}\,|\psi\rangle = \\
&\frac{1}{2}(|00\rangle\langle 00| + |01\rangle\langle 00| + |10\rangle\langle 00| + |11\rangle\langle 00| + |00\rangle\langle 01| - |01\rangle\langle 01| + |10\rangle\langle 01| - |11\rangle\langle 01| \\
&+|00\rangle\langle 10| + |01\rangle\langle 10| - |10\rangle\langle 10| - |11\rangle\langle 10| + |00\rangle\langle 11| - |01\rangle\langle 11| - |10\rangle\langle 11| + |11\rangle\langle 11|)|00\rangle \\
&= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \quad (10)
\end{aligned}$$

3. Caso de estudio: caminatas cuánticas

Un procedimiento que utiliza mecánica cuántica para hallar una solución se llama algoritmo cuántico, en tanto que un algoritmo convencional (también llamado clásico) es un procedimiento programado en una computadora como las que usted y yo ocupamos a diario. Crear un algoritmo cuántico no es tarea fácil, pues dicho algoritmo debe resolver el problema para el que fue diseñado y, además, ser más rápido que cualquier algoritmo convencional pensado para resolver el mismo problema. Entre las técnicas utilizadas para construir algoritmos cuánticos están la Transformada Cuántica de Fourier y las Caminatas Cuánticas. El objetivo principal de esta sección es presentar a usted los elementos fundamentales de las caminatas cuánticas y su empleo en el desarrollo de algoritmos.

Comenzaremos nuestra exposición repasando de manera sucinta las tres componentes fundamentales de la teoría de la computación, además de un área de la algorítmica esencial para nuestro análisis: los algoritmos estocásticos, esto es, los procedimientos que emplean distribuciones de probabilidad en su ejecución. Esta información servirá para presentar el concepto de caminata aleatoria y luego extenderlo al mundo de la mecánica cuántica, y así plantear los modelos discreto y continuo de las caminatas cuánticas. La última parte de esta sección consiste en la presentación de algunos algoritmos basados en caminatas cuánticas.

3.1. Modelos computacionales determinísticos y no-determinísticos

La teoría de la computación se divide en tres áreas de estudio, a saber:

- Teoría de autómatas , cuyo objetivo es la creación de modelos matemáticos de computadoras. Un ejemplo de estos modelos es la máquina de Turing.
- Teoría de la computabilidad . Dado un problema P y el modelo matemático M de una computadora, esta disciplina estudia si dicho problema P puede ser resuelto, en principio, con el modelo M, siendo válido suponer que se cuenta con una cantidad ilimitada de recursos (por ejemplo, tiempo o memoria).
- Teoría de la complejidad . Suponga que tenemos un modelo computacional M y un problema P que se puede resolver con un algoritmo A implantado en el modelo M. La pregunta que debe responder esta rama de la computación es: ¿cuántos recursos hay que invertir para ejecutar A en M? En otras palabras, la teoría de la complejidad cuantifica el costo (tiempo o energía, por ejemplo) de ejecutar un algoritmo.

Existen varias formas de ejecutar algoritmos en modelos computacionales. Uno de estos métodos, llamado *cmputación determinístico*, consiste en crear algoritmos que obedezcan la siguiente regla: para cualquier paso s_i de un algoritmo A, siempre es posible determinar, con toda certeza, el paso s_{i+1}. En otras palabras, un algoritmo determinístico tiene un comportamiento predecible y exacto (visto desde las matemáticas, la relación entre un nodo y sus hojas es siempre una función, pues sólo hay una hoja por nodo).

Otro método, llamado *computación no-determinístico*, consiste en obedecer la siguiente regla: para un paso arbitrario s_i del algoritmo A, existen varios pasos siguientes s_{i+1}^j, donde $j \in \{1, 2, \ldots, m\}$ es un índice que corre sobre el conjunto de los m pasos que siguen a s_i. En este caso, el nodo tiene una relación no funcional con sus hojas, pues *en general* hay más de una hoja por nodo.

Estos tipos de cómputo se pueden visualizar como árboles al estilo de la Fig. (2), en la que el método determinístico se representa con un árbol con una sola derivación, en tanto que los procedimientos no-determinísticos permiten que, de un nodo dado, aparezcan varias ramificaciones. Cada ramificación representa un proceso computacional que se ejecuta al mismo tiempo que todos los demás.

De los dos métodos presentados, el cómputo determinístico se ajusta perfectamente a la noción de disponibilidad de recursos, en tanto que en este mismo rubro, el cómputo no-determinístico se antoja irreal. Luego, ¿por qué es este método un tema de estudio en la ciencia computacional? La respuesta es que el cómputo no-determinístico no escatima la cantidad de recursos disponibles pues su objetivo es averiguar si es posible, al menos en principio, ejecutar un algoritmo dado, aunque ello implique suponer el uso de una cantidad infinita de recursos. No es lo mismo no poder resolver un problema que sólo tener que invertir muchos recursos en lograrlo.

Entre los diversos modelos computacionales sobresalen las máquinas de Turing, consideradas como el modelo computacional más poderoso creado a la fecha por las siguientes razones:

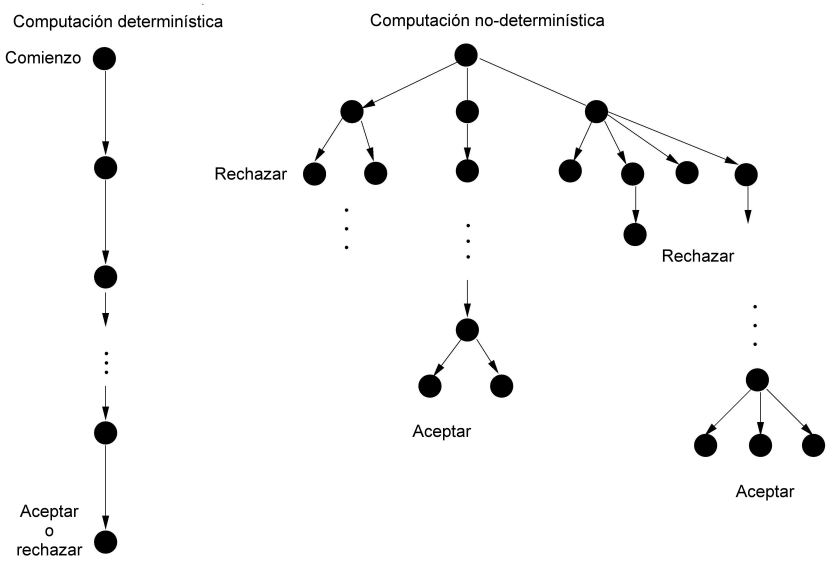


Figura 2. Computación determinístico y no-determinístico

- Cualquier problema resuelto por un modelo computacional distinto de la máquina de Turing (como los autómatas finitos) puede ser también resuelto por una máquina de Turing.
- En consecuencia, cualquier problema resuelto con una computadora construida al día de hoy también puede ser resuelto por una máquina de Turing.

Existen versiones determinística y no-determinística de las máquinas de Turing. Estas versiones son equivalentes en su capacidad para ejecutar algoritmos, pero difieren en el tiempo que tardan en hacerlo:

- Aquellos algoritmos que al ejecutarse en una máquina **determinística** de Turing efectúen una cantidad de pasos acotada superiormente por *una función polinomial en el número de datos de entrada*, i.e. $f(n) = \sum_i \alpha_i n^i$, donde n es el número de datos de entrada del algoritmo, reciben el nombre de algoritmos **P**.
- Los algoritmos que al ejecutarse en una máquina **no-determinística** de Turing consumen una cantidad de pasos acotada por una *función polinomial* $g(n)$ *en el número de datos de entrada*, i.e. $g(n) = \sum_k \beta_k n^k$, donde n es el número de datos de entrada del algoritmo, reciben el nombre de problemas **NP**.
- Por último, un algoritmo L es **NP-completo** si y sólo si L es **NP** y se cumple que, para todo problema L_i en **NP**, es posible transformar al algoritmo L_i en el algoritmo L usando solamente una cantidad polinomial de pasos.

Los algoritmos **P** son vistos con muy buenos ojos por la comunidad de científicos computacionales, pues utilizan una cantidad aceptable de tiempo en su ejecución. Para comprender mejor este concepto, analicemos el caso contrario, el de los algoritmos: **NP**

Dada la disparidad de recursos disponibles entre los modelos determinístico y no-determinístico, la ejecución de un algoritmo **NP** en una máquina determinística de Turing requiere una cantidad de recursos que crece exponencialmente (o factorialmente) en el número de datos de entrada (la excepción a esta regla es que se descubra que el problema asociado al algoritmo **NP** encuentra también solución con un algoritmo P. En este caso, el problema deja de pertenecer a la esfera de los **NP** y se vuelve un problema **P**). La explicación de este fenómeno radica en el hecho de que, para un problema **NP** y un **NP-completo**, el espacio de soluciones posibles es muy grande, y explorarlo *exhaustivamente* requiere muchos recursos.

3.2. Algoritmos estocásticos

Se han propuesto diversos caminos para hacer del cómputo no-determinístico algo más cercano a lo que es posible llevar a cabo en una computadora real. En uno de ellos, la computadora escoge *aleatoriamente* (i.e. usando una distribución de probabilidad) una de las ramas del árbol no-determinístico y la ejecuta. Esto es, si el algoritmo está en el paso s_i, entonces el siguiente y único paso s_{i+1} se escoge (usando una distribución de probabilidad) del conjunto de pasos $\{s_{i+1}^j | j \in \{1, 2, \ldots, m\}\}$. Este proceso se conoce con el nombre de cómputo probabilístico y, aunque no es precisamente equivalente al cómputo no-determinístico, su gran ventaja es que es posible implantarlo en una computadora convencional (el único problema práctico es que no es posible generar números totalmente aleatorios en una computadora convencional, mas los números pseudo-aleatorios son, en general, suficientemente buenos para muchas aplicaciones).

Estamos ya en posibilidad de definir un concepto crucial: un **algoritmo estocástico** es un algoritmo cuya sucesión de pasos (i.e. cuya evolución en el tiempo) se produce usando una distribución de probabilidad. Dicho de otra forma, un algoritmo estocástico es un procedimiento ejecutado en una máquina capaz de hacer cómputo probabilístico.

Los algoritmos estocásticos juegan un papel central en el estudio de los problemas NP-completos, pues gracias a ellos es posible encontrar soluciones, para dichos problemas, que consumen menos pasos que los que requeriría un algoritmo de fuerza bruta , i.e. un algoritmo que explorase, exhaustivamente, el espacio completo de posibles soluciones.

Un ejemplo de los problemas beneficiados por la existencia de algoritmos estocásticos es el 3-SAT, definido de la siguiente manera:

Problema 3-SAT. Sea $S = \{x_1, x_2, \ldots, x_n, \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$ un conjunto de variables booleanas $E = \{x_i\}$ y sus negaciones $\bar{E} = \{\bar{x}_i\}$. Construyamos ahora la proposición lógica P, definida por $P = \bigwedge_i [(\bigvee_{j=1}^{3} a_j)] = \bigwedge_i C_i$, donde $a_j \in S$, i.e. P es una conjunción de cláusulas C_i definida sobre el conjunto S, y donde cada cláusula se forma con la disyunción de tres variables booleanas.

La proposición P es una instancia del problema 3SAT y *satisfacer* la instancia P del 3SAT significa encontrar valores concretos para cada una de las variables booleanas (esto es, una cadena binaria) que componen a P, de tal suerte que al sustituir dichos valores obtengamos $P = 1$. Un ejemplo concreto del nivel de dificultad que acompaña satissfacer una instancia 3SAT se da a continuación.

Sea $E = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ un conjunto de variables binarias y considere la siguiente instancia P:

$$\begin{aligned} P = &(\bar{x_1} \vee \bar{x_4} \vee \bar{x_5}) \wedge (\bar{x_2} \vee \bar{x_3} \vee \bar{x_4}) \wedge (x_1 \vee x_2 \vee \bar{x_5}) \wedge (x_3 \vee x_4 \vee x_5) \wedge \\ &(x_4 \vee x_5 \vee \bar{x_6}) \wedge (\bar{x_1} \vee \bar{x_3} \vee \bar{x_5}) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_5}) \wedge (x_2 \vee \bar{x_3} \vee \bar{x_6}) \wedge \\ &(\bar{x_1} \vee \bar{x_2} \vee \bar{x_6}) \wedge (x_3 \vee \bar{x_5} \vee \bar{x_6}) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (x_2 \vee x_3 \vee \bar{x_4}) \wedge \\ &(x_2 \vee x_5 \vee \bar{x_6}) \wedge (x_2 \vee \bar{x_3} \vee \bar{x_5}) \wedge (\bar{x_2} \vee \bar{x_3} \vee \bar{x_4}) \wedge (x_2 \vee x_3 \vee x_6) \wedge \\ &(\bar{x_1} \vee \bar{x_2} \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_4} \vee \bar{x_5}) \wedge (\bar{x_3} \vee \bar{x_4} \vee \bar{x_6}) \wedge (\bar{x_4} \vee \bar{x_5} \vee x_6) \wedge \\ &(\bar{x_2} \vee x_3 \vee \bar{x_6}) \wedge (x_2 \vee x_5 \vee x_6) \wedge (x_3 \vee x_5 \vee \bar{x_6}) \wedge (\bar{x_1} \vee x_3 \vee \bar{x_6}) \wedge \\ &(x_3 \vee \bar{x_5} \vee x_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_1 \vee x_2 \vee \bar{x_3}) \end{aligned}$$

Este ejemplo sugiere que, aún en el caso de instancias creadas con un número limitado de variables booleanas, satisfacerlas puede convertirse en una tarea difícil (en este caso, P tiene una sola solución: $x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1, x_5 = 0, x_6 = 0$).

El mejor algoritmo conocido a la fecha para la solución de 3-SAT fue propuesto por U. Schöning en 1999 [67], el cual se construyó empleando un proceso estocástico (i.e. cuya evolución es función de una distribución de probabilidad) conocido bajo el nombre de caminata aleatoria. En [36] se presentó una mejora de [67], pero la idea fundamental es la misma: usar una caminata aleatoria para construir el algoritmo. Para estar en condiciones de presentar las ideas fundamentales de las caminatas cuánticas, permítame mostrar en detalle las ideas fundamentales de una caminata aleatoria.

3.3. Caminatas Aleatorias

El modelo básico de las caminatas aleatorias es el movimiento de una partícula (llamado caminante) sobre puntos discretos distribuidos en una línea sin restricciones. El sentido del movimiento del caminante (izquierda o derecha) depende de un sistema bivaluado (como una moneda) cuyos valores, para cada paso, dependen de la probabilidad.

Para ejemplificar jocosamente el concepto anterior, suponga que tenemos a la rana Froggy y una moneda, como se muestra en la Fig. (3) Froggy se desplazará sobre una línea y su movimiento dependerá del resultado de tirar volados (Froggy es una rana obediente). Si el resultado del volado es 'sol' entonces Froggy da un brinco a la derecha (por ejemplo, si la rana está en '0' antes del volado, entonces se mueve al sitio marcado con '1') y si el resultado es 'águila' entonces Froggy se mueve a la izquierda (del sitio '0' al sitio '-1'). Después de muchos volados (digamos, un millón), uno puede hacer varias preguntas interesantes, por ejemplo: ¿cuál es la probabilidad de que Froggy esté en el lugar '100'?

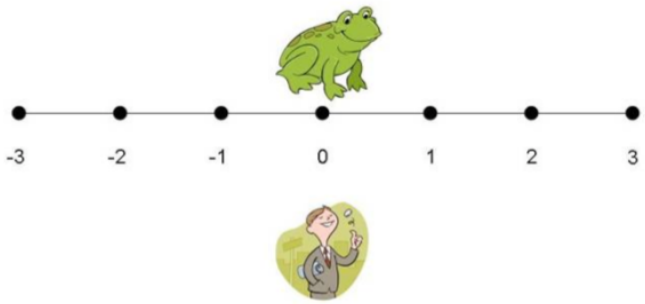


Figura 3. Cada paso de la caminata aleatoria consiste en que Froggy se mueva a la izquierda o derecha. El sentido de este movimiento depende del resultado del volado.

La ecuación que nos permite calcular la probabilidad de encontrar a nuestra rana en el lugar k, suponiendo que el movimiento comenzó en la posición 0 y que Froggy se ha movido n veces (esto es, que se han tirado n volados) está dada por la distribución binomial [Fig. (4)]

$$P_{ok}^{n} = \binom{n}{\frac{1}{2}(k+n)} p^{\frac{1}{2}(k+n)} q^{\frac{1}{2}(n-k)}$$

Dos propiedades importantes de la caminata aleatoria sobre una línea son: 1) la varianza de la distribución binomial es proporcional al número de pasos ejecutados, i.e. $\sigma^2 = O(n)$; 2) *la forma* de la distribución binomial no depende del punto de partida. Lo que sucede al cambiar el punto de partida (por ejemplo, poner a Froggy en 10 en vez de 0), es que la gráfica se desplazará a la izquierda o derecha, pero la forma será la misma. Esta invariancia de la forma de la distribución respecto del punto de partida es una característica fundamental de las cadenas de Markov, de las cuales las caminatas aleatorias son un caso especial.

Naturalmente, las caminatas aleatorias se pueden extender de varias maneras. Por ejemplo, es posible definir caminatas sobre líneas con barreras absorbentes o reflejantes, sobre grafos y, además, con movimientos hechos en tiempos infinitesimales ($t \geq 0$), en vez de tiempos discretos. Para el lector interesado en la formulación de procesos estocásticos en grafos y su aplicación en algoritmos, se recomienda consultar [50, 55, 79, 73] y demás fuentes citadas en el capítulo III de [73].

El éxito de varios algoritmos estocásticos en la solución de problemas NP, en particular algoritmos que emplean caminatas aleatorias, ha sido una importante fuente de inspiración para desarrollar nuevos modelos de caminatas, ahora bajo las leyes de la mecánica cuántica. En la siguiente sección exploraremos las definiciones y características principales de las caminatas cuánticas.

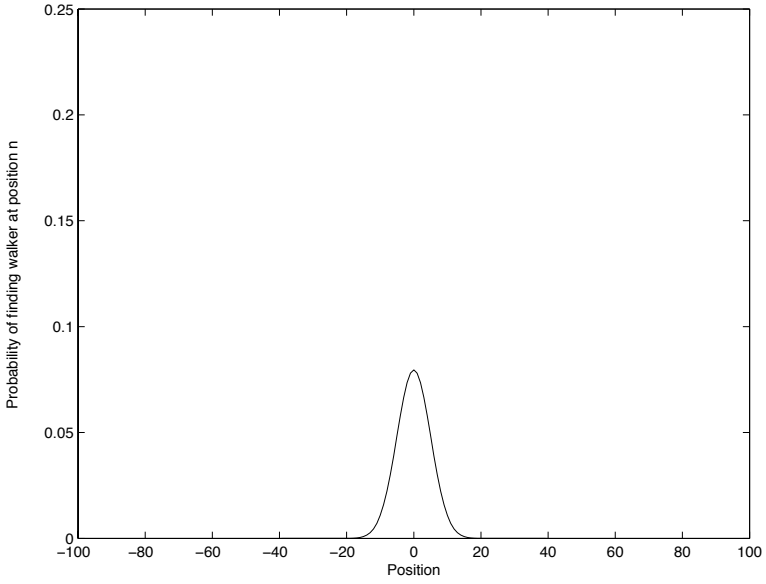


Figura 4. Distribución binomial

3.4. Caminatas Cuánticas

La existencia de caminatas aleatorias discretas (cadenas de Markov) y continuas (procesos de Markov) ha llevado también a sugerir dos tipos de caminatas cuánticas: discretas y continuas.

Antes de entrar en materia, subrayo que, a pesar de su aplicación común, existe una diferencia primera y fundamental entre las caminatas aleatorias y las cuánticas: las caminatas aleatorias son entes matemáticos que se utilizan para modelar fenómenos físicos, en tanto que las caminatas cuánticas son representaciones matemáticas de procesos físicos. Este origen físico de las caminatas cuánticas permite pensar en ellas no sólo como herramientas para la construcción de algoritmos, sino también como elementos de prueba para determinar si una computadora tiene, en efecto, propiedades cuánticas. Más aún, se ha demostrado ya que las caminatas cuánticas, tanto discretas como continuas, conforman un modelo universal de computación cuántica [11, 49]

Caminatas cuánticas discretas En este modelo participan dos elementos: el caminante y la moneda (la misma idea que con la caminata aleatoria). Ambos elementos son sistemas físicos cuyo comportamiento se modela y cuantifica mediante los principios y leyes de la mecánica cuántica [52, 58].

El modelo más sencillo de este tipo de caminata se ejecuta sobre un espacio discreto unidimensional (esto es, una recta con nodos). La evolución de esta caminata se lleva a cabo aplicando un operador de evolución consistente en dos

operaciones cuánticas (dos operadores unitarios): la primer operación hace que la moneda entre en un estado cuántico que asemeja un volado, y la segunda operación hace que los componentes cuánticos de la moneda interactúen con el caminante, de tal suerte que la probabilidad de encontrar al caminante en distintos puntos de la línea sea una función del tiempo. La aplicación de las dos operaciones cuánticas es equivalente a un paso algorítmico, una operación elemental.

La ecuación que define una caminata cuántica discreta es

$$|\psi\rangle_n = (\hat{U})^n |\psi\rangle_o \tag{11}$$

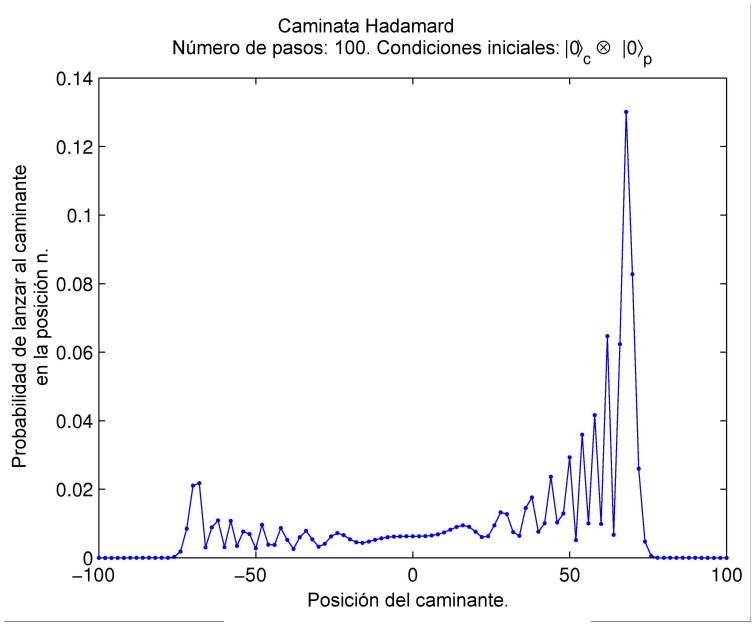


Figura 5. Distribución de probabilidad (posición *vs* probabilidad de encontrar al caminante en dicha posición) generada con $|\psi\rangle_0 = |0\rangle_{\text{moneda}} \otimes |0\rangle_{\text{caminante}}$. El operador de evolución de esta caminata cuántica es $[[\frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|)] \otimes \hat{1}]_{\text{moneda}}[|0\rangle_{\text{moneda}}\langle 0| \otimes \sum_i |i+1\rangle_{\text{caminante}}\langle i| + |1\rangle_{\text{moneda}}\langle 1| \otimes \sum_i |i-1\rangle_{\text{caminante}}\langle i|]$

Donde $|\psi\rangle_o$ es el símbolo que representa el estado inicial total de la moneda y el caminante, $\hat{U}^n$ representa n aplicaciones del operador de evolución $\hat{U}$ (i.e. de las dos operaciones cuánticas: volado más desplazamiento) y $|\psi\rangle_n$ es el símbolo que representa el estado de la caminata cuántica (moneda más caminante) después de n pasos. Un ejemplo del operador $\hat{U}$ es: $\hat{U} = [[\frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|)] \otimes \hat{1}]_{\text{moneda}}[|0\rangle_{\text{moneda}}\langle 0| \otimes \sum_i |i+1\rangle_{\text{caminante}}\langle i| + |1\rangle_{\text{moneda}}\langle 1| \otimes \sum_i |i-1\rangle_{\text{caminante}}\langle i|]$, donde:

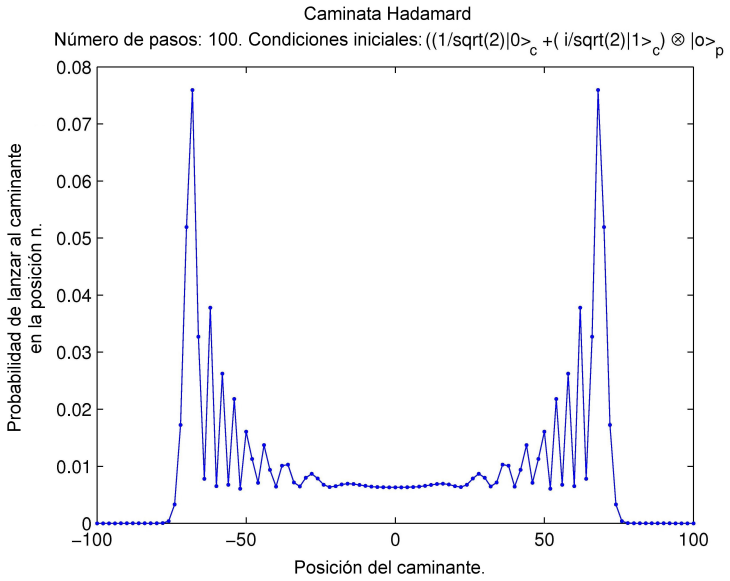


Figura 6. Distribución de probabilidad (posición *vs* probabilidad de encontrar al caminante en dicha posición) generada con $|\psi\rangle_0 = \frac{1}{\sqrt{2}}(|0\rangle_{\text{moneda}} + i|1\rangle_{\text{moneda}}) \otimes |0\rangle_{\text{caminante}}$. El operador de evolución de esta caminata cuántica es el mismo que el de la Fig. (5)

a) $[[\frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|)] \otimes \hat{1}]_{\text{moneda}}$ es el operador Hadamard que se aplicará a la moneda cuántica acompañado del operador identidad (para dejar intacto al caminante durante el volado), y

b) $[|0\rangle_{\text{moneda}}\langle 0| \otimes \sum_i |i+1\rangle_{\text{caminante}}\langle i| + |1\rangle_{\text{moneda}}\langle 1| \otimes \sum_i |i-1\rangle_{\text{caminante}}\langle i|]$ es el operador de desplazamiento, cuyo funcionamiento es: para las componentes de la moneda en superposición que están en $|0\rangle$ el caminante se moverá un paso a la izquierda, en tanto que las componentes de la moneda en superposición que están en $|1\rangle$ el caminante se moverá un paso a la derecha.

La ejecución de varias caminatas cuánticas discretas con estados iniciales idénticos y operaciones cuánticas iguales permite generar distribuciones de probabilidad como las mostradas en las Figs. (5) y (6). Estas gráficas ejemplifican algunas propiedades importantes de las caminatas cuánticas, a saber:

- Las caminatas cuánticas discretas tienen una varianza que crece proporcionalmente al cuadrado del número de pasos, i.e. $\sigma^2_{cc}(n) = O(n^2)$ [52, 58, 39, 74, 44]. Este hecho es importante por dos motivos: 1) la varianza de una caminata aleatoria es proporcional sólo al número de pasos ejecutados (i.e. $\sigma^2_{cc}(n) > \sigma^2_{ca}(n)$), y 2) esta diferencia entre las varianzas clásica y cuántica puede ser utilizada para aumentar la velocidad de ejecución de un algoritmo

basado en caminatas cuánticas, respecto del correspondiente algoritmo clásico diseñado con una caminata aleatoria. Al respecto, hemos de subrayar que la varianza proporcional al cuadrado del número de pasos puede ser también propiedad de sistemas clásicos (e.g. interferencia de campos electromagéticos clásicos [37], [42]), and [43], de ahí la necesidad de profundizar en las ventajas que el uso de sistemas cuánticos traería, unívocamente, a la ciencia computacional.

- La forma de la distribución de probabilidad generada con una caminata cuántica depende del estado inicial. Este hecho es importante pues el estado inicial del caminante y la moneda puede ser utilizado como un parámetro computacional. De hecho, la interacción de la moneda con el medio ambiente puede generar la distribución 'top-hat' [40], una gráfica cuasi uniforme muy agradable a la vista de un científico computacional.

Caminatas cuánticas continuas Este tipo de caminatas requiere un solo sistema físico, el caminante. La partícula que hace las veces de la moneda no es necesaria en este esquema.

En este modelo se aplica un hamiltoniano en cualquier momento, i.e. el tiempo de ejecución de la caminata es una variable real positiva, no más una variable discreta. Comencemos con algunas definiciones preliminares: en [12], Childs *et al* presentaron la siguiente formulación de una caminata aleatoria continua:

Definition 3.1. *Sea $G = (V, E)$ un grafo con $|V| = n$. Luego, una caminata aleatoria continua en G se puede describir empleando una matriz generadora M de orden n definida por:*

$$M_{ab} = \begin{cases} -\gamma, & a \neq b, (a,b) \in G \\ 0, & a \neq b, (a,b) \notin G \\ k\gamma, & a = b \text{ y } k \text{ es la valencia del vértice } a \end{cases} \tag{12}$$

Siguiendo a [12] y [23], la probabilidad de estar en el vértice a en el tiempo t es dada por:

$$\frac{\mathrm{d}p_a(t)}{\mathrm{d}t} = -\sum_b M_{ab} p_b(t) \tag{13}$$

Ahora, definamos un hamiltoniano cuya estructura está basada en la ecuación (12) [12, 23].

Definition 3.2. *Sea $\hat{H}$ un hamiltoniano cuyas entradas matriciales son dadas por:*

$$\langle a|H|b\rangle = \begin{cases} -\gamma, & a \neq b, (a,b) \in G \\ k\gamma, & a = b \text{ donde la valencia de } a \text{ es } k \\ 0, & \text{en cualquier otro caso} \end{cases} \tag{14}$$

Podemos ahora emplear el hamiltoniano $\hat{H}$ dado por la ecuación (14), definido en un espacio de Hilbert $\mathcal{H}$ sobre la base vectorial $\{|1\rangle, |2\rangle, \ldots, |n\rangle\}$, para construir la siguiente ecuación de Schrödinger:

$$i\frac{\mathrm{d}\langle a|\psi(t)\rangle}{\mathrm{d}t} = -\sum_b \langle a|H|b\rangle\langle b|\psi(t)\rangle \tag{15}$$

Finalmente, tomando las ecuaciones (14) y (15), el operador unitario $\hat{U}$

$$\hat{U} = exp(-i\hat{H}t) \tag{16}$$

define una **caminata cuántica continua** en el grafo G. Esta definición la emplearemos en la descripción del último problema de la sección 3.4.

Aleatoriedad de las caminatas cuánticas Posiblemente la primera pregunta que surge en torno a las caminatas cuánticas es: ¿por qué se ha eliminado el adjetivo *aleatorias*? La razón es que la evolución de un sistema cuántico cerrado (esto es, que no interactúa con el medio ambiente) es un proceso *determinístico*. Lo probabilístico llega cuando se intenta averiguar el lugar en el que se encuentra el caminante (o la cara de la moneda), pues en la lógica de la mecánica cuántica, conocer la posición del caminante es equivalente a *medir* una propiedad de la partícula que hace las veces del caminante, y la medición en mecánica cuántica es un proceso inherentemente probabilístico. El mismo argumento se aplica a la moneda.

¿Qué significa que la medición en mecánica cuántica sea un proceso inherentemente probabilístico? Que los resultados posibles de una medición aparecerán (serán revelados al científico) de acuerdo a una distribución de probabilidad, sin importar lo cuidadoso que sea el investigador ni la precisión o calibración de los instrumentos.

Esta naturaleza probabilística de la mecánica cuántica dista mucho de ser una propiedad intuitiva para el científico computacional o cualquier otro humano; de hecho, ha sido motivo de controversia desde el nacimiento de la teoría cuántica hasta nuestros días (por ejemplo, revisar [22, 32, 59, 26]). Para un científico computacional interesado en desarrollar algoritmos cuánticos, aprender estas nuevas formas de razonar será parte fundamental de su proceso educativo.

Algunos algoritmos basados en caminatas cuánticas Diversos algoritmos basados en caminatas cuánticas resuelven varias instancias y formulaciones de un problema de búsqueda, que en forma abstracta se plantea así: dado un espacio de estados traducible a un grafo G, encuentre un estado particular, el cual tiene una marca distintiva, a través de la ejecución de una caminata cuántica en G. El planteamiento se generaliza fácilmente para localizar un conjunto de estados marcados, en vez de uno solo.

Por supuesto, con la lectura del párrafo anterior, una pregunta asalta la mente: ¿es razonable suponer que el nodo buscado tendrá siempre una marca distintiva? La respuesta se puede dar en dos planos distintos:

- Para aplicaciones concretas de algoritmos de búsqueda, en común estar en posibilidad de distinguir el nodo que buscamos del resto.
- En algunos problemas cuya solución algorítmica está inspirada en procesos físicos, es posible garantizar que el nodo buscado está marcado por el valor mínimo (o máximo) de la propiedad física incorporada en el algoritmo.

A efecto de caracterizar estos problemas en los que hay que buscar elementos reconocibles (i.e. marcados), la ciencia computacional provee de una abstracción llamada oráculo:

Un **oráculo** es una máquina abstracta utilizada para estudiar problemas de decisión. Se puede pensar en esta máquina como una caja negra.

Los oráculos son elementos utilizados ampliamente en la construcción de algoritmos basados en caminatas cuánticas. A continuación, se presentan algunos algoritmos basados en caminatas cuánticas y que emplean oráculos.

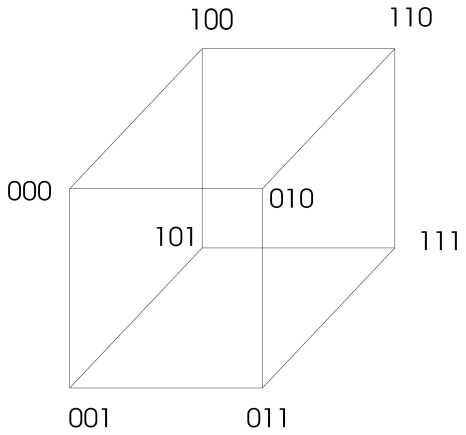


Figura 7. Hipercubo con $n = 3$

1. **Viaje a través de un hipercubo**. Para estudiar este algoritmo, definimos antes lo siguiente:
   Un hipercubo es un grafo G con 2^n nodos, donde cada nodo lleva por etiqueta un número binario de bits. Dos nodos a, b del hipercubo están conectados por una arista (a, b) si y sólo si las etiquetas de a y b difieren en un solo bit, i.e. $|a - b| = 1$, donde $|a - b|$ es la distancia de Hamming entre a y b.
   Un ejemplo de hipercubo con $n = 3$ se muestra en la Fig. (7).
   Pensemos ahora en el siguiente problema: dado un hipercubo, calcule el tiempo (i.e. el número de pasos) que tardaría un algoritmo en cruzar la

distancia entre dos nodos arbitrarios (a, b). Este problema tiene solución a través de un algoritmo clásico [12] y otro cuántico [68], en ambos casos polinomiales (i.e. son algoritmos P). El algoritmo cuántico propuesto en [68] emplea una caminata cuántica discreta y un oráculo.

Otro problema de búsqueda, propuesto y solucionado en [2] empleando una caminata cuántica discreta, se define en el siguiente párrafo:

2. **Elementos distintos** (en inglés, *element distinctness problem*). Sea S una lista de cadenas de caracteres (*strings*) definidos sobre el conjunto $\{0, 1\}$ separados entre sí por el símbolo #, i.e. $S = s_1 \# s_2 \# s_3 \# \ldots$ donde $s_{\in}\{0, 1\}^*$. Determine si todos los strings son distintos entre sí.

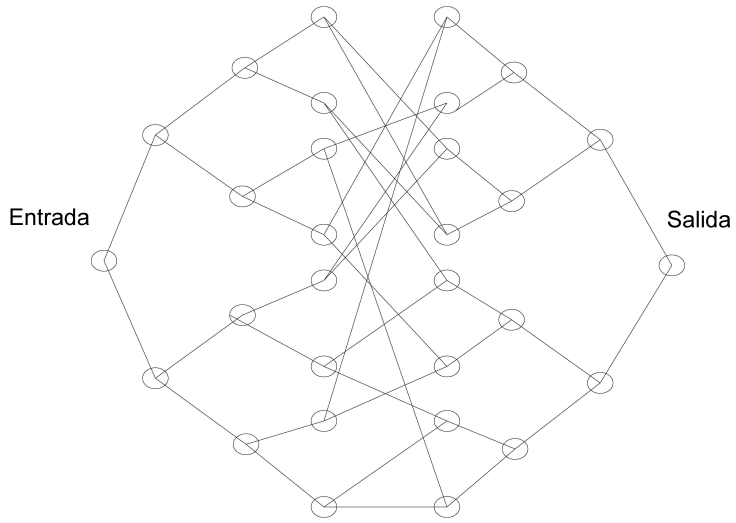


Figura 8. Árbol con uniones intermedias aleatorias.

3. **Aceleramiento exponencial usando una caminata cuántica**. Para terminar esta sección, deseo presentar a usted un último algoritmo, éste basado en una caminata cuántica continua y desarrollado en [12]. El problema a resolver se puede visualizar en la Fig. (8) y consiste en comenzar una caminata en el nodo *entrada* para terminar en el nodo *salida*. Dada la estructura irregular del centro de este grafo, formada por uniones aleatorias entre las hojas de los árboles izquierdo y derecho, es posible demostrar dos cosas [12]: 1) no es posible construir un algoritmo clásico que haga el recorrido solicitado en tiempo polinomial, y 2) es posible construir un algoritmo cuántico, basado en una caminata cuántica continua (sección 3.4) que, con alta probabilidad, logre hacer el recorrido solicitado en tiempo polinomial.

4. Conclusiones

La computación cuántica es una disciplina llena de ideas y múltiples enfoques, esto último debido a la plétora de profesiones e intereses científicos que habitan su espacio. Además de ser tierra fértil para la elaboración de nuevas teorías y estudios, el cómputo cuántico es tiene amplias oportunidades de desarrollo profesional para científicos e ingenieros. Más aún, la computación cuántica es una oportunidad para el crecimiento tecnológico y la generación de riqueza material, a través de empresas de alta tecnología que desarrollen productos de alto valor agregado nacidos de la ingeniería cuántica [16].

Agradecimientos

Mi sincero agradecimiento para el Dr. Genaro Juárez Martínez y el Dr. Héctor Zenil por su invitación, apoyo y paciencia en la elaboración de este manuscrito. Agradezco también al Sistema Nacional de Investigadores (SNI) por la beca (número de expediente 41594) con la que se pagó parte del tiempo que utilicé en la elaboración de este documento.

Referencias

[1] Altenkirch, T. & Grattage, J. (2005). A functional quantum programming language, *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*, 249-258.

[2] Ambainis, A. (2004). Quantum walk algorithm for element distinctness, *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, 22-31.

[3] Arndt, M., Juffmann, T. & Vedral, V. (2009). Quantum physics meets biology, *HFSP Journal*, 3(6), 386-400.

[4] Banuls, M. C., Orús, R., Latorre, J. I., Pérez, A., & Ruiz-Femenía, P. (2006). Simulation of many-qubit quantum computation with matrix product states, *Phys. Rev. A*, 73, 022344.

[5] Benioff, P. A. (2002). Space searches with a quantum robot, In *Quantum Computation and Quantum Information: A Millenium Volume AMS Contemporary Mathematics Series*, Amer. Math. Soc., Providence, RI, (J. S. J. Lomonaco & H. E. Brandt (Eds.)), 1-12.

[6] Bettelli, S., Calarco, T., & Serafini, L. (2003). Toward an architecture for quantum programming, *The European Physical Journal D - Atomic, Molecular, Optical and Plasma Physics*, 25(2), 181-200.

[7] Bouwmeester, D., Ekert, A., & Zeilinger, A. (Eds.) (2010). *The Physics of Quantum Information*, Springer Verlag.

[8] Caraiman, S. & Manta, V. I. (2009). New applications of quantum algorithms to computer graphics: the quantum random sample consensus algorithm, *Proceedings of the 6th ACM conference on Computing frontiers*, 81-88.

[9] Caraiman, S. &. Manta, V. (2010). Parallel simulation of quantum search, *International Journal of Computers, Communications and Control*, V(5), 634-641.

[10] Caruso, F., Chin, A. W., Datta, A., Huelga, S. F., & Plenio, M. B. (2010). Entanglement and entangling power of the dynamics in light-harvesting complexes, *Physical Review A*, 81, 062346 .

[11] Childs, A. M. (2010). Universal Computation by Quantum Walk, *Physical Review Letters*, 102, 180501.

[12] Childs, A. M., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., & Spielman, D. A. (2003). Exponential algorithmic speedup by quantum walk, *Proc. 35th ACM Symposium on Theory of Computing*, 56-68.

[13] Childs, A. & van Dam, W. (2010). Quantum algorithms for algebraic problems, *Review of Modern Physics*, 82, 1-51.

[14] Cohen-Tannoudji, C., Diu, B., & Laloe, F. (1977). *Quantum Mechanics, Vols. 1 & 2*, Wiley-Interscience.

[15] Cooper, W. G. (2011). Accuracy in Biological Information Technology Involves Enzymatic Quantum Processing and Entanglement of Decohered Isomers, *Information Journal*, 2(1), 166-194.

[16] Corker, D., Ellsmore, P., Abdullah, F., & Howlett, I. (2005). Commercial Prospects for Quantum Information Processing, *Technical report, Saïd Business School, The University of Oxford*.

[17] Deutsch, D. (1985). Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer, *Proceedings of the Royal Society of London A*, 400(1818), 97-117.

[18] Diamantini, M. C. & Trugenberger, C. (2006). Quantum Pattern Retrieval by Qubit Networks with Hebb Interactions, *Physical Review Letters*, 97, 130503.

[19] Dirac, P. A. M. (1930). *The Principles of Quantum Mechanics*, Oxford University Press.

[20] DiVincenzo, D. P. (2000). The Physical Implementation of Quantum Computation, *Fortschritte der Physik*, 48, 771-784.

[21] Díaz-Pier, S., Venegas-Andraca, S. E. & Gómez-Muñoz, J. L. (2011). Classical Simulation of Quantum Adiabatic Algorithms using Mathematica on GPUs, *International Journal of Unconventional Computing*, in press.

[22] Einstein, A. (1954). *Ideas and Opinions*, Wing Books.

[23] Farhi, E. & Gutmann, S. (1998). Quantum computation and decision trees, *Phys. Rev. A*, 58, 915-928.

[24] Feynman, R. P. (1982) Simulating Physics with Computers, *International Journal of Theoretical Physics*, 21(6-7), 467-488.

[25] Feynman, R. P. (1999). *The Feynman Lectures on Computation*, Penguin Books.

[26] Feynman, R. P., Leighton, R. B., & Sands, M. (1965). *The Feynman Lectures on Physics, vol. III*, Addison-Wesley Publishing, Co..

[27] Fraenkel, A. S. (1993). Complexity of Protein Folding, *Bulletin of Mathematical Biology*, 55(6), 1199-1210.

[28] Gauger, E. M., Rieper, E., Morton, J. L., Benjamin, S. C. & Vedral, V. (2011). Sustained Quantum Coherence and Entanglement in the Avian Compass, *Physical Review Letters*, 106, 040503.

[29] Grover, L. K. (1996). A fast quantum mechanical algorithm for database search, *Proc. 28th Annual ACM Symposium on the Theory of Computing*, 212-219.

[30] Gruzka, J. (2000). *Quantum Computation*, McGraw Hill.

[31] Gupta, S. (2001). Quantum Neural Networks, *Journal of Computer and System Sciences*, 63, 355-383.

[32] Heisenberg, W. (1962). *Physics and Philosophy*, Penguin Group.

[33] Hollenberg, L. C. L. (2000). Fast Quantum Search Algorithm in Protein Sequence Comparison- Quantum Biocomputing, *Physical Review E*, 62, 7532-7535.

[34] Hoyer, S., Sarovar, M. & Whaley, K. B. (2010). Limits of quantum speedup in photosynthetic light harvesting, *New Journal of Physics*, 12, 065041.

[35] Imre, S. & Balázs, F. (2005). *Quantum Computing and Communications*, John Wiley and Sons.

[36] Iwama, K. & Tamaki, S. (2003). Improved upper bounds for 3-SAT, *Electronic Colloquium on Computational Complexity*, 53, 328.

[37] Jeong, H., Paternostro, M. & Kim, M. S. (2004). Simulation of quantum random walks using the interference of a classical field, *Physical Review A*, 69, 012310.

[38] Kassal, I., Whitfield, J. D., Perdomo-Ortiz, A., Yung, M. H., & Aspuru-Guzik, A. (2011). Simulating Chemistry with Quantum Computers, *Annual Review of Physical Chemistry*, 62, 185-207.

[39] Kempe, J. (2003. Quantum random walks - an introductory overview, *Contemporary Physics*, 44(4), 307-327.

[40] Kendon, V. (2006). A random walk approach to quantum algorithms, *Philosophical Transactions of the Royal Society A*, 364, 3407-3422.

[41] Kitaev, A. Y., Shen, A. H., & Vyalyi, M. N. (2002). *Classical and Quantum Computation*, American Mathematical Society.

[42] Knight, P. L., Roldán, E. & Sipe, J. E. (2003). Quantum walk on the line as an interference phenomenon, *Physical Review A*, 68, 020301.

[43] Knight, P. L., Roldán, E., & Sipe, J. E. (2004). Propagating quantum walks: the origin of interference structures, *Journal of Modern Optics*, 51(12), 1761-1777.

[44] Konno, N. (2008). Quantum Walks, In *On Quantum Potential Theory (Lecture Notes in Mathematics)*, U. Franz & M. Schuermann (Eds.), Springer Verlag.

[45] Lanzagorta, M. (2011). Biologically Inspired Quantum Sensor for Magnetic Anomaly Detection in Anti-Submarine Warfare, *Cuarta Reunión de la División de Información Cuántica, Sociedad Mexicana de Física*.

[46] Lanzagorta, M. & Ullman, J. (2009. *Quantum Computer Science*, Morgan and Claypool Publishers.

[47] Le, P. Q., Doyng, F., & Hirota, K. (2010). A flexible representation of quantum images for polynomial preparation, image compression, and processing operations, *Quantum Information Processing*, 10(1), 63-84.

[48] Le, P. Q., Iliyasu, A. M., Doyng, F., & Hirota, K. (2011). Strategies for designing geometric transformations on quantum images, *Theoretical Computer Science*, 412(15), 1046-1418.

[49] Lovett, N. B., Cooper, S., Everitt, M., Trevers, M., & Kendon, V. (2010). Universal quantum computation using the discrete-time quantum walk, *Physical Review A*, 81, 042330.

[50] Lovász, L. (1996). Random Walks on Graphs: A Survey, *Combinatorics, Paul Erdös is Eighty, Vol. 2 (ed. D. Miklós, V. T. Sós, T. Szönyi), János Bolyai Mathematical Society, Budapest*, 353-398.

[51] Metodi, T. S. & Chong, F. T. (2006). *Quantum Computing for Computer Architects*, Morgan and Claypool Publishers.

[52] Meyer, D. A. (1996). From quantum cellular automata to quantum lattice gases, *Journal of Statistical Physics*, 85, 551-574.

[53] Mitchell, M. (2009). *Complexity: a guided tour*, Oxford University Press.

[54] Mohseni, M., Rebentrost, P., Lloyd, S., & Aspuru-Guzik, A. (2008). Environment-assisted quantum walks in photosynthetic energy transfer, *Journal of Chemical Physics*, 129, 174106.

[55] Motwani, R. & Raghavan, P (1995). *Randomized Algorithms*, Cambridge University Press.

[56] Munakata, T. (2007). Beyond Silicon: New Computer Paradigms, *Communications of the ACM*, 50(9), 30-72.

[57] Murg, V., Legeza, Ö., Noack, R.M., & Verstraete F. (2010). Simulating Strongly Correlated Quantum Systems with Tree Tensor Networks, *Physical Review B*, 82, 205105.

[58] Nayak, A. & Vishwanath, A. (2000). Quantum walk on the line, arXiv:quant-ph/0010117v1.

[59] Nielsen, M. A. & Chuang, L. I. (2000). *Quantum Computation and Quantum Information*, Cambridge University Peess.

[60] Nyman, P. (2009). A Symbolic Classical Computer Language for Simulation of Quantum Algorithms, *Lecture Notes in Computer Science*, 5494, 158-173.

[61] Ömer, B. (2000). Quantum Programming in QCL, *MSc Thesis*, The Technical University of Vienna.

[62] Ömer, B. (2005). Classical Concepts in Quantum Programming, *International Journal of Theoretical Physics*, 44(7), 943-955.

[63] Perdomo, A., Truncik, C., Tubert-Brohman, I., Rose, G., & Aspuru-Guzik, A. (2008). Construction of model Hamiltonians for adiabatic quantum computation and its application to finding low-energy conformations of lattice protein models, *Physical Review A*, 78, 012320–15.

[64] Preston, D. (2005). *Before the Fall-out: From Marie Curie to Hiroshima*, Doubleday.

[65] De Raedt, K., Michielsen, K., De Raedt, H., Trieu, B., Arnold, G., Richter, M., Lippert, Th., Watanabe, H., & Ito, N. (2007). Massively parallel quantum computer simulator, *Computer Physics Communications*, 176(2), 121-136.

[66] Rieffel, E. & Polak, W. (2000). An introduction to quantum computing for non-physicists, *ACM Computing Surveys*, 32(3), 300-335.

[67] Schöning, U. (1999). A probabilistic algorithm for k-sat and constraint satisfaction problems, *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS), IEEE*, 410-414.

[68] Shenvi, N., Kempe, J., & Whaley, B. (2003). A Quantum Random Walk Search Algorithm, *Physical Review A*, 67(5), 050237.

[69] Shor, P. (1994). Polynomial-Time Algorithms for Prime Factorization and Discrete Algorithms on a Quantum Computer, *Proc. 35th Annual Symposium on Foundations of Computer Science*, 124-134.

[70] Trugenberger, C. (2001). Probabilistic Quantum Memories, *Physical Review Letters*, 87, 067901.

[71] Trugenberger, C. (2002). Phase Transitions in Quantum Pattern Recognition, *Physical Review Letters*, 89, 277903.

[72] Trugenberger, C. (2002). Quantum Pattern Recognition, *Quantum Information Processing*, 1(6), 471-493.

[73] Venegas-Andraca, S. E. (2006). Discrete Quantum Walks and Quantum Image Processing, *PhD Thesis*, The University of Oxford.

[74] Venegas-Andraca, S. E. (2008). *Quantum Walks for Computer Scientists*, Morgan and Claypool Publishers.

[75] Venegas-Andraca, S. E. & Ball, J. L. (2010). Processing Images in Entangled Quantum Systems, *Quantum Information Processing*, 9(1), 1-11.

[76] Venegas-Andraca, S. E. & Bose, S. (2003). Storing, processing and retrieving an image using Quantum Mechanics, *Proc. SPIE Conference Quantum Information and Computation*, 137-147.

[77] Viamontes, G. F., Markov, I., & Hayes, J. P. (2003). Improving Gate-Level Simulation of Quantum Circuits, *Quantum Information Processing*, 2(5), 347-380.

[78] Weinstein, Y. S., Hellberg, C. S., & Levy, J. (2004). Quantum-Dot Cluster-State Computing with Encoded Qubits, *Physical Review A*, 72(2), 020304.

[79] Woess, W. (2000). *Random walks on infinite graphs and groups*, Cambridge University Press.

[80] Zhang, C. Y., Yeh, H. C., Kuroki, M. T., & Wang, T. H. (2005). Single Quantum-Dot-Based DNA nanosensor, *Nature Materials*, 4, 826-831.

Hacia una descripción realista del tráfico vehicular basada en autómatas celulares

María Elena Lárraga Ramírez, Luis Alvarez-Icaza

Instituto de Ingeniería
Universidad Nacional Autónoma de México, 04510, Coyoacán, México, D.F.
mlarragar@iingen.unam.mx, alvar@pumas.iingen.unam.mx

Resumen Los modelos para tránsito vehicular basados en autómatas celulares (AC) han llegado a ser un método bien establecido para modelar, analizar y entender el tráfico vehicular real. Sin embargo, un defecto común que todavía no se ha logrado rectificar es la desaceleración abrupta cuando los vehículos se encuentran con obstáculos fijos o estancamientos de tráfico. En la búsqueda de un modelo más acorde con el movimiento vehicular en el mundo real, recientemente desarrollamos un modelo de AC que toma en cuenta las políticas de espaciamiento de los conductores normales y prácticas de ingeniería de transporte para su definición [18]. Las reglas del modelo toman en cuenta el espaciamiento vehicular y las velocidades relativas entre dos vehículos e incorpora capacidades de aceleración/desaceleración vehicular con un valor límite que se establece con base en el desempeño de los vehículos reales. Así, el modelo corrige los desempeños de desaceleración irreales y puede reflejar más fielmente el desempeño del conductor real. Además de mantener la simplicidad computacional de los modelos de AC. Este trabajo describe en forma más detallada tal modelo, con la finalidad de enfatizar sus características que lo hacen diferente de otros modelos basados en AC previos y presenta algunos los resultados de simulación previamente publicados, que validan su desempeño.

1. Introducción

En los países desarrollados, los sistemas de transporte se basan principalmente en el uso de vehículos automotores, lo que ha originado un incremento continuo de la demanda vehicular y por lo tanto, que se sobrepase la capacidad para la cual fueron diseñadas sus vías, calles o autopistas. Como consecuencia, la contaminación ambiental y los congestionamientos vehiculares se incrementan día con día, mientras que la seguridad vial se decrementa. Por lo que el tráfico vehicular ha llegado a ser uno de los problemas sociales y económicos más importantes de la vida diaria. Aunque la construcción de nuevas vías de transporte o la modificación de las vías existentes pueden ser un método simple y efectivo para disminuir las consecuencias inducidas por la alta demanda vehicular, debido a diversas restricciones espaciales, sociales y económicas, no es fácil de implementar en la actualidad. La alternativa es buscar nuevas soluciones orientadas

a un uso más eficiente de las infraestructuras existentes, que permitan mejorar el desempeño de las mismas. Sin embargo, probar los impactos de estas nuevas soluciones en el mundo real antes de su implementación final puede ser muy costoso y no factible. Esto ha motivado el desarrollo continuo de modelos de tráfico vehicular orientados para el análisis y entendimiento del comportamiento del tráfico vehicular y la valoración de las alternativas para mejorar su desempeño.

La modelación del tráfico vehicular tiene una historia muy amplia, se ha desarrollado desde el punto de vista de la ingeniería, la física y las matemáticas aplicadas [24, 5, 8, 26]. Las clases de modelos usados para su descripción se pueden clasificar en dos grupos principales, los modelos microscópicos y los modelos macroscópicos. Los modelos microscópicos representan a cada vehículo en forma separada, lo que permite considerar diferentes tipos de vehículos o conductores con propiedades individuales. En los modelos macroscópicos, el estado del sistema se describe a través de densidades, por ejemplo, la densidad de masa derivada de las posiciones de los vehículos. A lo largo de los años se han desarrollado diversas aproximaciones para modelación [5, 8]. Los modelos hidrodinámicos ven el tráfico como un fluido compresible formado por varios vehículos. Esta aproximación es macroscópica dado que se basa en densidades más que distinguir entre vehículos. Los modelos de cinética de gas tratan de derivar modelos macroscópicos a partir de ecuaciones microscópicas [24]. El tráfico vehicular se trata como un gas de partículas que interactúan, que se describe por una función de distribución con la evolución temporal dada por una ecuación de Boltzmann. Por otra parte los modelos microscópicos se clasifican principalmente en tres: los modelos de seguimiento del vehículo (car-following), los modelos de velocidad óptima y los modelos basados en AC. Los modelos de seguimiento del vehículo son aproximaciones microscópicas que usan ideas de la mecánica de Newtoniana para describir el tráfico vehicular [3]. La aceleración se determina, por ejemplo, por la diferencia de velocidad al vehículo precedente. Mientras que en los modelos de velocidad óptima [1], los vehículos no tratan de adoptar la velocidad de su predecesor, sino una velocidad óptima la cual depende del espaciamiento existente. Por su parte, los modelos basados en autómatas celulares (AC) son modelos microscópicos, en los cuales la dinámica vehicular depende de un conjunto de reglas de evolución locales y simples, fáciles de entender, computacionalmente eficientes y suficientes para emular el desempeño que se observa en el tránsito vehicular.

En los modelos para tráfico vehicular basados en AC, el espacio que se simula, el tiempo y las variables de estado que se usan son discretas. Así, la red de transporte se parte en una malla con una topología ordenada, inducida por la topología real, que respeta las relaciones de conectividad y los sentidos de circulación de la red original. El estado de los vehículos se caracteriza por su posición y velocidad. La primera está determinada por su ubicación dentro de la malla y la segunda surge de las relaciones que el vehículo bajo análisis guarda con su entorno, determinado por los vehículos vecinos y la presencia de elementos externos (intersecciones, semáforos, etc.). La dinámica vehicular de los modelos de AC se basa usualmente en reglas intuitivas y locales, que permite

reproducir las decisiones que los conductores toman basados en su situación actual, la relación con sus vecinos, su metas, etc. Este hecho es importante ya que permite tomar en cuenta por ejemplo, aspectos psicológicos o de comportamiento de los conductores en una forma natural y eficiente. Además, la interacción local entre los vehículos permite capturar dinámicas a un nivel microscópico y propagarlas a un nivel macroscópico. Por lo que los modelos para tránsito vehicular basados en AC han llegado a ser un método bien establecido para modelar, analizar, entender y aun para pronosticar el desempeño del tránsito vehicular [22, contiene un resumen de modelos de AC]

El primer modelo que mostró las bondades de los AC para simular el tráfico vehicular surgió al inicio de los años 90s, cuando los alemanes [23] propusieron un modelo (referido como NaSch) para la simulación del tráfico vehicular de carreteras. Aunque el modelo NaSch reproduce la estructura básica de la relación densidad-flujo observada empíricamente y la formación espontáneas de estancamientos vehiculares, no exhibe otras características del tráfico vehicular como metaestabilidad, el flujo sincronizado, decaimiento de la capacidad vehicular, etc. Por lo que desde su creación se han desarrollado un número considerable de modificaciones ó extensiones del modelo NaSch (ver por ejemplo, [6, 2, 9, 14, 17, 12, 19, 20, 11, 22, 10]).

Sin embargo, la mayoría de los modelos se han sido orientados a reproducir los fenómenos que ocurren en el tráfico vehicular real y raramente han considerado una velocidad de desaceleración con un valor límite semejante al de los vehículos reales. De hecho, la mayoría de los modelos existentes han considerado explícitamente criterios libres de colisión, mediante la imposición de desaceleraciones arbitrariamente grandes (de 100 km/h a 0 en 1s) que se alejan de la capacidad práctica de frenado en pavimento y las condiciones de los neumáticos. Desde luego, que estas desaceleraciones abruptas exceden las capacidades de desaceleración reales en condiciones normales.

Recientemente, en la búsqueda de nuevos modelos de AC que en su definición tomen en cuenta los mecanismos que conllevan a los fenómenos que suceden en el tráfico real y no solamente a su reproducción, desarrollamos un nuevo modelo de AC para un solo carril [18]. El modelo (que aquí referiremos como modelo LAI) trata de capturar las reacciones de los conductores a las condiciones de tráfico vehicular, mientras se preserva la seguridad en las carreteras. De tal manera que el desempeño humano se modela como la respuesta del conductor a las condiciones de tráfico locales. Para este propósito, el modelo toma en cuenta para la definición de la dinámica vehicular el espacio existente entre dos vehículos, su velocidad relativa y las capacidades de aceleración/desaceleración con un valor límite acorde al de los vehículos reales. La determinación de las capacidades de aceleración/desaceleración se derivan de principios de conducción segura para los conductores normales y en acuerdo con las prácticas de transporte y las reacciones humanas [7, 4]; tal que los vehículos no puedan cambiar las velocidades abruptamente (como en la realidad). De tal manera. que el modelo LAI rectifica la desaceleración irreal de los vehículos cuando se enfrentan a aun estancamiento o incidente, como ocurre en la mayoría de los modelos de AC previos. Además,

el modelo es muy simple y computacionalmente eficiente. En este trabajo se describe en una forma forma más detallada el modelo LAI presentado en [18]. El objetivo principal es enfatizar las características del modelo que lo hacen diferente de otros modelos basados en AC existentes en la literatura, así como mostrar algunos resultados de simulación previamente publicados, los cuales validan la eficiencia del modelo para reproducir fenómenos que ocurren en el tráfico real.

El resto de este trabajo está organizado de la siguiente manera. En la sección 2, se presenta una introducción al modelo NaSch y algunos modelos previos que introducen el concepto de desaceleración con un valor límite en su definición. En la sección 3, se presenta una descripción del modelo desarrollado recientemente. En la sección 4 se presentan resultados de simulación del modelo para un sistema de un solo carril, con condiciones de frontera periódica. Finalmente, en la sección 5 se presentan las conclusiones de este trabajo.

2. El modelo de Nagel-Schreckenber (NaSch)

El modelo NaSch es un modelo de autómata celular probabilista para el tráfico vehicular. La carretera se divide en celdas de igual tamaño, las cuales pueden estar vacías u ocupadas por un vehículo con una velocidad $v = 0, 1, \ldots, v_{max}$. Los vehículos se mueve desde el extremo izquierdo al extremo derecho de la carretera. En cada paso de tiempo discreto $t \leftarrow t + 1$, la actualización del sistema se desempeña en paralelo de acuerdo a las siguientes cuatro reglas:

R1 Aceleración: $v_n(t + 1/3) = min(v_n(t) + 1, v_max)$;
R2 Desaceleración : $v_n(t + 2/3) = min(v_n(t + 1/3), d_n(t))$;
R3 Frenado aleatorio: $v_n(t + 1) = max(v_n(t + 2/3) - 1, 0)$ con probabilidad p;
R4 Cambio de posición: $x_n(t + 1) = x_n(t) + v_n(t + 1)$;

Aquí, v_n y x_n denotan la velocidad y posición del vehículo n respectivamente; v_{max} es la velocidad máxima y $d_n = x_{n+1} - x_n - 1$ denota el número de celdas vacías en frente del vehículo n; p es la probabilidad de frenado aleatorio. Es importante mencionar que un cambio en el orden las reglas de transición definidas cambiaría las propiedades del modelo; en otras palabras, las reglas del modelo NaSch no conmutan.

Todas la reglas tienen una interpretación simple. La regla R1 expresa el deseo de los conductores para moverse tan rápido como sea posible. La regla R2 refleja las interacciones entre vehículos consecutivos y garantiza la ausencia de colisiones en el modelo. La velocidad del vehículo precedente no se toma en cuenta. La regla R3 introduce una asimetría entre aceleración y desaceleración e incorpora fluctuaciones naturales inherentes al conductor. Finalmente, la regla R4 mueve los vehículos hacia adelante con base en la velocidad adquirida como resultado de la aplicación de las tres reglas previas. La longitud de una celda, Δx, corresponde a 7.5 m en la realidad, entonces para $v_{max} = 5$ y $p = 0{,}5$. Un paso de tiempo corresponde aproximadamente a 1 s en unidades de tiempo reales y corresponde al tiempo de reacción de un conductor.

Aún tratándose de un modelo elemental (ya que quitándole cualquiera de sus componentes, el modelo ya no reflejaría la realidad), el modelo NaSch únicamente sirve para modelar autopistas congestionadas, con vehículos uniformes y en un único carril. Para modelar otros aspectos del tráfico, como variedad de tipos de vehículos, múltiples carriles, u otros fenómenos como distintas formas de interacción entre los vehículos, se requiere modificar este para captar otras realidades que puedan modelarse. Por lo que a parir del modelo NaSch se han propuesto diversas modificaciones o extensiones. Sin embargo, como la mencionamos previamente, la mayoría de estos modelos se han orientado a reproducir los fenómenos que ocurren en el tráfico vehicular real y que no se pueden reproducir con el modelo NaSch (como la metaestabilidad, el efecto de histéresis, decaimiento de la capacidad, el flujo sincronizado, etc.). La idea de la desaceleración con un valor limite se ha considerado raramente. En la siguiente subsección se describen brevemente algunos modelos de AC que consideran en su definición una desaceleración vehicular con un valor acotado.

2.1. Algunos modelos con desaceleración acotada

Uno de los primeros esfuerzo en introducir la capacidad de desaceleración con un valor límite en la modelación basada en autómatas celulares fue el modelo propuesto por [15] (modelo KW). Ellos introdujeron el término llamado velocidad segura a través del siguiente concepto.

$$v^{(safe)}\tau^{(safe)} + X_d(v^{(safe)}) \leq g_n + X_d(v_{l,n})$$

donde g_n denota la brecha espacial.

$$X_d(u) = (u - b\tau) + (u - 2b\tau) + \cdots + \beta b\tau = b\tau^2(\alpha\beta + \frac{\alpha(\alpha-1)}{2})$$

representa la distancia esperada de viaje, con la velocidad original u, el intervalo de desaceleración b. Y el intervalo de tiempo seguro para los conductores, que es determinado de la siguiente manera.

$$\tau^{(safe)} = v^{(safe)}/b = \alpha_{safe} + \beta_{safe}$$

La velocidad del vehículo precedente se representa por la siguiente ecuación

$$\alpha_{safe} = \sqrt{2\frac{X_d(v_{l,n}) + g_n}{b}}\frac{1}{4} - \frac{1}{2}\beta_{safe}$$

Además de la complejidad con la que el modelo simula la dinámica vehicular, los resultados del modelo de Krauss mostraron claramente desaceleraciones que exceden las correspondientes a la realidad.

Recientemente, [16] introdujeron además capacidades de aceleración (a) y desaceleración (D) en su modelo y propusieron el siguiente criterio de seguridad para el movimiento de los vehículos, que es muy similar al propuesto por Krauss y Wagner.

$$x_n^t + \Delta + \sum_{i=1}^{\tau_f(c_n^{t+1})} (c_n^{t+1} - D_i) \leq x_{n+1}^t + \sum_{i=1}^{\tau_l(v_{n+1}^t)} (v_{n+1}^t - D_i)$$

donde n (n+1) denota al vehículo seguidor (conductor). c_n^{t+1} denota la velocidad segura al tiempo $t+1$. x_{n+1}^t (v_{n+1}^t) denota la posición (velocidad) del vehículo conductor. x_n^t (v_n^t) denotan la posición (velocidad) del vehículo seguidor al tiempo t. τ_f (τ_l), denota los pasos de tiempo requeridos por el vehículo seguidor (conductor) para desacelerar hasta parar; $i = 0, 1, \ldots \tau_f$ para el vehículo seguidor e $i = 0, 1, \ldots, \tau_l$ para el vehículo conductor; D es la capacidad de frenado máxima y Δ es la distancia mínima con respecto al vehículo conductor.

Tanto el modelo KW como el modelo de Lee et al. antes mencionados, se establecen bajo la suposición que el vehículo siguiente siempre tendrá conocimiento de la velocidad del vehículo que va adelante y así, mantendrá continuamente una distancia adecuada para evitar colisiones en caso de que el vehículo precedente desacelere a un paro total en el siguiente paso de tiempo. Sin embargo, la distancia de seguimiento segura que un vehículo debe mantener respecto al vehículo que le precede sigue siendo sobre-conservativa debido al uso de las velocidades absolutas para su determinación. Además, con la idea de incorporar capacidades de desaceleración limitadas, estos modelos más sofisticados utilizan un conjunto de reglas complejo, con un número de parámetros muy grande en comparación con el modelo original NaSch.

El modelo LAI

Recientemente, en la búsqueda de un modelo basado en AC para el tráfico vehicular más acorde con el desempeño microscópico real, desarrollamos un nuevo modelo para tráfico vehicular basado en AC [18]. El modelo enfatiza el conflicto entre el desempeño humano, las políticas espaciales del conductor normal y las capacidades de aceleración y desaceleración de los vehículos, como origen de los congestionamientos vehiculares. Como resultado, las reacciones de los conductores se basan en un análisis de seguridad que determina la acción más apropiada a tomar. El modelo introduce un conjunto de reglas nuevo para cambiar la velocidad de los vehículos, mediante la incorporación de tres umbrales importantes que determinan las distancias de seguimiento seguro que deben existir entre un vehículo y su predecesor para acelerar (d_{acc}), desacelerar (d_{dec}) o mantener su velocidad (d_{keep}) en el siguiente paso de tiempo. Para la definición de estos umbrales de seguridad, el modelo toma en cuenta el espaciamiento existente entre dos vehículos, su velocidad relativa y capacidades de aceleración/desaceleración con un valor límite acorde al comportamiento de los vehículos en la realidad. La definición de los mismos se explicará en forma detallada más adelante.

El modelo se define sobre un arreglo uni-dimensional de celdas de longitud L, donde cada celda puede estar vacía u ocupada por un solo vehículo. La velocidad de cada vehículo puede tomar uno de los (v_{max+1}) valores enteros permitidos, $v = 0, 1, \ldots, v_{max}$. Los vehículos pueden ocupar más de una celda. Debido a que

se considera solamente un carril, solamente un tipo de vehículo se considera en el artículo y por lo tanto, se usa el mismo valor para la velocidad máxima de todos los vehículos. Para reproducir los efectos estocásticos del desempeño del conductor individual, el modelo considera capacidades de aceleración y desaceleración con base en los vehículos individuales.

Un paso de cambio del sistema consiste de los siguientes cuatro pasos, los cuales se aplican en paralelo a todos los vehículos cada paso de tiempo.

S1 : Distancias seguras. Obtener el valor para $d_{dec_n} = d_{dec}(t, v_n(t), v_{n+1}(t))$, $d_{acc_n} = d_{acc}(t, v_n(t), v_{n+1}(t))$, y $d_{keep_n} = d_{keep}(t, v_n(t), v_{n+1}(t))$

S2 : Aceleración retardada. Obtener el parámetro de ruido estocástico R_a, basado en la velocidad del vehículo v_n.
$R_a = min(R_d, R_0 + v_n(t) \cdot (R_d - R_0)/v_s)$
donde la velocidad v_s es una constante ligeramente mayor a 0.

S3 : Sea Δv que denota la máxima magnitud en celdas para acelerar/desacelerar a un vehículo en una situación normal. La actualización de la velocidad de los vehículos a lo largo de la carretera se lleva a cabo al actualizar simultáneamente todos los sitios del arreglo de acuerdo a las siguientes reglas:

S3a : Aceleración. Si $d_n(t) \geq d_{acc_n}$, la velocidad del vehículo n es incrementada aleatoriamente en Δv con probabilidad (R_a), i.e.,

$$v_n(t+1) = \begin{cases} min(v_n(t) + \Delta v, v_{max}), & \text{si randf()} \leq (R_a) \\ v_n(t), & \text{en otro caso} \end{cases}$$

donde randf()$\in [0, 1]$ denota un número uniformemente aleatorio (específicamente para el vehículo n al tiempo t)

S3b : Desaceleración aleatoria. Si $d_{acc_n} > d_n(t) \geq d_{keep_n}$, la velocidad del vehículo n se decrementa con probabilidad R_s, i.e.,

$$v_n(t+1) = \begin{cases} max(v_n(t) - \Delta v, 0), & \text{if randf()} \leq (R_s) \\ v_n(t), & \text{en otro caso} \end{cases}$$

S3c : Desaceleración. Si $d_{keep_n} > d_n(t) \geq d_{dec_n}$ and $v_n(t) > 0$, la velocidad del vehículo n se reduce en Δv

$$v_n(t+1) \rightarrow max(v_n(t) - \Delta v, 0)$$

S3d : Frenado de emergencia. Si $v_n(t) > 0$ y $d_n(t) < d_{dec_n}(t)$, la velocidad del vehículo n se reduce en M, dado que no disminuye a menos de 0:

$$v_n(t+1) \rightarrow max(v_n(t) - M, 0)$$

donde M es el máximo decremento de velocidad en un paso de tiempo

S4 : Movimiento de los vehículos. Cada vehículo se mueve hacia adelante de acuerdo a su nueva velocidad determinada con las reglas S3a-S3d:

$$x_n(t+1) \rightarrow x_n(t) + v_n(t+1)$$

donde $x_n(t)$ y $v_n(t)$ respectivamente, denotan la posición y velocidad del vehículo n al paso de tiempo t (se asume que el vehículo $n+1$ precede al vehículo n). Entonces, el espacio al frente del vehículo n, es decir, la distancia desde la defensa delantera del vehículo n a la defensa trasera del vehículo $n+1$, se define como $d_n(t) = x_{n+1}(t) - x_n(t) - l_s$; donde l_s denota el tamaño del vehículo (en celdas) y se asume que la posición de un vehículo es la celda que contiene su

defensa trasera. El parámetro M representa la máxima capacidad de desaceleración de un vehículo en un paso de tiempo. Los parámetros estocásticos R_s, R_0 y R_d controlan las fluctuaciones de velocidad de los vehículos y se explicarán posteriormente.

Finalmente, Δv denota la magnitud para incrementar/decrementar la velocidad de un vehículo en un paso de tiempo bajo situaciones normales y se fija a un valor dado:

$$\Delta v = \lfloor 2{,}5\text{m}/\Delta x \rfloor$$

donde Δx denota el tamaño de la celda (en metros) que se utilizan para la discretización del sistema, que en este modelo es $\Delta x = 2{,}5$

Las reglas de S3a a S3d se diseñaron para actualizar la velocidad de los vehículos; la regla S4 actualiza la posición. De acuerdo a estas reglas, la actualización del estado se divide en 2 etapas, primero la velocidad y segunda la posición. En lo siguiente, se discute cada paso del modelo.

2.2. Entendiendo las reglas

S1: El punto inicial del modelo es el cálculo de las tres umbrales, que determinan las distancias de seguimiento que un vehículo debe tener con respecto al vehículo que le precede para acelerar, mantener su velocidad, o acelerar, en forma segura. Las distancias de seguimiento se definen por las siguientes ecuaciones:

$$d_{acc} = max(0, \sum_{i=0}^{(v_n(t)+\Delta v)_{div}M} [(v_n(t)+\Delta v)-i*M] - \sum_{i=0}^{(v_{n+1}(t)-M)_{div}M} [(v_{n+1}(t)-M)-i*M]) \tag{1}$$

$$d_{keep} = max(0, \sum_{i=0}^{(v_n)(t)_{div}M} [v_n(t)-i*M] - \sum_{i=0}^{(v_{n+1}(t)-M)_{div}M} [(v_{n+1}(t)-M)-i*M]) \tag{2}$$

$$d_{dec} = max(0, \sum_{i=0}^{(v_n(t)-\Delta v)_{div}M} [(v_n(t)-\Delta v)-i*M] - \sum_{i=0}^{(v_{n+1}(t)-M)_{div}M} [(v_{n+1}(t)-M)-i*M]) \tag{3}$$

donde $X_{div}Y$ denota la división entera, es decir, $X_{div}Y = \lfloor X/Y \rfloor$, donde "/" denota la división normal y $\lfloor z \rfloor$ es la función piso.

El primer término del lado derecho de las ecuaciones (1)-(3) determina la distancia que el vehículo seguidor viajaría si éste acelera ($v_n(t+1) = v_n(t)+\Delta v$), mantiene su velocidad ($v_n(t+1) = v_n(t)$)o desacelera ($v_n(t+1) = v_n(t) - \Delta v$), respectivamente, en el paso de tiempo $t+1$ y en el siguiente paso de tiempo empieza a desacelerar abruptamente (con una capacidad de frenado máxima M)

hasta que este se detiene. Mientras que el segundo término de las ecuaciones (1)-(3) se refiere a la distancia que viajaría el vehículo predecesor, si a partir del paso de tiempo $t+1$ empieza a desacelerar con la máxima velocidad de desaceleración posible (frenado de emergencia) M, hasta parar. Así, este término cuenta las desaceleraciones sucesivas del vehículo precedente durante los pasos de tiempo $i = 0, 1, \ldots, (v_{n+1}(t) - M)_{div} M$, considerando que desacelerará abruptamente con capacidad de frenado máxima, M.

Por lo tanto, la substracción de los dos términos del lado derecho de las ecuaciones (1)-(3) representa la distancia de seguimiento segura requerida para parar un vehículo en una situación de emergencia sin colisionar con su vehículo precedente, considerando que acelerará, mantendrá su velocidad o desacelerará respectivamente, en el paso de tiempo $t + 1$. Note que estas distancias siempre se consideran positivas.

Es importante hacer notar que todos los cálculos involucrados en las ecuaciones (1)-(3) pueden ser desempeñados fuera de línea. Después de que estos cálculos se hayan realizado, es posible generar tres tablas fijas de tamaño $(v_{max} + 1) \times (v_{max} + 1)$ que contendrán las distancias que requieren los vehículos para acelerar, mantener su velocidad o desacelerar. Con el uso de tablas de búsqueda, el costo computacional resultante de calcular las distancias de seguimiento seguro es muy bajo.

S2: Este paso obtiene el valor correspondiente para el parámetro estocástico R_a, el cual denota la probabilidad para acelerar, R_a, con base en la velocidad actual del vehículo bajo consideración. El cálculo del valor correspondiente se basa en suponer que un vehículo cuya velocidad sea menor que v_s en el paso de tiempo previo tiene una probabilidad menor de acelerar que el resto de los vehículos con una velocidad mayor a v_s. De esta manera los vehículos más lentos deben esperar más tiempo antes de continuar su jornada. Usando una idea similar a la descrita en [16], el parámetro estocástico R_a (¡1) en S2 interpola linealmente entre R_0 y R_d ($R_0 < R_d$), si v_n es menor que una velocidad dada v_s. Es importante hacer notar que para un valor dado para R_0, R_d y v_s, el cálculo que implica el paso S2 para la probabilidad de aceleración R_a también se puede desempeñar fuera de línea y el resultado almacenarse en una tabla fija de tamaño $1 \times (v_{max} + 1)$, la cual relaciona el valor de R_a con la velocidad del vehículo v_n. En la práctica, ésto reduce el uso de tres parámetros, R_0, R_d y v_s a sólo R_a.

S3a: Esta regla postula que todos los conductores intentan alcanzar la máxima velocidad siempre que le sea posible. Esto está en acuerdo con otras políticas de velocidad, como la política greedy. La regla S3a toma en cuenta la aceleración de los vehículos no uniforme, debido a que los conductores actúan en forma distinta. Por lo tanto, el proceso de aceleración introduce un elemento del comportamiento humano basado en el hecho que el desempeño del conductor puede variar en función de la situación de tráfico local y las fluctuaciones del tráfico resultantes del factor humano, en una forma estocástica. El factor estocástico considerado en esta regla es a través del parámetro R_a, definido en el paso S2. Es importante notar que la regla S3a sugiere alternativamente, que los vehículos que salen de los frentes de los estancamiento aceleran gradualmente. así, el

desempeño de los vehículos que dejan un estancamiento es más en acuerdo con el desempeño del tráfico real: un conductor necesita un instante de tiempo para acelerar su vehículo.

S3b: Esta regla refleja el hecho que los conductores tratarán de mantener su velocidad, si perciben la distancia al vehículo de enfrente como segura. Además esta regla también introduce disturbios de tráfico que ocasionan que los conductores reduzcan su velocidad sin razón aparente. Esta desaceleración aleatoria, controlada con el parámetro R_s, se aplica solamente a vehículos que están en condiciones para mantener y no requieren desacelerar. Así, el frenado doble, el cual caracteriza a los modelos de AC existentes, se evita. De tal manera que, un vehículo desacelera en forma aleatoria siempre y cuando tenga condiciones para mantener su velocidad.

S3c: Esta regla requiere que el conductor aplique frenado en forma moderada cuando el espacio que separa su vehículo del vehículo de enfrente es pequeño. Note que la desaceleración máxima que un conductor individual desea usar como desaceleración confortable, en situaciones que no son de emergencia, está acotada por Δv (en unidades de AC). Así, las desaceleraciones extremas se evitan.

S3d: Esta regla enfatiza la aproximación tomada en el modelo: las decisiones de los conductores más importantes se relacionan a la seguridad. Esta regla permite a los conductores reaccionar ante un frenado de emergencia (debido a que el vehículo precedente frena inesperadamente o el vehículo seguidor se aproxima a un vehículo parado) del vehículo precedente, lo que genera perturbaciones en las otras reglas. En este trabajo, el frenado de emergencia toma un valor de $-5{,}00$ m/s (que se considera un valor aceptable para esta maniobra [4, 21] y se alcanzaré en un paso de tiempo. Así, el parámetro M tomará valores iguales a $5{,}00/\Delta x$. Es importante enfatizar que los resultados que se presentan en este trabajo corresponden a los obtenidos de considerar un solo tipo de vehículos, es decir, vehículos con el mismo valor para la capacidad de frenado máxima M. Sin embargo, el modelo permite que el parámetro M tome valores diferentes de acuerdo al tipo de vehículo bajo consideración (automóvil, camioneta, autobús, etc), sin necesidad de modificar el modelo. De tal manera que es posible considerar distintos tipos de vehículos con capacidades de frenado diferentes, lo cual será determinante en una forma implícita en la determinación de las distancias requeridas por un vehículo para acelerar, mantener su velocidad o desacelerar: Un valor de M más pequeño (una capacidad de frenado más baja) implica una distancia de seguridad más grande para desacelerar, como ocurre en la realidad. Así, una de las contribuciones principales de este nuevo modelo es el garantizar que el desempeño vehicular microscópico sigue capacidades semejantes a la de los vehículos reales.

Además, el modelo tiene dos ingredientes que lo hacen diferente de otros modelos para tráfico vehicular basados en AC. Primeramente, en situaciones normales los vehículos no pueden cambiar sus velocidades en forma instantánea, en su lugar, los vehículos tratan de frenar con una desaceleración cuyo valor se limita por Δv (en unidades de celdas). Desaceleraciones con un valor mayor a Δv sólo son posibles en situaciones de emergencia, aunque el valor correspondiente

también se limita con base en los valores que usan los vehículos reales. Por otra parte, los conductores sobrereaccionan a las condiciones de tráfico locales de acuerdo a las tres distancias de seguimiento seguro, que toman en cuenta el hecho que un vehículo acelerará, mantendrá su velocidad o desacelerará normalmente en el siguiente paso de tiempo, considerando que el vehículo que le precede empezará a desacelerar con la capacidad de desaceleración máxima M hasta parar y las colisiones se evitan.

El valor más apropiado para Δv se determinó con base en las referencias del libro titulado *The Traffic Engineering Handbook* [27], que indican que el valor de la aceleración (desaceleración) bajo condiciones normales es acerca de $2-3$ m/^{2} (3.1 m/^{2}). Como en el modelo se considera que un paso de tiempo corresponde a 1 s, un valor aceptable para el tamaño máximo de una celda, el cual conduce a un mejor acuerdo con una aceleración y desaceleración confortable es $\Delta x = 2{,}5$ m. Es importante hacer notar que diferentes valores para Δv de acuerdo al tipo de vehículo bajo consideración (automóvil, camioneta, etc.), los cuales toman en cuenta las longitudes de los vehículos podrían ser considerados por el model (como sucede en la realidad). Además, no es necesario modificar ninguna de las reglas del modelo para ello.

Los parámetros del modelo son los siguientes: la velocidad máxima, v_{max}, la velocidad lenta v_s, la longitud del vehículo l_s (en celdas), el decremento de la velocidad máximo en un paso de tiempo M, la probabilidad de desaceleración aleatoria R_s y las probabilidades R_0 y R_d. Sin embargo, tomando en cuenta la consideración hecha para el parámetro R_a en el paso S2, en la práctica, el número efectivo de parámetros necesarios para las simulaciones cada paso de tiempo es cinco: v_{max}, l_s, R_d, R_a y M. Los parámetros adicionales l_s y M se requieren para permitir la consideración de vehículos con diferentes longitudes y capacidades de frenado. Además, mientras es claro que hay más parámetros para sintonizar que en el modelo NaSch, existen herramientas estadísticas para automatizar esta sintonización con base en datos de tráfico los cuales pueden simplificar el proceso de sintonización; sin embargo, en este trabajo no lo hicimos así.

3. Resultados de simulación

En esta sección se presentan resultados de simulación del modelo LAI tomados de [18], los cuales se obtuvieron considerando una carretera circular con condiciones de frontera periódicas. La simulaciones se realizaron sobre una carretera de $L = 2 * 10^4$ celdas. cada vehículo tiene una longitud de 5.0 m y por lo tanto, cada celda corresponde con una longitud de 2.5 m. La longitud del vehículo se determinó con base en datos empíricos que indican que la densidad de estancamiento máximo es acerca de 200 veh/km. El paso de tiempo t se toma igual a 1 s, por lo tanto, las transiciones del tiempo son de $t \longrightarrow t+1$. Este paso de tiempo es del orden del tiempo de reacción humana [11], sin embargo, puede modificarse fácilmente. Las velocidades se cambian de acuerdo a las reglas de cambio S3a-S3d y entonces, todos los vehículos se mueven hacia adelante con base en el paso S4.

Para cada simulación, inicialmente se distribuyen en forma aleatoria N vehículos, con una velocidad que toma valores entre 0 y v_{max}. Debido a que el sistema bajo consideración es cerrado, la densidad vehicular, $\rho = N/L$ se mantiene constante en el tiempo. Los valores de los parámetros se establecieron de la siguiente manera: $\Delta x = 2{,}5$ m, $R_d = 1{,}0$, $R_0 = 0{,}8$, $R_s = 0{,}01$, $l_s = 2$, $M = 2$ y $v_s = 3$. Estos valores se establecieron debido a que conducen a un acuerdo óptimo con datos empíricos. Note que considerar $R_d = 1{,}0$ significa que solamente los vehículos cuya velocidad sea más pequeña que v_s deben esperar más tiempo antes de que puedan continuar su jornada (aceleración retardada). Cada simulación se realizó para $T = 5 * 10^4$ pasos de tiempo posteriores a un periodo de transición de $10 * 10^4$ pasos de tiempo.

Actualmente se distinguen tres fases del tráfico vehicular distintas, el flujo libre, flujo sincronizado y flujo estancado[25], aunque algunos puntos son controversiales aún. En la fase de flujo libre la interacción entre los vehículos puede despreciarse. Cada vehículo se puede mover con su velocidad deseada. Por lo tanto, el flujo vehicular (cantidad de vehículos que pasan por un punto por unidad de tiempo) se incrementa linealmente con la densidad vehicular (número de vehículos por unidad de longitud). La fase de estancamientos amplios, donde los estancamientos vehiculares se pueden formar de manera espontánea, es decir, sin una razón externa obvia como un accidente o construcción de la carretera. Los estancamientos amplios son regiones con una densidad vehicular muy alta y flujo vehicular y velocidad promedio despreciables. Estos estancamiento se mueven en dirección opuesta al flujo vehicular con una velocidad característica $v_{jam} \approx 15$km/h. La fase de flujo sincronizado se forma del tráfico congestionado, el cual no puede clasificarse como estancamiento amplio. En esta fase, la velocidad promedio es significativamente más baja que en flujo libre. Sin embargo, el flujo vehicular es mucho más alto que en la fase correspondiente a estancamientos amplios. la característica principal de esta fase es la ausencia aparente de un forma funcional de la relación flujo-densidad, es decir, los puntos de los datos correspondientes se dispersan irregularmente sobre un área bidimensional amplia.

En la figura 1 se muestra el flujo promedio de 1 min con respecto a la densidad (conocido como diagrama fundamental) resultante del modelo LAI con condiciones de frontera periódicas. El diagrama fundamental se obtuvo variando la densidad global ρ entre 0 y 196 veh/km, con incremento de 2 veh/km. Para cada una de las densidades consideradas se midió el flujo local J y la velocidad local promedio, v_{loc}, a través de detectores de medición virtuales sobre una localidad específica. Entonces, la densidad local espacial, ρ_{loc}, se obtuvo a través de la relación hidrodinámica $J = \rho_{loc} * v_{loc}$; por lo que cualquier densidad considerada se relaciona diversos puntos en el diagrama fundamental. Como puede observarse de la Fig. 1, el modelo LAI reproduce las tres fases de tráfico existentes. La línea recta con pendiente positiva corresponde a la fase de flujo libre. La fase sincronizada forma una región bidimensional en medio del diagrama fundamental. Mientras que la fase de estancamientos amplios produce puntos distribuidos en la parte baja del diagrama fundamental. Es importante enfatizar que debido

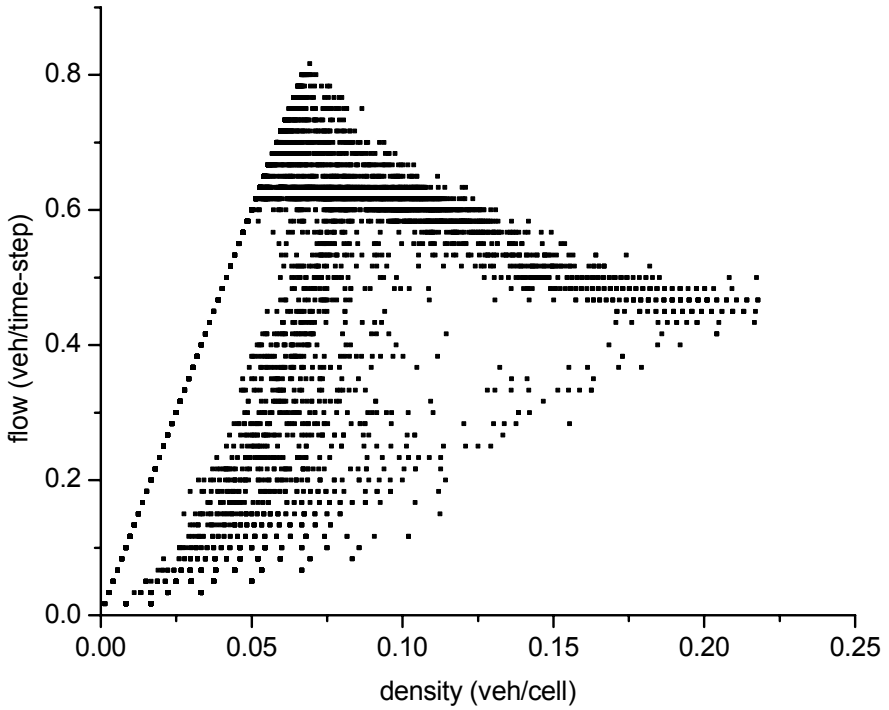


Figura 1. Flujo promedio de 1 min. con respecto a la densidad resultante del modelo propuesto para $\Delta x = 2{,}5m$, $R_d = 1{,}0$, $R_0 = 0{,}8$, $R_s = 0{,}01$, $v_s = 3$, $M = 2$, and $l_s = 2$.

a que solamente se consideran vehículos moviéndose en la medición de los detectores, la velocidad promedio local correspondiente a los estancamientos amplios se sobreestima y así, la densidad se subestima. Además, las densidades resultante de la simulación son menos distribuidas que aquellas correspondientes a encuentros empíricos, debido a un artefacto de discretización de las velocidades, las cuales determinan el límite superior de las densidades detectables.

Aunque la reproducción del diagrama fundamental es importante para validar el modelo, no es suficiente para identificar de manera aproximada las diferentes fases del tráfico. En la Fig. 2 se muestran los diagramas espacio-tiempo de las diferentes fases del flujo vehicular. Las figuras 2(a)-2(c) muestran las características espacio temporales para las fases correspondiendo a flujo libre, flujo sincronizado y estancamientos amplios, respectivamente. Cada columna vertical de puntos representan las posiciones instantáneas de los vehículos moviéndose hacia arriba; mientras que las columnas sucesivas de puntos representan las posiciones de los mismos vehículos en pasos de tiempo sucesivos. Los puntos negros

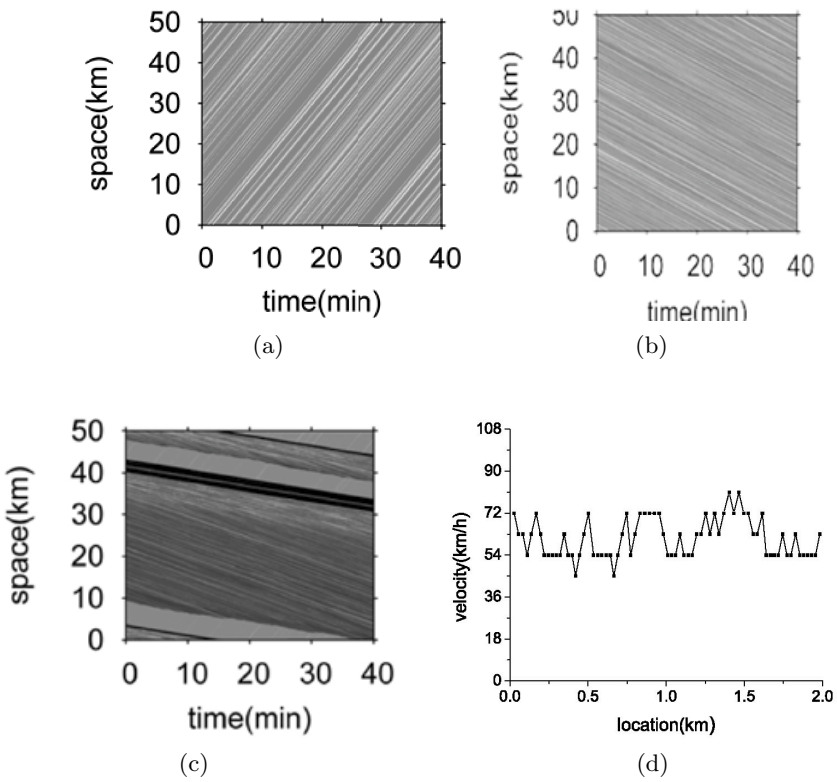


Figura 2. Diagrama espacio tiempo para diferentes fases del tráfico vehicular: flujo libre (a), flujo sincronizado (b) y flujo estancado (c), para valores de densidad de 14, 36, y 54 veh/km respectivamente (0.035, 0.090, and 0.135 veh/celda, respectivamente). (d) Imagen ampliada del flujo sincronizado para una densidad de 36 veh/km.

representan vehículos con velocidad cero. Como puede notarse de la Fig. 2(b) los vehículos se mueven con una velocidad menor (tono de gris más oscuro) que la correspondiente a la fase de flujo libre, pero no existen vehículos parados. Mientras que en la Fig 2(c) correspondiente a la fase de estancamientos amplios, se puede notar que la presencia de ondas denominadas stop-and-go y la presencia de vehículos con velocidad cero. Con la finalidad de verificar la existencia de la fase sincronizada en la Fig. 2(d) se presenta una imagen instantánea de una parte de la carretera seleccionada e forma aleatoria, que es resultado de una densidad inicial de 36 veh/km. En esta figura, los círculos sólidos representan vehículos moviéndose de la izquierda a la derecha. Nótese que la figura no corresponde a un periodo de transición y exhibe una velocidad promedio intermedia. Así, los puntos correspondientes al área de flujo sincronizado en el diagrama fundamen-

tal de la Fig. 1 no se atribuyen a los efectos promedio de las fluctuaciones fuertes, sino que son consecuencia de la relación especial velocidad-espaciamiento.

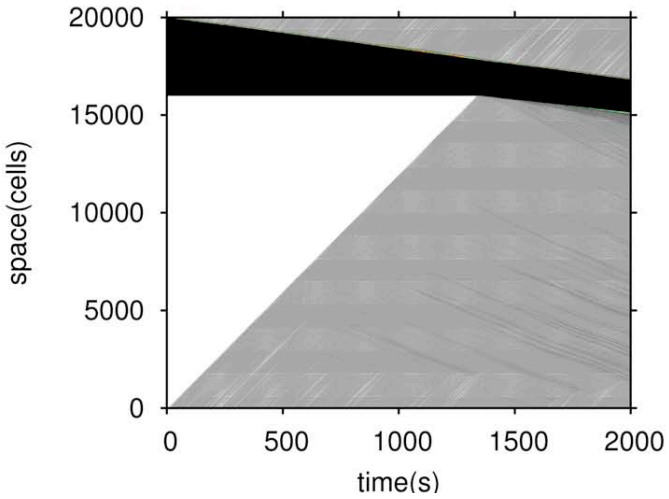


Figura 3. Diagrama espacio-tiempo para una densidad inicial de 40 veh/km. El eje horizontal representa el tiempo transcurrido en segundos (s), mientras que el eje vertical representa las posiciones de los vehículos (celdas). El tráfico vehicular inicia de un megaestancamiento, alineando todos los vehículos consecutivamente al final de la carretera

Por otra parte, con la finalidad de mostrar que el modelo es capaz de reproducir la velocidad de propagación de un congestionamiento, en la Fig. 3 se muestra el diagrama espacio-tiempo resultante de alinear los vehículos al final de la carretera, uno tras otro con velocidad cero, al inicio de la simulación. Con base en este diagrama, la velocidad hacia atrás del frente del estancamiento es aproximadamente 14.3 km/h, muy cercana a la observación de campo cuyo valor es alrededor de 15 km/h [28, 13]. La reproducción de la velocidad hacia atrás de los estancamiento, se debe al hecho que los vehículos que salen del frente de un estancamiento en la dirección del flujo vehicular son obligados a esperar una pequeña cantidad de tiempo, cuyo valor se determina como una función de su velocidad actual, es decir, aceleran lentamente. Por lo que la inclusión de una aceleración retardada en el modelo LAI permite además, reproducir la velocidad hacia atrás de un estancamiento.

Finalmente, uno de los objetivos principales del modelo LAI es evitar los desempeños de desaceleración irreales, de tal manera que el modelo debe ser capaz de reproducir el desempeño del flujo vehicular a un nivel macroscópico, con base en un desempeño vehicular microscópico aceptable. Para evidenciar este desempeño, en la Fig. 4(a) se despliegan las gráficas ampliadas correspon-

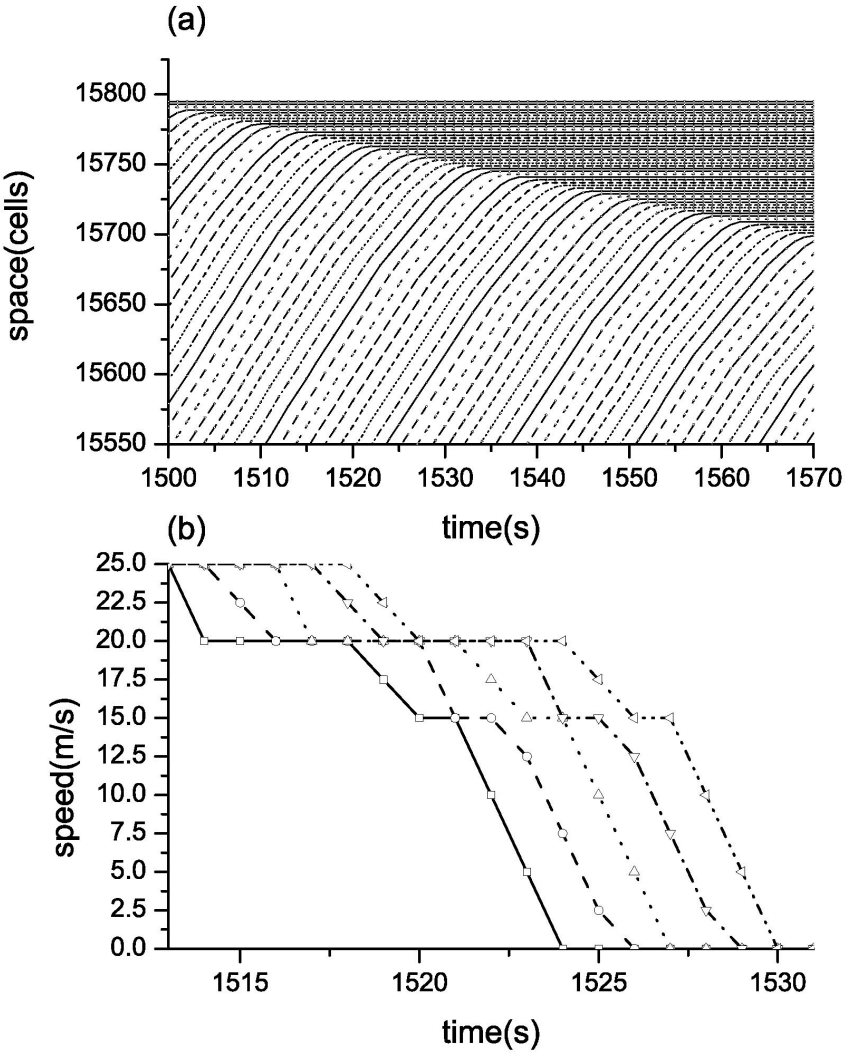


Figura 4. (a) Trayectorias vehiculares cuando aproximan el frente aguas arriba de un estancamiento vehicular (b) Variaciones de velocidad cuando los vehículos alcanzan un estancamiento vehicular.

dientes a las trayectorias de los vehículos cuando se aproximan al frente aguas arriba de un estancamiento vehicular, bajo las reglas de cambio del modelo LAI. Como puede notarse de esta figura, el decaimiento de la velocidad es en una forma gradual, debido a que los vehículos desaceleran de una manera oportuna, semejante al desempeño de los conductores en el mundo real. Estas variaciones de velocidad cuando los vehículos alcanzan el estancamiento vehicular pueden apreciarse de una manera más clara en la Fig. 4(b), que corresponde a las variaciones de velocidad de unos cuantos vehículos moviéndose consecutivamente. Por lo tanto, el modelo LAI suaviza el desempeño de desaceleración abrupto e irreal encontrado en la mayoría de los modelos para tráfico vehicular basados en AC y refleja el desempeño el desempeño del conductor como en el mundo real, donde los conductores decrementan sus velocidades de una forma suave bajo condiciones de conducción normales.

4. Conclusiones y trabajo futuro

4.1. Conclusiones

En este trabajo se describió un modelo para tráfico vehicular basado en AC que desarrollamos recientemente, el modelo LAI. El modelo introduce un nuevo conjunto de reglas que incorpora la definición de tres umbrales importantes requeridos por un vehículo seguidor para acelerar, desacelerar o mantener su velocidad, en una forma segura. Estos umbrales permiten determinar la acción más apropiada para un conductor con base en el estado del tráfico vehicular actual. Además, el modelo introduce en la definición de su dinámica capacidades de aceleración y desaceleración con un valor límite, cuya definición se deriva de principios de conducción segura y de acuerdo con prácticas de ingeniería de transporte, las características de los vehículos individuales y las reacciones humanas.

El modelo además incluye un parámetro para determinar la capacidad de frenado máximo que un vehículo puede aplicar en condiciones de emergencia, el cual puede sintonizarse de acuerdo al tipo de vehículo bajo consideración (automóviles, camionetas, autobuses, etc), como en el mundo real y sin necesidad de modificar el modelo actual. De tal manera que diferentes distancias de seguimiento seguro como una función de las características físicas de los vehículos (tamaño, peso) son posibles: capacidades de desaceleración menores implican distancias de seguimiento seguro mayores, como en la realidad. De tal manera que un vehículo puede desacelerar sin colisionar, en forma segura.

Resultados de simulación del modelo obtenidos previamente de un sistema con condiciones de frontera periódicas muestran que el modelo LAI puede suavizar el decaimiento de la velocidad cuando los vehículos se aproximan al frente aguas arriba de un estancamiento vehicular. Por lo tanto, el modelo evita el desempeño de desaceleración abrupto e irreal encontrado en la mayoría de los modelos de AC existentes en la literatura. Además, el modelo es también capaz de reproducir diversos encuentros empíricos incluyendo las tres fase del tráfico vehicular y la velocidad de propagación hacia atrás de un estancamiento. Cabe

mencionar que en [18], ya se mostró que el modelo también reproduce diferentes patrones espaciales del tráfico congestionados inducidos por un sistema con condiciones de frontera abierta con una rampa de entrada

Finalmente, el modelo presentado en este trabajo es muy simple. Con la perspectiva de tablas de búsqueda, el costo computacional no se incrementa substancialmente y el conjunto de reglas que definen la dinámica vehicular hace posible el uso de cómputo paralelo en una forma simple. Así esta característica de los modelos de AC se preserva.

4.2. Trabajo Futuro

Realizar la extensión del modelo para el estudio del tráfico vehicular de varios carriles, considerando vehículos heterogéneos, de distinta longitud y capacidades de frenado, los cuales requieren diferentes distancias de seguimiento seguro. Con el modelo extendido, se espera realizar una validación con datos de campo. Así como extender el modelo para simular tráfico vehicular de Sistemas de Carretera Automatizados.

5. Agradecimientos

Este trabajo fue soportado parcialmente por DGAPA-UNAM bajo el proyecto IN107909.

Referencias

[1] Bando, M., Hasebe, K., Nakayama, A., Shibata, A., & Sugiyama, Y. (1995). Dynamical model of traffic congestion and numerical simulation, *Physical Review E*, 51, 1035-1042.

[2] Barlovic, R., Santen, L., Schadschneider, A., & Schreckenberg, M. (1998). Metastable states in cellular automata for traffic flow, *The European Physical Journal B - Condensed Matter and Complex Systems*, 5(3), 793-800.

[3] Brackstone, M. & McDonald, M. (1999). Car-following: a historical review, *Transportation Research Part F: Traffic Psychology and Behaviour*, 2(4), 181-196.

[4] Carbaugh, J., Godbole, D., & Sengupta, R. (1997). automata, Tools for safety analysis of vehicle automation systems, In *Proceedings of the American Control Conference, 1997*, 2041-2045.

[5] Chowdhury, D., Santen, L., & Schadschneider, A. (2000). Statistical physics of vehicular traffic and some related systems, *Physics Reports*, 329(4-6), 199-329.

[6] Fukui, M. & Ishibashi, Y. (1996). Traffic flow in 1D cellular automaton model including cars moving with high speed, *Journal of the Physical Society of Japan*, 65(6), 1868-1870.

[7] Godbole, D. N. & Lygeros, J. (1994). Longitudinal control of the lead car of platoon, *IEEE Transactions on Vehicular Technology*, 43(4), 1125-1135.

[8] Helbing, D. (2001). Traffic and related self-driven many-particle systems, *Reviews of Modern Physics*, 73(4), 1067-1141.

[9] Helbing, D. & Schreckenberg, M. (1999). Cellular automata simulating experimental properties of traffic flow, *Physical Review E*, 59(3), R2505-R2508.

[10] Hsu, C. C., Lin, Z. S., Chiou, Y. C., & Lan, L. W. (2007). Dynamical model of traffic congestion and numerical simulation, *Journal of the Eastern Asia Society for Transportation Studie*, 7, 2502-2516.

[11] Jiang, R. & Wu, Q-S. (2005). First order phase transition from free flow to synchronized flow in a cellular automata model, *The European Physical Journal B - Condensed Matter and Complex Systems*, 46(4), 581-584.

[12] Kerner, B. S., Klenov, S. L., & Wolf, D. E (2002). Cellular automata approach to three-phase traffic theory, *Journal of Physics A: Mathematical and General*, 35(47), 9971-10014.

[13] Kerner, B. S. & Rehborn, H. (1996). Experimental Features and Characteristics of Traffic Jams, *Physical Review E*, 53(2), R1297-R1300.

[14] Knospe, W., Santen, L., Schadschneider, A. & Schreckenberg, M. (2000). Towards a realistic microscopic description of highway traffic, *Journal of Physics A: Mathematical and General*, 33(48), L477-L485.

[15] Krauss, S., Wagner, P., & Gawron, C. (1997). Metastable states in a microscopic model of traffic flow, *American Physical Society*, 55(5), 5597-5602.

[16] Lee, H. K., Barlovic, R., Schreckenberg, M., & Kim, D. (2004). Mechanical restriction versus human overreaction triggering congested traffic states, *Physical Review Letters*, 92, 238702-1–238702-4.

[17] Li, X. B., Wu, Q-S., & Jiang, R. (2001). Cellular automaton model considering the velocity effect of a car on the successive car, *Physical Review E*, 64, 066128-1–066128-4.

[18] Lárraga, M. E. & Álvarez-Icaza, L. (2010). Cellular automaton model for traffic flow based on safe driving policies and human reactions, *Physica A*, 389(23), 5425-5438.

[19] Lárraga, M. E., del Río, J. A., & Schadschneider, A. (2004). New kind of phase separation in a CA traffic model with anticipation, *Journal of Physics A: Mathematical and General*, 37, 3769-3782.

[20] Lárraga, M. E., del Río, & Álvarez-Icaza, L. (2005). Cellular automata for One-Lane traffic flow modeling, *Transportation Research Part C: Emergent Technologies*, 13(1), 63-74.

[21] Álvarez-Icaza, L. & Horowitz, R. (1999). Safe Platooning in Automated Highway Systems, Part I: Safety Regions Design, *Vehicle System Dynamics*, 32(1), 23-56.

[22] Maerivot, S. & De Moor, B. (2005). Cellular automata models of road traffic, *Physics Reports*, 419, 1-64.

[23] Nagel, K. & Schreckenberg, M. (1992). A cellular automata model for traffic flow, *Journal of Physique I*, 2, 2221-2229.

[24] Prigogine, I. & Herman, R. (1971). *Kinetic Theory of Vehicular Traffic*, American Elsevier Pub. Co.

[25] Schadschneider, A. (2006). Cellular automata models of highway traffic, *Physica A: Statistical Mechanics and its Applications*, 372, 142-150.

[26] Schadschneider, A., Pöschel, T., Kuhne, R., Schreckenberg, M. & Wolf, D. E. (2007). *Traffic and Granular Flow '05*, Springer Verlag.

[27] Pline, J. L. (1999). *Traffic Engineering Handbook, 5th Edition*, Institute of Transportation Engineers.

[28] Treiterer, J. (1975). Investigation of traffic dynamics by aerial photogrammetry techniques, *Technical Report PB 246 094*, Ohio State University.

Estudio de la dinámica y análisis de complejidad de la regla espiral

Paulina Anaid León Hernández, Rogelio Basurto Flores

Centro de Investigación y de Estudios Avanzados
Instituto Politécnico Nacional, México.
{pleon,rbasurto}@computacion.cs.cinvestav.mx

Resumen La regla espiral de autómatas celulares hexagonales ha resaltado por sus similitudes con Life, dando paso a su estudio con resultados alentadores. El objetivo de este documento es ahondar en dos de las vertientes de estudio que se han venido dando: por un lado, un estudio experimental llevado acabo con simulaciones computacionales que entregue resultados sobre el comportamiento del sistema y las interacciones existentes entre sus partículas; y por otro lado, un análisis exhaustivo de la regla mediante los diagramas de de Bruijn para estudiar las interacciones locales de sus células.

1. Introducción

La regla espiral pertenece a la familia de los autómatas celulares hexagonales totalísticos y fue introducida por Andrew Adamatzky y Andrew Wuensche en 2005 [3]. Esta regla muestra un comportamiento complejo a través de una dinámica con un alto grado de emergencia, produciendo partículas de especial interés para la computación teórica, tales como *gliders*, *glider-guns* y *still-lifes*.

La dinámica de la regla ha probado ser capaz de realizar computaciones mediante la colisión de sus partículas [4] [5], de manera similar a como se realizó en los años 80's con la regla *Life* [7]; además, se ha mostrado que se puede crear una lógica universal [9], lo que permite pensar en implementaciones más complejas, por lo cual es conveniente un estudio más profundo de la regla.

Dado lo anterior, el estudio de la regla ha tomado dos vertientes, la primera es la experimental, auxiliada por simuladores como el DDLab o Spiral simulator [24] [8], en los cuales se configura el espacio de evolución con condiciones iniciales aleatorias y propuestas, donde se espera encontrar comportamientos similares a *Life* [7], ejemplo de ellos se pueden ver en [9] y [12]; y la segunda, es la analítica, apoyada por herramientas matemáticas para el estudio de la interacción entre células vecinas, como lo son los diagramas de *de Bruijn*.

El presente trabajo, habla sobre los resultados encontrados para ambos estudios, por lo que el documento está dividido de la siguiente manera: la segunda sección presenta el análisis de la regla mediante los diagramas de *de Bruijn*; la tercera sección nos habla del estudio experimental y de las estructuras complejas encontradas; para finalmente, en la cuarta sección presentar el uso de dichas partículas para demostrar la existencia de una lógica universal mediante compuertas lógicas básicas.

2. Diagramas de *de Bruijn*

La necesidad de entender el comportamiento general de un autómata celular ha dado paso al uso de teorías matemáticas, tales como la teoría de campo promedio o la teoría de estructura local; sin embargo, para comprender el comportamiento de la interacción entre células, herramientas como los diagramas de *de Bruijn* han mostrado ser de utilidad.

En los años 80's Harold V. McIntosh utilizó los diagramas de *de Bruijn* para analizar la familia de autómatas celulares de una dimensión [19], y posteriormente hizo el análisis para el autómata del juego de la vida en dos dimensiones [17], [18].

En la presente sección se hablará del estudio realizado a la regla Espiral con los diagramas de *de Bruijn* y los resultados obtenidos.

2.1. Notación básica

Los diagramas de *de Bruijn* son la representación mediante grafos dirigidos de las secuencias de *de Bruijn*. Las secuencias de *de Bruijn* nacen a partir del siguiente problema:

Problema 1. Dado $m+1$ símbolos y un entero positivo n, encontrar un algoritmo para generar una secuencia de símbolos que tenga una longitud mínima y además, que cuando sean colocados en un círculo contengan como subsecuencias de símbolos consecutivos todas las secuencias de símbolos de longitud n.

Este problema ha sido resuelto en más de una ocasión desde 1894, cuando A. de Rivière encontró una solución para $m = 1$; posteriormente con las contribuciones de C. Flye Sainte-Marie y W. Mantel en la misma época, después en los años 30 M. H. Martin demostrando la existencia de las secuencias para cualquier m y n dando un algoritmo para la creación de tales secuencias, y finalmente *de Bruijn* solucionó el problema mediante grafos para $m = 1$, dejando ver como factible la extensión de su trabajo a cualquier m [21].

Definicón 1. Se define como una secuencia de *de Bruijn* a toda secuencia de longitud $(m+1)^n$ que permite generar $(m+1)^n$ subsecuencias de longitud $n-1$ al colocarse en una circunferencia.

Como se puede apreciar en la figura 1 los elementos de la secuencia están ordenados de tal manera que no existe un inicio ni un fin, con lo cual se puede obtener el número total de subsecuencias, mismas que están encerradas en rectángulos rojos y verdes; las subsecuencias generadas que tengan una longitud de $(n+1)^n$, se llamaran secuencias de *de Bruijn*. Los diagramas de *de Bruijn* permiten dar una solución al problema 1.

Para generar un diagrama es necesario obtener todas las secuencias de longitud $(m+1)$ posibles, con los $(m+1)$ símbolos; éstas secuencias se definen como:

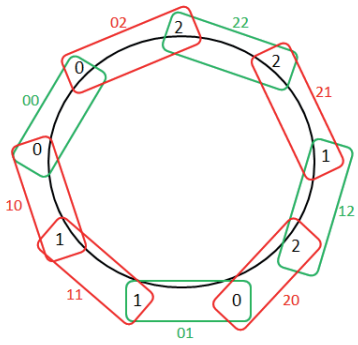


Figura 1. Subsecuencias para la secuencia 221201100.

S_A		S_B		Subsecuencia
0	0	0	0	000
0	0	0	1	001
0	1	1	0	010
0	1	1	1	011
1	0	0	0	100
1	0	0	1	101
1	1	1	0	110
1	1	1	1	111

Cuadro 1. Tabla de subsecuencias formadas mediante la intersección para $m = 1$ y $n = 3$

Definicón 2. Un nodo es una subsecuencia de longitud n que representa un número en base $(m+1)$. Existen en un diagrama de *de Bruijn* $(m+1)^n$ nodos.

Por ejemplo, para $m = 1$ se tienen secuencias de longitud 2 con 2 símbolos; para ello se usa una numeración en base 2, de longitud 2, esto es: 00, 01, 10 y 11; estas secuencias serán los nodos dentro del diagrama; y las aristas representarán la forma de unir estas secuencias y son definidas como:

Definicón 3. La unión de dos nodos a través de un traslape de símbolos de las secuencias de nodos, forma una arista, es decir, que ambas secuencias compartan un símbolo para poder ser unidas y formar una subsecuencia de longitud n.

En la tabla 1 se muestran los diferentes traslapes que existen para el caso de $m = 1$ y $n = 3$, donde en color gris se visualizan los símbolos que comparten las secuencias y que representan el traslape. Finalmente, en la figura 2 se representa el diagrama de *de Bruijn* para el ejemplo de la tabla 1.

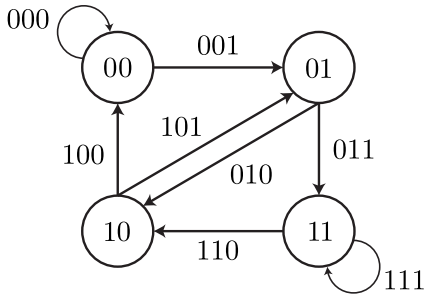


Figura 2. Diagrama de de Bruijn para $m = 1$ y $n = 3$.

El caso de $m = 1$ y $n = 3$ es de sumo interés debido a la relación existente con los autómatas celulares lineales de 2 estados y radio de vecindad 1, dicha relación se presenta en un autómata celular ya que son secuencias de células, donde las células pueden tener algún estado y así una célula forma parte de tres vecindades al mismo tiempo, tanto de la vecindad en la que es central, como de las que es vecino izquierdo o derecho, de tal manera se observa un traslape entre células, mismo que en los diagramas de *de Bruijn* aparece y es por esto que son utilizados para analizar el comportamiento del autómata de manera local.

2.2. Análisis de la regla espiral a través de *de Bruijn*

La regla espiral es una regla de AC de dos dimensiones donde cada célula tiene forma de hexágono; además, cada célula tiene 6 vecinos inmediatos y puede presentar uno de tres estados $\{0, 1, 2\}$; la vecindad es mostrada en la figura 3; la regla espiral es totalística, lo que quiere decir que para que una célula en un tiempo t, evolucione al tiempo $t + 1$, dependerá de su propio estado así como el de sus vecinos.

	j							
	0	1	2	3	4	5	6	7
0	0	1	2	1	2	2	2	2
1	0	2	2	1	2	2	2	
2	0	0	2	1	2	2		
i 3	0	2	2	1	2			
4	0	0	2	1				
5	0	0	2					
6	0	0						
7	0							

Cuadro 2. Matriz de evolución de la regla espiral.

La matriz de transición, que se muestra en la tabla 2, está dada en función del número de células en estado 1 y 2 dentro de la vecindad a evaluar; siendo las columnas el número de células en estado 1 y las filas el número de células en estado 2; por ejemplo, al tener una vecindad en un tiempo t con 3 células en estado 1, y 2 células en estado 2, la célula central evolucionará y para el tiempo $t+1$, pasará a estado 1. El número de células en estado 0 se obtiene mediante la operación $n_0 = 7 - (n_1 + n_2)$, donde n_0, n_1 y n_2 son el número de células en estado 0, 1 y 2, respectivamente.

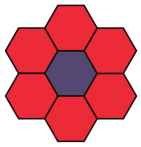


Figura 3. Vecindad para la regla espiral.

En los estudios que se han hecho a la regla anteriormente en [3], [4], [24], [9], [12] se ha observado que existe una diversidad de partículas tanto estáticas como movibles, que al interactuar muestran un comportamiento complejo.

La problemática radica en que las búsquedas que se han hecho de partículas no han sido de manera sistemática, para ello en el presente documento se realizará un análisis a través de diagramas de *de Bruijn* con el fin de conocer, primero el tipo de interacción entre células y posteriormente entre los conjuntos de células para así encontrar patrones bien definidos que formen partículas concretas.

2.3. Diagrama de *de Bruijn* en $2D$

Para poder realizar un diagrama de *de Bruijn* es importante conocer el número de estados que tiene el autómata y el número de vecinos que tendrá la subvecindad.

Una subvecindad se puede definir por sus características:

- *Forma:* Depende de la vecindad original, dado que al traslaparse al menos una célula de la subvecindad con otra célula de otra subvecindad se forma una vecindad completa.
- *Número de células*: Es el número n de células que contiene la subvecindad, cada célula tiene una etiqueta que corresponde a la posición que ocupa esa célula en dicha subvecindad; esto es importante para que se pueda definir de manera correcta las condiciones de traslape.
- *Número de células que traslapan:* Son el número n_t de células que al unirse con otra subvecindad para formar una vecindad completa, ocupan "la misma posición"; y debe cumplir la condición de $n_t \geq 1$.

- *Condición de traslape:* Para que dos subvecindades se puedan traslapar, las células que traslapan en A y las células que traslapan en B deben tener el mismo estado, y una de las células que se traslapan deberá ser la célula central.

Cabe mencionar que para cada subvecindad con sus diferentes número de células traslapándose se formaran diagramas diferentes.

La vecindad de la regla espiral se compone de 7 células, figura 3, donde al analizarla se observó que una de las subvecindades, posiblemente la única para este caso en partícular, que cumple con las características antes mencionadas es mostrada en la figura 5 inciso 1; donde el número de células de la subvecindad está dado por $n = 4$, el número de células que traslapan es $n_t = 1$ y las condición de traslape está dada por: $C = B'$.

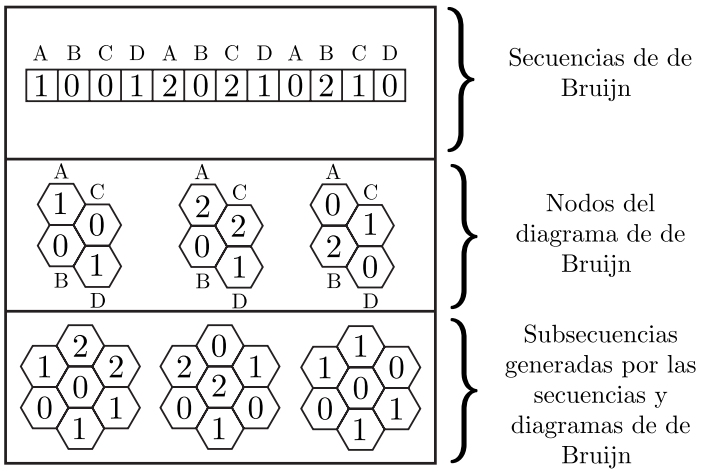


Figura 4. Relaciones entre diagramas y secuencias con un autómata celular hexagonal.

Análogamente a lo que se hizo para relacionar los diagramas de *de Bruijn* con un autómata celular unidimensional, los nodos son subsecuencias de símbolos, los cuales representan, los estados 0, 1, 2, de longitud 4, mismos que en la figura 4 se observan como A, B, C y D; y se derivan de una secuencia de nodos, donde el traslape entre dos nodos, da como resultado una vecindad.

Si se tienen dos nodos como los de la figura 5 incisos 1 y 2, entonces la condición de traslape se cumplirá cuando el símbolo C del nodo A sea igual al símbolo B del nodo B. Esta relación se visualiza en una matriz de *de Bruijn*.

Definicón 4. Una matriz de *de Bruijn* es una matriz booleana que representa las aristas, la existencia de un traslape entre dos nodos, de un diagrama de *de*

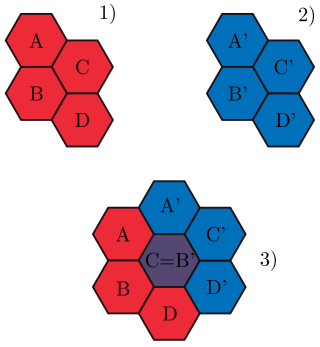


Figura 5. Subvecindades para la regla espiral. Los incisos 1) y 2) corresponden a las subvecindades A) y B) respectivamente; el inciso 3) representa el traslape de la célula central.

Secuencia de Símbolos	Dígito en sistema base 3
00	0
01	1
02	2
10	3
11	4
12	5
20	6
21	7
22	8

Cuadro 3. Relación entre una secuencia de símbolos y la numeración base 3.

Bruijn a través de la relación entre las filas con las columnas.

Donde las filas, o nodos A, y las columnas, o nodos B, se relacionan solamente de $A \to B$.

Con la finalidad de hacer más sencilla la manipulación de las subvecindades, se utiliza un sistema base 3 para poder representar a las secuencias de las subvecindades de una manera más compacta; para ello se tomaron 2 células, y sus posibles combinaciones, lo cual, da como resultado un dígito en numeración base 3, para poder tener la representación de las 4 células que tiene la subvecindad, se concatenaron dos dígitos que representan una cadena de símbolos de una subvecindad; la tabla 3 muestra la relación entre la secuencia de símbolos y el dígito del sistema base 3.

Para representar toda la cadena se divide en dos partes, las células A, B y las C, D de esta manera, la secuencia 0221 se divide en 02 y 21, entonces convirtiendo en decimal la numeración base 3, será: 27.

2.4. Trabajando con la matriz de *de Bruijn*

La matriz de *de Bruijn* muestra todas las posibles vecindades que pueden generarse al traslapar dos subvecindades, por lo tanto se puede realizar una evolución para conocer cual será el estado de la célula central, célula traslapada, en un tiempo $t + 1$.

El comportamiento que muestra la célula central al evolucionar es representado mediante otra matriz de *de Bruijn*; a esta nueva matriz se le puede aplicar "filtros" para que la información presentada sea más concreta y específica. En este sentido, los diferentes filtros que pueden aplicarse a la matriz ya evolucionada pueden ser tan variados como lo que se desee encontrar; los filtros aplicados a la regla espiral y que son de un interés general en los autómatas celulares son:

- *Permanencia*, muestra todas las relaciones entre nodos donde, después de aplicar la regla de evolución a las vecindades formadas, la célula central continua en el estado que tenía antes de aplicar la regla.
- *Corrimiento*, muestra todas las relaciones entre nodos donde, después de aplicar la regla de evolución a las vecindades formadas, la célula central presenta el estado que tenía un vecino antes de aplicar la regla de evolución.

Para el caso de la regla espiral las matrices de permanencia son 3, debido a los tres diferentes estados que pueden "permanecer"; el corrimiento que se presenta es a partir del vecino suroeste a la célula central, dicho de otra forma, el estado de la célula central después de evolucionar deberá ser el mismo que el de su vecino suroeste antes de la evolución.

Cabe aclarar que no todas las relaciones serán de interés debido a que no es posible la formación de patrones concretos mediante secuencias de nodos, pero si es notable un conjunto de posibilidades más reducido a estudiar; además, el estudio realizado fue para una evolución, sin embargo, si se desea conocer para n evoluciones, se deben formar vecindades que tengan un radio $r = n$.

Las relaciones que se pueden ver en la matriz de *de Bruijn* se dan por parejas, no obstante, es posible realizar una secuencia de nodos, la cual se define como:

Definicón 5. Si $s_1, s_2 \dots s_n$ es una secuencia donde s_i representa un nodo entonces s_i, s_{i+1} deberá cumplir con la condición de traslape.

De estas secuencias de nodos se pueden realizar los diagramas de *de Bruijn*. Dado que el diagrama completo es muy amplio se recomienda hacer solo secciones reducidas del mismo, sobre todo de aquellos nodos que pueden representar algo importante dentro de la regla, como los estados activadores.

3. Dinámica compleja en la regla espiral

La regla espiral tiene un universo de partículas complejas que emergen de la evolución de un estado inicial aleatorio en el espacio de evoluciones hexagonal. En esta sección se presenta un número de nuevas estructuras en la regla espiral.

Eventualmente, dichas partículas llegan a ser útiles para el desarrollo de nuevas configuraciones que permitan la computación.

3.1. Partículas movibles: *gliders*

La regla espiral tiene una gran diversidad de partículas que se desplazan en el espacio de evolución; las cuales son conocidas como *gliders*, en la literatura de los AC son descritos por un número de propiedades particulares, tales como: masa, volumen, período, desplazamiento y velocidad.

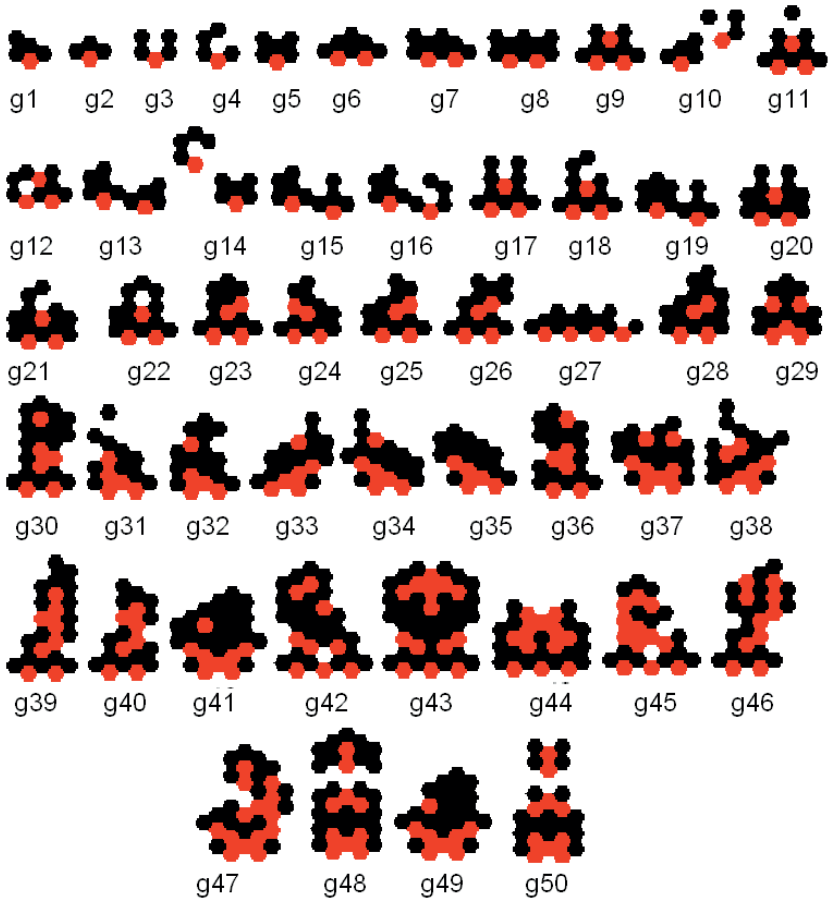


Figura 6. Gliders en la regla espiral; el estado 2 es representado en negro, estado 1 en rojo y el estado 0 en blanco.

Actualmente se han encontrado 50 gliders con sus respectivas propiedades [12]. La figura 6 muestra todos los gliders conocidos en la regla espiral, se enu-

meran de las formas básicas o primitivas, hasta los compuestos y con extensiones. Experimentalmente se ha observado que los gliders con mayor masa no tienen alta probabilidad de emerger de alguna configuración inicial aleatoria, ni de sobrevivir a muchas generaciones, debido a su alta sensibilidad a pequeñas perturbaciones.

La tabla 4 muestra las propiedades generales de los gliders en la regla espiral; donde la *masa* representa el número de células en estado 1 y 2 dentro del volumen del glider, si el glider tiene más de una forma, se toma la forma más grande; *período* es el número de evoluciones necesarias para que el glider regrese a su forma original; *desplazamiento* es el número de células que avanza el glider por período y finalmente la *velocidad* de las partículas es calculada como el período entre desplazamiento.

glider	masa	periodo	desplazamiento	velocidad
g_1	5	1	1	1
g_2	5	2	2	1
g_3	5	2	2	1
g_4	5	2	2	1
g_5	6	1	1	1
g_6	8	1	1	1
g_7	9	1	1	1
g_8	10	1	1	1
g_9	10	1	1	1
g_{10}	10	4	4	1
g_{11}	11	1	1	1
g_{12}	11	4	4	1
g_{13}	11	4	4	1
g_{14}	11	4	4	1
g_{15}	11	4	4	1
g_{16}	11	4	4	1
g_{17}	12	1	1	1
g_{18}	12	2	2	1
g_{19}	12	4	4	1
g_{20}	14	2	2	1
g_{21}	14	2	2	1
g_{22}	14	2	2	1
g_{23}	15	2	2	1
g_{24}	16	2	2	1
g_{25}	16	2	2	1

Cuadro 4. Las propiedades de los gliders (primera parte de dos tablas, continua en la tabla 5).

glider	masa	periodo	desplazamiento	velocidad
g_{26}	16	2	2	1
g_{27}	17	2	2	1
g_{28}	17	4	4	1
g_{29}	17	4	4	1
g_{30}	18	4	4	1
g_{31}	18	4	4	1
g_{32}	19	4	4	1
g_{33}	19	8	8	1
g_{34}	20	8	8	1
g_{35}	22	4	4	1
g_{36}	23	4	4	1
g_{37}	24	4	4	1
g_{38}	25	4	4	1
g_{39}	25	8	8	1
g_{40}	26	8	8	1
g_{41}	29	4	4	1
g_{42}	29	8	8	1
g_{43}	31	4	1	4
g_{44}	31	4	1	8
g_{45}	32	4	1	4
g_{46}	32	4	1	4
g_{47}	33	8	1	4
g_{48}	36	4	1	4
g_{49}	43	4	1	4
g_{50}	47	4	1	4

Cuadro 5. Propiedades de los gliders (parte final).

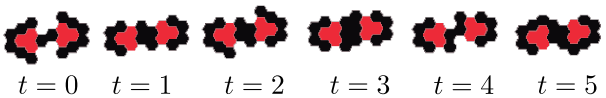
Con la diversidad de los gliders, se puede definir una clasificación por familias o especies. De esta manera la tabla 6 presenta tres principales tipos de especies de gliders en la regla espiral.

Especie	glider
Primitivos o básicos	g_1, g_2, g_3, g_4, g_5, g_{29}
Compuestos	g_6, g_7, g_9, g_{10}, g_{12}, g_{13}, g_{14}, g_{15}, g_{16}, g_{17}, g_{19}, g_{27}, g_{35}
Con extensiones	g_8, g_{11}, g_{18}, g_{20}, g_{21}, g_{22}, g_{23}, g_{24}, g_{25}, g_{26}, g_{28}, g_{30}, g_{31}, g_{32}, g_{33}, g_{34}, g_{36}, g_{37}, g_{38}, g_{39}, g_{40}, g_{41}, g_{42}, g_{43}, g_{44}, g_{45}, g_{46}, g_{47}, g_{48}, g_{49}, g_{50}

Cuadro 6. Especies de gliders en la regla espiral.

(a) (b)

Figura 7. Configuraciones still life en la regal espiral.



$t = 0$ $t = 1$ $t = 2$ $t = 3$ $t = 4$ $t = 5$

(a)

$t = 0$ $t = 1$ $t = 2$ $t = 3$ $t = 4$ $t = 5$

(b)

$t = 0$ $t = 1$ $t = 2$ $t = 3$

(c)

$t = 0$ $t = 1$ $t = 2$ $t = 3$

(d)

$t = 0$ $t = 1$ $t = 2$

(e)

Figura 8. Configuraciones de osciladores en la regla espiral.

3.2. Partículas estáticas: *Still-life*

La regla espiral tiene partículas estáticas primitivas conocidas como *still life*, [18]. Dichas partículas pueden vivir en el espacio de evoluciones sin alteraciones. La figura 7 muestra estas partículas.

El *still life* 'e1' (Fig. 7a) tiene una masa de 12 células; mientras que el segundo *still-life* 'e2' (Fig. 7b) tiene una masa de 13. El segundo puede ser usado como un contador binario para un dispositivo de memoria [2, 26], produciendo una familia de configuraciones del *still-life*.

Una característica importante es que ambas configuraciones de *still-life* trabajan como "eaters". Un eater es un tipo de *still-life* con la capacidad de eliminar los glides no importando la dirección de donde provengan. Este tipo de partícula, eventualmente llega a ser útil para el control de número de señales o valores en un proceso específico.

3.3. Partículas estáticas periódicas: *Osciladores*

Los osciladores son capaces de emerger en la regla espiral con facilidad; por lo que es posible ver una interesante diversidad de partículas estáticas periódicas. Estas son frecuentemente una composición de configuraciones de *still life*, alternándose en ON y OFF periódicamente.

La figura 8 presenta seis tipos de osciladores en la regla espiral. Estos son compuestos por configuraciones fundamentales de *still life*; todos ellos oscilan y cambian muy poco sus valores y sus estructuras.

oscillator	mass	period
o_1	20	6
o_2	20	6
o_3	24	4
o_4	24	4
o_5	20	3

Cuadro 7. Propiedades de los osciladores: o1 (a), o2 (b), o3 (c), o4 (d), y o5 (e).

La tabla 7 muestra propiedades generales para cada oscilador de la figura 8. Además estos osciladores son capaces de trabajar como configuraciones de eaters.

3.4. Glider guns

Una de las características más notables en la regla espiral, es la diversidad de *glider gun* que pueden aparecer en el espacio de evoluciones. Un *glider gun* es una configuración que genera gliders periódicamente. En la literatura de los

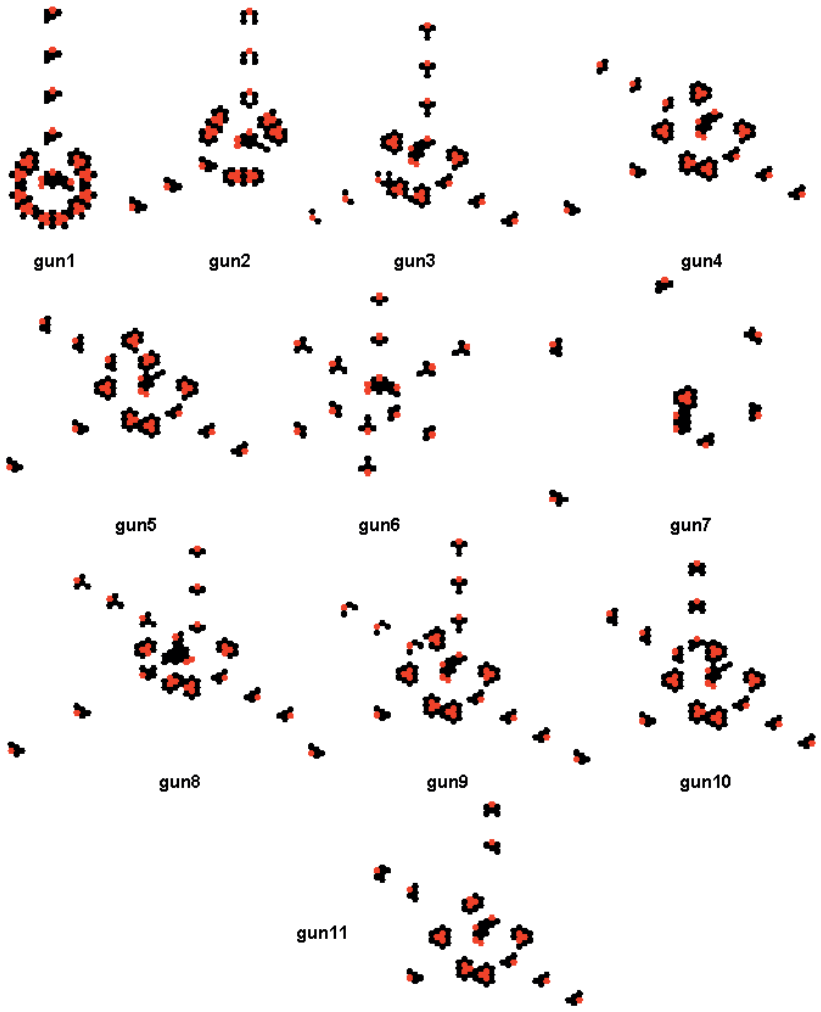


Figura 9. Glider gun fijos en la regal espiral. Un número de guns no naturales son presentados.

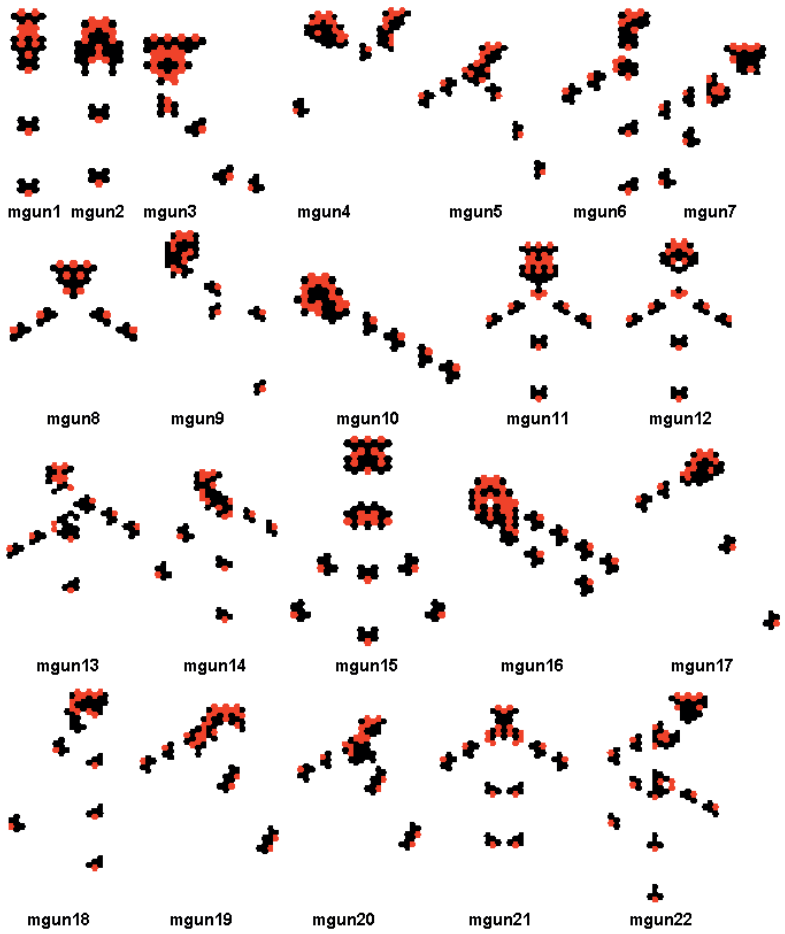


Figura 10. Glider Gun movibles en la regla espiral.

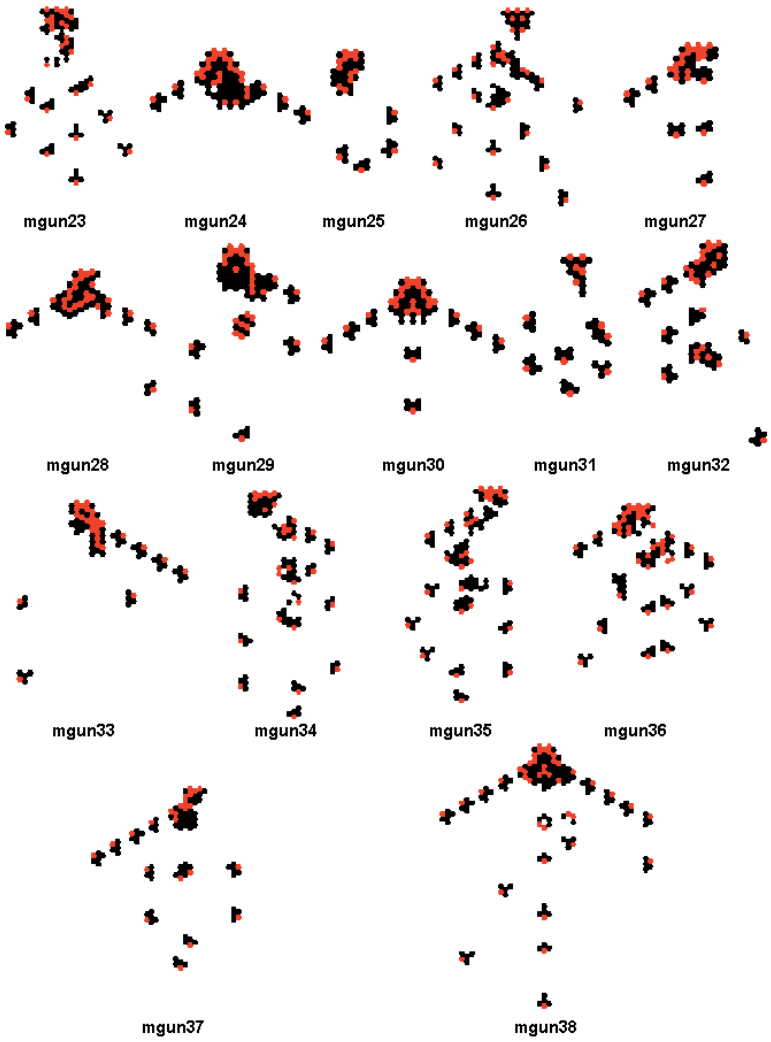


Figura 11. Glider gun movibles en la regla espiral.

autómatas celulares, la existencia de un glider gun representa la solución al problema de crecimiento ilimitado [7].

La regla espiral tiene dos tipos de glider guns: fijos y en movimiento. Un gun fijo no puede cambiar de posición dentro del espacio de evolución; mientras un gun movible puede viajar a lo largo del espacio en una dirección generando gliders.

La tabla 8 y Fig. 9 muestran las propiedades generales y la configuracón de cada gun fijo en la regla espiral. Los guns producidos con mayor frecuencia en la regla espiral con condiciones iniciales aleatorias son los gun6 y gun7. Estos tienen una alta y baja frecuencia para generar gliders g_2 y g_1 respectivamente. Mientras el gun6 produce seis g_2 cada 6 generaciones, el gun7 emite seis g_1 cada 22 generaciones (ver la tabla8).

gun	producción	frecuencia	periodo	volumen	gliders emitidos
gun1	g_1	1	6	15×15	1
gun2	g_1, g_3	2	6	15×15	2
gun3	g_1, g_2, g_3	3	6	14×15	3
gun4	$3g_1$, $2g_2$	5	12	16×17	3
gun5	$5g_1$	5	12	19×17	3
gun6	$6g_2$	6	6	8×9	6
gun7	$6g_1$	6	22	12×12	6
gun8	$3g_1$, $4g_2$	7	12	14×14	4
gun9	$3g_1$, $2g_2$, $2g_4$	7	12	15×17	4
gun10	$5g_1$, $2g_5$	7	12	15×15	4
gun11	$13g_1$, $4g_5$	17	30	15×17	4

Cuadro 8. Propiedades de los glider gun fijos en la regla espiral.

También la regla espiral tiene un número de glider guns movibles, generalmente están formados por estructuras complejas. No obstante, los guns son muy sensibles a cualquier perturbación, con su destrucción por consecuencia. Las figuras 10 y 11 presentan la gran diversidad de glider gun movibles en la regla espiral, existiendo 38 tipos diferentes.

4. Computación en la regla espiral

Mediante la manipulación de las partículas básicas que presenta la regla espiral es posible implementar computación lógica universal, por medio de choques de partículas, como se mostrará en la presente sección.

La característica de los glider guns de este autómata que les permite lanzar gliders en 6 direcciones puede llegar a ser una ventaja, pues se podrían procesar 6 señales al mismo tiempo, no obstante, por ahora sólo se considerará un solo

flujo; los flujos de gliders que no se utilicen serán eliminados mediante un eater E_1. Un glider gun limitado en 5 de sus 6 flujos se puede apreciar en la figura 12. Esto mismo se puede extender para el glider gun G_2.

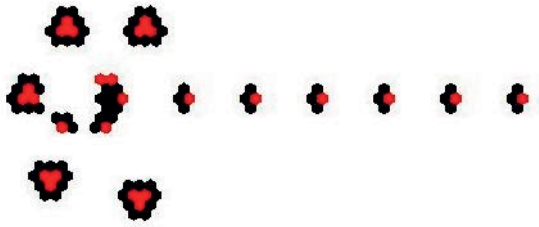


Figura 12. Glider gun G_1 con 5 flujos eliminados.

De la misma manera que en Life [7], en la regla espiral se representan unos lógicos, '1', con la presencia de gliders y ceros lógicos, '0', con la ausencia de los mismos. Así, utilizando el glider-gun G_1 se puede representar una cadena constante de información con 1's. La manera de cambiar esta cadena y poder hacerla más diversa es mediante el glider gun G_2. Al lanzar los gliders a una frecuencia más baja es posible modificar el flujo de gliders del G_1 para generar cadenas de unos y ceros, un ejemplo de esto se muestra en la figura 13.

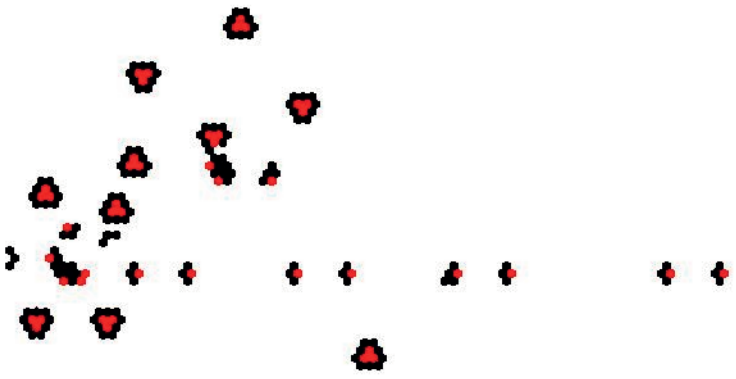


Figura 13. Flujo de gliders modificado.

La sincronización entre glider guns es una de las bases primordiales para la creación de compuertas lógicas, no solo en la regla espiral, sino en cualquier autómata, pues la computación está basada en las colisiones entre las partículas y la reacción que estas colisiones producen [5], [15], [**?**], [13], otros expererimentos recientes demuestran la implementacion de computaciones a través de la interacción en la propagación de patrones [16], [15]. Después de observar la regla espiral se notó una similitud entre las colisiones existentes entre gliders, dichas colisiones se muestran en la figura 14. En la imágen se observan tres colisiones diferentes, la colisión del inciso A tiene como reacción el cambiar de dirección el glider proveniente del suroeste; en el inciso B se observa la aniquilación de ambos gliders; el resultado de la colisión del inciso C es la eliminación de un solo glider, mientras el otro sigue su curso normalmente. Estas y otras colisiones se utilizan como función de procesamiento para la construcción de las compuertas lógicas.

Otra característica más a considerar son las partículas "excedentes", es decir, gliders que se generan y no son útiles para la computación propuesta, dichos gliders son eliminados mediante eaters. Finalmente, para construir una compuerta lógica y que su resultado sea fácilmente verificable es necesario construir un flujo de entrada que contenga los bits requeridos para comprobar la tabla de verdad de la compuerta implementada.

Dado que no es posible predecir el comportamiento general del autómata, se realizaron una serie de pruebas empíricas para poder encontrar la implementación de las compuertas lógicas; ésto con ayuda de un simulador que permitía la manipulación de los estados mediante una interfaz gráfica que fue desarrollado para este fin.

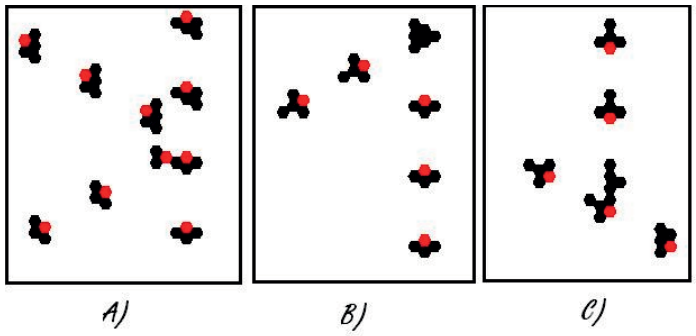


Figura 14. Tipos de colisiones entre gliders.

Las compuertas implementadas mediante la regla espiral son: AND, OR y NOT así, con estas compuertas la regla espiral posee una lógica universal. A lo largo de la búsqueda se logró encontrar otras compuertas más, estas son: NOR, XOR y XNOR.

A	S
0	1
1	0

Cuadro 9. Tabla de verdad de la compuerta NOT.

[th]

Entradas		Salidas				
A	B	AND	OR	NOR	XOR	XNOR
0	0	0	0	1	0	1
0	1	0	1	0	1	0
1	0	0	1	0	1	0
1	1	1	1	0	0	1

Cuadro 10. Tablas de verdad para las compuertas AND, OR, NOR, XOR y XNOR.

t

El orden cronológico de creación es el siguiente: NOT, AND, NOR y XNOR. Debido a que se tenían las compuertas NOR y XNOR se hizo uso de la compuerta NOT para de esta manera crear las compuertas OR y XOR. La interpretación en el autómata celular de las compuertas lógicas se presenta a continuación:

- La compuerta NOT está formada por dos glider-gun G_1 y un glider-gun G_2, este último se utiliza para modificar la señal de entrada de la compuerta; se observa la compuerta en la figura 15, donde A es el glider-gun que tiene el flujo de entrada y S es el flujo de salida de la compuerta. La señal de entrada es: 1100110; por lo que su flujo de salida es: 0011001.
- En la compuerta AND, que se puede ver en la figura 16, se utiliza un glider-gun G_1 por cada señal de entrada, de igual manera, para modificar el flujo de gliders se requiere un glider-gun G_2 por cada flujo de entrada; el flujo A es: 1111010; para la entrada B se utiliza: 1101100; el resultado de aplicar la operación AND se muestra con la salida S y es: 1101000.
- Para construir la compuerta OR se partio de la NOR, misma que requiere tres glider-guns G_1, dos para las entradas A y B, y uno que forme parte del proceso de transformación de los gliders para generar el resultado; también se utilizan cuatro G_2 para modificar los flujos de entrada, dos por cada flujo. Las cadenas de bits que representan los flujo de los gliders de entrada son: 1100100 para la entrada A y 1101110 para la entrada B, siendo el resultado: 1101110; posteriormente se paso a utilizar la compuerta NOT, así obteniendo la compuerta OR. En la figura 17 se puede observar la compuerta OR.

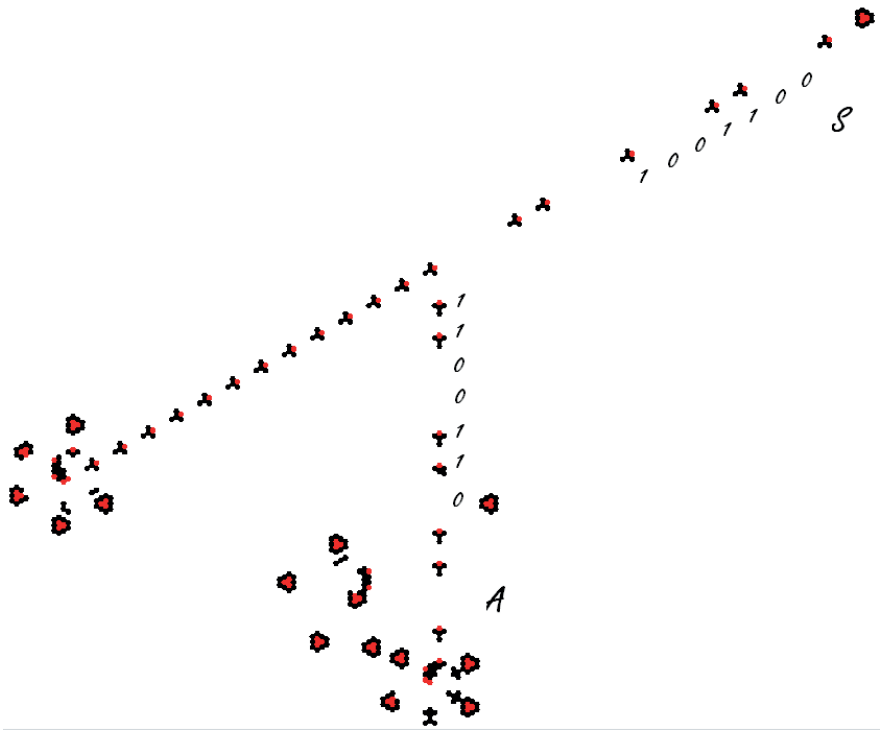


Figura 15. Compuerta NOT en la regla espiral.

5. Conclusiones

A través de los diagramas de *de Bruijn* se realizó una búsqueda exhaustiva de configuraciones resultantes de la interacción entre células, obteniendo como resultado matrices de *de Bruijn*, las cuales muestran las interacciones locales de las células en el autómata, para arrojar posibles partículas existentes.

Con lo anterior, se muestra que los diagramas de *de Bruijn* son una herramienta muy eficaz en el análisis de una regla de autómata celular, no obstante, la complejidad computacional que implica para obtener resultados es lo que deja abierto el siguiente problema:

Considerando que los diagramas de *de Bruijn* muestran el comportamiento local de la regla y con el fin de obtener resultados de mayor interés, es necesario realizar traslapes de un mayor número de células, o hacia una evolución para un tiempo n, lo que a su vez eleva exponencialmente el aspecto computacional, así como la generación y visualización de los diagramas.

Obteniendo resultados en este sentido, los diagramas de *de Bruijn* podrían llegar a ser utilizados de manera eficiente para encontrar todas las configuracio-

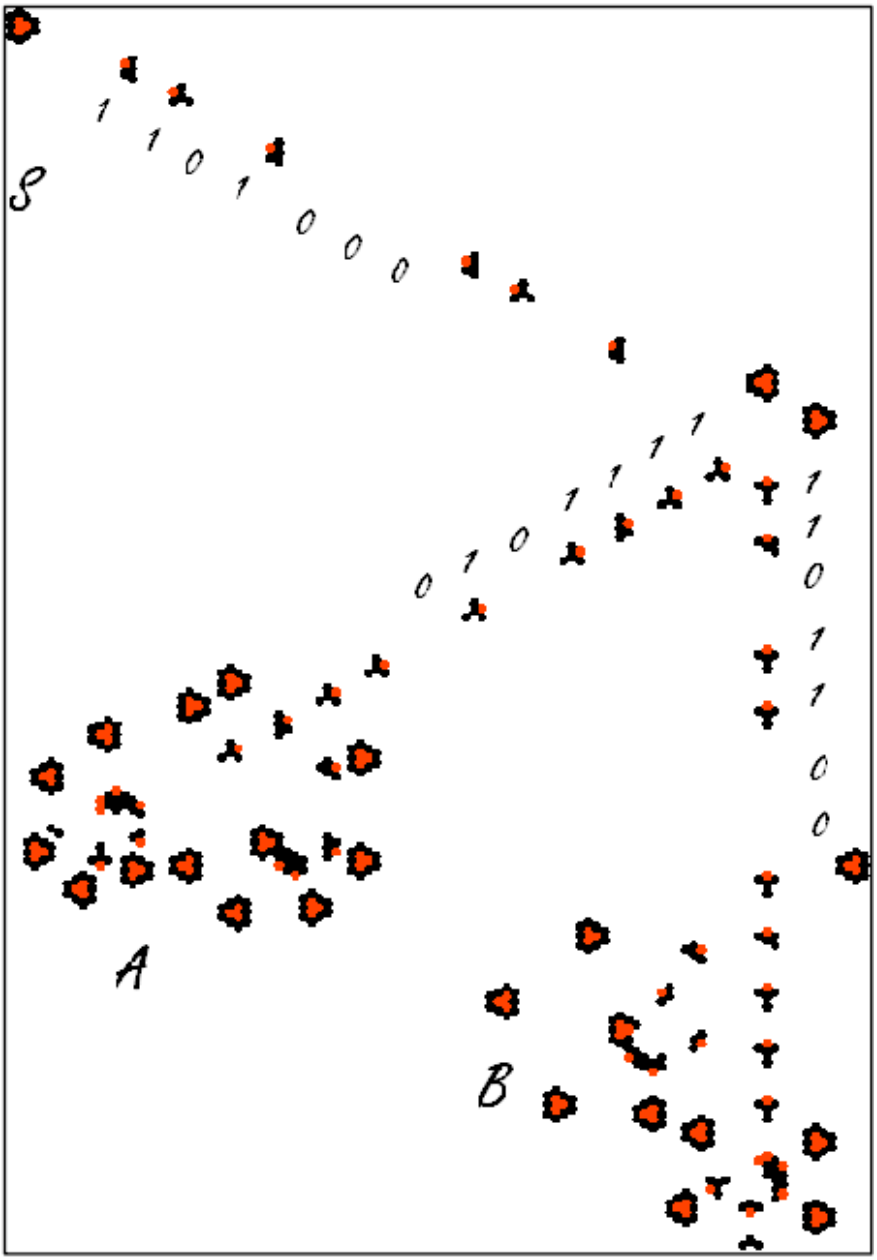


Figura 16. Compuerta AND en la regla espiral.

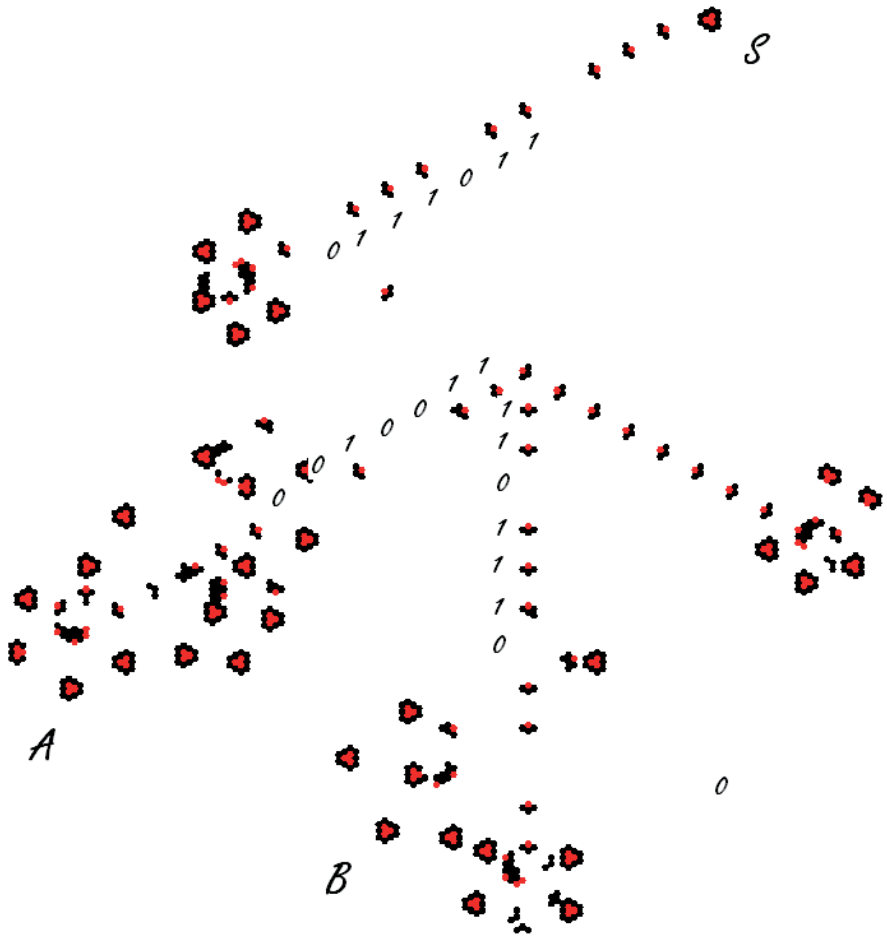


Figura 17. Compuerta OR en la regla espiral.

nes tanto movibles como estáticas del autómata, siempre limitados al tamaño de las subvecindades estudiadas.

Por otro lado, el análisis de la regla espiral de manera experimental ha encontrado una diversidad de partículas las cuales muestran que las colisiones entre ellas, así como sus reacciones, hacen posible implementar computación; también, se ha demostrado la lógica universal de la regla al tener las tres compuertas lógicas básicas: AND, OR y NOT. Derivado de estas construcciones se logró implementar otras compuertas que serían de ayuda en la búsqueda de construcciones más complejas. Por lo que un objetivo a futuro es la búsqueda de la construcción de un medio sumador utilizando la compuerta XOR y AND que se tienen actualmente. Incluso con la posibilidad de utilizarlas para construir dispositivos más complejos como simular una función computable completa; incluso cualquier otro sistema no-lineal con dicha dinámica.

Los espacios de evolución utilizados durante las pruebas y construcciones mostradas en el presente trabajo son de 160×160 y 240×240 células, lo que hace pensar que al realizar construcciones derivadas de las actuales, sería necesario utilizar espacios de evoluciones más amplios, lo que conlleva un mayor procesamiento y una visualización menos clara; es por eso, que se propone el diseño de un simulador con herramientas de computación de alto rendimiento.

Referencias

[1] Adamatzky, A. (2002). *Collision-Based Computing*, Springer-Verlag. London.

[2] Adamatzky, A., Martínez, G. J., Zhang, L., & Wuensche, A. (2009). Operating binary strings using gliders and eaters in reaction-diffusion cellular automaton, *Mathematical and Computer Modelling* 52, 177-190.

[3] Adamatzky, A. & Wuensche, A. (2006). On spiral glider-guns in hexagonal cellular automata: activator-inhibitor paradigm. *International Journal of Modern Physics C*, 17(7), 1009-1026.

[4] Adamatzky, A. & Wuensche, A. (2006). Computing in Spiral Rule Reaction-Diffusion Hexagonal Cellular Automaton, *Complex Systems*, 16(4), 277-297.

[5] Adamatzky, A., Wuensche, A., & Costello, B.De Lacy . (2006).Glider-based computing in reaction-diffusion hexagonal cellular automata. *Chaos, Solitons & Fractals*, 27(2), 287-295.

[6] Adamatzky, A. & Teuscher, C. (2006). *From Utopian to Genuine Unconventional Computers: Splendeurs et miseres du calcul peu usuel*, Luniver Press.

[7] Berlekamp, E. R., Conway, J. H., & Guy, R. K. (1982). *Winning Ways for your Mathematical Plays*, Academic Press, Volumen 4, capítulo 25, 927-961.

[8] Basurto, R. y León, P. A. (2009). Spiral Simulator, `http://uncomp.uwe.ac.uk/genaro/Papers/Thesis.html`.

[9] Basurto, R. y León, P. A. (2009). Computación basada en reacción de partículas en un autómata celular hexagonal, *Tesis de Licenciatura*, Escuela Superior de Cómputo del Instituto Politécnico Nacional, México, D.F. `http://uncomp.uwe.ac.uk/genaro/Papers/Thesis.html`.

[10] Gutowitz, H. (1991). *Cellular Automata, Theory and Experiment*, The MIT Press.

[11] Ilachinski, A. (2001). *Cellular Automata: A Discrete Universe*, World Scientific Press, Singapore.

[12] León, P. A., Basurto, R., Martínez, G. J., & Seck-Tuoh-Mora, J. C. (2011). Complex Dynamics in a Hexagonal Cellular Automaton, *Proceedings of the 2011 International Conference on High Performance Computing & Simulation (HPCS 2011)*, 750-756.

[13] Martínez, G. J., Adamatzky, A., & McIntosh, H. V. (2006). Phenomenology of glider collisions in cellular automaton Rule 54 and associated logical gates. *Chaos, Fractals and Solitons*, 28, 100-111.

[14] Martínez, G. J., McIntosh, H. V., Seck-Tuoh-Mora, J. C., & Vergara, S. V. C. (2007). Rule 110 objects and other constructions based-collisions, *Journal of Cellular Automata*, 2(3), 219-242.

[15] Martínez, G. J., Adamatzky, A., McIntosh, H. V., & Costello, B. D. L. (2008). Computation by competing patterns: Life rule B2/S2345678. *Automata 2008: Theory and Applications of Cellular Automata*, Luniver Press, 356-366.

[16] Martínez, G. J., Adamatzky, A., & Costello, B. D. L. (2008). On logical gates in precipitating medium: cellular automaton model. *Physics Letters A*, 1(48), 1-5.

[17] McIntosh, H. V. (1988). A Zoo of Life Forms, `http://delta.cs.cinvestav.mx/~mcintosh/cellularautomata/Papers.html`.

[18] McIntosh, H. V. (1998). Life's Still Lifes, `http://delta.cs.cinvestav.mx/~mcintosh/cellularautomata/Papers.html`.

[19] McIntosh, H. V. (2009). *One Dimensional Cellular Automata*, Luniver Press.

[20] Toffoli, T. & Margolus, N. (1987). *Cellular Automata Machines*, The MIT Press, Cambridge, Massachusetts.

[21] Ralston, A. (1982). De Bruijn sequences-A Model Example of the Interaction of Discrete Mathematics and Computer Science. *Mathematics Magazine*, 55(3), 131-143.

[22] von Neumann, J. (1966). *Theory of self-reproducting automata*, Urbana and London, University of Illinois.

[23] Wuensche, A. (2005). Glider dynamics in 3-value hexagonal cellular automata: the beehive rule. *Int. J. of Unconventional Computing*, 1(4), 375-398.

[24] Wuensche, A. (2011). *Exploring Discrete Dynamics*, Luniver Press. Software "Discrete Dynamics Lab (DDLab)", `http://www.ddlab.org`.

[25] Wolfram, S. (2002). *A New Kind of Science*, Champaign, Illinois, Wolfram Media Inc.

[26] Zhang, L. (2010). The extended glider-eater machine in the Spiral rule, *Lecture Notes in Computer Science*, 6079, 175-186.

Algebraic relations for computations with Rule 110 cellular automaton

José Manuel Sausedo Solorio

Laboratorio de Física Avanzada
Universidad Autónoma del Estado de Hidalgo, Hidalgo, México.
sausedo@uaeh.edu.mx

Resumen This work[1] deals with collisions of periodic structures (known as *gliders*) generated by the evolution of the one-dimensional Rule 110 cellular automaton. A specific value associated with each glider and an algebraic equation that describes the collision between two gliders are shown. Because the products of the collision between two gliders may result in no gliders or one, two or more gliders, that equation states that the total sum of the associated values corresponding to colliding gliders equals the sum of the values of the gliders which are products of the collision. Moreover, an analogy is proposed between the glider collisions and the collisions of physical particles with the equation corresponding to colliding gliders being similar to the equation of energy conservation in physics. In this scheme, even without carrying out the temporal evolution for a collision, it can be determined if a possible combination of resulting gliders accomplishes the equation corresponding to that collision.

1. Introduction

In recent years, cellular automata (CA) have gained attention by proving their capacity for analyzing complex systems, generating new concepts, universal computations, and even their application to physical systems. Hence the characterization of these types of systems is very important, with specific examples of CA applications as [1]: The characterization of complex dynamic systems based on statistical properties, proving criteria for self-organization using statistical complexity in models of excitable media and the behavior of physical systems without taking into account small-scale details [2, 3]. Also, there have been reports of particle-like objects that propagate in several spatially-extended dynamic systems and interact among them [10]. In particular, the one-dimensional Rule 110 cellular automaton[2] has been widely studied in the last decade because of its capacity to produce universal complex behaviors; however a cell takes into account the actual state of just three neighborhoods and each cell has only two states [5]. Moreover, it was first conjectured by Stephen Wolfram that this

[1] Main results already have been published in: IJMPC Vol. 21 No. 7, 2010.
[2] From here onwards, we will just use "Rule 110" to refer to Rule 110 cellular automaton.

cellular automaton may be universal. This statement was proved by Matthew Cook implementing a cyclic tag system using Rule 110 [10, 5].

A distinctive feature of the Rule 110 is the formation of a periodic background in space and time which is called *ether*. In conjunction with this regular mosaic, other periodic structures known as *gliders* are formed as time evolves. Such gliders move with constant lateral displacement. However, such displacement may be different between one glider and another, resulting in collisions between them.

Collisions may yield other or even the same combinations of gliders also called *products* here. This feature has been studied to both obtain a theoretical understanding of this behavior and implement unconventional computer systems [6, 7].

Most of the work of Rule 110 has been done from the perspective of Computer Theory or from using Complex Systems analysis [8]. Moreover, there are previous research findings for Rule 110 with a general scope, which have considered algebraic features of cellular automata to provide invariant attributes in the sense of group theory [9, 10]. Collisions among gliders have been analyzed by controlling their relative period as a way of producing them more easily [11]. However, to date there has been a lack of published research on the characterization of gliders in this cellular automaton, which considers them as interacting objects.

In general according to Rule 110, gliders may be generated with specific initial conditions or as products of collisions with other gliders [5, 12]. However in this work, it is considered gliders which have been created from initial conditions only. The aim of this work is to state a quantitative characterization of the structures from the Rule 110 and to establish relations derived from collisions among them. The result is stated as an algebraic system capable to elucidate computations. The whole analysis is based on computational experimentation by causing two gliders to collide and observing the products of the collision.

The remainder of the paper is organized as follows: Section 2 is devoted to exposing the basic concepts of one-dimensional cellular automata. Section 3 explains how collisions among gliders are expressed in terms of algebraic equations. Section 4 provides values for the found constants as well as their interpretation. Section 5 states the conclusions reached about the utility of the constant associated with each glider.

2. Theory

In general, cellular automata are defined by means of a tuple $\{\Sigma, r, \phi, C\}$, where Σ is a finite set of allowed states for each cell, $r \in \mathbb{Z}^+$ is the number of neighbors with respect to each side of a cell, $\phi : \Sigma^{2r+1} \to \Sigma$ is the evolution rule determining the next state for every cell as a function of its own state and the states of its 2r neighboring cells at current time, and $C : \mathbb{Z}_m \to \Sigma$ is the initial configuration, $\mathbb{Z}_m = \{0, \ldots, m-1\}$, and $m \in \mathbb{Z}^+$ is the size of C. Hence, C contains the initial state of every cell at the starting time of the evolution. In

this way, CA are dynamic systems, not only with discrete spatial domain, but also with discrete temporal domain, where their spatial evolution is carried out through interactions with their nearest neighbors.

Cuadro 1. Evolution for cellular automaton based on Rule 110.

Neighborhood	Evolution	Neighborhood	Evolution
000	0	100	0
001	1	101	1
010	1	110	1
011	1	111	0

The particular case being analyzed is a cellular automaton whose rule of evolution is the Rule 110 defined in Table 9. For this rule, the set of states is $\Sigma = \{0, 1\}$ and $r = 1$ (a single neighbor to each side of the cell), therefore an initial configuration may be specified by a one-dimensional finite chain of 0's and 1's. A particular evolution can be seen in Fig. 1.

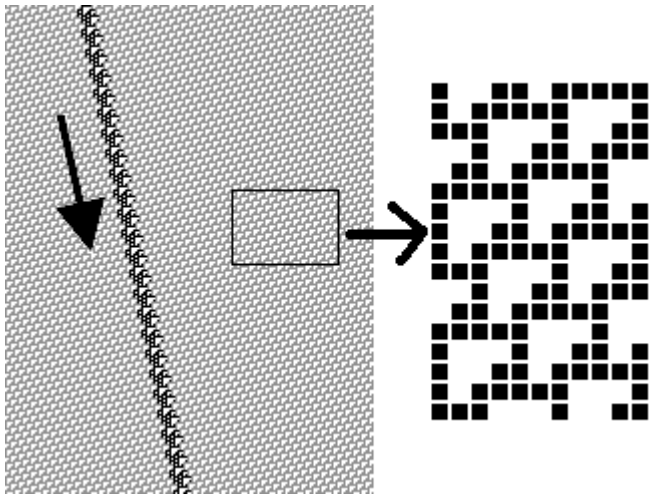


Figura 1. A typical glider (darker structure) moving through ether (light gray color). Temporal evolution follows the downward direction. The detail shows ether structure.

In this cellular automaton there are 14 known individual gliders represented by the set $M = \{A, B, \bar{B}, \bar{B}_8, C_1, C_2, C_3, D_1, D_2, E, \bar{E}, F, G, H\}$ plus a glider called *Gun*, which produces several of the gliders of M as time evolves. So this

glider is not considered as individual structure [5, 6]. Here, we consider only binary collisions among gliders belonging to M.

In order to generate ether as well as a specific glider, it is necessary to choose the appropriate initial conditions for the beginning of the evolution. In particular, the ether in Rule 110 is generated by a sequence consisting of 14 cells. In the case of gliders, the length of the sequence is variable, for example, to generate gliders A, C_1 and E, the lengths of the ships are 6, 23, and 29 cells respectively. It is possible to generate gliders with more than one set of initial conditions[3]; for example, glider A can be generated with the following two sequences (phases) of 6 cells: 111110, and 100011 [6].

Most of the possible combinations[4] of binary collisions among gliders of M have been studied and classified previously in atlases and catalogs [10, 5, 13]. However, up to now there has not been a collision-based analysis that provides features useful for exploring the underlying quantitative properties from the interaction among gliders.

3. Algebraic and production relations for collisions

A schematic representation of a binary collision is shown in Fig. 2, where initial (two) gliders μ_i and μ_j collide to produce the gliders labeled as μ_1, μ_2, μ_3 and so on, appearing at the bottom of the figure. A collision may generate no gliders, one, or several gliders as its *products*.

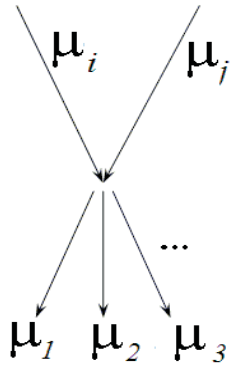


Figura 2. Schematic representation for a binary collision between two gliders μ_i and μ_j. The resulting products are labeled as μ_1, μ_2, and μ_3.

Equation (1) highlights the notation used to specify a collision between two incident gliders. Labels on the left-hand side correspond to the colliding gliders,

[3] In the terminology of cellular automata, it is called *phase* to each of those sequences.
[4] Although, it is highlighted bellow some collisions that are not found in catalogs yet.

whereas labels on the right-hand side indicate the products. Here, this equation is called a *production relation*.

$$\mu_i \oplus \mu_j \to \mu_1 + \cdots + \mu_n \tag{1}$$

where $\mu_k \in M$ and $k \in \mathbb{N}$. Symbol $\oplus$ indicates the interaction (collision) between gliders μ_i and μ_j, while the plus sign (+) represents the collection[5] of resulting products labeled as $\mu_1, \mu_2, \ldots, \mu_n$.

With 14 gliders in M, there are 91 ($14 \times 13/2$) possible results of binary collisions. This is because for two gliders $\mu_1, \mu_2 \in M$ under this notation, the result of $\mu_1 \oplus \mu_2$ is the same as for $\mu_2 \oplus \mu_1$. Furthermore, some gliders have the same *horizontal speed*, meaning that they travel in parallel, hence they will never collide. The subsets of gliders that move in parallel are $\{C_1, C_2, C_3\}$, $\{B, \bar{B}, \bar{B}_8\}$, and $\{D_1, D_2\}$. Additionally, there are some *soliton-like* binary collisions, i.e., interactions in which at least one of the gliders remains without change after collision. An example of this type of collision process is[6] $A \oplus \bar{E} \to A \ + \bar{E}$.

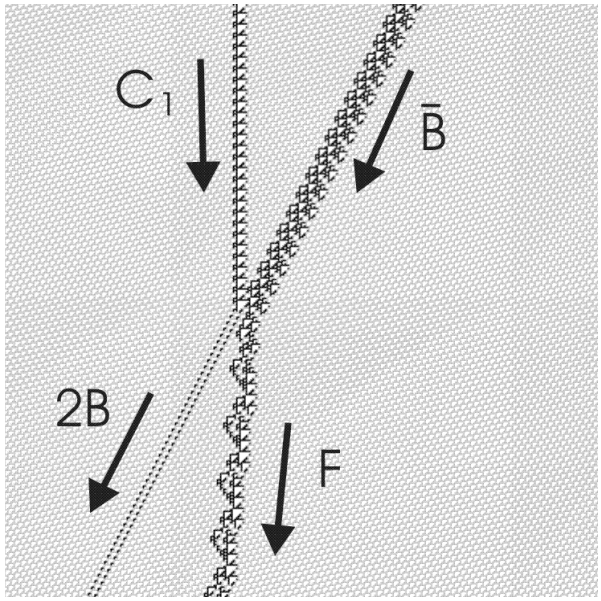


Figura 3. The binary collision between gliders C_1 and $\bar{B}$ with a productive relation denoted by $C_1 \oplus \bar{B} \to 2B + F$. The corresponding algebraic equation proposed is $\xi_{C_1} + \xi_{\bar{B}} = 2\xi_B + \xi_F$.

[5] Here, we use the sign of summation, instead of a comma as in the symbology of set theory.

[6] Hereafter the symbols corresponding to gliders on both sides of a production relation are written in alphabetical order, regardless of its position when a collision is observed in the time evolution graph.

A typical example of a binary collision can be observed in Fig. 3, where gliders C_1 and $\bar{B}$ collide, yielding as products two[7] B gliders and one F glider ($\bar{B} \oplus C_1 \rightarrow 2B + F$). In this work it is proposing constants ξ_X associated to each glider and an algebraic equation corresponding to each production relation. For the example in this paragraph, constants and relation can be written as $\xi_{\bar{B}} + \xi_{C_1} = 2\xi_B + \xi_F$.

After consideration, a total of 83 pairs of colliding gliders can be listed which are represented as production relations in Table 2. In this table, the Φ symbol is used to denote that no particle is obtained after a collision, being its *associated constant* equal to zero ($\xi_\Phi = 0$).

Cuadro 2. Production relations for the collisions between two gliders.

$A \oplus B \rightarrow \Phi$	$A \oplus \bar{B} \rightarrow \Phi$	$A \oplus \bar{B}_8 \rightarrow C_2$	$A \oplus C_1 \rightarrow F$
$A \oplus C_2 \rightarrow C_1$	$A \oplus C_3 \rightarrow C_2$	$A \oplus D_1 \rightarrow C_2$	$A \oplus D_2 \rightarrow D_1$
$A \oplus E \rightarrow D_1$	$A \oplus \bar{E} \rightarrow A + \bar{E}$	$A \oplus F \rightarrow 4B + C_2$	$A \oplus G \rightarrow \bar{E} + C_1$
$A \oplus H \rightarrow C_2$	$B \oplus C_1 \rightarrow C_2$	$B \oplus C_2 \rightarrow D_1$	$B \oplus C_3 \rightarrow E$
$B \oplus D_1 \rightarrow \bar{E}$	$B \oplus D_2 \rightarrow A + \bar{E}$	$B \oplus E \rightarrow E_2$	$B \oplus \bar{E} \rightarrow 2A + 3B + \bar{E}$
$B \oplus F \rightarrow 2A + D_1$	$B \oplus G \rightarrow G_2$	$B \oplus H \rightarrow \bar{E}$	$\bar{B} \oplus C_1 \rightarrow 2B + F$
$\bar{B} \oplus C_2 \rightarrow 3A + \bar{E}$	$\bar{B} \oplus C_3 \rightarrow 2A + \bar{E}$	$\bar{B} \oplus D_1 \rightarrow E$	$\bar{B} \oplus D_2 \rightarrow A + \bar{E}$
$\bar{B} \oplus E \rightarrow A + 4B + C_2$	$\bar{B} \oplus \bar{E} \rightarrow 4A + 5B + \bar{E}$	$\bar{B} \oplus F \rightarrow A + 4A + \bar{E}$	$\bar{B} \oplus G \rightarrow 4B$
$\bar{B} \oplus H \rightarrow A + 2C_2 + \bar{E}$	$\bar{B}_8 \oplus C_1 \rightarrow A + 2\bar{E}$	$\bar{B}_8 \oplus C_2 \rightarrow 2A + 3B + 2C_2$	$\bar{B}_8 \oplus C_3 \rightarrow 2A + 3B$
$\bar{B}_8 \oplus D_1 \rightarrow A + \bar{B} + \bar{B}$	$\bar{B}_8 \oplus D_2 \rightarrow 2A + 4B$	$\bar{B}_8 \oplus E \rightarrow 2A + B + G$	$\bar{B}_8 \oplus \bar{E} \rightarrow 3A + 2\bar{E}$
$\bar{B}_8 \oplus F \rightarrow 2A + 2B + C_2 + \bar{B} + F$	$\bar{B}_8 \oplus G \rightarrow 4A + 4B + \bar{E}$	$\bar{B}_8 \oplus H \rightarrow 2A + C_2 + \bar{E}$	$C_1 \oplus D_1 \rightarrow 4A + 3B$
$C_1 \oplus D_2 \rightarrow 2A + 2B$	$C_1 \oplus E \rightarrow A + \bar{E} + F$	$C_1 \oplus \bar{E} \rightarrow C_1 + \bar{E}$	$C_1 \oplus F \rightarrow C_1 + F$
$C_1 \oplus G \rightarrow 3A + F$	$C_1 \oplus H \rightarrow 3A + 3B + 2C_2$	$C_2 \oplus D_1 \rightarrow 2A + 2B$	$C_2 \oplus D_2 \rightarrow A + 2B$
$C_2 \oplus E \rightarrow A + 2B$	$C_2 \oplus \bar{E} \rightarrow 3B$	$C_2 \oplus F \rightarrow C_1 + \bar{B} + F$	$C_2 \oplus G \rightarrow 3B + C_2$
$C_2 \oplus H \rightarrow 3A + 2B + \bar{B}$	$C_3 \oplus D_1 \rightarrow A + 2B$	$C_3 \oplus D_2 \rightarrow A + 3B$	$C_3 \oplus E \rightarrow A + G$
$C_3 \oplus \bar{E} \rightarrow 2C_1$	$C_3 \oplus F \rightarrow C_1 + C_2$	$C_3 \oplus G \rightarrow \bar{E}$	$C_3 \oplus H \rightarrow 2B + \bar{B} + C_3 + F$
$D_1 \oplus E \rightarrow 2B$	$D_1 \oplus \bar{E} \rightarrow 4B$	$D_1 \oplus F \rightarrow 2A$	$D_1 \oplus G \rightarrow \bar{E}$
$D_1 \oplus H \rightarrow A + D_1 + E$	$D_2 \oplus E \rightarrow G$	$D_2 \oplus \bar{E} \rightarrow 5B$	$D_2 \oplus F \rightarrow 2A + B$
$D_2 \oplus G \rightarrow A + 3B + C_2 + G$	$D_2 \oplus H \rightarrow C_1 + \bar{E}$	$E \oplus F \rightarrow 4A + 3B$	$E \oplus G \rightarrow F$
$E \oplus H \rightarrow 5A + 2\bar{E}$	$\bar{E} \oplus F \rightarrow B$	$\bar{E} \oplus G \rightarrow 4A + \bar{E}$	$\bar{E} \oplus H \rightarrow A + 3B + C_1 + \bar{E}$
$F \oplus G \rightarrow 3A + \bar{E}$	$F \oplus H \rightarrow F + D_1$	$G \oplus H \rightarrow A + 2\bar{E} + F$	

Most pairs of colliding particles can be found to collide in more than one way. For example, the collision $A \oplus \bar{B}_8 \rightarrow C_2$ listed in Table 2, it can also be found as $A \oplus \bar{B}_8 \rightarrow 4A + \bar{E}$. Thus, it is possible at times for collisions to result in more than one combination of particles. For simplicity, in Table 2 only one possibility of these combinations has been written for each pair, the rest can be found elsewhere [13]. In order to get different results from the collision of a pair of gliders, they must collide with a different contact point or relative phase. This is achieved by generating the gliders with different initial conditions[8].

[7] To verify that the structure labeled with $2B$ actually consists of two B particles, there are two ways to proceed: One, by performing an amplification of the figure and comparing with the corresponding *tilling* of glider B [13]. Two, by causing a collision of a known particle (i.e. A) with this structure and observing that one of the B particles is eliminated with A (according to the production: $A \oplus 2B \rightarrow B$), while the remaining B glider continues its path without change.

[8] The ways that a glider can be generated are called *Periods*, which are used to control the spatial contact point at which gliders collide.

4. Results

In general, for the production relation in Eq. (1), the following corresponding algebraic equation is proposed:

$$\xi_{\mu_i} + \xi_{\mu_j} = \xi_{\mu_1} + \cdots + \xi_{\mu_n} \tag{2}$$

where each ξ_{μ_k} is the unknown glider constant associated to glider μ_k. In both sides of this equation, the plus sign means algebraic summation of the constants, so for each production relation in Table 2, it is possible to write a corresponding linear algebraic equation involving these unknowns ξ_μ. All of these algebraic equations form a system of linear equations (a total of 83). Since the system has more equations than variables, this is an over-determined system. The full solution was obtained by a trial-and-error procedure, assigning an arbitrary value[9] to some ξ_μ's, and then obtaining the values for the rest of them. The resulting values obtained for each unknown are shown in Table 3. For example, for the collision in Fig. 3 ($\xi_{C_1} + \xi_{\bar{B}} = 2\xi_B + \xi_F$), by replacing values from that table in the corresponding algebraic equation $\xi_{C_1} + \xi_{\bar{B}} = 2\xi_B + \xi_F$, we get 3+ (-2) on the left side and 2(-2)+5 on the right side, ensuring equality. It must be emphasized that the origin of each ξ_μ is geometric and it is related to a relative shift of the ether pattern in both sides of the traveling glider.

Cuadro 3. Values of the unknowns found for each glider of the set M.

$\xi_\Phi = 0$	$\xi_A = 2$	$\xi_B = -2$	$\xi_{\bar{B}} = -2$	$\xi_{\bar{B}_8} = -1$
$\xi_{C_1} = 3$	$\xi_{C_2} = 1$	$\xi_{C_3} = -1$	$\xi_{D_1} = -1$	$\xi_{D_2} = -3$
$\xi_E = -3$	$\xi_{\bar{E}} = -7$	$\xi_F = 5$	$\xi_G = -6$	$\xi_H = -1$

As a new result, included in Table 2 is the collision $B \oplus F \to 2A + D_1$ not previously catalogued in any atlas for Rule 110 [13]. Also two structures are not included in the set M: E_2 and G_2 (not catalogued as individual gliders), appearing as collision products in $B \oplus E \to E_2$ and $B \oplus G \to G_2$. Moreover, the values $E_2 = -5$ and $G_2 = -8$ have been found for these structures, both consistent with the system of linear algebraic equations proposed.

Table 2 shows 83 collisions, for 80 of them it is possible to write an algebraic equation in the form of Eq. (2). But for the remaining three ($A \oplus F \to 4B + C_2$, $E \oplus G \to F$, and $\bar{E} \oplus G \to 4A + \bar{E}$), no set of gliders has been found or reported whose constants fulfill the corresponding algebraic equations. For those cases it is proposed here a constant[10] $\alpha = 14$ to include in each equation in the following form,

[9] We start by considering the equation $\xi_A + \xi_B = 0$ corresponding to production relation $A \oplus B \to \Phi$, in order the numeric solution be self-consistent.

[10] Here, it is considered α as a *Structure Constant* due to its origin from the *ether structure*.

$$\begin{aligned} \xi_A + \xi_F &= 4\xi_B + \xi_{C_2} + \alpha \\ \xi_E + \xi_G &= \xi_F - \alpha \\ \xi_{\bar{E}} + \xi_G &= 4\xi_A + \xi_{\bar{E}} - \alpha. \end{aligned} \tag{3}$$

As a step in the process of solving this problem, we have determined that the respective balance equations are fulfilled if such constant is added to or subtracted according to these equations. In Eqs. (3), the value of α is equal to the length of the sequence that generates ether. In fact, the whole algebraic system has an infinite number of solutions, the shown in Table 3 is just one particular solution.

Whenever a glider goes through ether, the ether pattern has a relative displacement of one side of the glider with respect to the other side. Figure 4 shows the way such a displacement can be detected. In this figure there are three reference lines superimposed on the ether pattern. Two of the lines are vertical and one is horizontal. In Fig. 4a, the two intersections of the line match the ether pattern at the same relative point; whereas in Fig. 4b as glider C_1 travels down through the ether, there is a shift in the ether pattern. It can be seen as indicated by the arrow, that the ether pattern to the right of the figure is no longer in the same position as before, relative to the intersection of the corresponding lines. Such displacement depends on each glider. Moreover, if the two gliders have the same ξ_μ, the ether experiences the same shift.

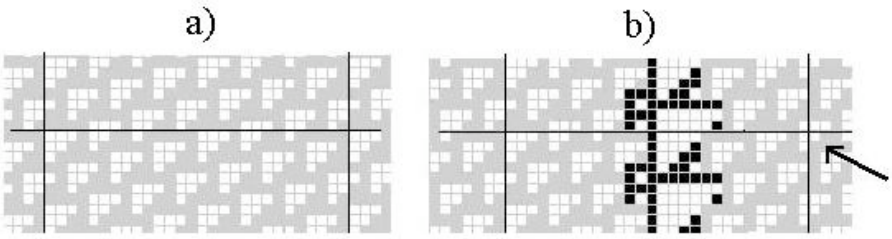


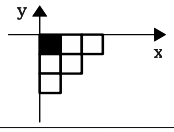
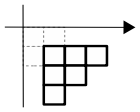
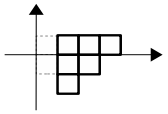
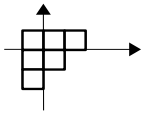
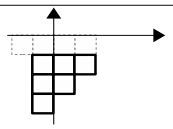
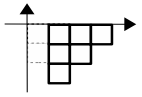
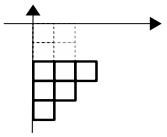
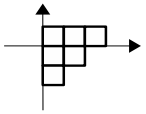
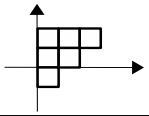
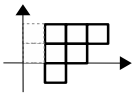
Figura 4. Comparison of the displacement produced in ether by a traveling glider. Intersections of horizontal and vertical lines serve as references to observe the shift on ether. On the left, ether is depicted alone. On the right, glider C_1 travels in a downward motion. The arrow points to the intersection which demonstrates the ether's shift.

Table 4 shows the horizontal an vertical displacements $(\delta x, \delta y)$ of each glider μ, the ether displacement graph and the corresponding constant ξ_μ. The same cell[11] was taken as reference per Fig. 4a, and the displacement is counted by the number of cells with the same signs as used in the *Cartesian Coordinate*

[11] Indicated by a small white square.

System to specify distances. The column of constants ξ_μ taken from the set of solutions already encountered by each glider 1, -1, 2, -2, 3, -3, 5, -6, -7 in Table 3. The gliders with the same value-ξ, displace the ether in the same magnitude and direction and are found in the table on the same line. This table shows only one of an infinite number of ether displacements. Because the ether pattern is periodic, a displacement, as an example, $(\delta x = +4, \delta y = +1)$, takes the reference cell to the same location as with $(\delta x = +2, \delta y = -3)$.

Cuadro 4. Displacement of the ether-pattern and values associated with each glider.

μ	(δx, δy)	Ether pattern	ξ_μ	μ	(δx, δy)	Ether pattern	ξ_μ
Reference	(0, 0)		0	D_2 E	(1, -1)		-3
$\overline{B}_8$ C_3 D_1 H	(1, 1)		-1	C_1	(-1, 1)		+3
C_2	(-1, -1)		+1	F	(1, 0)		+5
B $\overline{B}$	(0, -2)		-2	G	(0, 1)		-6
A	(0, 2)		+2	$\overline{E}$	(1, 2)		-7

Displacements behave in the same way as the components of a two dimensional vector. As an example, from Table 4, for glider C_1 (with $\xi_{C_1} = +3$) the cell of reference, undergoes a displacement of $(\delta x_1 = -1, \delta y_1 = +1)$, and for D_2 (with $\xi_{D_2} = -3$) the displacement is $(\delta x_2 = +1, \delta y_2 = -1)$. If we cause C_1 to collide with D_2, the rule of production for this collision and the corresponding algebraic equation are $C_1 \oplus D_2 \rightarrow 2A + 2B$, and $\xi_{C_1} + \xi_{D_2} = 2\xi_A + 2\xi_B$. By

replacing the values of every ξ on the last equation, (+3)+(-3)=2(2)+2(-2), resulting $0 \equiv 0$. Observe that the summation of the displacements of the initial gliders vanishes too [(+3)+(-3)=0]. Because values cancel on both sides of the algebraic equation, it is expected ether does not suffer any displacement, this can be seen in Fig. 5. Therefore displacements $(\delta x, \delta y)$ behave in an arithmetical manner consistent with the ξ_μ associated with glider-μ. This is the way in which each displacement connects with its corresponding glider constant.

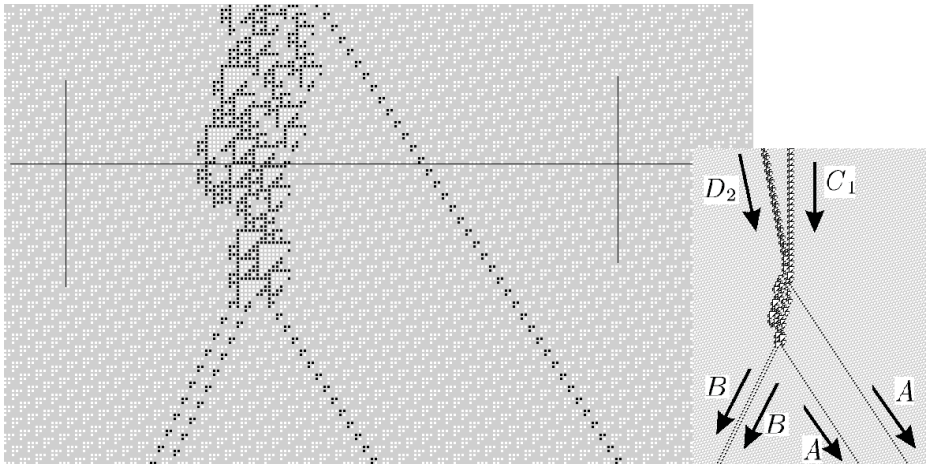


Figura 5. Collision between gliders C_1 and D_2, the evolution is in accordance with the rule of production: $C_1 \oplus D_2 \rightarrow 2A + 2B$ as seen completely in the box. In the enlarged part of the figure, with the help of the previously calibrated lines, one can observe that the ether is not displaced.

Although previously mentioned that the origin of the constants associated with each glider is of a geometric character, there is no metric discovered yet to establish a formal relationship a displacement $(\delta x_\mu, \delta y_\mu)$ and the corresponding ξ_μ in the form $F(\delta x_\mu, \delta y_\mu) \sim f(\xi_\mu)$.

5. Concluding remarks

In this paper, we suggest that:

- For each glider $\mu \in M$, there is a quantity ξ_μ which fulfills a balanced algebraic equation.
- Each collision between two gliders corresponds to a balanced equation. This is a linear algebraic relationship with the unknowns being the ξ_μ's. On the left-hand side of the equation, the incident gliders are represented; on the right, the resulting gliders.

- The numeric quantity ξ_μ associated with each glider generated by Rule 110 represents a shift in the ether pattern.
- Constants and algebraic relations can be useful for establish a systems capable to perform computations.

It is possible to use the value found for each glider and its related equations to construct an algebraic system valid for collisions among gliders. It is noteworthy that, with this tool and even without the temporal evolution, the manner in which the ether is displaced can be established, merely by knowing the gliders involved in the collision.

With respect to the balanced equations contained in Eq. (3), the addition or subtraction of a constant should not affect the validity of the solution seen in Table 3, because this is a particular solution. We consider the failure to find such sets of gliders does not detract from the usefulness of these results. We must wait for the necessary collisions to complete this scheme can be found.

Acknowledgments

The author wishes to express his gratitude to U.A.E.H. for the support given to realize this work; to Genaro J. Martínez for his interest in this problem and his many useful suggestions and a special thanks to J. C. Seck-Tuoh-Mora, for discussing the results and reviewing the entire manuscript.

Referencias

[1] A. Shreim, P. Grassberger, W. Nadler, B. Samuelsson, J. E. S. Socolar, & M. Paczuski, *Phys. Rev. Lett.* **98**, 198701 (2007).
[2] C. R. Shalizi, K. L. Shalizi, & R. Haslinger, *Phys. Rev. Lett.* **93**, 118701 (2004).
[3] N. Israeli & N. Goldenfeld, *Phys. Rev. E* **73**, 026203 (2006).
[4] S. Wolfram, *A New Kind of Science* (Wolfram Media, Illinois, 2002).
[5] M. Cook, *Complex Systems* **15** (2004), p.1.
[6] G. J. Martínez, H. V. McIntosh, & J. C. Seck-Tuoh-Mora, *Int. J. Unconv. Comp.* **2** (2006).
[7] A. Adamatzky (Ed.), *Collision-Based Computing* (Springer, 2002).
[8] L. Landau & E. Lifshitz, *Curso abreviado de física teórica*, Ed. Mir (Vol. 1, Moscu 1971).
[9] M. Pivato, *Nonlinearity* **15** (2002) p. 1781.
[10] M. Pivato, *Ergod. Th. and Dynam. Sys.* **26** (2006), p. 1.
[11] G. J. Martínez, H. V. McIntosh, & J. C. Seck-Tuoh-Mora, *Lect. Notes in Comp. Sc.*, 2801 (2003) p. 175.
[12] X. B. Li, R. Jiang, & Q. S. Wu, *Phys. Rev. E*, **68** 016117 (2003).
[13] G. J. Martínez & H. V. McIntosh (2001) ATLAS: Collisions of gliders as phases of ether in rule 110, `http://uncomp.uwe.ac.uk/genaro/Papers/Papers_on_CA_files/ATLAS/bookcollisions.html`.

Modelando la evolución de una red compleja con autómatas celulares

Andrés Anzo Hernández, Juan Gonzalo Barajas Ramírez

IPICYT, División De Matemáticas Aplicadas
San Luis Potosí, S. L. P., México. C.P. 78216.
{andres.anzo,jgbarajas}@ipicyt.edu.mx

Resumen Las redes del mundo real evolucionan: nodos y enlaces aparecen y desaparecen, los enlaces cambian su peso, dirección, etc. En este trabajo proponemos un modelo de evolución en redes que consiste en cambiar la conexión de los enlaces mediante reglas de Autómatas Celulares (AC), unidimensionales con frontera periódica. La idea principal en este modelo es asociar en cada paso del tiempo, el estado binario de las celdas del AC y los enlaces de la red (presente/ausente). De esta forma se establece un proceso de cambio estructural en la red en el cual, el estado de cada enlace al tiempo $t + 1$, depende de su propio estado y el del sus vecino al tiempo t.

1. Introducción

Una de las propiedades más importantes de las redes del mundo real es la capacidad de cambiar su estructura. Estos cambios llamados evolución estructural, incluyen entre otros procesos el aumento y disminución de nodos y enlaces, el ajuste en el peso y la dirección de sus enlaces, así como procesos de recableado. Todos estos mecanismos de evolución estructural pueden ocurrir de manera simultánea lo que resulta complicado de describir en forma precisa. Es particularmente difícil determinar cuales son las condiciones necesarias para disparar estos cambios estructurales en un tiempo específico. Las modificaciones estructurales a lo largo del tiempo permite a las redes complejas del mundo real evolucionar hacia formas que beneficien alguna funcionalidad dada. Estudios empíricos de redes del mundo real de diferentes naturalezas, como el Internet, WWW, las redes de colaboraciones científicas, etc., han mostrado que independientemente de su naturaleza, estas redes comparten características estructurales tales como los efectos de mundo pequeño [1] y de escala-libre [2]. Esto último sugiere, más no demuestra, la existencia de mecanismos comunes que generan algunos de los fenómenos observados en estas redes.

Para capturar los mecanismos esenciales que dan lugar a las propiedades observadas en redes del mundo real se proponen modelos matemáticos que permiten estudiar teóricamente la evolución estructural de una red. Estos modelos no tratan de emular con exactitud todos los procesos ocurridos durante su evolución, si no que, proponen mecanismos hipotéticos que al ser aplicados en forma

recursiva un cierto número de veces, se obtienen propiedades estructurales similares a las redes estudiadas. Cuando estos modelos simplificados capturan alguna de las características comunes de las redes del mundo real se obtiene una pista sobre cuales son los mecanismos de cambio que rigen la evolución de las redes complejas. Un modelo de evolución de redes particularmente importante es el propuesto por Barabasi-Albert (BA) en 1999 [2]. El modelo BA inicia con un número pequeño (m_0) de nodos, y en cada paso de tiempo son aplicados los siguientes mecanismos: crecimiento y enlace preferencial. El crecimiento consiste en incrementar un nuevo nodo con m ($\leq m_0$) enlaces que conectarón al nuevo nodo con m nodos presentes en la red. El mecanismo de enlace preferencial establece que el nuevo nodo se conectará con una mayor probabilidad con aquellos nodos que tengan un número grande de conexiones, es decir, la probabilidad de conexión entre el nuevo nodo y el nodo i presente en la red, tienen una dependencia lineal con el grado de nodo k_i. Las simulaciones numéricas muestran que al aplicar estos mecanismos de forma recursiva un número grande de pasos de tiempo, la red evoluciona hasta alcanzar una estructura tal que, la distribución de la probabilidad del grado de nodo sigue una ley de potencias con exponente $\gamma = 3$. Las redes con esta estructura particular son llamadas redes de escala-libre y se distinguen principalmente por la presencia de nodos concentradores (nodos con un gran número de conexiones).

Un gran número de las redes estudiadas empíricamente son de escala-libre, por lo que los mecanismo de crecimiento y enlace preferencial, propuestos en el modelo BA, nos dan una pista sobre los procesos evolutivos en redes del mundo real. Sin embargo, estos mecanismos no toman en cuenta los procesos dinámicos internos para determinar como evoluciona la red, en cierto sentido, los mecanismos del modelo BA pueden verse como reglas externas a la red. En contrasentido, recientemente se han propuesto modelos de evolución de redes basados precisamente en los procesos dinámicos internos. Un ejemplo es el modelo propuesto por H. Sayama y C. Laramee en [4], en el cual la evolución es determinada por mecanismos locales en términos de la dinámica de cada nodo en una vecindad. En este modelo se establece un ciclo de retroalimentación entre la dinámica de los nodos y los mecanismo de evolución estructural. A este tipo de modelos se les ha dado el nombre de Redes Co-Evolutivas Adaptables (RCA) [3].

En un modelo RCA, la dinámica de un nodo depende del estado de sus vecinos, lo cual guarda una estrecha similitud con el concepto de autómata celular (AC). En particular, [4] modela una red con una estructura dada mediante un AC en el cual los nodos se representan como celdas, y las conexiones entre los nodos describen las vecindades. En cada instante de tiempo, conforme se actualizan los estados de los nodos, la co-evolución de la red se describe mediante reglas locales que determinan cuando y como se activan mecanismos de cambio estructural como el aumento o disminución de nodos, aparición y desaparición de enlaces (realambrado), el aumento o disminución de la fuerza de conexión, entre otros. Desde otra perspectiva, Smith et al en [5], propone un modelo donde la co-evolución de la red se basa en las reglas de un AC. En el modelo propuesto, para una red con un número fijo de nodos, el enlace que une al i-ésimo con

el j-ésimo nodo al tiempo $t+1$ se conecta o no de acuerdo con una regla local, por ejemplo, si la suma de los correspondientes grados de nodo al tiempo t es m_c, no se conectan, y en caso contrario se conectan. El autor llama a este modelo Red Autómata (RA). Cabe mencionar que las reglas locales se pueden definir en términos de propiedades de nodos (grado de nodo, centralidad, etc.) o propiedades de enlaces (peso o dirección).

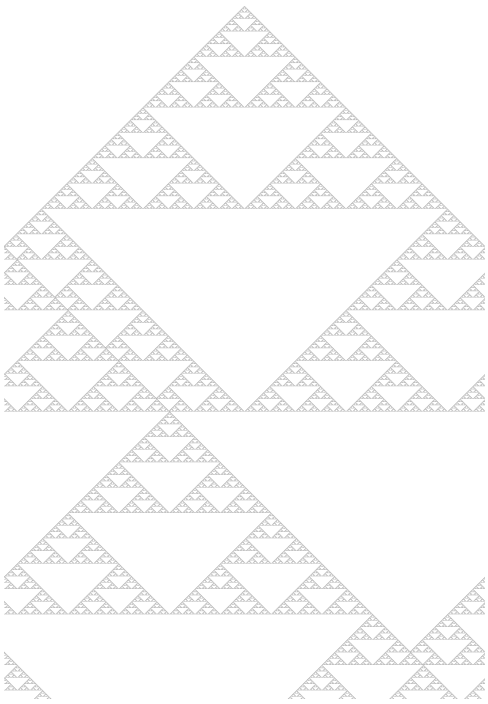


Figura 1. Regla 210 de un AC.

Inspirados en los trabajos discutidos arriba, en esta contribución proponemos un modelo de evolución semejante al propuesto en [5], en el sentido de que reglas locales son utilizadas para determinar la forma en que cambian los enlaces de la red. En nuestro modelo sin embargo, optamos por utilizar reglas de evolución tradicionalmente asociadas con AC de una dimensión, tales como las descritas en la clasificación de Stephen Wolfram [7]. La idea principal de nuestro modelo es asociar el estado binario de cada una de las celdas del AC con la presencia o ausencia de los enlaces de la red. De esta forma, al evolucionar un AC unidimensional binario, se determinan los cambios estructurales de la red en cada instante de tiempo.

En este artículo hemos seleccionado la regla 210 (en código Wolfram) para ejemplificar el funcionamiento de nuestro modelo. Después de aplicar recursi-

vamente esta regla un número grande de pasos de tiempo, se observa que la estructura de la red cambia drásticamente durante pequeños periodos de tiempo, en los cuales el número de nodos y enlaces se incrementan, y la estructura de la red es tal que la distancia geodésica entre los nodos es grande. Después de este periodo, la red cambia nuevamente a una estructura donde hay pocos enlaces, y en cada paso de tiempo ocurren pequeños cambios hasta llegar a un periodo de tiempo donde nuevamente la red cambia su estructura en formas similares a las descritas anteriormente. Este comportamiento de la red a lo largo de su evolución es consecuencia de los patrones generados por la regla 210 del AC.

2. Descripción del modelo de evolución

Considere una red de N nodos que en el instante inicial, todos excepto un par están aislados. Suponga también que los enlaces de la red son bidireccionales, no tienen peso y no hay enlaces bucles. En este modelo, la evolución de la red se realiza en tiempos discretos y suponemos que todas las posibles conexiones son permitidas, por lo que, el número máximo de enlaces de la red será $N(N-1)/2$. En cada paso de tiempo un enlace dado puede aparecer o desaparecer de acuerdo a reglas locales descritas en términos de su vecindad. Cuando el estado del enlace entre los nodos i y j al tiempo t, que denotaremos con $e_{ij}(t)$, sea 1, significará que estos nodos están conectados, y cuando el estado de $e_{ij}(t)$ sea cero, los nodos no están conectados. Definamos la configuración del cableado de la red al tiempo t como la $N(N-1)/2$-tupla $e(t) = (e_{1,2}(t), e_{1,3}(t), \ldots, e_{1,N}(t), e_{2,3}(t), \ldots, e_{N-1,N}(t))$.

Construyamos un AC de una dimensión con frontera periódica y con $N(N-1)/2$ celdas binarias. El estado binario de la n-ésima celda al tiempo t lo denotaremos con $c_n(t)$. En este trabajo consideraremos que la regla de AC ϕ_W (donde W es el número de la regla en código Wolfram) de la n-ésima celda al tiempo $t+1$ es función de $c_n(t)$ y del estado de las $r = 1$ celdas vecinas más cercanas a la izquierda y a la derecha al tiempo t, i.e. $c_n(t+1) = \phi_W(c_{n-1}(t), c_n(t), c_{n+1}(t))$. Por lo tanto habrá $2^{2^3} = 256$ posibles reglas de AC [7]. Definamos la configuración de la AC al tiempo t como la $N(N-1)/2$-tupla $c(t) = (c_1(t), c_2(t), \ldots, c_{N(N-1)/2}(t))$.

La idea principal en este modelo es asociar el estado de los elementos de la tupla $c(t)$ con los correspondientes estados (presente/ausente) de los enlaces en la tupla $e(t)$, de tal forma que si el estado de la n-ésima celda es uno, entonces el n-ésimo enlace de $e(t)$ estará presente en la red, y dicho enlace estará ausente en caso contrario. Como primer paso se define el estado de las celdas al tiempo cero, y mediante la asociación entre las tuplas $c(t)$ y $e(t)$ descrita con anterioridad, se construye la correspondiente red inicial. Evolucionamos después el estado de la celdas de acuerdo a la regla de AC seleccionada. En cada paso de tiempo $t+1$ se establece la configuración del cableado de la red a partir de los estados de las celdas en $c(t)$.

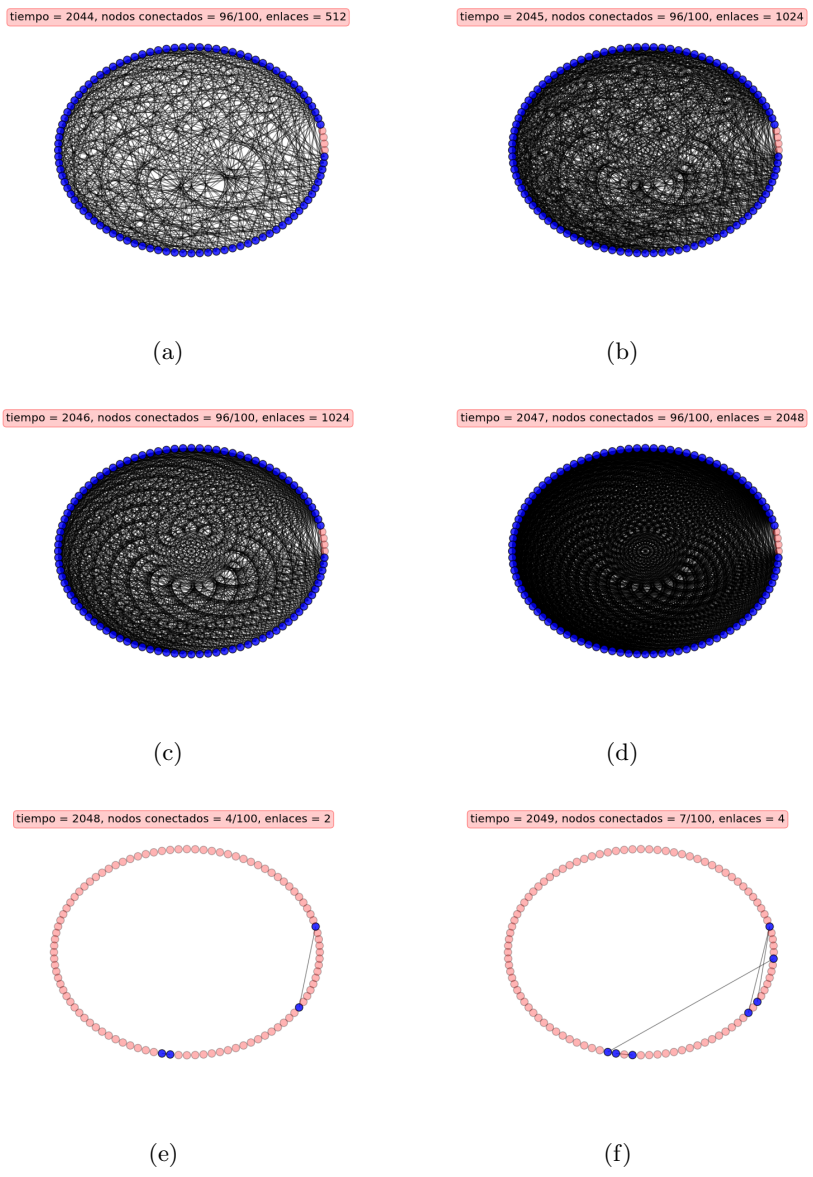


Figura 2. a) Estructura de la red en distintos pasos de tiempo de la evolución de acuerdo a la regla 210 de la AC. En azul los nodos con almenos una conexión, y en rojo los nodos sin conexión.

3. Resultados

Con el objetivo de ejemplificar el funcionamiento de este modelo, consideremos la regla 210 (código Wolfram) de una AC unidimensional con frontera periódica, $r = 1$ vecinos y celdas binarias (figura (1)). Supongamos que el número de nodos en la red es $N = 100$ por lo que el AC tendrá 4950 celdas. El estado de las celdas al tiempo cero será uno para la celda c_{2475} y cero para el resto de las celdas, por lo que de acuerdo con el modelo, a esta configuración inicial de celdas le corresponde la red con sólo un enlace entre los nodos 30 y 41. A continuación aplicamos la regla 210 en $T = 7000$ pasos de tiempo, actualizamos el estado de las celdas y establecemos la correspondiente asociación entre $c(t)$ y $e(t)$ para determinar la configuración del cableado de la red en cada instante de tiempo de la evolución. En la figura (2) podemos observar la estructura de la red en distintos pasos de tiempo.

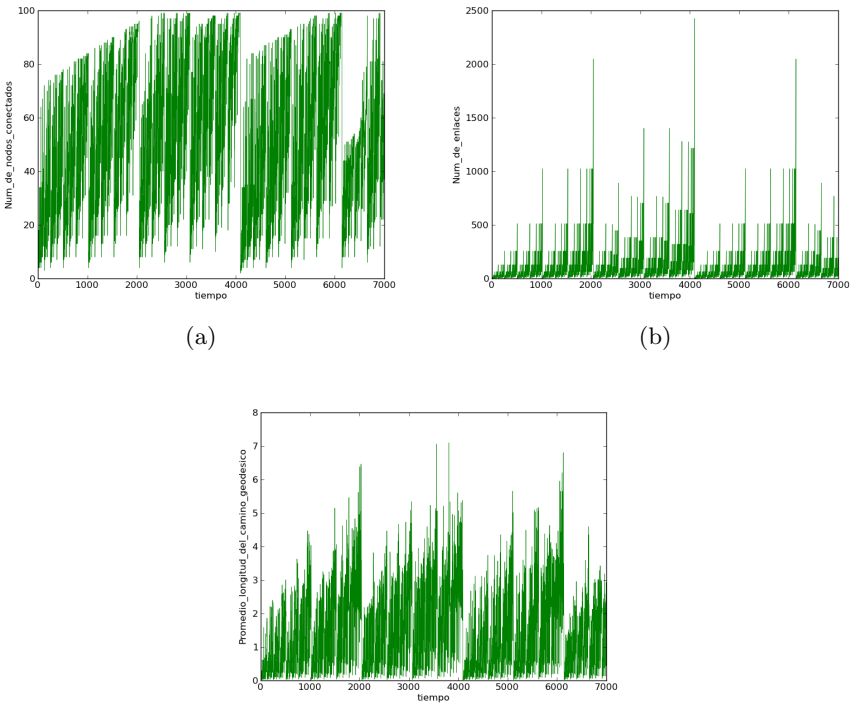


Figura 3. a) Estructura de la red en distintos pasos de tiempo de la evolución de acuerdo a la regla 210 de la AC. a) Número de nodos conectados, b) Número de enlaces c) Promedio de la longitud del camino geodésico.

En las figuras (3a) y (3b) podemos observar respectivamente, el valor del número de nodos conectados y el número de enlaces presentes en la red. En $T = 7000$ pasos de tiempo, el número de nodos conectados y el número total de enlaces presentes en la red aumentan drásticamente durante breves periodos de tiempo y se generan estructuras como las mostradas en la figura (2). En la figura (3c) podemos observar que el valor del promedio de la longitud del camino más corto es grande cuando el número de enlaces es también grande.

4. Discusiones

En este artículo hemos propuesto un modelo de evolución de redes en el cual, las reglas locales determinan la forma en que cambian los enlaces. Dichas reglas están asociadas con las reglas de evolución de AC unidimensionales, con celdas binarias y con frontera periódica. De esta forma, la regla que define la presencia de cada enlace en el tiempo $t + 1$ depende de la presencia o ausencia de dicho enlace y de sus enlaces vecinos adyacentes a la derecha e izquierda en la tupla $e(t)$.

Para ejemplificar el funcionamiento de nuestro modelo, en este trabajo hemos seleccionado la regla 210 (en código Wolfram) de un AC para simular computacionalmente la evolución de una red con un sólo enlace al tiempo cero. Los resultados muestran que bajo esta regla, la evolución de la red es tal que su estructura cambia en distintos periodos de tiempo, y oscila entre escenarios donde el número de nodos conectados y de enlaces es muy pequeño, y escenarios donde estos números se incrementan de manera contondente, generando estructuras como las mostradas en las figuras (2), en las cuales, en su gran mayoría, todos los nodos tienen muchas conexiones. Este comportamiento de la red a lo largo de su evolución es consecuencia de los patrones generados por la regla 210 del AC.

En este trabajo hemos utilizado como una primera aproximación, las reglas típicas para la evolución de los AC unidimensionales con celdas binarias y con frontera periódica. Sin embargo, este primer modelo puede ser extendido para incluir situaciones en las que por ejemplo, un AC cambia el estado de sus celdas de forma asíncrona, cada celda sigue una regla de cambio distinta, e inclusive, una situación donde las celdas tienen más de dos estados, lo cual representa un escenario donde los enlaces tienen distintos pesos. De igual forma, en este modelo se establece la vecindad de cada enlace de acuerdo a su posición en la tupla $e(t)$, lo cual puede ser modificado para incluir una metodología que nos permita definir dicha vecindad. Consideramos importante señalar que varias avenidas de investigación en torno al modelado de la evolución de las redes usando reglas de AC's, están siendo atendidas actualmente.

Referencias

[1] Watts, D. J. & Strogatz, S. H. (1998). Collective Dynamics of Small World Networks, *Nature*, 393, 440-442.

[2] Barabási, A. L. & Albert, R. (1999). Emergence of scaling in random networks, *Science*, 286, 509-512.

[3] Gross, T. & Blasius, B. (2008). Adaptive Coevolutionary Networks: A Review, *Journal of The Royal Society Interface*, 5, 259-271.

[4] Sayama, H. & Laramee, C. (2009). Generative Network Automata: A Generalized Framework for Modeling Adaptive Network Dynamics Using Graph Rewritings, In. *Adaptive Networks: Theory, Models and Applications* (T. Gross & H. Sayama (Eds.)), Springer, 311-330.

[5] Smith, D. M. D., Fricker, M., Johnson, N. F., Lee, C. F. & Onnela, J. P. (2007). Network automata and the functional dynamic network framework, arXiv:0701.307v2.

[6] Wolfram, S. (2002) *A New Kind of Science*, Wolfram Media Inc.

Buscando complejidad y computación en el espacio de polinomios

Todd Rowland

Wolfram Research, Inc.

Resumen Reconsideramos el comportamiento de los polinomios desde la perspectiva de la metodología de Wolfram y mostramos evidencia de que podrían tener posibles usos computacionales del mismo modo que otros sistemas de reglas simples lo tienen.

1. Introducción

En matemáticas, los polinomios tienen un larga historia en la que han sido utilizados como computaciones, y hoy se pueden encontrar en muchos programas modernos. Este trabajo busca, sin embargo, otras maneras en que los polinomios puedan considerarse como modelos de computación, del mismo modo en que puede considerarse que un autómata celular calcula algo más (por ejemplo una función) que su simple evolución [12]. Esto involucra buscar complejidad y estructura en el espacio de polinomios.

Siguiendo la metodología de Wolfram [12] estudiaremos el comportamiento de polinomios como reglas simples. Desde la perspectiva de la aritmética, esto es relativamente sencillo porque sólo hay dos tipos de primitivas: los números y las variables; y dos tipos de operadores: la suma y la multiplicación. Se puede argumentar que incluso éstas son algo complicadas debido a que sabemos que con menos primitivas se puede alcanzar el *umbral de Wolfram* en donde el comportamiento interesante aparece. Debido a esto podría haber alguna duda de si los polinomios forman parte de la categoría de objetos que tiene demasiado diseño como para ser capaces de desplegar máxima sofisticación o la capacidad de computación universal.

Los polinomios han jugado un papel clave en los sistemas complejos. Los mapas de polinomios iterados son objetos de estudio en dinámica compleja [4] y, en particular, el conjunto de Mandelbrot [2, 9] consiste en iterar un polinomio, e involucra sensibilidad a condiciones iniciales. Aquí estamos tratando de tener un enfoque aún más elemental, y dado que estamos buscando computación, nos interesa algo más que la sensibilidad a las condiciones iniciales.

Hay tres ideas principales sobre el trabajo de Wolfram [12] que motivan una nueva búsqueda de los polinomios como modelo de cálculo. La primera es que la noción de computación se extiende a todos los procesos, tanto los de la teoría como los de la naturaleza, y esto implica la posibilidad de cálculos en curso. La atención se centra más en lo que hacen que en lo que producen. En el caso de los

polinomios el reto es encontrar formas de visualizar y comprender sus cálculos internos.

En segundo lugar estamos interesados en el espacio de polinomios, no en cuanto a si tienen la estructura subyacente como espacio, sino en cuáles son los cálculos que realizan y cual es el tipo de espacio computacional que pueden sostener.

Luego está la metodología de búsqueda de reglas que puedan ser utiles para cálculos mediante el uso del *filtro de Wolfram* para distinguir los que son interesantes de los que son triviales. En pocas palabras uno visualiza el comportamiento de una regla simple y si su evolución parece complicada entonces probablemente esa regla tenga capacidad de computación universal (de acuerdo al principio de equivalencia computational o PCE de Wolfram [12]), más aún si se ven estructuras emergentes. De esta manera, este estudio se realiza en el espíritu de la metodología propuesta por Wolfram.

Otros han también investigado la matemática elemental a lo largo de líneas similares, en particular Stedman Wilson en su proyecto de la Escuela de Verano de NKS sobre la suma de senos [10]. Inspirados por Wolfram, han habido otros enfoques sobre polinomios. Una idea ha sido buscar reglas simples que calculen aritmética [5]. Johan Veerman ha también trabajado en autómatas celulares que computan aritmética basados en partículas [6, 7, 8].

Otra idea anterior al trabajo de Wolfram parte de los fundamentos de las matemáticas, donde la indecidibilidad se ha demostrado para la resolución de ecuaciones diofantinas de la forma polinomio=0, donde el polinomio tiene coeficientes enteros y las soluciones deben ser enteros positivos [1, 3].

Ha habido un considerable esfuerzo para la construcción de la ecuación más simple que cumpla dichas condiciones, pero Wolfram tomó la dirección opuesta y buscó a los candidatos que están por arriba de lo que llamamos el *umbral de Wolfram*, la frontera entre la sofisticación de cálculo trivial y máxima. Investigamos seis polinomios de su lista, uno sin soluciones, dos con pequeñas soluciones, una de las soluciones de tamaño mediano, y dos cuya estado de solución se desconoce. Estos últimos son candidatos a ser la más pequeña ecuación diofantina indecidible:

etiqueta	**polinomio**	**ecuación diofantina**	**solución**
a	$x^2 - y^2 - 1$	$x^2 - y^2 - 1 = 0$	None
b	$x^2 - 2y^2 - 1$	$x^2 - 2y^2 - 1 = 0$	$x = 3, y = 2$
c	$x^2 - y - y^5 - 3$	$x^2 - y - y^5 - 3 = 0$	$x = 2537, y = 23$
d	$x^2 - 2y - y^5 - 3$	$x^2 - 2y - y^5 - 3 = 0$	?
e	$x^3 - xy + 1$	$x^3 - xy + 1 = 0$	$x = 1, y = 1$
f	$x^3 + xy - y^4 + 1$	$x^3 + xy - y^4 + 1 = 0$	?

Los polinomios **d** y **f** son los candidatos a ser las más pequeña ecuaciones diofantinas con soluciones indecidibles. Naturalmente, si ambas son indecidibles, sería una cuestión de convención denominar cual es en realidad la más simple. Lo primero que observamos son los valores producidos (Fig. 1).

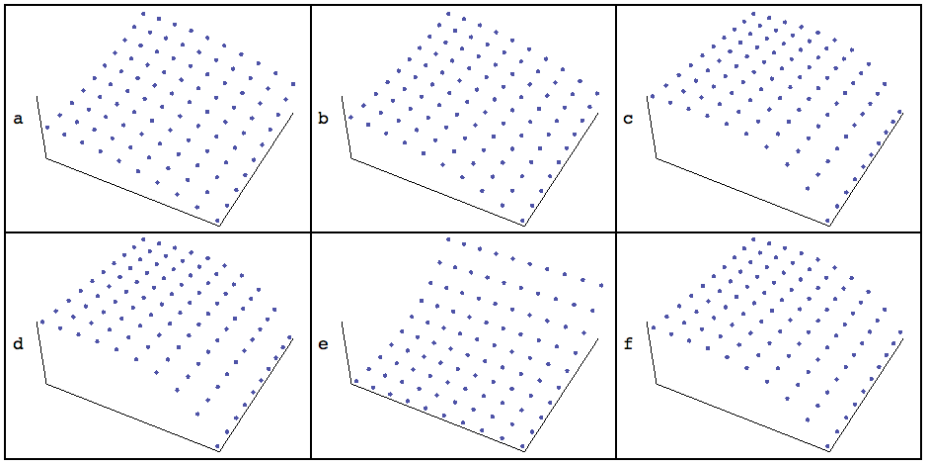


Figura 1. Estos gráficos no muestran nada interesante, lo cual no es sorprendente ya que este es un método tradicional para la visualización de polinomios.

2. Computaciones parciales

A continuación vamos a ver los dígitos binarios que los polinomios producen con sus valores. Si sólo nos interesan los primeros n dígitos que el polinomio produce entonces sólo necesitamos los primeros n dígitos de sus argumentos x y y. De esta manera, sólo hay un número finito de casos a considerar para discusión.

Las imágenes a continuación son apiladas como las imágenes en ([12], p.249). Se alimentan los dígitos de x y y en el polinomio $p(x, y)$ y se determina si los primeros n dígitos son iguales a cero. Si lo son entonces se colorea el cuadrado en (x, y) (ver Fig. 2 y 3).

Esto muestra una cierta cantidad de complejidad. Pero si cambiamos la visualización también podemos ver que aparecen estructuras.

Hacemos esto mediante la inversión de los dígitos de x y y. Uno puede pensar en esto motivado por la norma p-ádica donde los dígitos menos significativos tienen la mayor importancia, pero también desde el punto de vista de la evolución de los autómatas, que comienza con los dígitos menos significativos. Es sólo otro aspecto de la computación polinómica.

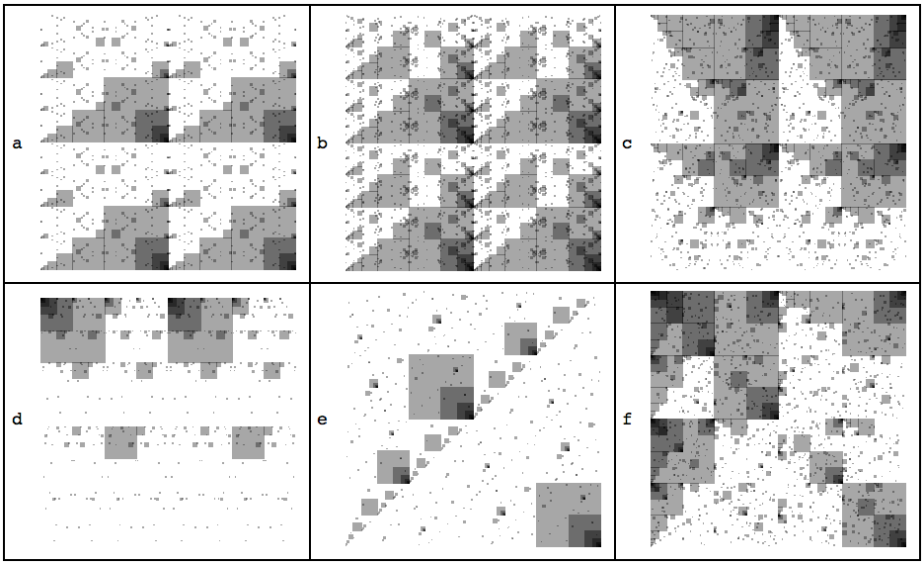


Figura 2. Las celdas grises representan soluciones para dígitos inferiores y las más oscuras son las de soluciones fijas. Para hasta los primeros 9 dígitos.

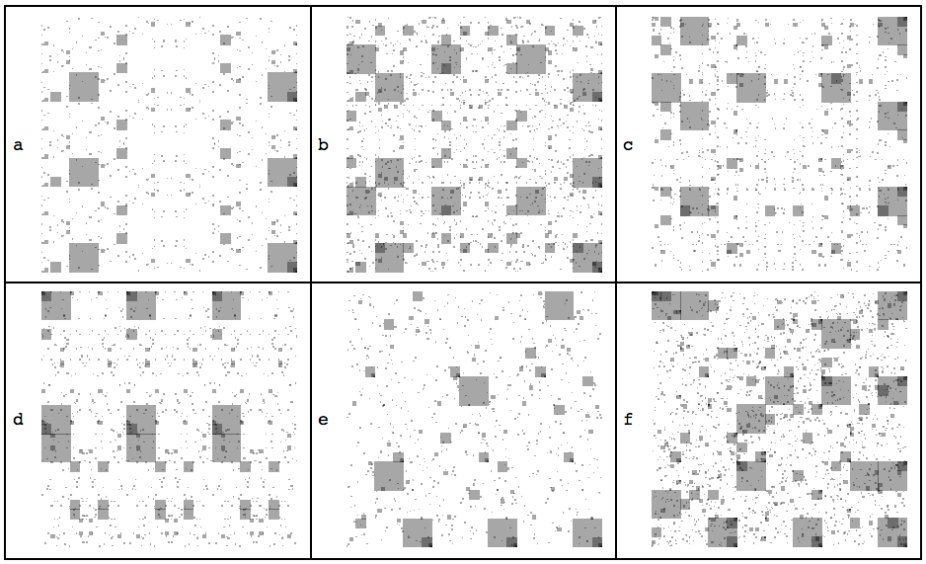


Figura 3. La imagen para base 3, con celdas grises para las soluciones, esta vez hasta 6 dígitos.

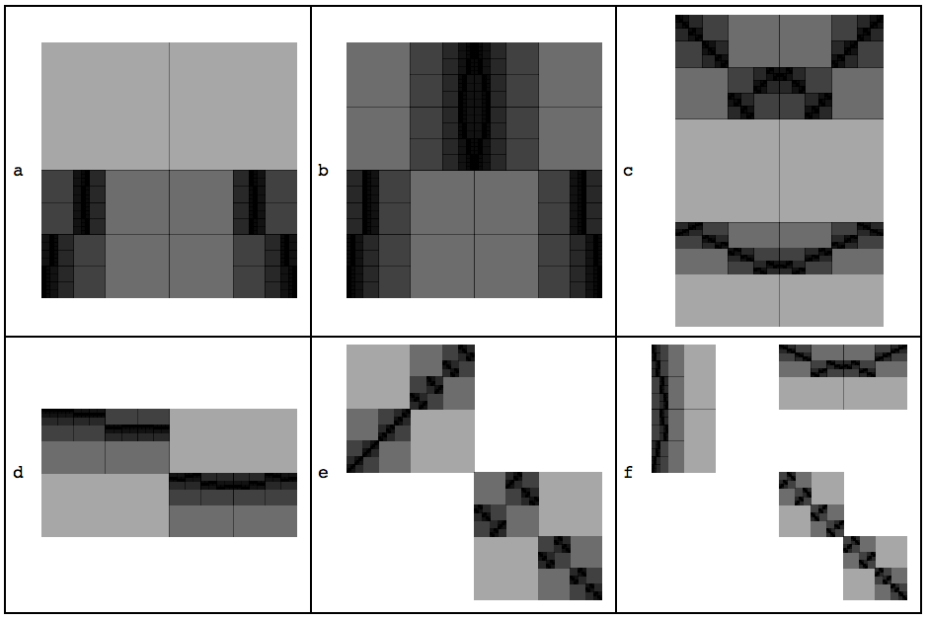


Figura 4. Invierte dígitos binarios soluciones a $n = 9$.

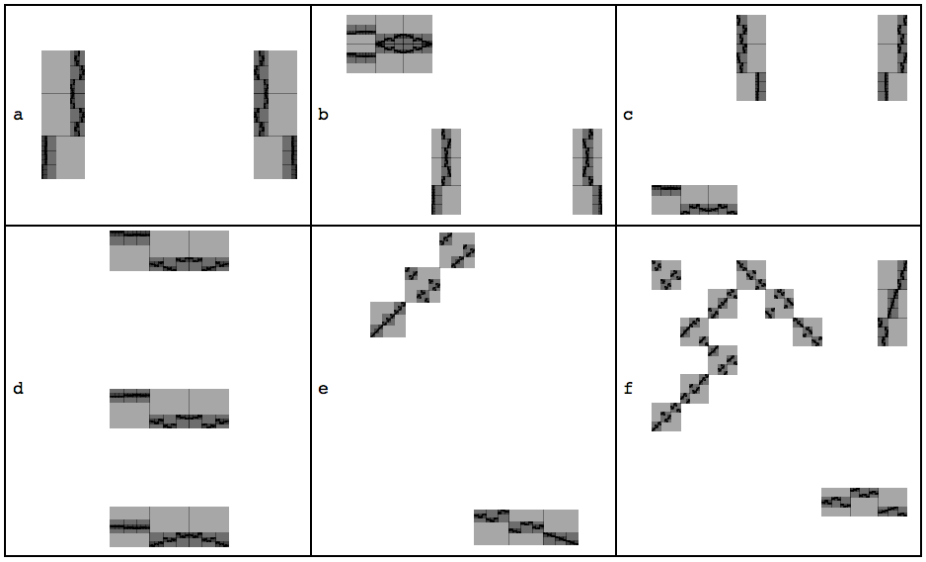


Figura 5. Invierte dígitos ternarios soluciones a $n = 6$.

Estas imágenes (Fig. 4 y 5) tienen una estrecha relación con aquellas producidas por autómatas finitos, por ex. ([12], pp.608-609) y, sin embargo, parecen comportarse de manera muy diferente, lo que justifica la afirmación de que los polinomios pertenecen a las herramientas de los programas de computación simples y pueden considerarse modelos de computación.

3. Emulaciones algebraicas

Recordemos que un proceso computacional puede emular a otro. En el software disenãdo por una persona, un programa comúnmente contiene muchos otros programas. Aquí especulamos que algo similar ocurre con los polinomios.

Consideremos un enfoque ingenuo para encontrar soluciones a una ecuación diofantina escribiendo las soluciones como raíces. Aquí tomamos la raíz real del candidato **f**.

$$\textbf{Solve}\left[1+x^3+xy-y^4==0,x\right][[1,1,2]]$$

$$-\frac{\left(\frac{2}{3}\right)^{1/3}y}{\left(-9+9y^4+\sqrt{3}\sqrt{27+4y^3-54y^4+27y^8}\right)^{1/3}}+\frac{\left(-9+9y^4+\sqrt{3}\sqrt{27+4y^3-54y^4+27y^8}\right)^{1/3}}{2^{1/3}3^{2/3}}$$

Esto equivale a resolver una ecuación cúbica (esta es la solución aportada por *Mathematica* [11]).

Con el fin de obtener una solución entera estas raíces tiene que cancelarse (lo cual es improbable si no imposible) o sus argumentos deben ser potencias perfectas, en otras palabras, este enfoque ha dado lugar a nuevas ecuaciones diofantinas. De esta manera el problema original genera nuevos problemas.

Es conocido que la solución por raíces de polinomios no funciona para ecuaciones de grado mayor a 4. Es posible que al considerar las raíces enésimas como una solución a una ecuación diofantina en particular y extendiendo esto a una clase un poco más general, estos cálculos podrían ser calculadoras universales para polinomios, en analogía con la emulación cómputacional, apoyado por la evidencia de la complejidad de las soluciones de estos polinomios.

4. Conclusión

Sólo le hemos echado una mirada a una pequeña muestra del espacio de polinomios pero aun así podemos creer que comportamiento complejo es común. Aún desde un punto de vista básico, polinomios hacen algo nuevo y entendido poco. Esos comportamientos deben ser accesibles con la metodología de Wolfram. En la última sección sugerimos que las emulaciones algebraicas de los comportamientos de polinomios pueden dar una perspectiva nueva sobre emulación computacional. Así que hemos mostrado que los polinomios merecen un lugar en el universo de programas simples.

5. Agradecimientos

Quiero expresar mi agradecimiento a Johan Veerman y Hector Zenil.

Referencias

[1] Chaitin, G., Foundations of Mathematics, arXiv:math.HO/0203002v2, 2002.
[2] Mandelbrot, B.B., *The Fractal Geometry of Nature,* New York: W. H. Freeman, pp. 188-189, 1983.
[3] Matiyasevich, Yu., *Hilbert's 10th Problem,* MIT Press, 1993.
[4] Milnor, J., *Dynamics in One Complex Variable: Introductory Lectures,* Friedrick Vieweg & Son, 2000.
[5] Rowland, T., "Graphic Addition", posting on forum.wolfram.science `http://forum.wolframscience.com/showthread.php?threadid=262`, March 21, 2004.
[6] Veerman, J., "Two methods for finding cellular automata that perform simple computations" NKS Midwest conference, Bloomington, 2005.
[7] Veerman, J., "Arithmetical Cellular Automata" NKS conference, Washington, D.C., 2006.
[8] Veerman, J., "Further Results in Arithmetical Cellular Automata" NKS conference, Burlington, 2007.
[9] Weisstein, E., "Mandelbrot Set." From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/MandelbrotSet.html`
[10] Wilson, S., "The Distribution of Zeros of Periodic and Aperiodic Sinusoidal Sums" NKS conference, Burlington, 2007.
[11] Wolfram Research, Inc., Mathematica, Version 8.0, Champaign, IL, 2010.
[12] Wolfram, S., *A New Kind of Science,* Wolfram Media, 2002.

Un método estable para la evaluación de la complejidad algorítmica de cadenas cortas

Héctor Zenil[1], Jean-Paul Delahaye[2]

[1] Dept. of Computer Science, University of Sheffield, Reino Unido.
Centro de Ciencias de la Complejidad, UNAM, México.
h.zenil@sheffield.ac.uk
[2] Laboratoire d'Informatique Fondamentale de Lille (LIFL)
Université de Lille 1, Francia.
delahaye@lifl.fr

Resumen Se discute y revisa un método numérico propuesto que, de manera alternativa (y complementaria) al método tradicional de compresión, permite aproximar la complejidad algorítmica de cadenas, particularmente útil para cadenas cortas para las cuales los métodos tradicionales de compresión no son efectivos y dependen de los algoritmos de compresión. El método muestra ser estable ya que produce clasificaciones razonables a partir de modelos de computación razonables, incluidos sistemas de etiquetas de Post, autómatas celulares y máquinas de Turing. Además, permite la concepción (y comparación) de un modelo que predice la distribución de patrones en un mundo algorítmico.[3]

Palabras clave: complejidad de Kolmogorov, probabilidad algorítmica de Solomonoff, teorema de codificación de Chaitin-Levin, semimedida de Levin, máquinas de Turing pequeñas, problema del castor atareado.

1. Introducción

En el estudio de sistemas complejos, es fundamental contar no sólo con definiciones precisas sino también con herramientas para evaluar la complejidad de sus objetos de estudio. El trabajo que hemos desarrollado y publicado en [7], presenta una alternativa, confiable y estable [9], para evaluar la complejidad algorítmica (o de Kolmogorov) de cadenas de caracteres, en particular de cadenas cortas, para las cuales el método tradicional de compresión es, en la práctica, inútil para aproximar su complejidad algorítmica. La complejidad algorítmica de un objeto es la descripción más corta posible que regenera el objeto.

El método introducido en [8, 7] y recientemente difundido en la versión francesa de la revista Scientific American *Pour La Science* [9], provée un nuevo método numérico y efectivo (hasta cierto punto) para la evaluación de la complejidad algorítmica de cadenas. En este artículo lo describimos brevemente en el contexto de su relevancia como herramienta en el estudio e investigación en el

[3] Para WCSCM2011. El autor H. Zenil no actuó como editor activo para este artículo.

área de sistemas complejos con su amplia y diversa gama de posibles aplicaciones debido a que, por primera vez, permite aproximar la complejidad de objetos que generalmente se utilizan en aplicaciones prácticas, esto es, de objetos pequeños, cadenas de longitud corta. Por ejemplo en la compresión de datos, algoritmos de optimización, problemas de reconocimiento y clasificación, por mencionar algunas. La mejor referencia introductoria al tema y aplicaciones es [14]

1.1. Pseudomedidas de complejidad

Algunos métodos atractivos para calcular la complejidad de cadenas cortas son insuficientes porque no coinciden con la complejidad de Kolmogorov de cadenas cuando aumenta la longitud que se requiere para que la medida sea consistente. Por ejemplo, una medida largamente pero erroneamente utilizada es la entropía de Shannon [10] que se define como $-\sum p_i \log p_i$ (donde p_i es una frecuencia). Con esta medida, la cadena 01010101010101010101 es la cadena con mayor entropía de Shannon posible para una cadena de longitud 20 (ya que tiene tantos "0"s como "1"s). Mientras que, la cadena 10010111010100001011, también de longitud 20, tiene la misma entropía de Shannon pero nuestra intuición nos dice que debería ser más compleja.

La entropía de Shannon no hace mas que contar el número de "0"s y "1"s en una cadena ya que nunca fue diseñada para medir la complejidad (o el "orden") de la información contenida en la cadena. Es una medida estadística que ni siquiera es capaz de considerar las repeticiones y que hereda los problemas de la teoría clásica de probabilidades que, precisamente, la teoría de la complejidad algorítmica (la complejidad de Kolmogorov) resuelve. De hecho, la entropía de Shannon es simplemente un corolario de la complejidad de Kolmogorov: si un objeto es más complejo su transmisión toma, potencialmente, más tiempo.

Otras medidas, como la complejidad por factores, por ejemplo, son erroneamente utilizadas cuando no hacen más que cuantificar el número de posibles maneras de ordenar un sistema, son medidas probabilistas que en nada (o muy poco) se relacionan con una medida de complejidad. Si los investigadores en sistemas complejos están interesados en una medida combinatoria (por ejemplo, el número de capas, de elementos en interacción, etc.), con las limitaciones producto de las bases probabilistas en las que se funda (como es el caso, por mencionar otro ejemplo, del parámetro lambda de Langton [11]), los investigadores pueden continuar utilizando medidas *ad-hoc* en el entendido de que no son una medida de complejidad universal (es decir, una medida de complejidad general y objetiva que pueda aplicarse en cualquier situación y a cualquier sistema).

Hoy en día, una amplia variedad de conceptos, basados en medidas como la entropía de Shannon (que Shannon diseño con el propósito de cuantificar el ancho de banda necesario para un canal de comunicación) y otras falsas medidas de complejidad, se emplean para calcular y comparar la complejidad de objetos discretos bajo la falsa idea de que es el número de elementos o interacciones en un sistema hacen que un sistema sea complejo (por ejemplo, en la teoría de sistemas dinámicos, sistemas incluso muy simples resultan comportarse caóticamente, incluso en sistemas de computación deterministas y extremadamente

simples sin interacción con el medio, producen aleatoriedad aparente y impredictabilidad [10]). Algunos autores podrían argumentar que hay otras medidas de complejidad, pero medidas de complejidad como la profundidad lógica de Bennett [2], están fundadas, o bien son variaciones de la complejidad de Kolmogorov que toman en cuenta otros parámetros como el tiempo, la geometría de la evolución de un sistema o son versiones computables [14] (que asumen recursos finitos) e interesantes de la complejidad de Kolmogorov, mientras que la mayoría del resto de las pseudo medidas de complejidad están fundadas en distribuciones de probabilidad o densidad, como la medida de Shannon o el parámetro de Langton.

1.2. Complejidad algorítmica de Kolmogorov-Chaitin

Imaginemos que se nos proporcionan dos cadenas cortas y se nos pregunta cuál de ellas parece ser el resultado de un proceso que genera cada símbolo de la cadena al azar. Digamos que las cadenas son binarias y cortas, por ejemplo 0101 y 1011. A simple vista, la primera cadena tiene un patrón, aunque se repita sólo dos veces, y que podría ser aprovechado para generar una descripción de la cadena. En español, por ejemplo, la primera cadena podría ser descrita como "dos veces cero y uno" (aunque el lenguaje se presta a confusiones, ya que la misma descripción puede interpretarse como 001 si no se conoce la longitud de la cadena[4]). Por otro lado, la segunda cadena parece necesariamente requerir una descripción ligeramente más larga. La primera podría describirse también como "cero seguido de uno seguido de cero seguido de uno". Descripciones de la segunda pueden incluir "uno y cero seguido de dos unos" o "uno, cero, uno, uno", que no parece ésta última una versión comprimida de la cadena, sino más bien una traducción a una forma expandida del idioma. De hecho, pareciera que cadenas con patrones permiten menos descripciones distintas (inténtese, por ejemplo, con cadenas más largas).

Para resolver si alguna de las dos cadenas es, sin lugar a dudas, más sencilla que la otra, o si la aparente repetición de la primera cadena puede realmente aprovecharse a pesar de repetirse sólo dos veces, es necesario fijar un lenguaje objetivo (y que no permita las ambigüedades del lenguaje coloquial). Para determinar cuál de las cadenas parece más aleatoria que la otra bastaría, entonces, comparar sus respectivos valores de complejidad. La complejidad algorítmica de una cadena es el programa más corto, medido en número de bits, que produce una cadena dada cuando se ejecuta en una máquina universal de Turing. Asumimos que el lector está familiarizado con el concepto de máquina de Turing y de máquina universal de Turing. Para una buena introducción véase [16].

[4] En inglés, éste tipo de ambigüedades regularmente se pueden evitar con la introducción de una coma. Así "zero, and one twice" y "zero and one twice", engendran 011 y 0101 respectivamente. En español, la solución es "cero y uno dos veces" versus "cero y uno, dos veces", pero en general la gramática en español (y otros idiomas, por ejemplo, francés) no permite comas antes de la conjunción 'y' lo que no permite resolver todos los casos de ambigüedad de este tipo.

El concepto de complejidad, introducido por Andrei Kolmogorov y Gregory Chaitin define la complejidad $K(s)$ de un objeto s como el tamaño del programa más corto de computadora que genera s. Formalmente,

$$K_U(s) = \text{mín}\{|p|, U(p) = s\}$$

donde $|p|$ es la longitud de p medido en bits. En otras palabras, el tamaño de un archivo comprimido s es la complejidad de s. La complejidad de Kolmogorov (o Kolmogorov-Chaitin, para ser justos) proporciona una medida de *aleatoriedad*.

La complejidad algorítmica es considerada la medida universal de complejidad. Sin embargo, no existe algoritmo efectivo que, para una cadena, el algoritmo produzca el entero que corresponda a la longitud del programa más corto (la mejor compresión posible) que genere la cadena como salida (el resultado se debe al problema de la detención de las máquinas de Turing). Lo que significa que uno no puede medir con absoluta certeza la complejidad algorítmica de una cadena.

El que sea no computable no significa, sin embargo, que no se le pueda utilizar ya que en realidad a menudo se le puede aproximar de manera eficaz. El cálculo del valor aproximado de la complejidad de Kolmogorov, gracias a algoritmos de compresión sin pérdida, hacen del concepto una herramienta de gran utilidad usado en diversas aplicaciones. De hecho existen aplicaciones de la teoría de la complejidad algorítimca que han resuelto problemas de clasificacón de todo tipo de objetos [14, 22], para estudiar la similitud de ciertos idiomas, especies de animales, para detectar fraudes (por ejemplo, plagios) y caracterizar imágenes [24].

1.3. El problema de las cadenas cortas

La complejidad de Kolmogorov permite una caracterización matemática del azar: una cadena aleatoria s de n bits de información es una cadena cuya complejidad de Kolmogorov $K(s)$ es cercana a n. Es decir, el programa que lo produce es de más o menos el mismo tamaño en bits que la cadena original. Una cadena aleatoria infinita es tal que ningún proceso de compresión puede comprimir por más de una constante ningún segmento inicial de la cadena. Por ejemplo, la secuencia infinita 01010101... no es aleatoria, porque uno puede definir de forma concisa la "serie infinita de 01" y, sobre todo, escribir un programa muy corto basado en un bucle que genere la secuencia infinita. La secuencia compuesta de los dígitos de π tampoco son aleatorios: hay un programa más corto que genera todos sus decimales.

La forma de abordar la complejidad algorítmica de una cadena es por medio del uso de algoritmos de compresión sin pérdida. *Sin pérdida* significa que se puede recuperar la cadena original a partir de la versión comprimida por medio de un programa de descompresión. Entre más compresible se considera menos compleja la cadena. Por el contrario, si no es compresible, se le considera a la cadena como aleatoria o máximamente compleja. El resultado de un algoritmo de compresión es una cota superior de su complejidad algorítmica, por lo que se dice que la complejidad de Kolmogorov es computable por *arriba*. Esto quiere decir

que a pesar de que uno nunca puede decir cuando una cadena no es compresible, si uno tiene éxito en la reducción de la longitud de una cadena se puede decir que la complejidad algorítmica de esa cadena no puede ser mayor a la longitud de la versión comprimida.

Para evitar hacer trampa y decir que uno puede comprimir cualquier cadena con un algoritmo de compresión *ad hoc* (por ejemplo, codificando artificialmente ciertas cadenas complicadas con programas cortos interpretados en una máquina universal truqueada) la codificación de la máquina debe ser parte de la complejidad de un objeto cuando es medida con respecto a esa máquina. Un algoritmo de compresión transforma una cadena comprimida en dos partes: una parte es la versión comprimida del objeto original, y la otra las instrucciones para descomprimir la cadena. Ambas partes deben ser contabilizadas en el tamaño final de la cadena comprimida, debido a que se requieren las instrucciones de decompresión para obtener la cadena original sin necesidad de depender de la elección arbitraria del algoritmo (o de la máquina de Turing). En otras palabras, uno puede considerar que agrega el algoritmo de descompresión a la cadena comprimida de manera que la cadena comprimida sea *autodescomprimible* y venga con sus propias instrucciones de descompresión[5]. A la larga, un teorema [14] (llamado de *invarianza*) garantiza que los valores de la complejidad convergen a pesar de la elección arbitraria de lenguajes de programación o la utilización de máquinas de Turing truqueadas (en otras palabras, uno no puede seguir engañando por siempre).

El teorema de invarianza [19, 5] acota la diferencia entre evaluaciones de la complejidad de Kolmogorov calculadas con diferentes máquinas de Turing. Si U y U' son dos máquinas de Turing universales diferentes, el teorema estipula que si $K_U(s)$ es la complejidad algorítmica de una cadena s medida con respecto a una máquina universal U y $K_{U'}(s)$ es la complejidad algorítmica de la misma cadena s medida con respecto a otra máquina universal U' entonces $|K_U(s) - K_{U'}(s)| < c$ donde c es una constante que no depende de s. En otras palabras, la diferencia en las evaluaciones es a lo más la longitud finita de un compilador que pueda escribirse entre U y U'.

Uno requiere la utilización de máquinas universales porque es la única manera de garantizar que la máquina va a producir la cadena que elijamos evaluar y no se esté restringido a poder preguntarse sobre la complejidad de un conjunto limitado de cadenas (por ejemplo, cadenas producidas por lenguajes regulares).

El teorema de invarianza muestra que un sentido amplio la complejidad de Kolmogorov es una medida objetiva y universal. Aunque el teorema de invarianza le da estabilidad a la definición de complejidad de Kolmogorov, también hace evidente que, para cadenas cortas la medida es inestable porque la constante implicada (c), o sea el tamaño de la máquina universal (o las instrucciones de descompresión) dominan el resultado final en la evaluación de $K(s)$. Es decir,

[5] De hecho, algunos programas, como GZIP, permiten la generación de archivos comprimidos ejecutables, que empacan precisamente las instrucciones de descompresión en el programa mismo y no requiere ni siquiera de tener instalado GZIP para descomprimirlo

incluir las instrucciones de descompresión afecta la complejidad relativa de una cadena si la complejidad de la cadena es más pequeña que la longitud de las instrucciones de descompresión, lo que resulta en evaluaciones inestables cuando se trata de cadenas cortas, es decir, cadenas de longitud cercana o menor a la longitud típica de las instrucciones de descompresión (en el orden del tamaño en bits del algoritmo de descompresión).

Hasta ahora, a diferencia de cadenas suficientemente largas para las cuales los método de compresión funcionan, no existía un método para evaluar la complejidad algorítmica de cadenas cortas, y por lo tanto una manera objetiva de determinar si una cadena como 000 es más simple que 01101001 a pesar de que la intuición nos sugiere que la primera parece más simple y la segunda más aleatoria. Nos gustaría decir objetivamente que, por ejemplo, la cadena de 7 bits 1001101 parece más compleja que la cadena 0000000, o que 0001000 tiene una complejidad intermedia a las dos anteriores. Así que tenemos una idea intuitiva de una clasificación de complejidad pero no una medida objetiva que valide la intuición. ¿Cómo hacer que la teoría confirme la intuición, y que sea universal y consistente tanto para cadenas cortas como largas?

1.4. El problema del método de compresión

Para cadenas cortas (que son a menudo las usadas en aplicaciones prácticas), la adición de las instrucciones de descompresión de la versión comprimida hace que la cadena comprimida, con frecuencia, resulte más larga que la versión original. Si la cadena es, por ejemplo, más corta que el tamaño del algoritmo de descompresión, no habrá forma de comprimir la cadena en algo más corto que la longitud original de la cadena, simplemente porque las instrucciones de la descompresión rebasan la longitud de la cadena original (Figura 1). Por otra parte, el resultado depende tanto del tamaño del algoritmo de descompresión (porque en estos casos es el mayor contribuyente a la longitud total) y por lo tanto la longitud (y aproximación de la complejidad algorítmica) es demasiado inestable.

A manera de ilustración, si se trata de comprimir una cadena corta con, digamos, el lenguaje de programación *Mathematica*, se obtiene que la longitud de la versión comprimida de la cadena de longitud 0101 es:

```
StringLength@Compress[‘‘0101’’] = 30
```

(incluso antes de que la versión comprimida sea transformada a bits para que el resultado esté en el mismo lenguaje de la cadena misma)

Esto significa que la compresión de la cadena 0101 requiere de un programa de 46 caracteres (aún más en bits) para ser generada, lo que no tiene sentido alguno, pues la simple descripción en español es más corta que la versión comprimida con *Mathematica*. En *Mathematica*, las cadenas comienzan a ser mejor comprimidas (en caracteres) cuando las cadenas tienen una longitud de 30 bits. Si se trata de comprimir 1011 se llega nada menos que al mismo valor que para 0101, es decir:

```
StringLength@Compress[‘‘1011’’] = 30
```

Éste no es, sin embargo, un fallo de *Mathematica* sino el resultado de lo que hemos explicado. La función Compress en *Mathematica* en realidad está basada en el algoritmo de compresión sin pérdida Deinflate, que es una combinación del algoritmo LZ77 y Huffman, dos de los algoritmos de compresión sin pérdida más populares disponible del mercado, utilizados en formatos públicos como ZIP, GZIP, GIF y PNG.

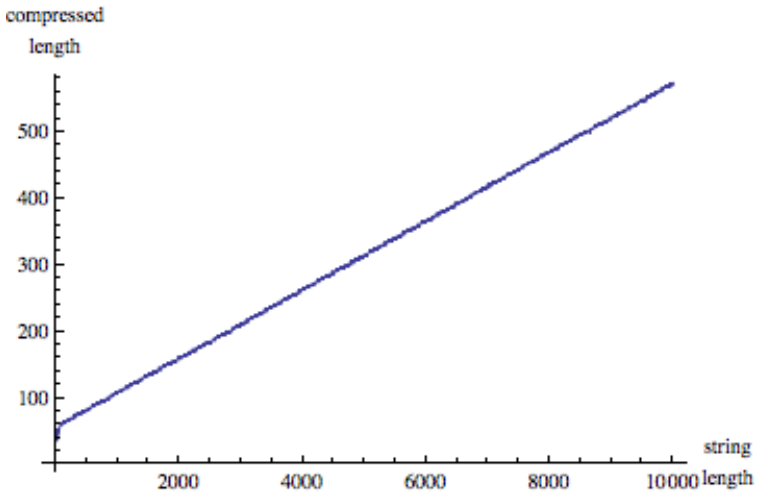


Figura 1. Gráfica de compresión de una cadena de n bits (eje x) contra su versión comprimida (eje y) usando un típico algoritmo de compresión de datos. Al principio de la línea de compresión se observa que el origen no pasa por $y = 0$, incluso cuando $x = 0$, lo que significa que cadenas cortas comprimidas resultan más largas que su tamaño original.

Las instrucciones obviamente ocupan un espacio del valor final de la longitud comprimida y no pueden ser ellos mismos (las instrucciones) comprimidas (si lo fueran, sería en todo caso una longitud constante para todas las cadenas, que nos remiten a la misma situación). En resumen, hay un límite para los algoritmos de compresión para comprimir cadenas cortas. Así que si se quisiera decir cuál de las dos cadenas son objetivamente más o menos complejas por medio de la aproximación de su complejidad algorítmica mediante un algoritmo de compresión, resulta que no hay manera de obtener una respuesta, por el contrario, se encuentra una medida inestable y generalmente sin sentido (Figura 2).

El problema del bit isolado Por ejemplo, dada la definición de la complejidad algorítmica basada en la compresibilidad, si una cadena no es compresible entonces es aleatoria, de donde de inmediato se podría decir que un bit isolado, 0

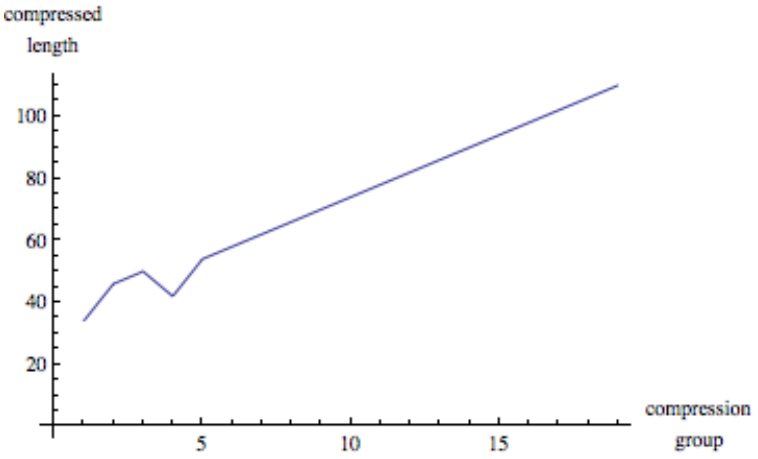


Figura 2. Al hacer un acercamiento al origen de esta gráfica de compresión de una cadena de n "1"s (eje x), contra las diferentes longitudes de sus versiones comprimidas (eje y), se verifica que el inicio es más inestable que el resto.

ó 1, son cadenas al azar con toda certitud, ya que tienen complejidad algorítmica máxima, dado que no hay manera de comprimir un solo bit en algo más corto (siendo el bit indisociable, la unidad mínima y básica). En otras palabras, no hay un programa de menos de 1 bit que produzca 0 o 1. La mejor descripción de 0 y 1 son, por lo tanto, 0 y 1 mismos. Por lo tanto, la teoría dice que son automáticamente cadenas aleatorias. Evidentemente, esto puede chocar con nuestra intuición de azar si se les compara con cadenas más largas y que parecen más el resultado de azar, o al menos se les puede considerar más complejas (por ejemplo, 0110101).

Por un lado, un bit solo no contiene información alguna, y por este motivo uno podría pensar que representa de alguna manera al azar. Si uno piensa si uno habría sido capaz de predecir 0 o 1 como el resultado de un proceso, dado que no hay ningún contexto, ya que se producen solos, se podría concluir que su ocurrencia es de alguna manera el resultado (aparente o no) del azar. En otras palabras, si uno ve una cadena como 010101, uno apostaría fácilmente que el siguiente bit es un 0, pero si no se le proporciona mas que un bit no hay manera de favorecer uno u otro resultado siguiente.

Es difícil, sin embargo, justificar cómo la cadena de un solo bit "0"podía parecer más aleatoria que, digamos, cualquier otra cadena posible, sino es bajo el razonamiento descrito anterior, que se refiere a contextos y no la noción de compresibilidad. La intuición nos dice que las cadenas cortas (incluido "0.º "1") no parecen más aleatorias que cualquier otra cadenas posible, y que si un bit representa la máxima complejidad entre todas las cadenas finitas, y otras cadenas cortas como 000 no son aleatorias, entonces hay una fase de transición

particularmente abrupta entre las cadenas de 1 bit y cadenas de unos cuantos bits más, lo que parece contraintuitivo.

El problema a resolver es, como Delahaye ha señalado en [9], un problema de termómetro: entre todos los instrumentos de medición que conducen a la evaluación de la complejidad de Kolmogorov, pero que difieren por constantes aditivas ¿cuál es el mejor? ¿cómo elegir? Una solución es utilizar el método que hemos diseñado. En lugar de elegir un sólo termometro, o una sola medida, enumeramos todas y nos fijamos en un conjunto suficientemente grande de ellas. El método le va a dar sentido también al problema del bit isolado.

2. La probabilidad algorítmica de Solomonoff-Levin

La intuición nos dice que algo aleatorio también debe ser raro y poco común. Si uno se pregunta qué tan común es 0 o 1 como resultado de la ejecución de un programa elegido al azar, hay una medida que indica la probabilidad de que un programa produzca una cadena determinada si se ejecuta en una máquina universal de Turing. Ésta es la medida que utilizamos para presentar un nuevo método para evaluar la complejidad algorítmica de cadenas, incluyendo cadenas cortas incluso de un bit, como alternativa al uso tradicional de los algoritmos de compresión. El nuevo método tiene como objetivo para resolver el problema de la evaluación de la complejidad de las cadenas cortas, como hemos discutido; y resuelve el problema del bit isolado. Se basa en el concepto de probabilidad algorítmica de Solomonoff -Levin y se conecta con la complejidad algorítmica (Kolmogorov-Chaitin) por medio del teorema de codificación de Chaitin-Levin.

Este nuevo método que resuelve varios problemas que la teoría y los métodos actuales no permitían resolver, tiene una limitación: es muy costoso en términos de computación. Como la duración o las longitudes muy cortas, los objetos de complejidad muy débil son muy difíciles de evaluar y, paradójicamente, el método de evaluación requiere de un cálculo masivo. En la práctica, sólo proporciona resultados para cadenas muy cortas y desde este punto de vista los métodos de compresión siguen siendo esenciales para complementar la necesidad de aproximar la complejidad algorítmica de cadenas largas. Delahaye hace una analogía interesante: Al igual que en astronomía, donde, dependiendo del tipo de objetos y la distancia, se utiliza uno u otro método para calcular distancias, en las medidas de complejidad, nuestro método proporciona una alternativa para cadenas cortas y se pueden adoptar métodos híbridos con la utilización de algorithmos de compresión y técnicas de concatenación (en particular para cadenas de tamaño mediano).

La idea que subyace nuestro nuevo termómetro de baja complejidad para cadenas cortas se basa en una propiedad notable de la complejidad de Kolmogorov: entre más un objeto es simple, más se produce con frecuencia cuando se utiliza una computadora ejecutando programas al azar.

En un lenguaje de programación Turing completo (es decir, en el que cualquier función computable puede implementarse) si cada secuencia de instrucciones es generada de manera aleatoria y ejecutada, muy frecuentemente pro-

ducirá un programa gramaticalmente inválido que no puede si quiera ejecutarse. Otras muchas veces el programa va a comenzar a ejecutarse y no se detendrá jamás. Estos programas no nos interesan, sólo nos interesan aquellos que producen como salida una cadena finita de "0"s y "1"s y se detienen.

Evidentemente si se ejecutan varios programas distintos algunos van a producir la misma salida (para la misma entrada). Si ejecutamos tantos programas como podamos podemos generar una clasificación de frecuencia en donde a cada cadena se le asigna una frecuencia de repetición r de entre todos los programas t ejecutados. Esto define una distribución de probabilidad sobre $\{0,1\}^n$, es decir, sobre todas las cadenas binarias.

Como resultado, obtenemos que cadenas como 0000000 son mucho más frecuentes que cadenas como 1001101. El resultado es una distribución de potencia en donde las cadenas más frecuentes tienen baja complejidad y las menos frecuentes mayor complejidad (son más aleatorias). Esta probabilidad es la medida que Solomonoff [18] y Levin [12] caracterizaron matemáticamente mediante un razonamiento relativamente sencillo (tirar programas al azar). En resumen, para Kolmogorov, la complejidad de una cadena es una longitud mientras que para Solomonoff es una probabilidad.

Formalmente, si $m(s)$ es la probabilidad de producción de s,

$$m(s) = \sum_{p:U(p)=s} 2^{-|p|} = pr(U(p) = s).$$

Es decir, la suma sobre todos programas p que al ejecutarse en una máquina (autodelimitada) universal de Turing U generan s y se detienen, o si se prefiere, la probabilidad de que U corriendo p produzca s.

Una máquina de Turing autodelimitada o prefix-free es una máquina cuyas entradas forman una codificación prefix-free, esto quiere decir que ninguna entrada es el principio de ninguna otra. Esto es para garantizar ciertas propiedades de la medida por varias razones. Más detalles pueden encontrarse en [4, 14]. Una codificación prefix-free es, por ejemplo, el sistema mundial de números telefónicos. Si un número telefónico fuera el principio de uno otro, nunca le sería posible a uno comunicarse con el del número de teléfono más largo pues como funciona la red de telefonía mundial, la secuencia de números que formen un número de teléfono valido es inmediatamente utilizado para realizar la conexión. Imagina por un momento que mi número de teléfono fuera 558973213 y que el de otra persona fuera 558973. Evidentemente, el que intente marcar mi número siempre acabará comunicándose con 558973. Una codificación prefix-free permite poder hablar de la probabilidad de un conjunto de programas que produzcan cierta cadena sin que el conjunto pueda acotarse de alguna forma.

2.1. El teorema de codificación de Chaitin-Levin

Los valores de m están relacionados con la complejidad algorítmica porque el término más grande en la sumatoria es el programa más corto, y por tanto, es $K(s)$ quien domina el total. Un teorema, clave para el método que propusimos

para evaluar la complejidad de Kolmogorov, relaciona matemáticamente $m(s)$ y $K(s)$. El teorema de codificación de Chaitin-Levin establece que $K(s)$ es aproximadamente igual a $-\log_2(m(s))$. En otras palabras, $K(s)$ y $m(s)$ difieren de una constante c, independiente de la cadena s tal que $|-\log_2(\mathrm{m(s)}) - K(s)| < c$.

A groso modo, la probabilidad algorítmica m dice que si hay muchas descripciones largas de cierta cadena, entonces también hay una descripción corta y por lo tanto con baja complejidad algorítmica y si hay pocas descripciones para una cadena, entonces difícilmente tendrá una descripción corta.

Que el logaritmo negativo de $m(s)$ coincida con la complejidad algorítmica de s con una diferencia de una constante significa que aproximar $m(s)$ nos aproxima a $K(s)$. Debido a que ni $K(s)$ ni $m(s)$ son computables, no hay programa que tome s como entrada y produzca m, m tiene que ser también aproximado en lugar de calculado con certeza absoluta.

3. Cálculo de la probabilidad de producción

El cálculo de $m(s)$ se obtiene mediante la ejecución de un gran número de programas que serán producidos al azar, o de un conjunto de programas que se enumeran de manera sistemática (que resulta en lo mismo). Al combinar los resultados teóricos discutidos anteriormente se obtiene una distribución que llamaremos $SL(s)$, y de donde podremos calcular $-\log_2(SL(s))$ para aproximar $K(s)$. SL es una distribución que puede escribirse en función del número de estados de las máquinas que se utilizan para generar la distribución de frecuencia de salida de las máquinas de Turing. Como función, SL no es computable ya que si lo fuera, es decir si se pudiera calcular numéricamente SL para cualquier número de estados de máquinas de Turing, se podría resolver el problema del castor atareado para cualquier número de estados, lo que se sabe es imposible por contradicción con el resultado de Rado de incomputabilidad de las funciones del castor ateareado.

3.1. Máquinas de Turing pequeñas

Usamos un método de cálculo tan primitivo como posible, pero lo suficientemente poderoso para que cualquier programa pueda ser ejecutado y cualquier cadena producida. Introducido por Alan Turing en 1936, el modelo de las máquinas de Turing ha desempeñado un papel fundamental en la ciencias de la computación y la lógica matemática, ya que ha permitido el estudio de lo que es un algoritmo y, en la práctica, del desarrollo de la computadora digital. El modelo de Turing puede verse como un lenguaje de programación; la descripción de una máquina de Turing es equivalente a escribir un programa de cómputación.

Las máquinas de Turing son el modelo de computación más conocido, debido a que es un modelo que tiene una representación física cuya motivación fue la descripción de un humano que calculara con lápiz y papel. Podemos verlas como una abstracción de nuestras computadoras. Disponen de una cinta de longitud ilimitada dividida en celdas discretas (análoga a la tira de papel donde escribía

el computador humano) sobre la que se sitúa una cabeza capaz de leer y escribir en la celda donde se encuentra. La máquina sólo lee y escribe un conjunto finito de símbolos conocido como su alfabeto. Entre estos símbolos hay uno llamado usualmente blanco que es el que por defecto llena todas las celdas de la cinta. Existe un conjunto finito de estados en los que puede encontrarse la máquina. Uno de tales estados es el estado inicial desde el que comienzan todas las computaciones. También suele haber un estado de detención, que cuando se alcanza se termina la computación. En cada paso de computación, la máquina de Turing:

1. Lee el símbolo escrito en la celda sobre la que se encuentra la cabeza.
2. En función del símbolo leído y del estado actual de la máquina:
   a) Escribe un nuevo símbolo en la celda (puede ser igual al que había).
   b) Se desplaza una posición a la izquierda o derecha sobre la cinta.
   c) Cambia de estado (o permanece en el mismo).

Así se continúa hasta llegar al estado de parada. Lo que caracteriza las computaciones de una máquina de Turing es su tabla de transiciones. Si vemos la enumeración anterior, el comportamiento en cada paso de computación dependerá del estado en que se encuentra la máquina de Turing y el símbolo leído.

Las máquinas de Turing constituyen el ejemplo más conocido de dispositivo de computación abstracto capaz de computación universal, lo que significa que para cualquier función efectivamente calculable existe una máquina de Turing que la calcula. Especialmente interesantes son las máquinas de Turing universales, capaces de simular la computación de cualquier otra máquina de Turing.

El número de estados de una máquina de Turing determina su poder de cálculo. Máquinas con un estado sólo pueden hacer cálculos sencillos, tales como invertir 0 a 1 y 1 a 0 a una cadena que se les presenten. Una máquina que dispone de 2 estados comienza a hacer cosas más interesantes. El número de máquinas con 2 estados es, curiosamente, 10 000 que por supuesto sólo pueden generar cadenas muy cortas (no más largas que el máximo número de pasos que una máquina que se detiene puede alcanzar y que es acotado por el número de estados). Al operar todas las máquinas de Turing de 3 estados, el número de máquinas comienza a crecer de manera colosal pero éstas generan cadenas más largas que permiten el cálculo de frequencia, y por tanto la probabilidad de producción de cadenas un poco más largas.

El problema de la detención Sin embargo, las máquinas de Turing pueden detenerse o no dependiendo si entran en el estado de detención de su tabla de instrucciones (y del contenido de la cinta, que en este caso es siempre blanco). Aunque Alan Turing demuestra la existencia de una máquina de Turing universal, es decir, una máquina de Turing capaz de simular cualquier otra máquina de Turing, también muestra que no existe una máquina de Turing que pueda determinar si cualquier otra máquina se detendrá. A este problema se le conoce como el problema de la detención.

Evidentemente, si uno está interesado en la salida de una máquina de Turing, definida como el resultado de lo que contiene su cinta una vez que se detiene. Si no es posible saber si una máquina se va a detener no hay manera de determinar con certeza su salida, ni la frecuencia de una cadena en general.

Una manera elegante y concisa de representar el problema de la detención es el número de Chaitin Ω [5] (un número irracional entre 0 y 1), cuyos dígitos en su expansión binaria es la probabilidad de detención de una máquina de Turing universal corriendo programas al azar. Formalmente,

$$0 < \Omega = \sum_{p \ se \ detiene} 2^{-|p|} < 1$$

con $|p|$ la longitud de p en bits[6].

3.2. El problema del castor atareado

De entre las máquinas que se detienen una pregunta, realizada por Rado [17], es cuál máquina de n estados (y 2 símbolos) escribe más símbolos o le toma más tiempo para detenerse a partir de una cinta en blanco. Al máximo número de pasos se le asigna un número $S(n)$ que depende solamente del número de estados n y se le llama a dicha máquina un *castor atareado* (o *busy beaver* en inglés) comúnmente denotado por $B(n)$.

Ahora bien, si se conoce el valor $S(n)$ para $B(n)$ cualquier máquina que corra más de $S(n)$ es una máquina que no se detendrá nunca. Así que basta ejecutar cada máquina para saber si se detiene o no. Rado demuestra, sin embargo, que la función $n \to S(n)$ no es computable, es decir, no existe un algoritmo (o máquina de Turing) que dado un número de estados produzca el número $S(n)$.

El número Ω de Chaitin, el castor atareado, la probabilidad algorítmica y nuestro método, están todos íntimamente relacionados. Para máquinas de Turing pequeñas, el problema de la detención se puede resolver porque, por un lado, porque no son relativamente muchas y uno puede ya sea ejecutar todas las máquinas y examinar su comportamiento o examinar la tabla de instrucciones de la máquina y decidir analíticamente si se detiene o no. Sin embargo, la secuencia de números del castor atareado, $S(1)$, $S(2)$, ... crece más rápido que cualquier secuencia computable. Porque si una máquina de Turing pudiese computar una sucesión que crece más rápido que el castor atareado, entonces dicha secuencia, paradójicamente, resolvería el problema del castor atareado.

Es fácil verificar que para $B(1)$, $S(1) = 1$ pues no hay mucho lugar para cualquier otro comportamiento más complicado. Con dos estados, Rado determina que $S(2) = 6$ y unos años después, junto con Lin [15], probaron que $S(3) = 21$ requiriendo un análisis exhaustivo y un importante poder computacional. Brady [3], usando técnicas de análisis más sofisticadas y aún un mayor poder computacional prueba que que $S(4) = 107$, pero el valor de $S(5)$ es desconocido, aunque se conocen algunas cotas.

[6] La definición precisa requiere que la máquina de Turing universal sea *prefix-free* (para mayor información véase [4])

Un programa que muestra los valores de los castores atareados y su evolución está disponible en línea [25].

4. Evaluando la complejidad de cadenas cortas

El hecho de conocer los valores del castor atareado permite acotar el cálculo sistemático y masivo de un gran número de máquinas de Turing para producir una clasificación de frecuencia de cadenas binarias. Esta consideración es, por supuesto, esencial para no perder tiempo innecesariamente en funcionamiento de máquinas que no contribuyen a los resultados deseados. Para máquinas de Turing con 3 estados, por ejemplo, cualquier máquina que ejecute más de 22 pasos, es una máquina que no se dentendrá nunca, ya que para $B(3)$, $S(3)$=21. Para 4 estados $S(4) = 107$, pero no se conoce $S(n)$ para $n > 4$ y por lo tanto nuestro experimento exhaustivo sólo puede realizarse a lo más para todas las máquinas de 4 estados.

Para las 7 529 526 máquinas de Turing con 2 símbolos (0 y 1) y 3 estados, los resultados de $SL(s)$ comienzan a arrojar indicios de un ordenamiento no trivial de una clasificación de complejidad para cadenas binarias. Por ejemplo, entre las cadenas binarias de longitud 6, el cálculo de $SL(s)$ produce, mediante la ejecución de máquinas de Turing con 3 estados, las cadenas 000000 y 111111 con la más alta probabilidad (y por lo tanto la más baja complejidad algorítmica), que es lo que uno podría esperar. Seguido en orden de las siguientes cadenas: 000001, 100000, 111110 y 011111 con igual frecuencia, seguidas de 000100, 001000, 111011 y 110111 con igual frecuencia, seguidas de 001001, 100100, 110110 y 011011, 010110 y finalmente por el conjunto 101001, 100101 y 011010. Esta clasificación es bastante sutil, pero natural para colocar la cadena que puede describirse como un 1 detrás de cinco 0 como más simple que un 1 con tres 0 delante y dos detrás. La clasificación obtenida con la máquinas de Turing tiene 3 estados producen sólo 128 diferentes cadenas con las cuales comparar. Esto deja a las demás cadenas un poco más largas, pero aún cortas, sin probabilidad. Por lo tanto, ejecutamos las máquinas de Turing con 4 estados para obtener un mayor número de cadenas y generar una clasificación más completa y fidedigna. Para ello fue necesario correr 11 019 960 576 máquinas de Turing, que a pesar de ciertos métodos para acortar su cálculo llevo casi 9 días (en una sola computadora portatil con un procesador Duo a 1.2 Ghz. y 4Gb de memoria RAM) usando un programa escrito en lenguaje C, mediante la libreria *bignum* para grandes números ya que las tablas de transición de cada máquina se generaban en tiempo real a partir de una enumeración, ya que generar las reglas de manera combinatoria y almacenarlas resultaba, para este número de máquinas, imposible para cualquier disco duro actualmente en el mercado (sin mencionar el tiempo de lectura que para cada máquina tomaría). Algunas simetrías pudieron ser explotadas (por ejemplo, para toda regla de una máquina de Turing existe una que es su complemento y basta calcular el complemento de su salida para conocer el resultado antes de correrla) reduciendo el tiempo de cómputo a los 9 días mencionados.

Las máquinas de Turing con 4 estados producen 1824 diferentes cadenas binarias que permiten la aproximación de $K(s)$ a través de $SL(s)$ y la fórmula $-\log_2(SL(s))$. Hay que tener en cuenta que la complejidad de Kolmogorov calculada a través de SL es un número real. Esto es realmente una ventaja ya que permite una clasificación más fina, pero si se quiere interpretar el resultado como la longitud de un programa basta tomar el siguiente entero.

Así que valores exactos pueden ser numéricamente aproximados mediante el uso delos valores conocidos del castor atareado hasta $n = 4$ y hemos publicado las tablas completas en línea [1] y tablas parciales en [7].

4.1. Un método estable, autómatas celulares y un modelo de distribución de patrones

Una pregunta fundamental, y evidente, es qué tan estable y robusto es el método si se utilizan diferentes formalismos de cómputo (por ejemplo, usando máquinas de Turing con una cinta ilimitada en una sola dirección o utilizando automatas celulares en lugar de máquinas de Turing). Hemos mostrado que formalismos de cómputo razonables producen clasificaciones de complejidad razonables [23]. Las mismas distribuciones de frecuencia fueron producidas ejecutando y explorando un espacio representativo de automatas celulares unidimensionales con 2 colores y rango 3/2, es decir, el espacio de autómatas celulares que utilizan en sus reglas de producción el estado de 2 celdas a la izquierda y una a la derecha, y con condición inicial la más simple posible (una celda negra). Este espacio de autómatas celulares, que es justo en tamaño el espacio siguiente más grande al de autómatas celulares elementales [10] (es decir, unidimensionales, con rango 1 y 2 colores posibles) nos permitió hacer una exploración del tipo de clasificaciones de complejidad de cadenas que producen. Se eligió este espacio porque el espacio siguiente más simple es el de los autómatas celulares elementales definidos por Stephen Wolfram [10] que no contiene mas que 256 autómatas celulares y por lo tanto un número no muy significativo. Evidentemente tanto para autómatas celulares, como máquinas de Turing o cualquier otro formalismo de computación, entre más número de autómatas explorados mejor. Sin embargo, las restricciones en tiempo y recursos de computación no permiten obtener resultados en un tiempo razonable, ni añaden necesariamente mayor información al modelo descrito.

A diferencia de las máquinas de Turing, los autómatas celulares favorecen ciertas simetrías por la manera en que los autómatas celulares evolucionan aplicando su regla en paralelo sobre todas las celdas al mismo tiempo. Además, también a diferencia de las máquinas de Turing, los autómatas celulares no tienen un estado de detención (las cadenas que producen no contienen, por lo tanto, la *información* de su los límites extremos que contiene una cadena producida por una máquina de Turing que se detiene) ya que un automata celular se detiene en un tiempo arbitrario decidido por el que lo ejecuta (en nuestro caso, cada autómata celular fue detenido arbitrariamente después de 100 pasos). Sin embargo, las clasificaciones producidas mediante autómatas celulares (y también sistemas de etiqueta de Post) resultaron parecerse unas a otras, la

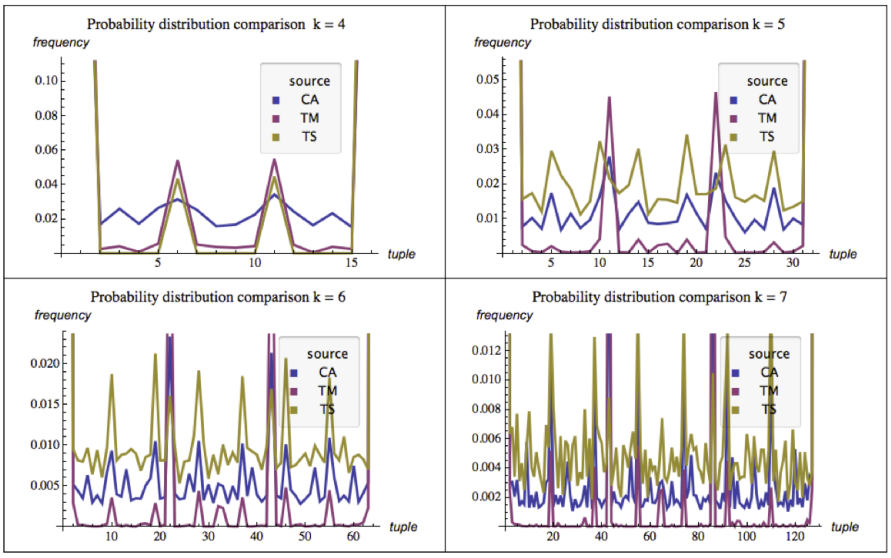


Figura 3. Comparación de k-tuplas generadas por autómatas celulares (CA), máquinas de Turing (TM) y sistemas de etiquetado de Post (TS). Las tuplas están ordenadas por orden lexicográfico.

similitud fue cuantificada estadísticamente mediante el coeficiente de correlación de Spearman [20] (coeficiente diseñado para comparar clasificaciones). El hecho de que las clasificaciones no sólo sean razonables con respecto a nuestra intiuición de lo que es complejo o simple (por ejemplo, las cadenas 000... y 010101... aparecen con baja complejidad aleatoria mientras que cadenas que parecen aleatorias intuitivamente, lo son también en las clasificaciones), sino además estén correlacionadas estadísticamente (Figura 4.1) y sean compatibles con la definición universal de complejidad algorítmica, proporcionan un método eficaz, general y estable (las tablas con las clasificaciones completas están disponibles en `http://www.algorithmicnature.org`).

El método descrito tiene, por un lado, la remarcable característica de resolver un problema teórico (el de la estabilidad de la definición y evaluación de la complejidad de cadenas cortas, por ejemplo, el problema del bit isolado) y permite, en la práctica y gracias a su estabilidad y consistencia, la comparación de la complejidad de distintas clasificaciones. Esta última ventaja permite, por ejemplo, comparar la distribución de patrones (cadenas con cierta complejidad) presente en el mundo físico (a partir de fuentes de información empírica) y la clasificación producida por medios algorítmicos (utilizando las máquinas de Turing o algún otro formalismo). Sus similitudes y diferencias podrían decirnos qué tanto las estructuras que se forman en el mundo real pueden ser el resultado de procesos algorítmicos a diferencia de, por ejemplo, procesos mayoritariamente aleatorios.

En [23] y [26] nos hemos formulado estas preguntas, y esbozado un inicio de ruta de investigación.

5. Comentarios finales

El avance del programa de investigación que se describe aquí pone a disposición un método general para calcular la complejidad de Kolmogorov.

En el artículo de *Pour La Science* [9] Delahaye señala:

> Pour les petites séquences, cette mesure est stable et conforme à notre idée de la complexité, et, pour les grandes, elle est, d'après le théorème mentionné conforme à la mesure de meilleure mesure de complexité unanimement admise, la complexité de Kolmogorov. Que demander de plus?
> (Para cadenas cortas, esta medida [la medida que describo en este artículo, nuestro comentario] es estable y se ajusta a nuestra idea de la complejidad, y, para largas cadenas, de acuerdo con el teorema mencionado [el teorema de invarianza, nuestro comentario], se ajusta a la mejor y universalmente aceptada medida de la complejidad, la complejidad de Kolmogorov. ¿Qué más se puede pedir?)

El método pretende ser utilizado para evaluar la complejidad de cadenas más largas mediante su descomposición en subcadenas más cortas para las cuales podemos calcular su complejidad y generar una aproximación de la complejidad de la cadena original. Tampoco es necesario recorrer espacios completos para aproximar un valor de complejidad. Muestreos del espacio de máquinas de Turing con 5 estados, espacios de autómatas celulares unidimensionales con rangos de vecindad más grandes y otros formalismos de computación, como sistemas de substitución, pueden utilizarse. De hecho una pregunta abierta, es qué tanto pequeñas diferencias en un mismo formalismo impactan las medidas de complejidad. Por ejemplo, si se les permite a las máquinas de Turing quedarse en la misma celda o no moverse más que en una dirección de la cinta (variantes que preservan universalidad).

A manera de conclusión, Chaitin ha expresado [6] que (hablando de los resultados de nuestro método):

> ...the dreaded theoretical hole in the foundations of algorithmic complexity turns out, in practice, not to be as serious as was previously assumed.
> (...el agujero teórico terrible en los fundamentos de la complejidad algorítmica resulta, en la práctica, no ser tan grave como se suponía anteriormente).

Sin embargo, lo cierto es que estamos muy lejos de haber sacado todas las conclusiones y aplicaciones posibles.

Agradecimientos

Los autores agradecen a los dos revisores cuyos comentarios y sugerencias fueron de gran valor para mejorar la manera de comunicar los resultados explorados en este artículo. Cualquier error en él, sin embargo, es exclusiva responsabilidad de los autores.

Referencias

[1] *Algorithmic Nature* research program `http://algorithmicnature.org`.
[2] Bennett, C.H. Logical Depth and Physical Complexity in *The Universal Turing Machine–a Half-Century Survey*, editado por Herken, R. Oxford University, pp. 227–257, 1988.
[3] Brady, A.H. The determination of the value of Rado's noncomputable function Sigma(k) for four-state Turing machines, *Mathematics of Computation*, Vol. 40, No. 162, pp. 647–665, 1983.
[4] Calude, C.S. *Information and Randomness: An Algorithmic Perspective.* (Texts in Theoretical Computer Science. An EATCS Series), Springer, 2nd. edition, 2002.
[5] Chaitin, G.J. A Theory of Program Size Formally Identical to Information Theory, *Journal of the ACM,* No. 22, 1975.
[6] Chaitin, G.J. Reporte de la tesis de H. Zenil, Université de Lille, 2011.
[7] Delahaye, J.-P. and Zenil, H. Numerical Evaluation of Algorithmic Complexity for Short Strings: A Glance Into the Innermost Structure of Randomness. arXiv:1101.4795v4 [cs.IT], 2011.
[8] Delahaye, J.-P. y Zenil, H. On the Kolmogorov-Chaitin complexity for short sequences, en Calude, C.S. (ed.) *Randomness and Complexity: from Chaitin to Leibniz.* World Scientific, p. 343–358, 2007.
[9] *Pour La Science* (edición francesa de Scientific American), No. 400, 2011.
[10] Shannon, C.E. A Mathematical Theory of Communication. *The Bell System Technical J.* 27, 379–423 and 623–656, 1948.
[11] Langton, C.G. Computation at the edge of chaos. *Physica D*, 42, 1990.
[12] Levin, L. Laws of information conservation (non-growth) and aspects of the foundation of probability theory, *Problems in Form. Transmission* 10, 206–210, 1974.
[13] L. Levin. Universal Search Problems. 9(3): 265-266, 1973 (c). (submitted: 1972, reported in talks: 1971). English translation in: B.A.Trakhtenbrot. *A Survey of Russian Approaches to Perebor (Brute-force Search) Algorithms.* Annals of the History of Computing 6(4): 384-400, 1984.
[14] Li, M. y Vitányi, P. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer, 3rd. Revised edition, 2008.
[15] Lin, S. y Rado, T. Computer Studies of Turing Machine Problems. *J. ACM* 12, 196–212, 1965.
[16] Minsky, M. *Computation: Finite and Infinite Machines.* Prentice–Hall, 1972.
[17] Rado, T. *On noncomputable Functions. Bell System Technical J.* 41, 877–884, May 1962.
[18] Solomonoff, R. J. A formal theory of inductive inference: Parts 1 and 2. *Information and Control*, 7:1–22 y 224–254, 1964.
[19] Kolmogorov, A. N. Three Approaches to the Quantitative Definition of Information. Problems of Information Theory 1, 1965.
[20] Kendall, M.G. Rank correlation methods. *Griffin*, 1962.

[21] Wolfram, S. *A New Kind of Science,* Wolfram Media, 2002.
[22] Zenil, H. Compression-based investigation of the behavior of cellular automata and other systems, *Complex Systems* (19)2, 2010.
[23] Zenil, H. y Delahaye, J.-P. On the Algorithmic Nature of the World, en Dodig-Crnkovic, G. and Burgin, M. (eds.) *Information and Computation.* World Scientific, 2010.
[24] Zenil, H., Delahaye, J.-P. and Gaucherel, C. Image Characterization and Classification by Physical Complexity, por aparecer en *Complexity.*
[25] Zenil, H. "Busy Beaver", Wolfram Demonstrations Project `http://demonstrations.wolfram.com/BusyBeaver/`.
[26] Zenil, H. The World is Either Algorithmic or Mostly Random, *the FXQi Contest: Is Reality Digital or Analog?* ensayo ganador del tercer premio, 2011.

Una nueva familia de sistemas tipo Collatz

Enrique Zeleny Vazquez

Wolfram Research, Inc.
ezelenyv@gmail.com

Resumen Se presentan sistemas similares al problema de Collatz, diferentes a generalizaciones conocidas y a las introducidas por Wolfram[1,2], cuya definición no es artimética. Se estudian los grafos dirigidos para entender cómo convergen y a que tipos de secuencias; se hallan recurrencias lineales y funciones generadoras; así como algunas propiedades y formulaciones alternativas útiles para realizar emulaciones.

1. Introducción

El problema de Collatz[3] establece que la iteración de la función

$$f(n) = \begin{cases} 3n+1 & \text{si } n \text{ es impar} \\ n/2 & \text{si } n \text{ es par} \end{cases} \tag{1}$$

siempre termina en 1, para cualquier número positivo; fue originalmente propuesto por Lothar Collatz en 1937 y hasta la fecha ha resistido a todo intento de demostración.

Una característica de este sistema es que en su evolución, en una serie de pasos aumenta y en otra disminuye de longitud la secuencia de dígitos de su expansión binaria repetidamente y de manera irregular.

El programa similar en su comportamiento al sistema de Collatz que se estudia en este trabajo y que se muestra a continuación y posteriormente se describe con palabras (que resultan más simples que tratar de escribirlo en pseudocódigo) es:

```
If[OddQ[FromDigits[#1, 2]], Mod[Differences[#1], 2],
  Prepend[RotateRight[#1], 1]] &
```

En el caso en que el valor inicial es impar, se toman las diferencias sucesivas de la secuencia, módulo 2, y si es par, mueve el dígito del final al principio, y se antepone un 1 a la secuencia resultante. Desglosando el programa, `If` es el condicional que evalúa si el resultadode `OddQ` es un valor impar del entero que `FromDigits` extrae de una secuencia en base 2 (`#1` representa el sitio donde se introduce como argumento la secuencia binaria), en caso afirmativo, `Differences` procesa las diferencias sucesivas módulo 2 usando `Mod`, en caso contrario, `RotateRight` mueve el dígito final de la derecha de la secuencia y lo rota a la primera posición y `Prepend` le agrega un 1 al comienzo.

Veamos un ejemplo: comenzamos con el valor inicial 13, que corresponde a la cadena 1101, como es impar, se restan sucesivamente 1-1, 1-0 y 0-1 módulo 2 (o valor absoluto, si se prefiere), que produce 011; después 0-1,1-1, produciendo 10, que es par, entonces el cero del final se agrega al principio anteponiendo un 1, quedando 101, que es igual a 5 que es impar y produce 3 (11, sin cero a la izquierda), que finalmente termina en 0, que jugaría el papel de nuestro estado de detención. Resumiendo, tenemos la evolución.

1101
011
10
101
11
0

El símbolo & denota simplemente que se trata de una función. Definimos el sistema como programa porque no existe en la notación matemática tradicional símbolos para operaciones como "diferencias" y otras operaciones muy simples y generales que existen en el lenguaje de *Mathematica*.[1]

En la figura 1 se puede ver un ejemplo del comportamiento de la secuencia, que va disminuyendo su longitud hasta llegar a una secuencia de 5 ceros, a partir de ese momento, el comportamiento se vuelve repetitivo y se forman una serie de barras en forma de triángulo que crecen cada vez más.

En el grafo de la figura 2, vemos las trayectorias que siguen algunos números menores que 100 como valor inicial. El grafo rápidamente se vuelve muy complejo y aparecen secuencias de números muy grandes a menudo disconexas del resto del grafo, pero que al incluir un mayor número de valores iniciales y aumentar el número máximo de pasos, se conectan; debido a esto, algunos números menores que 100 no aparecen en el grafo, además de otros detalles que se discuten en la sección 2.

En algunos casos, las trayectorias son muy largas y en algunos intervalos no parece claro si va a converger, por ejemplo, para un valor inicial grande como 8618126181, se requiere casi 9000 pasos para converger.

Otro detalle del grafo es que para algunos enteros, aparece más de una flecha hacia otros enteros porque también pueden generarse secuencias con ceros a la izquierda, que produce un número diferente que si no los tuviera.

Éste es otro ejemplo de un programa tipo Collatz,

```
If[OddQ[FromDigits[#1, 2]], Prepend[Accumulate[#1], 1],
 BitXor[Most[#1], 3]]&
```

similar al anteriormente presentado, pero que utiliza la operación `Accumulate`, donde se obtienen los totales acumulados de los dígitos de la secuencia, es decir, se toma el primer dígito, luego se suman los primeros dos, posteriormente los tres primeros y así sucesivamente; que en un caso como el de una cadena como

[1] `http://www.wolfram.com/mathematica/`

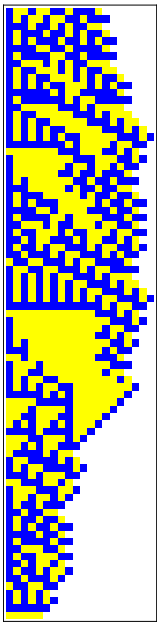


Figura 1. Evolución para la condición inicial 4718.

1011, produce la cadena 1123. También se introduce la operación lógica `BitXor` actuando sobre la secuencia de dígitos pero removiendo el último, que se obtiene con `Most`. Para ilustrar como actúa el comando `BitXor`, tomemos el 5 y el 3, que en binario serían 101 y 011 (los ceros a la izquierda son para que las secuencias tengan la misma longitud), 01 y 10 dan 1, y 00 y 11 dan 0 (0 es falso y 1 verdadero), entonces el resultado de concatenar las secuencias sería 110, que corresponde al 6. La evolución del sistema se puede ver en la figura 3.

Éstos y otros programas fueron hallados utilizando un programa que genera combinaciones de un conjunto de diferentes instrucciones de *Mathematica* con diferentes estructuras de árbol y ayuda a visualizar el tipo de secuencias que produce el programa. Incluso podrían generalizarse más estas secuencias considerando otras propiedades que ser par o impar.

2. Convergencia

El programa converge a una secuencia de uno o más ceros, que entran en un ciclo $0, 2, 5, 3, 0, ...$, en el caso del 1, se indefine porque se requieren al menos dos dígitos para poder realizar otra iteración. También existen casos donde la secuencia no converge a cero, donde utilizamos el término "converge.$^{\text{en}}$ el sentido de que llega al entero menor de la secuencia. Una vez alcanzado éste, se generan

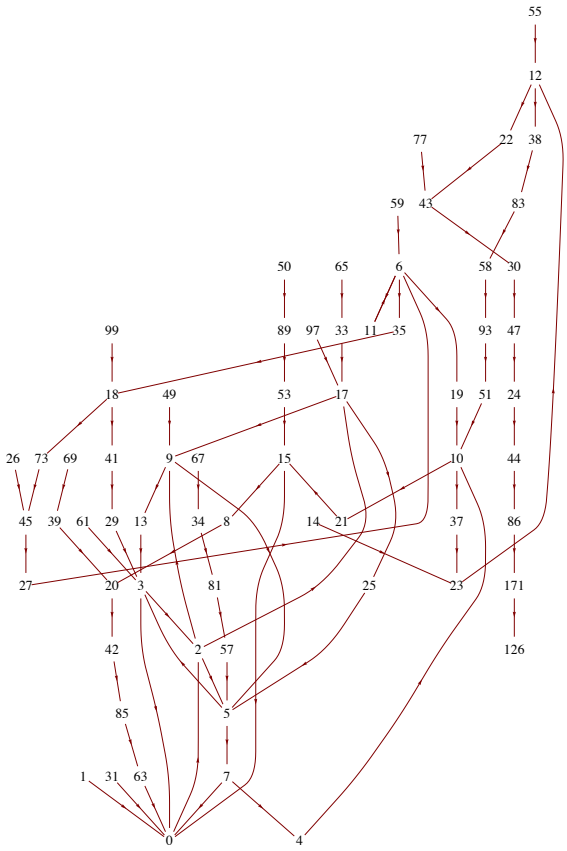


Figura 2. Grafo para números menores que 100.

estructuras repetitivas triangulares de tres tipos diferentes, generando enteros cada vez más grandes, como se muestra en la figura 4.

Ésta es la lista de los primeros cien enteros y los valores a los que converge.

0, 1, 0, 0, 0, 0, 6, 0, 0, 0, 0, 6, 12, 0, 12, 0, 0, 0, 0, 0,0,
0, 22, 12, 24, 0, 0, 0, 28, 0, 24, 0, 0, 0, 0, 0, 0, 12, 10,
0, 0, 0, 0, 24, 44, 0, 46, 24, 48, 0, 0, 10, 0, 0, 54, 10, 56,
0, 10, 0, 60, 0, 48, 0, 0, 0, 6, 0, 0, 0, 23, 0, 0, 0, 12, 46,
0, 24, 0, 0, 0, 0, 0, 10, 0, 0, 86, 60, 88, 0, 0, 54, 92, 10,
94, 48, 96, 0, 0, 0, 0.

Para valores más grandes, ver la figura 5.

En otros casos aparecen ciclos como es el caso de 6 y 11, y de 54 y 91. Para enteros de la forma 2^n y $2^n + 1$ la convergencia es muy simple, como se muestra

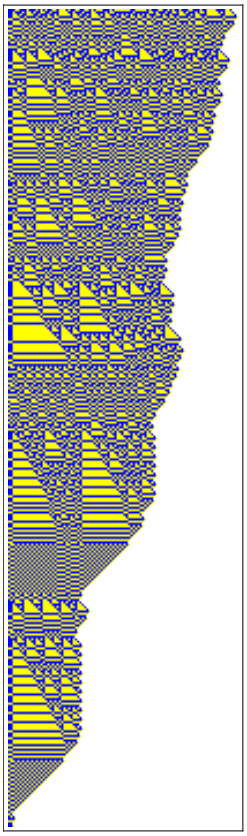


Figura 3. Otro ejemplo de programa tipo Collatz.

en la figura 6, como éstas estructuras aparecen repetidamente, juegan el papel de estructuras localizadas.

La cantidad de pasos que se requieren para que alcancen una longitud mínima los primeros 100 dígitos son:

1, 1, 4, 2, 5, 3, 1, 2, 6, 4, 4, 2, 1, 6, 3, 2, 7, 5, 9, 5, 5,
3, 1, 2, 1, 4, 9, 7, 1, 7, 3, 2, 8, 6, 10, 10, 12, 3, 6, 6, 6,
8, 4, 4, 1, 8, 1, 2, 1, 8, 10, 2, 16, 8, 1, 8, 1, 8, 4, 12, 1,
8, 3, 2, 9, 7, 7, 11, 17, 7, 6, 13, 15, 9, 4, 2, 17, 5, 21, 7,
7, 9, 11, 5, 5, 3, 1, 2, 1, 9, 15, 2, 1, 3, 1, 2, 1, 6, 9, 11,
23.

3. Definiciones alternativas

Un par de maneras de reescribir este sistema en el leguaje de *Mathematica* son:

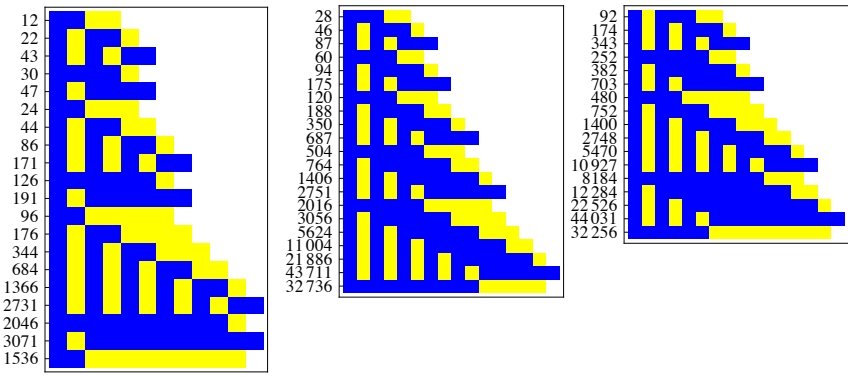


Figura 4. Tipos de estructuras que emergen cuando la secuencia alcanza su menor longitud.

```
If[Last[#] == 1, BitXor[Rest[#1], Most[#]],
 Join[{1, 0}, Most[#1]]] &
```

que en palabras podría expresarse como "Si el último dígito es 1 (es decir, si es impar), realizar la operación lógica `BitXor` con la secuencia de dígitos resultante de remover el primer dígito y la secuencia removiendo el último; si el último dígito es cero, agregar a la secuencia 1,0 al principio y remover el último dígito":

```
 If[Last[#] == 1,  Partition[#1, 2, 1] /.
{{i_, i_} -> 0, {i_, j_} -> 1}, #1 /. {l__, i_} -> {1, 0,  1}] &
```

que podría describirse como "Si el último dgito es 1, particionar la lista en pares, empezando por cada dígito, aquellos pares que sean iguales se reescriben como cero y los diferentes como 1; de ser cero, sustituir la secuencia con otra con un 1, 0 al principio y removiendo el último dígito". La manera en que actúa `Partition` en una secuencia como 1011101 genera pares recorriendo un sitio y produce 10, 01, 11, 11, 10, 01. El símbolo `/.` quiere decir reemplazar y las expresiones con un guión bajo como `i_` aparejan un dígito y expresiones como `l__` una secuencia de una cantidad arbitraria de dígitos; expresiones como `i_` y `j_` se refieren a diferentes dígitos. Los smbolos `->` asignan que es lo que se debe reemplazar en la secuencia. Puesto de ésta manera guarda un parecido a un sistema de sustitución dependiente de vecinos. Veremos que para secuencias de números pares, éstas pueden escribirse con fórmulas en términos de potencias de dos y relaciones de recurrencia en la sección siguiente.

También existe una relación con la máquina de Post (sistema de etiquetas), de hecho se conoce un sistema de Post que emula el problema de Collatz[4]. En nuestro caso, hay una reminiscencia con la máquina de Post en la parte donde se agrega un 1 al principio, sin embargo el sistema tratado aquí depende del

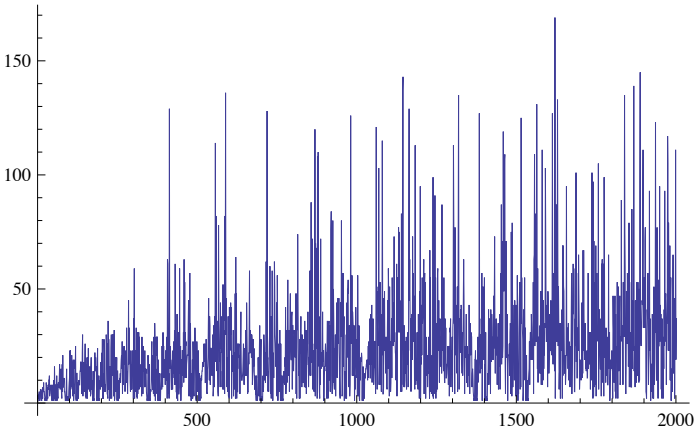


Figura 5. Cantidad de pasos para converger para valores hasta 2000.

vecino de la derecha, pero las reglas de producción deberían definirse de manera diferente a la habitual.

También es posbile definir el problema de Collatz[5,6] como máquina abstracta, con números racionales, como secuencia de paridad, con mapeos en los reales y complejos, usando la función de Siracusa y en una versión de autómata celular[7] en base 6 con 2 vecinos. En este caso se puede definir como función por partes para ternas de números *abc*:

$$f(n) = \begin{cases} 4 & \text{si } a \text{ es impar y } b = 6 \\ 6 & \text{si } a \text{ es par y } b = 6 \\ 3 \text{ a mod } 2{+}\lfloor \text{b}/2 \rfloor & \text{en cualquier otro caso} \end{cases} \tag{2}$$

en la última parte se debe sustituir los ceros por seis cuando $a = 6$.

Se han observado curiosos patrones[8,9] en objetos tales como la espiral de Ulam, así como en el problema de Collatz; para nuestro sistema, también aparecen patrones que nos ayudarían a entender cómo interactúan las estructuras (figura 7).

4. Expresiones analíticas

En ésta sección se muestran ejemplos de fórmulas para algunas secuencias, sin embargo no se intenta una derivación general de todas las posibles secuencias. Tomemos el caso de la secuencia creciente, obtenida del sistema que estamos estudiando:

576, 1312, 2704, 5448, 10916, 21842, 43689

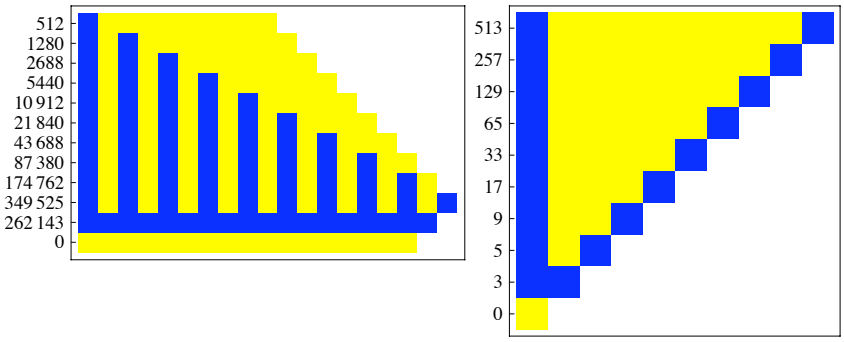


Figura 6. Convergencia de enteros de la forma 2^n y 2^{n+1}.

cuyos términos están dados por la expresión (comenzando con $n = 1$),

$$\frac{1}{3}2^{7-n}(2^{3+2n} - 5) \tag{3}$$

que también puede escribirse como relación de recurrencia,

$$a(n+2) = \frac{5}{2}a(n+1) - a(n), a(0) = 576, a(1) = 1312. \tag{4}$$

Observemos que el último entero de la secuencia resulta impar, que a su vez genera un racional para el siguiente término, entonces se aplica la otra condición para números impares de acuerdo a la definición del programa, entonces se reduce a un problema de relación de recurrencia que se evalúa en los números pares. En el caso de secuencias impares no existe una recurrencia lineal.

En el caso de las estructuras repetitivas cuando la secuencia no converge a cero, también pueden describirse por este tipo de expresiones, en el caso del 12, se genera ésta subsecuencia:

> 393216, 720896, 1409024, 2801664, 5595136, 11186176, 22370304, 44739584, 89478656, 178957056, 357913984, 715827904, 1431655776, 2863311536, 5726623064, 11453246124, 22906492246, 45812984491

que se pueden escribir como:

$$\frac{1}{3}2^{18-n}(1 + 2^{1+2n}) \tag{5}$$

la cual puede escribirse también como relación de recurrencia mostrada anteriormente, así como las secuencias siguientes, tomando los primeros dos enteros como valores iniciales.

Otro tipo de estructura que se genera, como es el caso del 28, genera la siguiente secuencia:

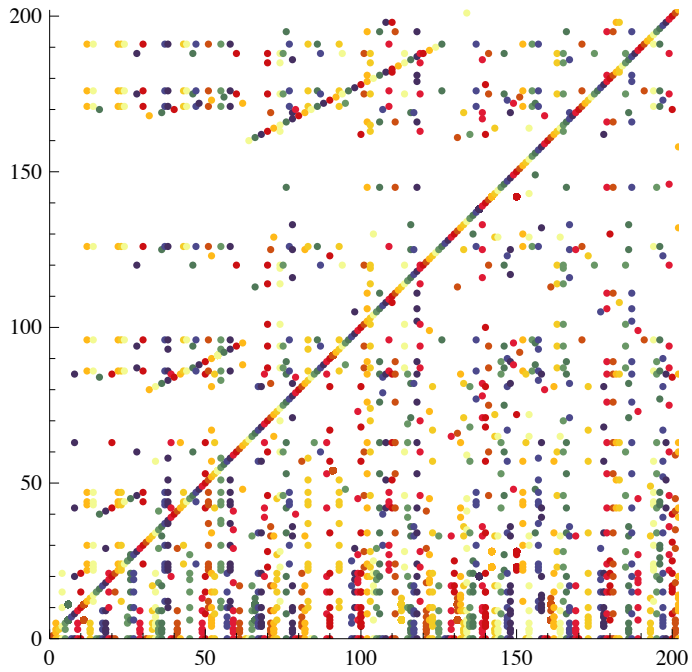


Figura 7. Evolución del sistema para los primeros 200 valores iniciales. Nótese la existencia de líneas horizontales y otras con una cierta inclinación.

523776, 786176, 1441664, 2817984, 5603296, 11190256, 22372344, 44740604, 89479166

que de manera similar queda:

$$\frac{1}{3}2^{10-n}(1021 + 2^{9+2n}). \tag{6}$$

Por útimo, otra estructura se produce a partir de enteros como el 92 de la siguiente manera:

32256, 48896, 89984, 176064, 350176, 699376, 1398264, 2796284, 5592446

o de la forma:

$$\frac{1}{3}2^{10-n}(61 + 2^{5+2n}). \tag{7}$$

5. Conclusiones

Esto muestra que este tipo de problemas no son aislados, como otros sistemas de tipo reversión-adición[10] también hallados por el autor y que sugieren que existe una mayor relación entre problemas de teoría de números e ideas de computación y complejidad, en relación con identificación de estructuras localizadas para allanar el camino hacia demostraciones de universalidad, así como la comprensión de límites de solubilidad y no solubilidad. Como trabajo posterior, se requiere un teorema que resuma los ejemplos de las fórmulas presentadas aquí para secuencias de números pares.

Referencias

[1] Wolfram, S. (2002). *A New Kind of Science*, Wolfram Media, p. 904.

[2] Weisstein, E. W. "Wolfram Sequences" From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/WolframSequences.html`

[3] Weisstein, E. W. "Collatz Problem" From MathWorld - A Wolfram Web Resource. `http://mathworld.wolfram.com/CollatzProblem.html`

[4] De Mol, L. (2008). Tag systems and Collatz-like functions, *Theoretical Computer Science*, 390(1), 92-101.

[5] Collatz conjecture (2011). Wikipedia, `http://en.wikipedia.org/wiki/Collatz_conjecture`

[6] Zeleny, E. (2011). "Other Formulations of Collatz Problem", submitted to Wolfram Demonstrations Project.

[7] Zeleny, E. "Collatz Problem as a Cellular Automaton" from the Wolfram Demonstrations Project. `http://demonstrations.wolfram.com/CollatzProblemAsACellularAutomaton/`

[8] Pickover, C. A. (2000) *Wonders of Numbers, Adventures in Math, Mind, and Meaning*, Oxford University Press.

[9] Zeleny, E. "Preferred Values of Collatz Paths" from the Wolfram Demonstrations Project. `http://demonstrations.wolfram.com/PreferredValuesOfCollatzPaths/`

[10] Zeleny, E. (2008). "Reversal-Addition Related Systems" from the Wolfram Demonstrations Project. `http://demonstrations.wolfram.com/ReversalAdditionRelatedSystems/`

Un algoritmo de encriptación basado en la composición de las reglas 30 y 86 del autómata celular elemental

Emmanuel Garcés Medina

Laboratorio de Ciencias de la Computación, UNAM.
Laboratorio de Dinámica No Lineal, Facultad de Ciencias, UNAM.
emmanuel.garces@ciencias.unam.mx

Resumen En este artículo se describe un algoritmo de encriptación basado en la dinámica de la composición alternada de los autómatas celulares elementales (ECA) reglas 30 y 86. La llave de encriptación k es una secuencia de bits de longitud $|k| \leq |x|$, siendo x los bits a cifrar. La llave, que es privada, define un camino hacia atrás en el tiempo desde el estado inicial que corresponde a los bits de los datos a encriptar alternando las reglas de ambos autómatas. El cifrado falla cuando el algoritmo alcanza un estado que no tiene predecesores de longitud $|x|$ con la regla 30, lo cual ocurre con probabilidad muy cercana a cero.

1. El autómata celular elemental

El autómata celular elemental es un sistema dinámico definido sobre un arreglo finito x con $n > 2$ elementos cuyo valor puede ser 1 o 0 ($x \in \{1,0\}^n$). La dinámica del sistema esta regida por una regla o función de transición f que actualiza el valor de cada elemento a través del tiempo t. La función de transición esta definida sobre una vecindad de tres elementos: el vecino izquierdo del elemento a actualizar, el elemento mismo y el vecino derecho. Para los elementos que se encuentran en la frontera del arreglo la vecindad se considera de la misma manera, pero suponiendo que el arreglo es circular. Por lo tanto, si x_i^t es el estado de la i-esima celda en el tiempo t, entonces:

$$x_i^{t+1} = \begin{cases} f(x_{i-1}^t, x_i^t, x_{i+1}^t) & \text{si } i > 1 \wedge i < n \\ f(x_n^t, x_1^t, x_2^t) & \text{si } i = 1 \\ f(x_{n-1}^t, x_n^t, x_1^t) & \text{si } i = n \end{cases} \quad (1)$$

Los autómatas celulares (AC), además de ser estudiados históricamente como sistemas dinámicos discretos, también han sido explorados como máquinas capaces de efectuar computación. Existen, inclusive, autómatas celulares que pueden ser configurados para efectuar computación de propósito general o universal. Otros han sido diseñados para propósitos específicos, entre ellos la encriptación de datos. Se han reportado varios algoritmos de encriptación basados en autómatas celulares [3, 6, 10] los cuales explotan diversas propiedades dinámicas de estos sistemas.

Existen 256 funciones de transición o reglas posibles para un autómata celular elemental. Muchas de ellas exhiben comportamiento dinámico con suficiente complejidad que se aprovecha para simular otros mecanismos. En este artículo se utilizan las reglas 30 y 86, las cuales se clasifican dentro de la clase III de Wolfram [7] exhibiendo pseudoaletoriedad, caos e irreduciblidad computacional.

La función de transición de la regla 30 se define como $f_{30}(v_1, v_2, v_3) = (v_1 + v_2 + v_3 + v_2v_3)\ mod\ 2$. La función de transición de la regla 86 se define como $f_{86}(v_1, v_2, v_3) = (v_1 + v_2 + v_3 + v_1v_2)\ mod\ 2$.

La regla 86 es la regla reflejada izquierda-derecha de la regla 30, es decir, si $f_{30}(v_1, v_2, v_3)$ es la función de transición de la regla 30, entonces $f_{86}(v_1, v_2, v_3) = f_{30}(v_3, v_2, v_1)$ es la función respectiva para la regla 86. La dinámica de ambos autómatas es equivalente.

2. Antecedentes

Los autómatas celulares como mecanismos de encriptación de información se han usado de diversa manera para la encriptación de datos. Olivera, Martins y Alt mencionan tres usos de los AC para encriptar datos: i) usando AC para generar buenas llaves de encriptación [6], ii) usando AC reversibles [3] y iii) usando AC no reversibles [2, 5, 10]. El algoritmo propuesto en este artículo reside en la tercera categoría.

Existen varios motivos por los cuales los autómatas celulares resultan atractivos para su uso como mecanismos de cifrado. En principio porque la computación de un autómata celular se desarrolla en paralelo y la programación de las funciones de transición es simple, pero sobre todo porque existen dinámicas que exhiben pseudoaletoriedad, caos e irreducibilidad computacional y otras que exhiben reversibilidad.

Los métodos de encriptación que emplean autómatas celulares no reversibles realizan la computación de preimagenes de estados para el proceso de cifrado y el computación hacia adelante para el proceso de descifrado. El algoritmo de Gutowitz [2] usa cierto tipo de reglas (toggle rules) y configuraciones de celdas no periódicas para sus procesos mientras que Wuensche [10] usa otra clase de reglas asociadas con ciertos valores del parámetro Z y usando configuraciones de celdas de tamaño fijo. Oliveira [5] extiende el método de Gutowitz para usar cualquier regla de AC.

En los métodos ii) y iii) mencionados anteriormente las llaves de encriptación corresponden a reglas de algún autómata celular. En el algoritmo propuesto en este artículo la llave corresponde a una secuencia de bits que define una composición funcional de la regla 30 y su equivalente reflejada izquierda-derecha, la regla 86.

La regla 30, es usada por Wolfram [6] como un proceso para encontrar una llave de tamaño adecuado para el método de encriptación basado en XOR o cifrado de Vernan a partir de la llave original y usando los valores de la celda central en la evolución de la regla 30. El método de Wolfram no emplea el cálculo de preimagenes como lo hace el algoritmo de Gutowitz [2], donde a partir de una

configuración de bits a encriptar se calcula algun predecesor a cierta distancia hacia atrás y de longitud mayor a la configuración inicial. En tal caso, la llave de encriptación es la regla de transición. En el método de este artículo se fijan dos reglas y el tamaño de la configuración de celdas.

3. Espacio de estados e irreducibilidad del autómata celular elemental regla 30

Dado un tiempo t en la evolución del autómata celular. Se conoce como *estado* a la configuración de los elementos del arreglo en el tiempo t. Es decir:

Si $x \in \{1,0\}^+$ con $|x| = n$ entonces $regla_R(x,t)$ es el estado del autómata celular regla R en el tiempo t para la condición inicial x de tamaño n. Se abrevia $regla_R(x)$ a $regla_R(x,1)$.

Por lo tanto, para un arreglo de n celdas con dos estados, el número máximo de posibles estados en el que se puede encontrar un autómata celular es 2^n.

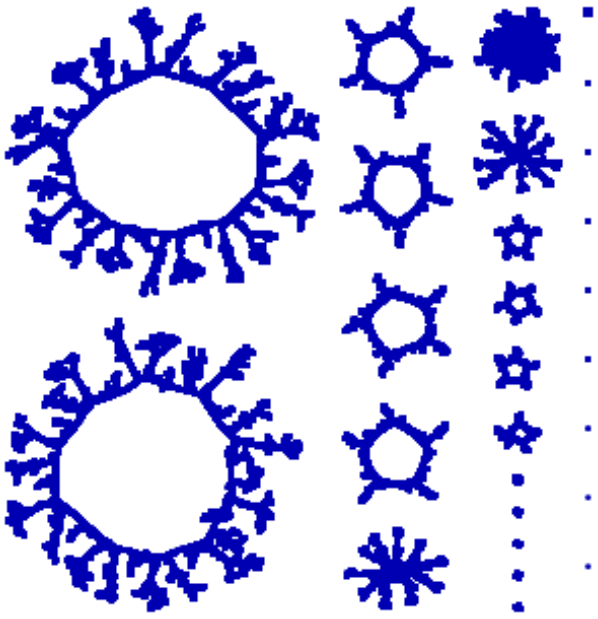


Figura 1. El espacio de transiciones del AC Elemental regla 30 para $n = 20$.

Para el caso de la regla 30, a partir de la mayoría de las 2^n condiciones iniciales posibles, el número de estados que son visitados durante la dinámica del autómata celular crece exponencialmente respecto al tamaño de la condición inicial [7]. Existen evidencias experimentales que apoyan el hecho de que la evolución de la regla 30 es una computación irreducible.

Por lo tanto, el espacio de transiciones entre estados de la dinámica del autómata celular elemental regla 30 contiene caminos que visitan una fracción muy importante de todos las configuraciones posibles de un arreglo de bits de n d elementos (2^n). En la Figura 1 se muestra el trazo general de todos los caminos realizados por la regla 30 en un arreglo de bits de 20 elementos. El número total de estados en la Figura 1 debe ser igual a $2^{20} \approx 1$ millón.

4. Densidad de estados no alcanzables de longitud n para la regla 30

Un estado no alcanzable se define como aquel que no es predecesor de algún otro estado. Es decir, son estados que solo pueden aparecer como condiciones iniciales en la evolución de un autómata celular y visualmente son las hojas de la gráfica de la Figura 1. Si $pred_R(x)$ es el conjunto de estados predecesores del estado x para la regla R, entonces:

$$pred_R(x) = \{p \in \{1,0\}^n | regla_R(p) = x\} \tag{2}$$

El conjunto de hojas o estados no alcanzables para el autómata celular regla R con n celdas esta determinado por:

$$hojas_R(n) = \{x \in \{1,0\}^n | pred_R(x) = \emptyset\} \tag{3}$$

Es importante resaltar que la densidad de estados que son alcanzables será de mucha importancia en la eficacia del algoritmo de encriptación que se propone. Existen evidencias [7, 9] que sugieren que para la regla 30, la fracción o densidad de estados no alcanzables tiende a cero conforme aumenta el número de elementos n del arreglo de bits del autómata celular. Esto se debe principalmente porque la regla 30 es suryectiva. Por lo tanto, $\forall x \in \{1,0\}^n$ se tiene que:

$$probabilidad(x \in hojas_{30}(n)) \to 0 \; cuando \; n \to \infty \tag{4}$$

La Figura 2 muestra la fracción de estados no alcanzables para 4 reglas de autómata celular elemental.

Debido a que la regla 86 es la regla reflejada izquierda-derecha de la regla 30, el conjunto de predecesores de los 2^n estados de ambos autómatas celulares respectivamente es el mismo [8]. Por lo tanto, se ha verificado que:

$$\forall x \in \{1,0\}^n$$

$$\exists p \in \{1,0\}^n \; t.q. \; regla_{30}(p) = x \Leftrightarrow \exists q \in \{1,0\}^n \; t.q. \; regla_{86}(q) = x \tag{5}$$

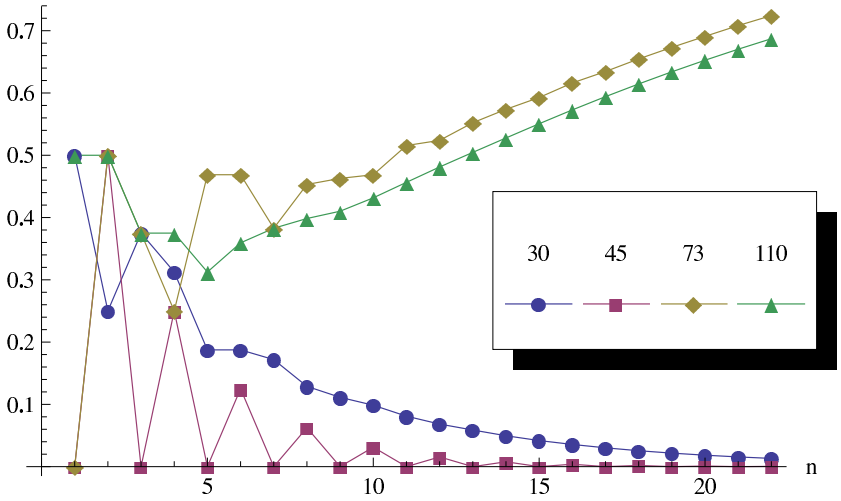


Figura 2. La densidad de estados de tamaño n sin predecesores de la misma longitud para las reglas 30, 45, 73 y 110 en función de n.

5. Composición de autómatas celulares determinadas por un vector

Para efectos del algoritmo, se define la composición de autómatas celulares como la aplicación alternada de dos reglas sobre algún estado del autómata celular. A diferencia de las definiciones usuales de composición de funciones, se puede definir un órden particular de aplicación de funciones determinadas por un vector. Por lo tanto, dada una condición inicial $x \in \{1,0\}^n+$, un vector de composición $v \in \{1,0\}^m+$ y dos reglas A yB, la composición se define como:

$$c_{A,B}(x,v) = regla_{r_1}(regla_{r_2}(...regla_{r_j}(regla_{r_{j+1}}(...regla_{r_m}(x)...))...)) \quad (6)$$

$$r_j = \begin{cases} A \text{ si v(j)=0} \\ B \text{ si v(j)=1} \end{cases} \quad (7)$$

Se define ahora al conjunto de estados distintos E a los que se puede llegar desde un estado x de tamaño n mediante la composición determinada por un vector de tamaño m descrita anteriormente:

$$E_{A,B}(x,m) = \{c_{A,B}(x,v) | v \in \{1,0\}^j, 1 \leq j \leq m\} \quad (8)$$

Para la composición con las reglas 30 y 86 en adelante se abrevia $c_{30,86} = \text{c}$ y $E_{30,86} = E$. Se observa que:

$$|E(x,m)| \approx \begin{cases} O(2^m) & \text{si } m < n \\ 2^n & \text{si } m \geq n \end{cases} \tag{9}$$

Es decir, fijando la condición inicial x de tamaño n, el número de estados visitados por la composición determinada por un vector de tamaño m es tan grande como el número de vectores explorados, pero lógicamente menor, al número total de estados posibles (Figuras 3 y 4).

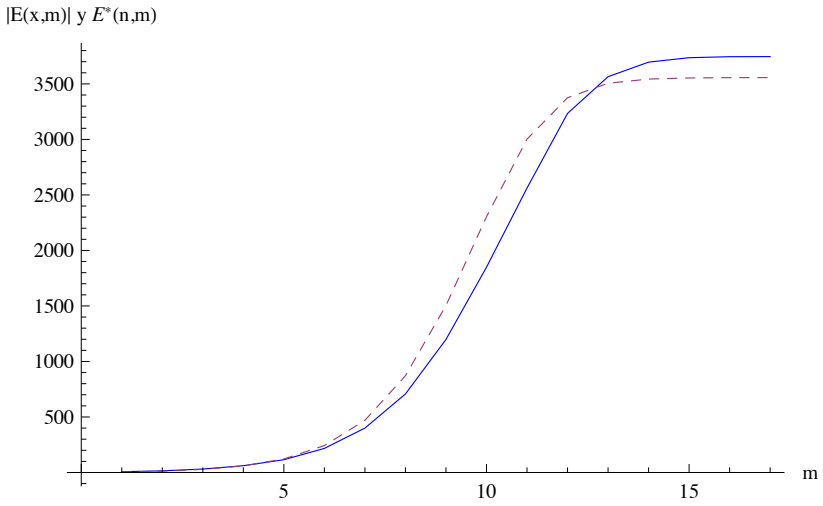


Figura 3. El número de estados distintos $|E(x,m)|$ (línea continua) visitados desde un estado fijo x (elegido aleatoriamente) de 12 bits en el proceso de composición de las reglas 30 y 86 determinada por un vector de tamaño m. $E^*(n,m)$ (l{́}nea discontinua) es el número de estados visitados en m pasos desde un vértice inicial para el proceso aleatorio de agregación de aristas a una gráfica de 2^{12} vértices descrito en esta sección. Se observa que ambas distribuciones son semejantes.

Para demostrar la proposición (9) se construye una gráfica diriga G cuyos vértices son los estados x de n bits y las aristas son las transiciones de la regla 30 y la regla 86 para todos los estados (Ver Figura 5). Todos los vértices $u \in G$ tienen grado de salida $d(u)^+ = 2$ ya que las aristas de salida son $u \to regla_{30}(u)$ y $u \to regla_{86}(u)$.

Por otro lado, debido a la proposición (4) casi todos los vértices $u \in G$ tienen grado de entrada $d(u)^- = 2$, ya que las aristas de entrada son todas las flechas $w \to u$ tal que $w \in G$ y $regla_{30}(w) = u$ o $regla_{86}(w) = u$ y en general, debido a

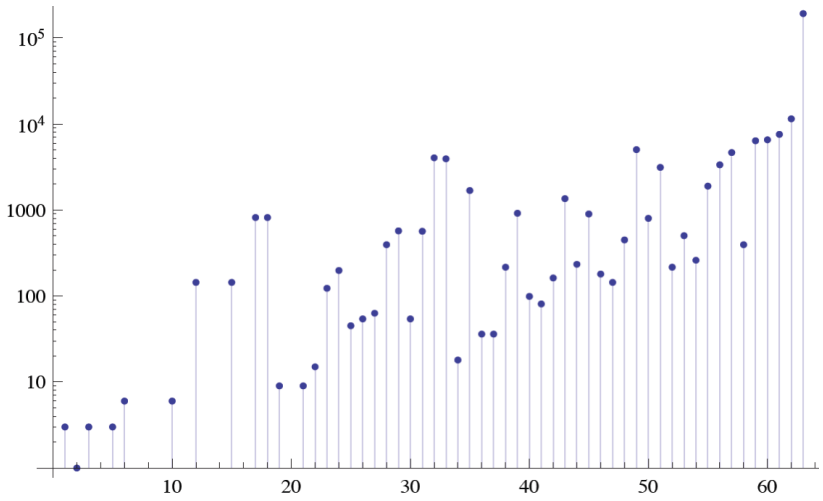


Figura 4. Para todas las configuraciones x de 16 bits se muestra la frecuencia de $|E(x,5)|$. Se observa que a partir de la mayoria de configuraciones x se exploran casi $2^{5+1} = 64$ estados diferentes.

la suryectividad de las reglas 30 y 86, la mayoria de los estados tienen solo un predecesor para la regla 30 y otro para la regla 86.

Por lo tanto, G es aproximadamente una gráfica dirigida 2-regular, y usando el teorema de conectividad en [1] se deduce que con muy alta probabilidad desde un vértice inicial x se puede llegar a la mayoria del resto de los 2^n vértices de G mediante alguna ruta determinada por algun vector v.

Sea G^* una gráfica con 2^n y sin aristas. Se describirá un proceso aleatorio para agregar aristas a G^* de tal manera que la gráfica final tendrá propiedades similares a G.

A continuación se define $E^*(n,m)$ que denota al número de estados a los que se puede llegar construyendo una gráfica mediante el proceso aleatorio mencionado y descrito como sigue: Sea $X^* \subseteq V(G^*)$ el conjunto de los vértices recientemente visitados. Ahora, $\forall x^* \in X^*$ se eligen dos vértices u^* y w^* no visitados durante el proceso y se agregan las aristas $x^* \to u^*$ y $x^* \to w^*$ a G^*. Luego se saca x^* de X^* y se meten u^* y w^* en X^*. Este proceso se hace m veces o hasta que ya no hay más vértices que procesar. La ecuación 10 muestra $e(n,m')$, que denota el número de vértices procesados en el m'-esimo paso del proceso aleatorio.

$$e(n,m') = e(n,m'-1)(2 - \frac{1}{2^{n-1}} \sum_{j=1}^{m'-1} e(n,j)) \tag{10}$$

$$E^*(n,m) = \sum_{j=1}^{m} e(n,j)) \tag{11}$$

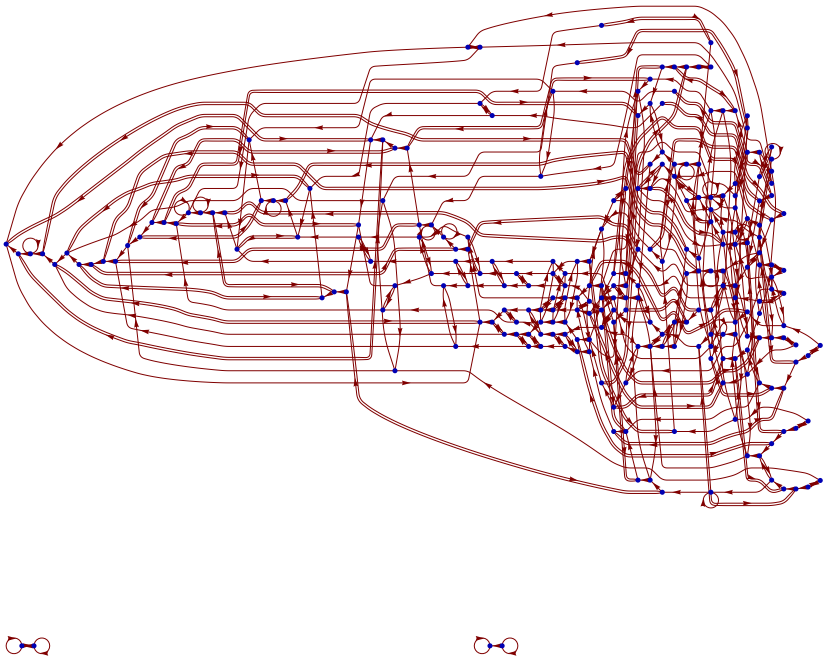


Figura 5. Todas las posibles transiciones con la regla 30 y regla 86 para todos los estados de longitud $n = 8$.

Informalmente, las aristas de G^* se generan a través una m-caminata aleatoria asegurando que cada vértice $x^* \in G^*$ tenga grado de salida $d^+(x^*) = 2$.

Experimentalmente se observa que la gráfica de transiciones en la composición de las reglas 30 y 86 es similar a G^*. Por lo tanto $|E(x,m)| \approx E^*(n,m)$ (ver Figura 3).

Supongamos ahora que $m < n$. Entonces, reemplazando la suma en (10) con la definición de (11), despejando y adecuando los índices adecuadamente tenemos que:

$$E^*(n,m) = \delta 2^{n-1} \tag{12}$$

$$\delta = \frac{2e(n,m) - e(n,m+1)}{e(n,m)}. \tag{13}$$

Se observa que δ es la fracción de vértices que se reeligen en el m'-esimo paso del proceso aleatorio que genera las aristas de G^*.

En cada paso m' de este proceso, δ crece proporcional a la fracción del número de vértices ya usados. Por lo tanto, $\delta \propto \frac{2^m}{2^n}$ y sustituyendo esta proporcionalidad en (12) tenemos que $E^*(n,m) \approx O(2^m)$.

Si $m \geq n$ entonces debido al crecimiento asintótico de $E^*(n,m)$, se tiene que $E^*(n,m) \approx O(2^n)$.

Finalmente, como $E^*(n,m) \approx |E(x,m)|$ entonces la proposión (9) queda demostrada.

La complejidad derivada de la pseudoaleatoriedad experimentada en la dinámica de la regla 30 y 86 se manifiesta en la Figura 5 donde se muestran las transiciones de un autómata celular con 8 celdas y todas las posibles composiciones de la regla 30 y 86 determinadas por algún vector.

6. Trazo del algoritmo

Si los datos a cifrar son una secuencia $x \in \{1,0\}^n$ de n bits, y la llave es otra secuencia $k \in \{1,0\}^m$ de m bits, la idea principal del algoritmo consiste en encontrar un estado $y \in \{1,0\}^n$ tal que $x = c(y,k)$. El estado y será entonces la secuencia de bits encriptados. Para lograrlo se calculan los predecesores comenzando desde el estado x, tomando, en cada paso del algoritmo algún predecesor del estado actual y repitiendo el proceso hasta terminar con todos los elementos de la llave k.

Por ejemplo, si los bits a encriptar son $x = 001001$, y la llave es $k = 001$ entonces la encriptación esta dada por $s_1 \in pred_{30}(s_2 \in pred_{30}(s_3 \in pred_{86}(001001))) = 101011$. El cálculo del algún predecesor se puede hacer usando el algoritmo para encontrar preimagenes descrito en [4]. La probabilidad de que no exista predecesor de alguna configuración de celdas en algún momento del proceso de encriptación es muy baja (Ecuación 4), tanto para la computación de predecesores de la regla 30 como para los de la regla 86 (Ecuación 5).

Si durante algoritmo se llegase a visitar un estado sin predecesores entonces el algoritmo falla y se puede solicitar una nueva llave al usuario del algoritmo. El algoritmo 1 muestra la definición formal del método de cifrado propuesto.

Inversamente, si $y \in \{1,0\}^n$ es una secuencia encriptada de bits bajo la llave $k \in \{1,0\}^m$ entonces su descifrado se logra mediante $c(y,k)$ (ver definición 6), es decir, el decifrado de y es la aplicación de la composición de la regla 30 y 86 determinada por la llave k. Para el ejemplo anterior se verifica que $c(101011, 001) = 001001$. El algoritmo 2 muestra la definición formal del decifrado.

Algorithm 1 Calcular $y = Cifrado(x, k)$

Require: $x, k \in \{1, 0\}^+$
Ensure: $y = cifrado\ de\ x\ con\ la\ llave\ k \Leftrightarrow hoja = false$
  $i \leftarrow 1$
  $p \leftarrow x$
  $hoja \leftarrow false$
  while $i \leq Length(k) \wedge hoja = false$ **do**
    if $pred_{30}(p) \neq \emptyset$ **then**
      if $k[i] = 0$ **then**
        $p \leftarrow s \in pred_{30}(p)$
      else
        $p \leftarrow r \in pred_{86}(p)$
      end if
    else
      $hoja \leftarrow true$
    end if
    $i \leftarrow i + 1$
  end while
  $y \leftarrow p$

Algorithm 2 Calcular $x = Descifrado(y, k)$

Require: $x, k \in \{1, 0\}^+$
Ensure: $x = descifrado\ de\ y\ con\ la\ llave\ k$
  $i \leftarrow Length(k)$
  $q \leftarrow y$
  while $i \geq 1$ **do**
    if $k[i] = 0$ **then**
      $q \leftarrow regla_{30}(q)$
    else
      $q \leftarrow regla_{86}(q)$
    end if
    $i \leftarrow i - 1$
  end while
  $x \leftarrow q$

7. Sobre la eficacia del algoritmo

La sensibilidad a condiciones iniciales y la pseudoaleatoriedad de la dinámica del autómata celular regla 30 son dos propiedades que refuerzan la eficacia del algoritmo. La sensibilidad a condiciones iniciales de las reglas 30 y 86 garantiza que cualquier cambio en la llave o en los datos encriptados derivará en resultados radicalmente distintos en el proceso de descifrado (Figuras 6 y 7) y la pseudaleatoriedad en la regla 30 (y regla 86) equipara el proceso de busqueda de llaves con un proceso aleatorio semejante que genera la gráfica G^* descrito en este artículo.

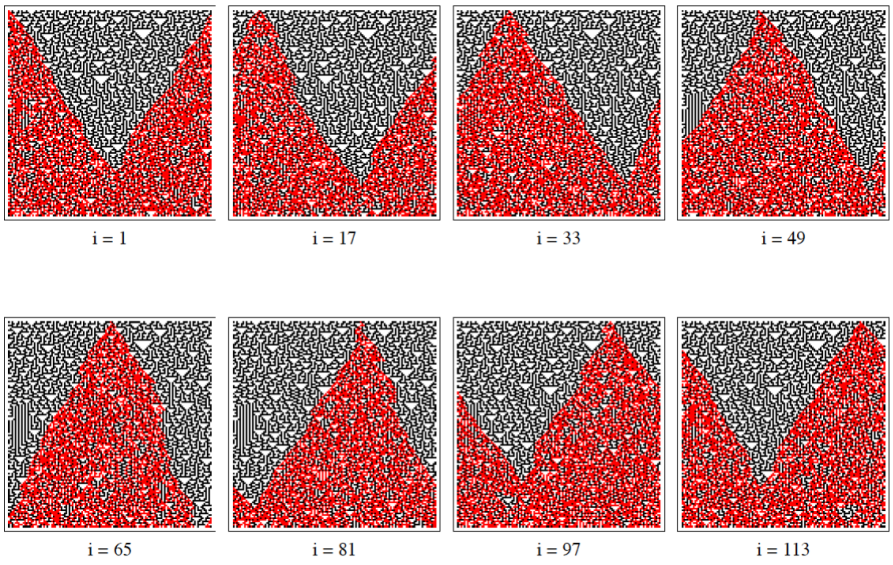


Figura 6. El proceso de descifrado $c(y, k)$ exhibe sensibilidad a condiciones iniciales para cambios en los bits encriptados y. Con $|y| = |k| = 128$ $bits$ cada figura muestra en rojo las diferencias entre dos evoluciones del proceso de descifrado cuyas condiciones iniciales se diferencían únicamente por el valor del i-esimo bit de y.

El tamaño de la llave $|k| = m$ puede ser tan grande como estados posibles explorados en el algoritmo de cifrado y descifrado (estados visitados en la evolución hacia adelante y hacia atrás de la composición de las reglas 30 y 86). Sin embargo, la proposición (9) asegura que cuando m tiende a n entonces el número de llaves que se requieren explorar para descifrar un dato encriptado es tan

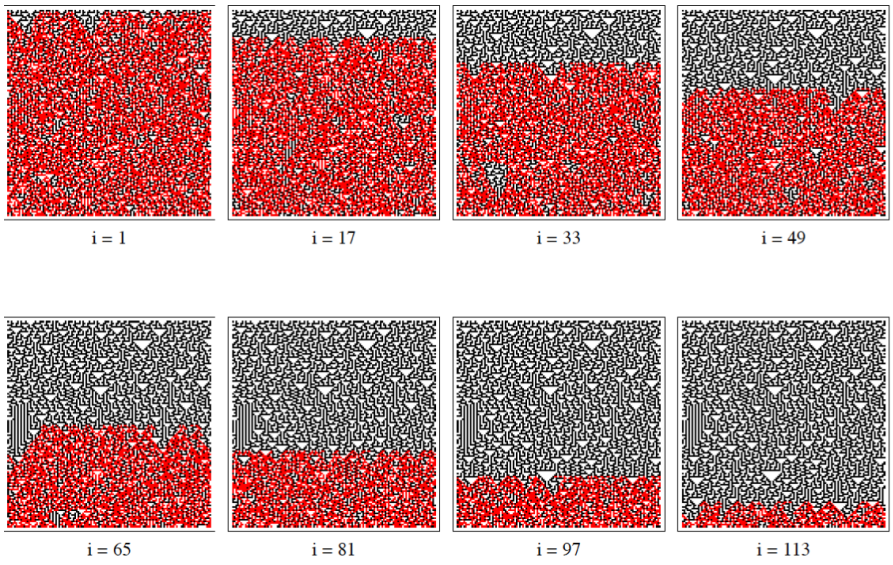


Figura 7. El proceso de descifrado $c(y, k)$ exhibe sensibilidad a condiciones iniciales para cambios en los bits de la llave k. Con $|y| = |k| = 128$ $bits$ cada figura muestra en rojo las diferencias entre dos evoluciones del proceso de descifrado cuyas condiciones iniciales se diferencían únicamente por el valor del i-esimo bit de la llave k.

grande como el número de datos de tamaño n. Por lo tanto, aunque la encriptación es posible para llaves con longitud $m > n$, el tamaño de las configuraciones a explorar en un intento de descifrado es tan grande como si la encriptación hubiese sido realizada usando una llave de longitud $m = n$.

La probabilidad de que la llave de cifrado no funcione y haga fallar el método de cifrado disminuye asintóticamnte a cero conforme aumenta el tamaño n del dato a encriptar debido a que que la probabilidad de que durante el proceso de cifrado se visite un estado sin predecesores tiende a cero cuando n tiende a infinito (ver proposición 4).

La computación de preimagenes de estados de la regla 30 puede ser realizado en tiempo lineal. Por lo tanto, la complejidad computacional del algoritmo de ciifrado es $O(nm)$. Análogamente, la computación de una transición de la regla 30 u 86 se hace en tiempo lineal, por lo que la complejidad computacional del algoritmo de descifrado es también $O(nm)$.

8. Ejemplos

A continuación se muestran tres ejemplos de encriptación usando el algoritmo propuesto y usando llaves de distinto tamaño relativo al tamaño de las secuencias que se encriptan. Cada una de estas secuencias son los bits de un texto plano que se obtuvieron usando la codificación UTF-8.

Debido a que el resultado de la encriptación puede generar secuencias de bits que no son mapeables a símbolos de la codificación UTF-8, entonces éste se muestra usando la codificación Radix-64. Los bits de cada llave de encriptación tambien se obtienen usando la codificación UTF-8.

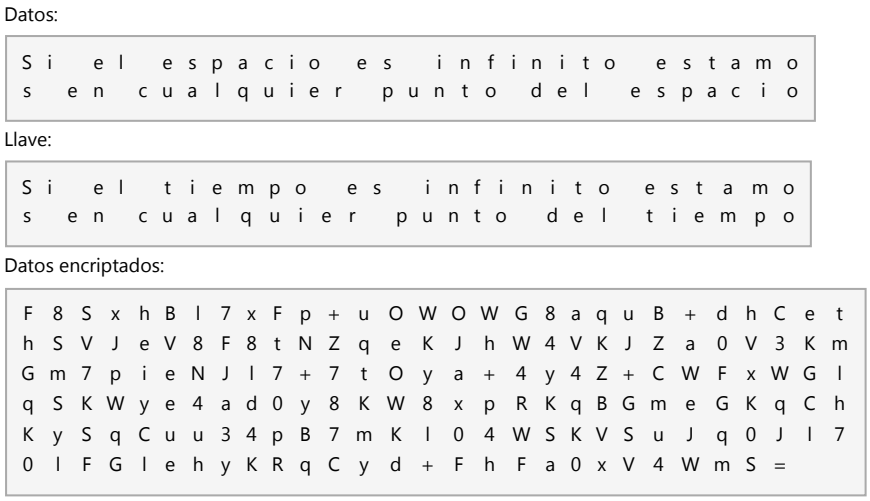


Figura 8. Primer ejemplo de texto encriptado.

Referencias

[1] Cooper, C. (1993) A note on the connectivity of 2-regular digraphs, *Random Structures & Algorithms*, 4(4), 469-472.

[2] Gutowitz, H. (1995) Cryptography with Dynamical Systems, In *Cellular Automata and Cooperative Phenomena*, E. Goles & N. Boccara (Eds.), 237-274, KluwerAcademic Press.

[3] Kari, J. (1992) Cryptosystems based on reversible cellular automata, *Technical report*, University of Turku, Finland.

[4] Mora, J. C. S. T., Martínez, G. J., & McIntosh, H. V. (2004) Calculating Ancestors in One-Dimensional Cellular Automata, *International Journal of Modern Physics C*, 15(8), 1151-1169.

[5] Oliveira, G. M. B., Martins, L. G. A., Ferreira, G. B., & Alt, L. S., Secret Key Specification for a Variable-Length Cryptographic Cellular Automata Model, *PPSN'10 Proceedings of the 11th international conference on Parallel problem*

Datos:

```
s a t u r n o
```

Llave:

```
S i  e l  t i e m p o  e s  i n f i n i t o  e s t a m o
s  e n  c u a l q u i e r  p u n t o  d e l  t i e m p o
```

Datos encriptados:

```
y u J m t a m i + S S t N Z p B y R u =
```

Figura 9. Segundo ejemplo de texto encriptado.

Datos:

```
S i  e l  e s p a c i o  e s  i n f i n i t o  e s t a m o
s  e n  c u a l q u i e r  p u n t o  d e l  e s p a c i o
```

Llave:

```
s a t u r n o
```

Datos encriptados:

```
i h a 8 4 l Z K K h 7 e N p m Z 0 e K J + p G i h J S x Z
B l N i 7 e 7 W 3 4 p 4 0 O q h 0 u 3 0 q m N K t 8 V y i
p 3 x h W 7 F R F F u 3 a u C 7 d h S h G 7 V J l V a B K
q B m F K x l O u O 7 8 R B 4 t 0 y C + y d 3 4 u G 4 d e
0 3 4 B i l 0 W h 7 l p + d Z 7 u d a i d G p a p h y q S
O 7 G K V m R 7 8 3 p q a 8 i F N O t 0 C 0 R V Z 0 =
```

Figura 10. Tercer ejemplo de texto encriptado.

solving from nature: Part II, Krakow, 2010. *Lecture Notes in Computer Science*, 6239, 381-390. 2011.

[6] Wolfram, S. (1986) Cryptography with Cellular Automata, In *Cryptology*: Crypto '85 Proceedings, *Lecture Notes in Computer Science*, 218, 429-432.

[7] Wolfram, S. (2002) *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc.

[8] Wuensche, A. & Lesser, M. (1992) *The Global Dynamics of Cellular Automata; An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*. Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley.

[9] Wuensche, A. (1994) Complexity in one-D cellular automata: gliders, basins of attraction and the Z parameter, Santa Fe Institute *working paper* 94-04-025.

[10] Wuensche, A. (2008) Encryption using cellular automata chain-rules, In *Automata-2008: Theory and Applications of Cellular Automata*, A. Adamatzky, R. Alonso-Sanz, A. Lawniczak, G. J. Martínez, K. Morita, T. Worsch (Eds.), 126-138, Luniver Press.

Autómatas celulares elementales aplicados a la encriptación de datos

Elena Villarreal Zapata, Francisco Cruz Ordaz Salazar

Universidad Politécnica de San Luis Potosí, San Luis Potosí, México.
elena.villarreal@cns-ipicyt.mx, francisco.ordaz@upslp.edu.mx

Resumen Para el cifrado de datos suele ser necesaria una llave como base, por lo que es indispensable tener una que sea robusta y confiable, para así evitar el acceso de terceros a la información cifrada. Esto requiere un generador de números pseudo-aleatorios que proporcionará dicha llave, por lo que se propone trabajar con autómatas celulares auxiliándose con *Mathematica*, para revisar qué reglas, y a qué nivel, son pseudo-aleatorias. Este proyecto se centra en la revisión de posibles reglas pseudo-aleatorias, analizando sus características detalladamente y sometiéndolas a un conjunto de pruebas de aleatoriedad con el fin de conocer cuales de ellas nos permitirán obtener los números pseudo-aleatorios que conformarán la llave para el cifrado de datos.
Keywords: autómatas celulares, pseudo-aleatoriedad, cifrado de datos.

1. Introducción

Esta investigación es parte complementaria de un proyecto que se está trabajando en la Universidad Politécnica de San Luis Potosí, en el que se pretende desarrollar un sistema de encriptación de datos basado en autómatas celulares. Como parte del proyecto inicial se debe comprobar que se está trabajando con reglas pseudo-aleatorias, ya que de comenzar a trabajar con reglas al azar se corre el riesgo de generar un cifrado que puede ser *hackeado* fácilmente. Por tanto, se buscan reglas de autómatas celulares que tengan un comportamiento pseudo-aleatorio, para después generar secuencias con cada una y probar su aleatoriedad.

2. Antecedentes

Los autómatas celulares fueron inventados a fines de los años cuarenta por Stanislaw Ulam y John von Neumann, quienes realizaron trabajos para crear un sistema que se replicara a sí mismo a partir de una abstracción matemática. Años después, Wiener y Rosenblueth desarrollaron un modelo de autómatas celulares que pretendía describir matemáticamente la conducción de impulsos en sistemas cardiacos. En los sesentas se empezaron a estudiar como un tipo de sistemas dinámicos, y para los setenta aparece el Juego de la Vida. Este fue inventado por John Conway y consistía en una colección de celdas las cuales, basadas en

reglas matemáticas, podían vivir, morir o mutiplicarse, todo esto dependiendo de las condiciones iniciales [6]. En 1983, Stephen Wolfram publicó algunos escritos sobre una clase de autómatas que el llamaba autómatas celulares elementales y sobre su comportamiento y las reglas que los definían. Para el 2002, Wolfram publicó su libro *A New Kind of Science* [10] en el cual explica ampliamente sobre ellos, su trabajo y su importancia en todas las ramas de la ciencia. En cuanto a la encriptación, Olu Lafe [5] nos explica que existen un numero de patentes dadas y literatura sobre ello que incluye los trabajos de Wolfram (1985) [9], Delahaye[1] (1991), Guan [8] (1987) y Gutowitz [2] (1994). En los cuales, Wolfram hace uso de la regla 30 de los autómatas celulares para generar números pseudo-aleatorios; Guan usa un sistema dinámico invertible; Gutowitz (U.S. Patent 5,365,589) usa sistemas dinámicos irreversibles; y Lafe (U.S. Patent 5,677,956 el 14 de octubre de 1997) utiliza operaciones simples de transformación, lo cual implica una enorme biblioteca de llaves o códigos criptográficos derivados de los autómatas celulares[1, 2, 3].

3. Autómatas celulares

Un autómata celular, en su versión más simple, es una línea unidimensional de sitios o celdas, donde cada una es blanca o negra. El color o estado de esta celda puede cambiar conforme al tiempo. Con cada paso discreto de tiempo, las celdas se actualizan (ya sea para mantener o cambiar su color previo) de acuerdo a la función de su estado anterior y al de las dos celdas vecinas a ella (una por el lado izquierdo y otra por el lado derecho). Existen además, otros espacios disponibles de autómatas celulares, donde se consideran más parámetros como lo son el número de estados en las celdas, vecindarios mayores, plantillas más amplias y dimensiones adicionales, colores, entre otros.

3.1. Reglas

A las condiciones de vecindad de un autómata celular se le conoce como "regla". Existen 256 (2^8) reglas para los autómatas celulares con un estado binario variable (0,1) y una vecindad de 1 con longitud de tres. Cada una de ellas está especificada por un código decimal obtenido a partir de las ocho permutaciones para la vecindad 1 en orden descendiente y los leemos como un código binario de ocho dígitos, lo cual nos da el número de la regla. La regla 30, por ejemplo, está definida por la configuración dada en la figura 1. Nótese que la secuencia 00011110 es la representación binaria del número 30.

Wolfram propone un esquema de clasificación, el cual divide las reglas de autómatas celulares en cuatro categorías de acuerdo a sus evoluciones a partir de una condición inicial "desordenada.º aleatoria. La clase 1, también conocida como de tipo fijo, la cual evoluciona rápidamente a un estado estable y homogéneo en el que todos los sitios tienen el mismo valor y cualquier aleatoriedad en el patrón inicial desaparece; la clase 2, también conocida como de tipo periódico, en la cual se repite un mismo patrón como un bucle donde su

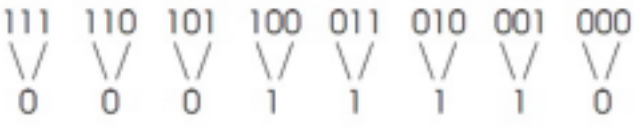


Figura 1. Representación binaria de la regla 30.

evolución es a gran velocidad y cualquier aleatoriedad en el patrón inicial solo dejaría restos que complementarían el bucle; la clase 3, también conocida como de tipo caótico o pseudo-aleatorio, en donde su evolución conduce a un patrón caótico donde cualquier estructura estable es rápidamente destruida por el ruido circundante y los cambios tienden a extenderse de manera indefinida; y la clase 4, de tipo complejo, la cual presenta comportamientos tanto de la clase 2 y 3 y suelen presentar una evolución más lenta. Teniendo una condición inicial simple, existen 13 reglas de autómatas celulares en las clases 3 y 4 calculadas en [11], que son las siguientes: 30, 45, 75, 79, 86, 89, 101, 110, 124, 135, 137, 149, 193. Éstas se amplían a 38 reglas de clase 3 si se tienen condiciones iniciales aleatorias [9], las cuales son las siguientes: 18, 22, 30, 45, 54, 60, 73, 75, 86, 89, 90, 101, 102, 105, 106, 109, 110, 120, 122, 124, 126, 129, 135, 137, 146, 147, 149, 150, 151, 153, 161, 165, 169, 182, 183, 193, 195 y 225. Según un estudio realizado en Brasil [7], las reglas de clase 3 pueden ser clasificadas en cuatro distintas subclases: Depósito Aleatorio (Declaración), representada por las siglas RD; Percolación Dirigida, representada por las siglas DP; Percolación Compacta Dirigida, de siglas CDP; y autómatas celulares Domany-Kinzel, de siglas DKCA y donde pueden ser simétricos o asimétricos. Siendo que las reglas de clase 3 presentan comportamientos caóticos y pseudo-aleatorios, se eligieron cuatro reglas. La regla 30 perteneciente a la subcategoría RD; la regla 54 perteneciente a la subcategoría DKCA (asimétrica); la regla 73 perteneciente a la subcategoría CDP; y la regla 110 perteneciente a la subcategoría DP y DKCA (simétrica).

4. Pseudo-aleatoriedad

La necesidad de obtener números aleatorios y pseudo-aleatorios se plantea en muchas aplicaciones criptográficas, pues se emplean llaves que deben ser generadas con dichas características. Por ejemplo, para cantidades auxiliares usadas en generación de firmas digitales, ó para generar desafíos en autentificación de protocolos. El Instituto Nacional de Estándares y Tecnología (NIST) proporciona un conjunto de pruebas estadísticas de aleatoriedad y considera que estos procedimientos son útiles en la detección de desviaciones de una secuencia binaria en la aleatoriedad [4]. Existen dos tipos básicos de generadores usados para producir secuencias aleatorias: Generadores de Números Aleatorios (RNGs) y Generadores de Números Pseudo-Aleatorios (PRNGs). Para aplicaciones criptográficas, ambos tipos de generadores producen un flujo de ceros y unos que pueden ser

divididos en sub-flujos ó bloques de números aleatorios. Nuestro interés está en la revisión de un generador tipo PRNGs, en este caso, si la semilla (línea inicial) es desconocida, en el paso siguiente el número producido en la secuencia debe ser impredecible a pesar de todo conocimiento de números aleatorios anteriores en la secuencia. Esta propiedad se conoce como imprevisibilidad siguiente, y es lo que se presume que obtenemos mediante autómatas celulares de clase 3. El conjunto de pruebas de NIST es un paquete estadístico que consiste en 15 pruebas que se desarrollaron para probar la aleatoriedad de (arbitrariamente largas) secuencias binarias producidas por hardware y software basado en generadores criptográficos de números aleatorios o pseudo-aleatorios. Dichas pruebas se enfocan en diversos tipos de no aleatoriedad que pueden existir en una secuencia. Las 15 puebas son:

- Prueba de frecuencia (Monobit).
  – Esta prueba mide la proporción de ceros y unos de toda una secuencia.
- Prueba de frecuencia dentro de un bloque.
  – Esta prueba mide la proporción de unos dentro de un bloque de M bits.
- Prueba de corridas.
  – Esta prueba mide el total de corridas en una secuencia, donde una corrida es una secuencia interrumpida de bits idénticos.
- Prueba de la más larga corrida de unos en un bloque.
  – Esta prueba mide la corrida más larga de unos dentro de un bloque de M bits.
- Prueba de rango de la matriz binaria.
  – Esta prueba mide el rango de sub-matrices disjuntas de toda la secuencia.
- Prueba de la transformada discreta de Fourier (Espectral).
  – Esta prueba mide las alturas de los picos en las transformadas discretas de Fourier de las secuencias.
- Prueba de la no acumulación de coincidencia de plantilla.
  – Esta prueba mide el número de ocurrencias de cadenas destino pre-especificadas. Una ventana de m bits es usada para buscar un patrón específico de m bits.
- Prueba de acumulación de coincidencia de plantilla.
  – Esta prueba también mide el número de ocurrencias de cadenas destino pre-especificadas. La diferencia con la prueba anterior reside en la acción realizada al encontrar un patrón.
- Prueba de Estadística Universal de Maurer.
  – Esta prueba mide el número de bits entre los patrones de juego (una medida que está relacionada con la longitud de una secuencia comprimida).
- Prueba de complejidad lineal.
  – Esta prueba mide la longitud de un Registro de Desplazamiento con Retroalimentación Lineal (LFSR). Una baja longitud LFSR implica no aleatoriedad.
- Prueba de serie.
  – Esta prueba mide la frecuencia de todos los posibles patrones de m bits acumulados a través de la secuencia completa.

- Prueba de entropía aproximada.
  – Esta prueba tiene el mismo enfoque que la anterior, con el propósito de comparar la frecuencia de bloques acumulados de dos consecutivas/adyacentes longitudes (m y $m+1$).
- Prueba de sumas acumulativas.
  – Esta prueba mide la excursión máxima (desde cero) del paseo aleatorio definido por la suma acumulada de ajustados (-1, +1) dígitos en la secuencia.
- Prueba de excursiones aleatorias.
  – Esta prueba mide el número de ciclos teniendo exactamente k visitas en una suma acumulativa de un paseo aleatorio.
- Prueba variante de excursiones aleatorias.
  – Esta prueba mide el total de veces que un estado particular es visitado (es decir, se produce) en una suma acumulada de un paseo aleatorio.

5. Metodología

Primeramente se recopiló información sobre las clases que propone Wolfram para clasificar las reglas del autómata celular. Fue con esta recopilación, que se encontró que existían subcategorías propuestas dentro de la clase 3. Y, al encontrar estas subcategorías, se decidió realizar pruebas de aleatoriedad a una regla por división, como se mencionó anteriormente.

Cuadro 1. Resultados de las pruebas aplicadas a reglas representantes de cada una de las 4 clases de Wolfram (A significa que la prueba fue aprobada y R que fue reprobada).

Prueba	R30	R54	R73	R110
Frecuencia (Monobit)	A	R	R	R
Frecuencia dentro de un bloque	A	R	R	R
Corridas	R	R	R	R
Más larga corrida de unos en un bloque	A	R	R	R
Rango de la matriz binaria	A	R	A	R
Transformada discreta de Fourier (Espectral)	R	R	R	R
No acumulación de coincidencia de plantilla	A	A	A	A
Acumulación de coincidencia de plantilla	A	A	R	R
Estadística Universal de Maurer	A	A	R	R
Complejidad lineal	A	R	A	R
Serie	A	R	R	R
Entropía aproximada	A	R	R	R
Sumas acumulativas	A	R	R	R
Excursiones aleatorias	A	R	R	R
Variante de excursiones aleatorias	A	R	R	R

Por tanto, para cada una de las reglas elegidas, se generaron mediante *Mathematica* 1000 archivos con 10000 datos. Estos 10000 datos son conformados a partir de una 'cadena inicial de 100 caracteres, la cual es generada aleatoriamente y se compone únicamente de 0s y 1s. Después de generar los archivos para cada regla, estos se juntaron en un solo archivo, que posteriormente se analizaría mediante la Suite de Pruebas de la NIST. Al finalizar el análisis de cada archivo final (uno por regla), se obtuvo un archivo con los resultados del análisis, lo cual nos permite ver si la regla tiene o no características que la avalen como pseudo-aleatorias o no. En el Cuadro 1, podemos ver una comparación de las reglas y su pase en cada una de las pruebas.

Los resultados y calificaciones de la tabla se obtuvieron después de realizar varias veces el procedimiento de generación y prueba de datos y promediar los resultados por prueba estadística y por intento.

6. Conclusiones

Como podemos ver, la regla que más propiedades de pseudo-aleatoriedad presenta es la regla 30, por lo que podemos concluir que se puede considerar que es pseudo-aleatoria. Es importante notar que las dos pruebas que reprueba no se les considera que afecten a los resultados, puesto que se notó que sus reprobaciones son debido a que las corridas son consideradas perfectas, lo cual es poco probable en un generador de números pseudo-aleatorios. Por el contrario, la enorme falta de propiedades básicas de aleatoriedad en las otras reglas, nos permite pensar que es posible que solo las reglas de clase 3 que pertenezcan a la subcategoría RD sean las que presenten pseudo-aleatoriedad. Se continúa realizando pruebas nuevamente, con reglas distintas a las elegidas, para comprobar si es la subcategoría o si solamente fue una coincidencia entre las reglas elegidas de cada subcategoría que solo aquella perteneciente a la subcategoría RD sea pseudo-aleatoria. Consideramos que después de realizar estas pruebas se podría continuar con el trabajo enfocándose a la encriptación y recomendamos que se pruebe cada una de las reglas pseudo-aleatorias encontradas, como llave de un sistema simple de encriptación y, posteriormente, en uno más complejo para verificar el funcionamiento de las mismas como llaves y su utilidad.

Referencias

[1] J-P. Delahaye, Les Automates, *Pour La Science,* pp.126–134, 1991.

[2] H. A. Gutowitz, Cellular Automata: Theory and Experiment; proceedings of an interdisciplinary workshop, Editor. vol. 45, *Physica D,* 1990.

[3] H. A. Gutowitz, Artificial Life Simulators and Their Applications, *DRET Technical Report,* 1994.

[4] A.L. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, *NIST Special Publication 800-22 A*, (2001) revised 2010.

[5] O. Lafe, Cellular Automata Transforms, *Kluwer Academia Publishers,* 2000.
[6] E. Martin, John Conway's Game of Life. Recuperado el Julio de 2010, de http://www.bitstorm.org/gameoflife/
[7] T.G. Mattos y J. G. Moreira. Universality Classes of Chaotic Cellular Automata, *Brazilian Journal of Physics,* vol. 34, núm. 02A, pp. 448–451, 2004.
[8] P. Guan, Cellular automaton public-key cryptosystem, *Complex Systems,* vol. 1, pp. 51–57, 1987.
[9] S. Wolfram, Theory and Applications of Cellular Automata, *Rev. Mod. Phys.* 55, 601, 1983.
[10] S. Wolfram, *A New Kind of Science,* Wolfram Media Inc., 2002.
[11] H. Zenil, Compression-based Investigation of the Dynamical Properties of Cellular Automata and Other Systems, *Complex Systems,* 19(1), pages 1-28, 2010.

Modelación de una red de Petri mediante un autómata celular

Carlos Adrián Jaramillo Hernández, Juan Carlos Seck Tuoh Mora,
Joselito Medina Marín

Centro de Investigación Avanzada en Ingeniería Industrial
Universidad Autónoma del Estado de Hidalgo
Pachuca de Soto, Hidalgo, México
uaeh86@hotmail.com

Resumen En el presente proyecto de investigación se llevó a cabo la modelación de una red de Petri (RdP) mediante un autómata celular (AC), así como el análisis de dicha red por medio de la evolución del autómata celular desde una configuración inicial apropiada, con el fin de conocer su dinámica y propiedades más relevantes.

1. Introducción

Las RdP son una herramienta gráfica y matemática que se han aplicado en el estudio de sistemas que se caracterizan por ser concurrentes, asíncronos, distribuidos, paralelos, no-deterministas, y/o estocásticos [5]. Se han aplicado ampliamente en la modelación y análisis de sistemas de eventos discretos, sistemas de manufactura flexible, protocolos de comunicación, sistemas de base de datos distribuidos, programación paralela y concurrente, sistemas operativos y compiladores, lenguajes formales, entre algunas otras aplicaciones [5]. Con una RdP se puede estudiar la dinámica de operación del sistema que está siendo modelado, mediante el uso de herramientas de análisis, tales como el árbol de alcanzabilidad y la ecuación de estado [5][9].

Por otro lado, los AC pueden ser utilizados en la modelación y análisis de sistemas dinámicos, entre los que existe una interacción local y una evolución de forma paralela. Los AC han sido aplicadas en teoría de la computación, biología, física y en ciencias sociales, en la modelación de individuos y sociedades. Además, se han utilizado para el estudio de comportamiento de incendios, planeación urbana, tránsito vehicular, entre muchas otras aplicaciones [2][4][1].

Dado que un AC evoluciona a partir del estado de los vecinos que cada célula tiene y el disparo de una transición en la RdP está condicionado al estado de sus lugares de entrada, es factible considerar segmentos de la RdP como una célula en el AC, y llevar a cabo la modelación de la dinámica de la RdP en el AC.

La mayor debilidad que tienen las RdP es el problema de complejidad, es decir, los modelos basados en RdP tienden a ser demasiado grandes para analizarlos, aun en sistemas de tamaño moderado.

Por lo anterior, en este trabajo, se propone llevar a cabo la modelación de RdP mediante un AC, ya que la complejidad que presenta un AC debido al número

de células es menor a la que presentaría el modelo de RdP en la modelación del mismo sistema. Además, si se desea modelar el sistema con más elementos, en el AC solamente se agregan más células, mientras que en la RdP se tendrían que agregar los lugares, transiciones y arcos correspondientes a los elementos nuevos.

Existe poco trabajo de investigación relacionado a la interacción de una RdP y un AC, además, en el análisis de sistemas de manufactura flexible se ha aplicado en mayor medida teoría de RdP, y no se ha aprovechado lo que un AC ofrece para el análisis de tales sistemas.

En el presente trabajo se vincularon éstas herramientas, utilizadas en la modelación y análisis de sistemas de eventos discretos.

1.1. Redes de Petri

Las RdPs fueron introducidas por medio de la tesis doctoral de Carl Adam Petri [6]. Se describen como una herramienta de naturaleza gráfica para el diseño y análisis de sistemas dinámicos de eventos discretos [3]. Una RdP se representa gráficamente por un grafo dirigido bipartito. Los dos tipos de nodos, lugares y transiciones, representan las variables que definen el estado del sistema. Los lugares se representan por círculos, las transiciones por barras y el marcado M se representa por una distribución en los lugares denominados marcas, como se muestra en la figura 1.

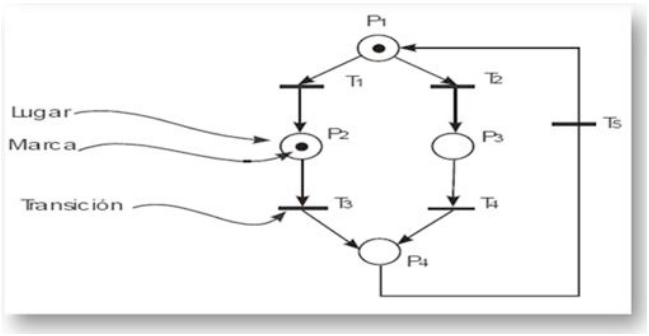


Figura 1. Red de Petri y sus componentes.

Formalmente, una RdP puede definirse como una tupla de 5 elementos $PN = \{P, T, F, W, M_0\}$, donde:
$P = \{p_1, p_2, \ldots, p_m\}$ es un conjunto finito de lugares,
$T = \{t_1, t_2, \ldots, p_n\}$ es un conjunto finito de transiciones,
$F \subseteq \{P \times T\} \cup \{T \times P\}$ es un conjunto de arcos,
$W : F \to \{1, 2, 3, \ldots\}$ es una función de asignación de peso a los arcos,
$M_0 : P \to \{0, 1, 2, 3, \ldots\}$ es la marca inicial.
$P \cap T = \emptyset$ and $P \cup T \neq \emptyset$

La dinámica del sistema modelado con una RdP se basa en la siguiente regla de disparo:

1. una transición t se habilita si cada lugar de entrada tiene al menos $w(p,t)$ tokens.
2. Una transición habilitada puede disparar o no, dependiendo si el evento ocurre o no.
3. El disparo de una transición t habilitada elimina $w(p,t)$ tokens de cada lugar de entrada p de t, y agrega $w(t,p)$ tokens a cada lugar de salida p de t, donde $w(t,p)$ es el peso del arco que va de t a p.

1.2. Autómatas celulares

Los ACs, que son una clase de sistemas matemáticos espacial y temporalmente discretos y determinísticos [7], caracterizados por una interacción local y una forma de evolución paralela, utilizados en el análisis de sistemas dinámicos en los cuales el comportamiento está regulado por la interacción local de células, las cuales pueden tener un conjunto finito de estados. El cambio de estado de cada célula dependerá del estado en que se encuentren sus vecinas y el de la misma célula. De esta manera, el vecindario de cada célula estará formado por ésta y las r células situadas tanto a la derecha como a la izquierda de la misma, teniendo un total de $2r+1$ células en el vecindario. Una de las aportaciones importantes de Stephen Wolfram [4] es la notación (k,r) para un AC de una dimensión, en la que k es el número de estados que puede tomar cada una de las células y r es el radio de vecindad e indica el número de células vecinas a ambos lados de las mismas. En conjunción estos parámetros representan el vecindario de las células del AC, de tal manera que el número de vecindarios diferentes posibles está dado por la expresión k^{2r+1}, formados por $2r+1$ células, como se muestra en la figura 2.

Para denotar los cambios de estado en la evolución del AC y de la RdP, se le asigna la letra griega τ para diferenciarlo de las transiciones t.

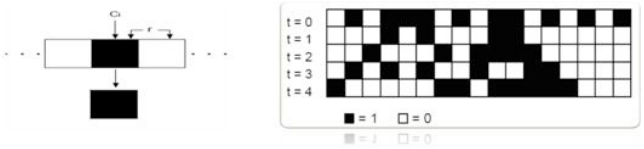


Figura 2. Autómata celular y sus componentes.

2. Modelación de una RdP mediante un AC

La importancia de modelar una RdP con un AC radica en las ventajas que tienen los ACs para ser analizados de manera gráfica, encontrando candados

mortales, ciclos, además de poder modificar rápidamente el estado inicial del sistema y ver la evolución resultante.

Para llevar a cabo ésta modelación se utilizó una RdP ordinaria y un AC de una dimensión.

Teniendo como referencia la *célula de manufactura flexible* encontrada en [8] que es simulada por una RdP, se aplica la siguiente metodología:

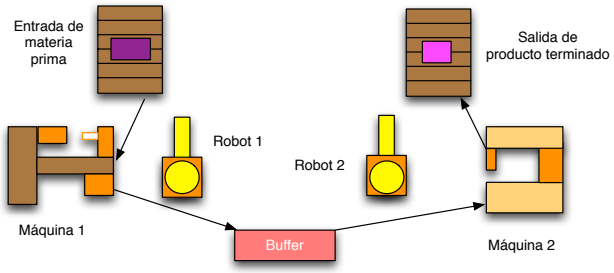


Figura 3. Célula de manufactura flexible.

1.- La RdP se divide en sub redes o módulos, que emularán cada célula del AC. Figura 4.

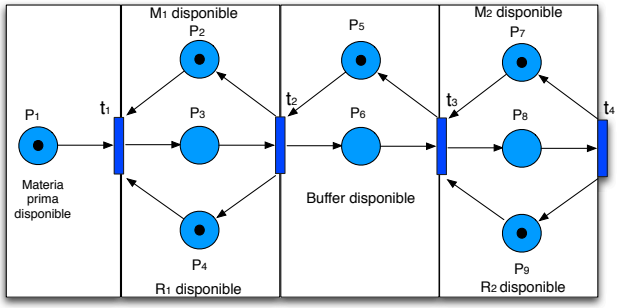


Figura 4. Division en subredes o módulos de la RdP.

2.- Se asigna un cero o un uno, dependiendo de la configuración, a cada subred o módulo que representa una célula, esta sólo tendrá dos estados posibles. Figuras 5, 6, 7 y 8.

De esta manera se hace un arreglo de ceros y unos que conforman el AC, y con una regla de evolución basada en la dinámica de la RdP se evolucionará el estado inicial del mismo para observar su comportamiento con las dos herramientas.

Figura 5. Asignación de valores para el primer módulo.

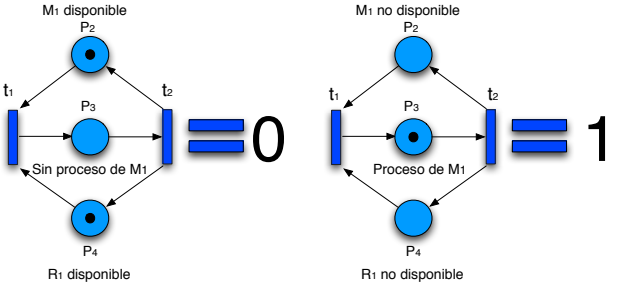


Figura 6. Asignación de valores para el segundo módulo.

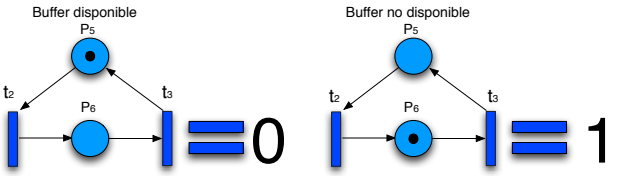


Figura 7. Asignación de valores para el tercer módulo.

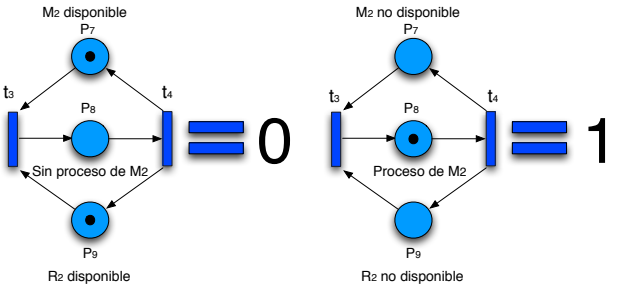


Figura 8. Asignación de valores para el cuarto módulo.

La regla de evolución queda determinada, como se muestra en la figura 9, donde la célula central es la que evolucionará dependiendo del estado de sus vecinas y con esto poder observar los cambios de estado partiendo de un estado inicial.

0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		1
1	0	0		1
1	0	1		1
1	1	0		0
1	1	1		1

Figura 9. Regla de evolución para el AC.

Ejemplificando lo anterior, se describe la evolución tanto del AC con su regla de evolución, como en la RdP. Figuras 10, 11, 12, 13, 14 y 15.

τ_0	1	0	0	0
τ_1	0	1	0	0
τ_2	0	0	1	0
τ_3	0	0	0	1
τ_4	1	0	0	0

Figura 10. AC y su evolución.

Como se observa, la regla de evolución dicta el cambio de estado del AC y a su vez se ve reflejado en la dinámica de la RdP. En este caso τ_4 es igual a τ_0 por lo que se observa que el AC se cicla y vuelve a su estado inicial.

Para fines prácticos se programó este AC en MATLAB que nos permite aumentar el número de células, así como el número de evoluciones. El siguiente ejemplo se ejecutó el programa con un AC de 10 células y 50 evoluciones. Figura 16.

La ventaja de programar el AC radica en que se puede ingresar una cantidad grande de células y evoluciones. Así, se le puede dar una interpretación de forma gráfica y describir la dinámica del sistema.

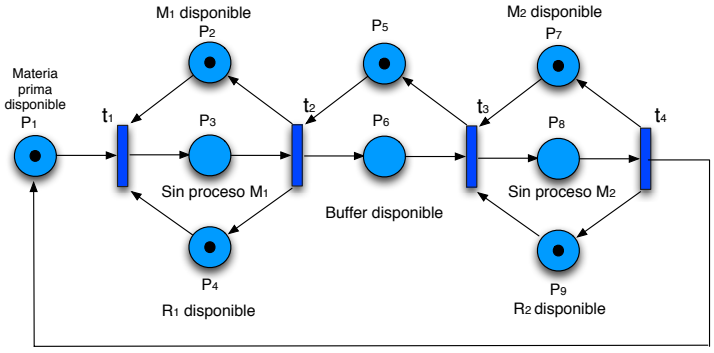


Figura 11. RdP en τ_0.

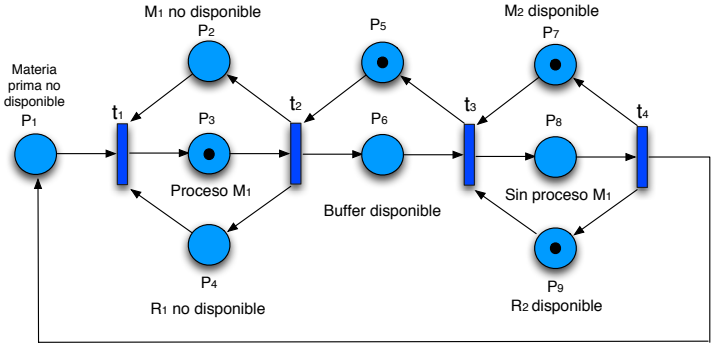


Figura 12. RdP en τ_1.

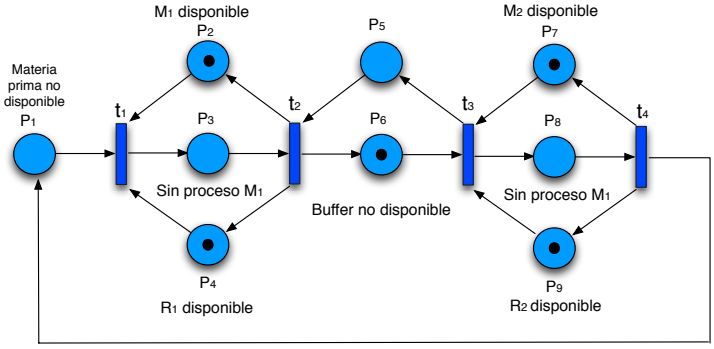


Figura 13. RdP en τ_2.

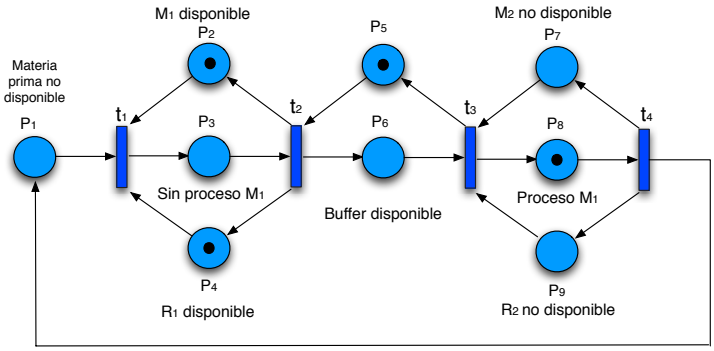


Figura 14. RdP en τ_3.

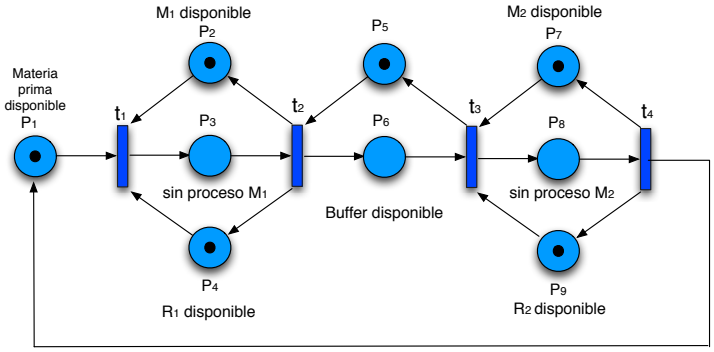


Figura 15. RdP en τ_4.

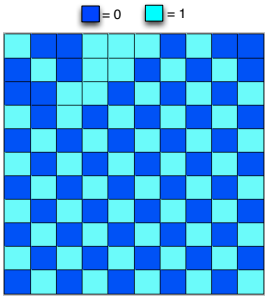


Figura 16. Evolución del AC con 10 células y 10 evoluciones.

3. Modelación de una RdP con recurso compartido mediante un AC

En los sistemas de producción flexible es común ver que se comparten ciertos recursos como máquinas, brazos de robot, inventarios temporales y/o recursos que hacen dos o más tareas.

En este caso, el sistema de producción estudiado consta de dos líneas de producción paralelas, donde cada una hace un trabajo diferente y ambas comparten dos máquinas para la ejecución de cada trabajo. Para la elaboración de ambos productos se tiene que pasar por dos procesos que son realizados por ambas máquinas, como se muestra en la figura 17.

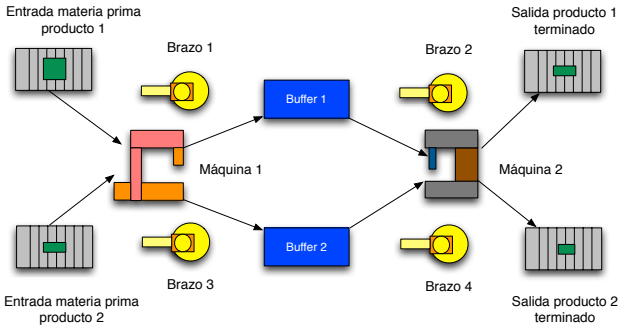


Figura 17. Célula de manufactura flexible con recurso compartido.

Cada línea inicia con la entrada de materia prima al sistema por medio de una banda transportadora. La materia prima del producto 1 y del producto 2, es llevada por los brazos de robot de las bandas a las máquinas 1 y 2 para su posterior procesamiento, por lo que ambas tienen que ser compartidas entre las dos líneas para realizar los trabajos y finalmente obtener un producto terminado. Los buffers se utilizan para desacoplar ambos procesos a fin de que exista menos interdependencia entre ambos, haciendo más eficiente el flujo de materiales.

Tomando en cuenta la configuración de este sistema de producción flexible con recurso compartido, se construye la RdP para describir la dinámica del sistema, dando como resultado la siguiente red. Figura 18.

La retroalimentación de P_4 a P_1 y de P_7 a P_{10} significan que los trabajos 1 y 2 fueron terminados y puede entrar nuevamente materia prima al sistema.

El conflicto en estos sistemas de producción con recurso compartido consiste en que, cuando existe materia prima disponible en las dos líneas para ser procesada por la máquina 1 o producto semiprocesado de los buffers hacia la máquina 2, se tiene que decidir cuál va a ser el producto que primero será procesado. En caso que suceda esto, la solución planteada es darle prioridad a la línea de producción superior y así hacer determinista la entrada de materia. Figura 19.

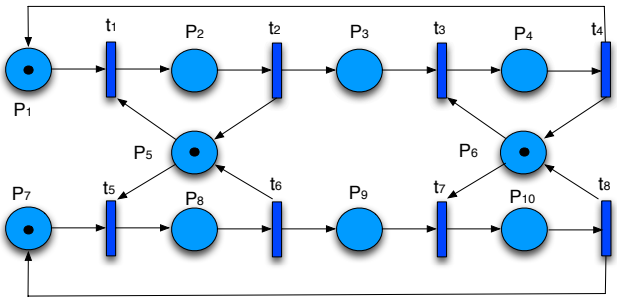


Figura 18. RdP con recurso compartido.

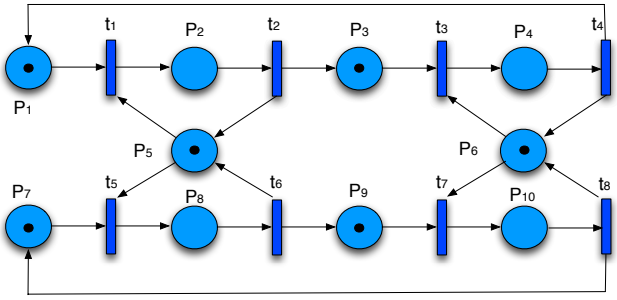


Figura 19. Conflicto en RdP con recurso compartido.

No existe este conflicto cuando la disponibilidad de materia prima solo se da en la línea 1 o en la línea 2, hacia la máquina 1 y de los buffers 1 y 2, hacia la máquina 2, ya que a la llegada de materia prima simplemente es procesado por la máquina correspondiente sin tener que decidir, como en el caso anterior. Figura 20.

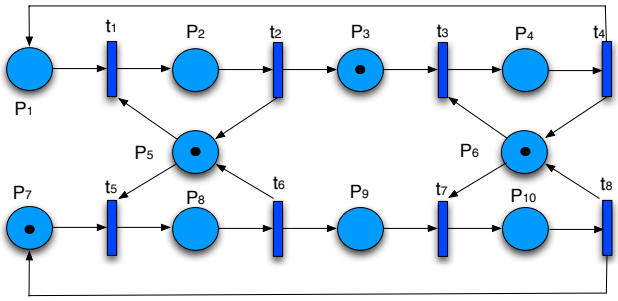


Figura 20. Sin conflicto en RdP con recurso compartido.

Para la descripción de cada lugar, se toma en cuenta que cuando se encuentra con token alguno de los lugares significa que está disponible o se esté llevando a cabo un proceso, y sin el que no esté disponible, como se muestra en la figura 21.

Lugar	Actividad
P_1	Trabajo 1 disponible.
P_2	Trabajo 1 procesado por la maquina 1.
P_3	Operación de trabajo 1 por la maquina 1 terminado
P_4	Trabajo 1 procesado por la maquina 2.
P_5	Maquina 1 disponible
P_6	Maquina 2 disponible
P_7	Trabajo 2 disponible.
P_8	Trabajo 2 procesado por la maquina 1.
P_9	Operación de trabajo 2 por la maquina 1 terminado
P_{10}	Trabajo 2 procesado por la maquina 2.

Figura 21. Lugares y actividades de la RdP.

Tomando como referencia la RdP que representa el sistema de producción antes mencionado, se lleva a cabo la división en sub redes o módulos de la red de la siguiente manera: como se puede observar los lugares P_1 y P_7 con P_3 y P_9, tienen la misma configuración, asi como P_2, P_5 y P_8 con P_4, P_6 y P_{10}, por lo que se asignaron las letras A y B para diferenciar entre las dos configuraciones de las subredes. Figura 22.

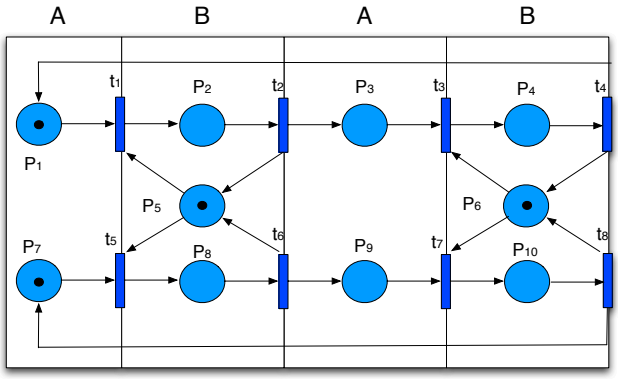


Figura 22. División de RdP.

Ahora cada una de las subredes se tomará como una célula, para que de esta manera se pueda conformar el AC. Para cada configuración de subred se le será asignado un número dependiendo del marcado de la misma, en el caso de la configuración A, se asignarán los siguientes números. Figura 23.

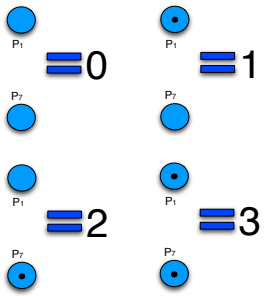


Figura 23. Asignación de valores para la configuración A.

Y para la configuración B como se muestra en la figura 24.

Como se puede observar la configuración A tiene cuatro estados 0, 1, 2 y 3 y la configuración B solo tres 0, 1 y 2.

Haciendo referencia de las asignaciones anteriores se hace un arreglo de AC con los números que dan como resultado cada una de las configuraciones. Figura 25.

Tomando en cuenta que se trabaja con un AC con un tamaño de vecindad de tres, es necesario tener en cuenta que para la evolución de la célula central, podría ser en un caso una configuración ABA o BAB, por lo que se determinó el desarrollo de dos reglas de evolución, una para cada caso.

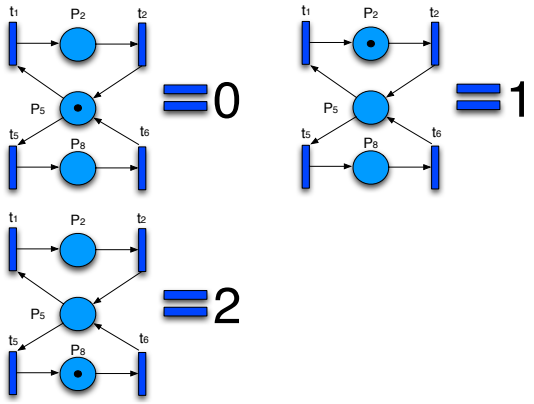


Figura 24. Asignación de valores para la configuración B.

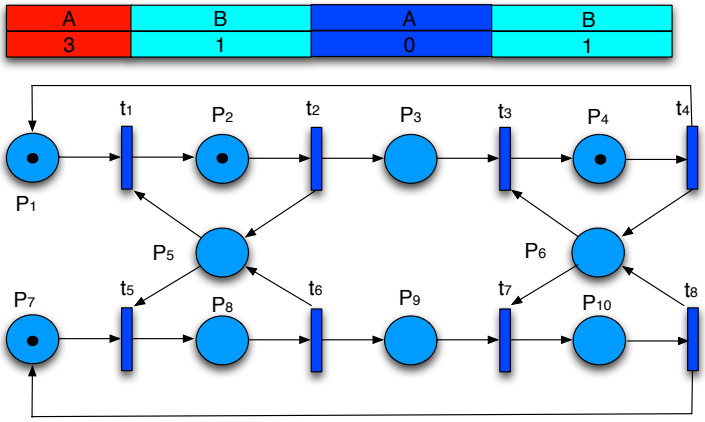
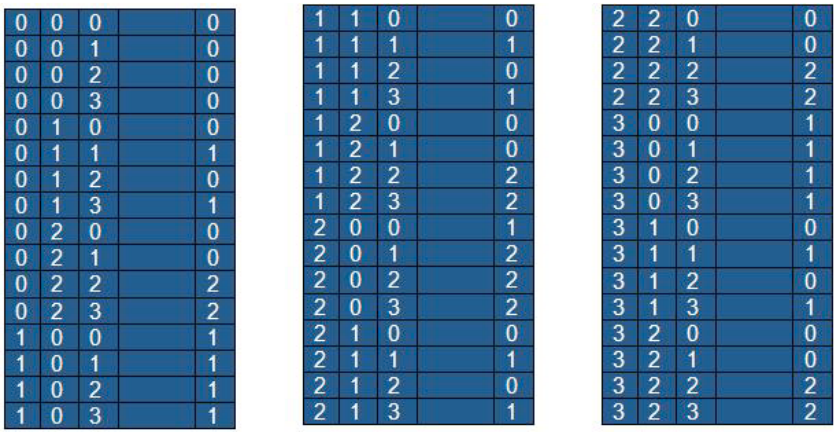


Figura 25. AC con su respectiva RdP.

Para la configuración ABA se muestra en la figura 26, y para el caso de la configuración BAB. Figura 27.



0	0	0		0
0	0	1		0
0	0	2		0
0	0	3		0
0	1	0		0
0	1	1		1
0	1	2		0
0	1	3		1
0	2	0		0
0	2	1		0
0	2	2		2
0	2	3		2
1	0	0		1
1	0	1		1
1	0	2		1
1	0	3		1

1	1	0		0
1	1	1		1
1	1	2		0
1	1	3		1
1	2	0		0
1	2	1		0
1	2	2		2
1	2	3		2
2	0	0		1
2	0	1		2
2	0	2		2
2	0	3		2
2	1	0		0
2	1	1		1
2	1	2		0
2	1	3		1

2	2	0		0
2	2	1		0
2	2	2		2
2	2	3		2
3	0	0		1
3	0	1		1
3	0	2		1
3	0	3		1
3	1	0		0
3	1	1		1
3	1	2		0
3	1	3		1
3	2	0		0
3	2	1		0
3	2	2		2
3	2	3		2

Figura 26. Reglas de evolución para la configuracion ABA.

De la misma manera se programó este AC en Matlab, con base a la reglas de evolución de ambas configuraciones. Se desarrolló este ejemplo con una AC de 10 células y 40 evoluciones. Figura 28.

4. Conclusiones

La investigación donde se combinan la teoría de RdP y AC es una área poco explorada, sin embargo, dentro de este trabajo se ha encontrado que es factible llevar a cabo la modelación y simulación de una RdP mediante un AC. En éste caso se representaron dos Sistemas de Manufactura Flexible, un sistema lineal de producción y otro con recurso compartido.

Las ventajas de modelar las RdPs con ACs radica en que permiten representar una RdP de una gran cantidad de lugares y transiciones con pocas células de un AC. Como se observó el análisis de la RdP se hace de manera más sencilla porque se visualizan de manera gráfica los cambios de estado en cada célula y se determinan ciertos compartamientos como un candados mortales, ciclos, flujo de material y estabilidad del sistema, que se presentan en una RdP.

Como trabajo futuro de ésta investigación se aplicarán las herramientas gráficas que ofrecen los ACs, tales como: diagramas de Bruijin, diagrama de pareja y diagramas de subconjuntos, para analizar la dinámica de los Sistemas de Manufactura Flexible que se representan como una RdP.

0	0	0		0
0	0	1		0
0	0	2		0
0	1	0		0
0	1	1		1
0	1	2		1
0	2	0		0
0	2	1		2
0	2	2		2
0	3	0		2
0	3	1		3
0	3	2		3
1	0	0		1
1	0	1		1

1	0	2		1
1	1	0		0
1	1	1		1
1	1	2		1
1	2	0		0
1	2	1		3
1	2	2		3
1	3	0		2
1	3	1		3
1	3	2		3
2	0	0		2
2	0	1		2
2	0	2		2
2	1	0		3

2	1	1		3
2	1	2		3
2	2	0		0
2	2	1		2
2	2	2		2
2	3	0		2
2	3	1		3
2	3	2		3

Figura 27. Reglas de evolución para la configuracion BAB.

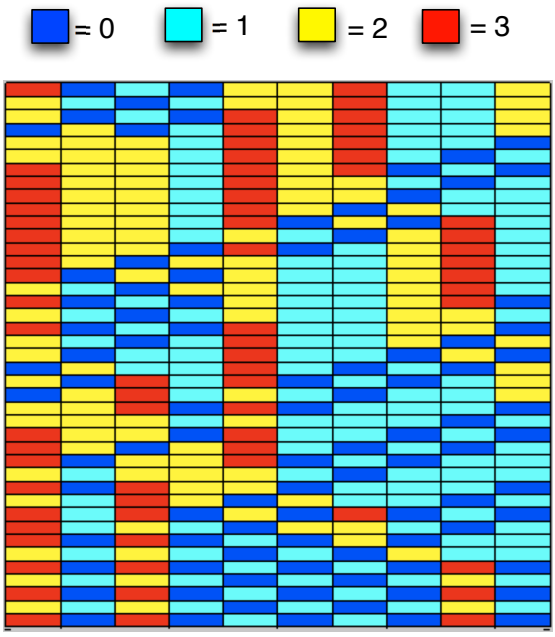


Figura 28. Evolución AC recurso compartido.

Referencias

[1] Back, T., Dornemann, H., Hammel, U., & Frankhauser P. (1996). Modeling urban growth by cellular automata, In *Parallel Problem Solving from Nature*, 636-645.

[2] Ganguly N., Sikdar B.K., Deutsch A., Canright G., & Chaudhuri P. P. (2003). A survey on Cellular Automata, *Technical Report 9*, Centre of High Performance Computing, Dresden University of Technology.

[3] Ilachinski, A. (2002). *Celullar Automata: A Discrete Universe*, World Scientific, USA.

[4] Karafyllidis, I. & Thanailakis, A. (1997). A model for predicting forest fire spreading using Cellular Automata, *Ecological Modelling*, 99:1, 87-97.

[5] Murata T. (1989) Petri Nets: Properties, Analysis and Apllications, *Proceedings of the IEEE*, 77:4, 541-580.

[6] Petri, C. A. (1966). Communication with automata, *Technical report RADC-TR-65-377*, Volume I, Final Report, Supplement I.

[7] Wolfram, S. (1983). Statical Mechanics of Cellular Atomata, *Reviews of Modern Physics*, 55:3, 601-644.

[8] Rodríguez, R. Z. (2002). Modelación de flujo de tránsito de autos utilizando autómatas celulares. *Tesis de Maestría*, CINVESTAV, México.

[9] Zurawski, R. & MengChu Zhou (1994). Petri Nets and Industrial Appications: A Tutorial, *IEEE Transactions on Electronics*, 41:6, 567-583.

Índice alfabético