Computación basada en reacción de partículas en un autómata celular hexagonal

Rogelio Basurto Flores*
Paulina Anaid León Hernández †
Miguel Olvera Aldana‡
Genaro Juárez Martínez‡

Centro de Estudios Avanzados del IPN
Depto. de Microcomputadoras, BUAP
Escuela Superior de Cómputo, IPN
* rbasurto@computacion.cs.cinvestav.mx
† pleon@computacion.cs.cinvestav.mx
‡ Director de tesis
Paper received on 31/08/10, Accepted on 27/09/10.

Resumen. Se implemento computación a través de compuertas lógicas mediante reacción de partículas dentro de un autómata celular hexagonal.

1 Introducción

En el año 2005 Andrew Wuensche descubrió un autómata celular hexagonal, que representa un sistema de reacción - difusión basado en una reacción química de tres sustancias: *activador, inhibidor y sustrato*; el autómata celular fue bien recibido dado que presentaba *gliders,* partículas que pueden viajar por el espacio de evoluciones; gracias a estas partículas y a las colisiones entre ellas se logróllegar a la construcción de las compuertas lógicas básicas, mostrando así su universalidad lógica; estos resultados fueron presentados en [2]; debido al comportamiento que presentóse le da el nombre de regla Beehive. Su estudio demostró que a pesar de que presentaba una lógica universal aún tenía detalles a superar, como la ausencia de "glider-guns" estáticos partículas básicas para la computación en autómatas celulares [1], estos resultados fueron un trabajo presentado en [3], que presenta una nueva regla derivada de Beehive. Esta nueva regla es llamada "Spiral rule" (regla espiral), en adelante regla Spiral, y tiene la ventaja de presentar glider-guns estacionarios y móviles, gracias a estoy al antecedente de la regla Beehive se sospecha que la regla Spiral podría soportar una lógica universal a través de compuertas lógicas.

Para obtener los resultados de este trabajo se hizo uso del simulador que se puede encontrar en [18].

Advances in Intelligent and Information Technologies.
Research in Computing Science 50, 2010, pp. 159.-168

2 Spiral rule

Un autómata celular se define omo l modelo de un *sistema complejo* que itera a través del tiempo, con una regla isotrópica de transición para el cambio de estados de las partículas individuales que lo componen, llamadas células, siendo el conjunto de estados que pueden tomar las células finito.

Tabla 1: Matriz de evolución de la regla Spiral

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

De manera formal un autómata celular está compuesto por la tupla:

- Q es un conjunto finito de estados.
- d, un número entero positivo que representa la dimensión del autómata
- e, es la configuración inicial de las células
- $f: S \rightarrow S$, es la función de transición local

En el particular caso de la regla Spiral se tiene un conjunto $Q = \{0, 1, 2\}$ representados mediante los colores blanco, rojo y negro respectivamente; la dimensión $d = 2$ y además, es importante mencionar que la forma de las células es hexagonal y debido a ello la vecindad utilizada será como la presentada en la figura 1, con 6 vecinos inmediatos; e representa la configuración inicial, es decir, los estados de las células para el tiempo $t = 0$; y finalmente, la función de transición estará dada mediante una matriz de transición, donde las columnas representan el número de células en estado 1 y las filas el número de células en estado 2, dicha matriz de transición es presentada en la tabla 1.

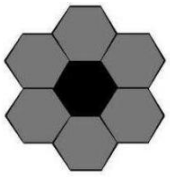

Figura 1: Vecindad hexagonal.

Lo anterior se ve directamente reflejado al hacer una evolución paso a paso de una configuración inicial sencilla como la mostrada en la figura 2, donde se presenta únicamente una célula en estado 1.

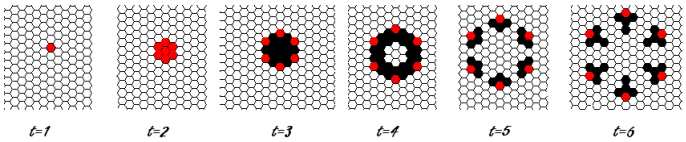

Figura 2: Evolución de una configuración inicial con sólo una célula en estado 1.

3 Dinámica y partículas básicas

La regla de evolución, representada con la matriz 1 permite observar la interacción entre células, lo que a través de las evoluciones hará que se formen estructuras de forma inherente llamadas partículas. Dichas partículas son *gliders*, *glider-guns* y *eaters*; un *glider* es una partícula que se puede mover a través del espacio de evoluciones; los *gliders-gun* son partículas más complejas que después de un determinado número de evoluciones lanzan un *glider*; el *eater* es una partícula estática capaz de eliminar otras partículas. En la figura 3 se muestran los tipos básicos de *gliders* que presenta el autómata, de ellos se podrán obtener diferentes datos relevantes al momento de utilizarlos con fines computacionales:

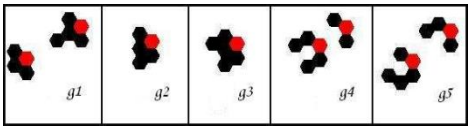

Figura 3: Tipos de gliders.

- *Peso:* se refiere a la cantidad de células que forman una partícula; de tener más de una fase (o forma) se toma la mayor como su peso.
- *Período:* es el número de evoluciones necesarias para que una partícula vuelva a tener su estructura inicial.
- *Desplazamiento:* se refiere al número de células que avanza el glider por período, se define su dirección en base al plano cartesiano.
- *Velocidad:* está determinada como P /D, es decir, Período/Desplazamiento.

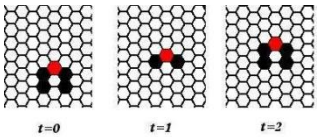

Figura 4: Glider g4.

Para explicar estos conceptos se puede tomar como ejemplo el *glider* g_4. Su peso es 5, pues tiene 5 células que lo conforman en su fase más grande; su período es 2, esto puede apreciarse de la figura 4, donde se observa

que le toma 2 tiempos regresar a su forma inicial. En la misma imagen se observa el desplazamiento que tiene el glider, donde avanza una célula cada tiempo y, sí para completar un período necesita de dos tiempos, entonces tiene un desplazamiento igual a 2; finalmente, la velocidad del glider g_4 se obtiene dividiendo su período entre su desplazamiento, es decir, $\frac{2}{2} = 1$. De esta manera se han analizado los 5 gliders, la información obtenida se muestra en la tabla 2.

Tabla 2: Características de gliders

Partícula	Peso	Velocidad	Período	Desplazamiento
g_1	5	1	2	2
g_2	5	1	1	1
g_3	6	1	1	1
g_4	5	1	2	2
g_5	5	1	2	2

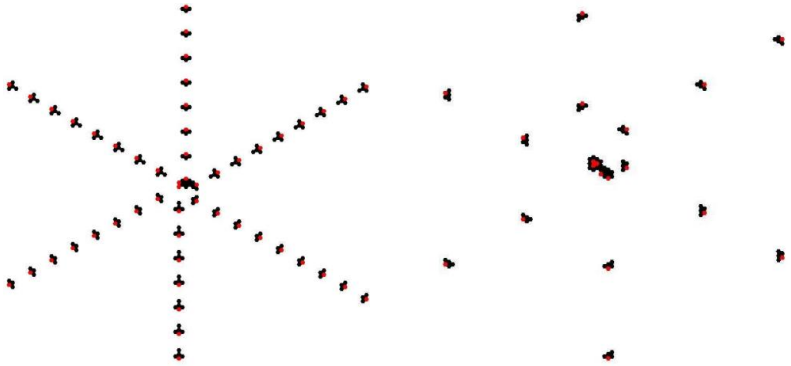

Figura 5: Glider-gun G1

Figura 6: Glider-gun G

La creación de gliders es muy importante, y para ello la regla Spiral tiene dos tipos de gliders-gun básicos, también llamados "guns", estos son mostrados en las figuras 5 y 6. La información que se puede obtener de estos guns está condensada en la tabla 3. De la tabla, se define la frecuencia como el número de gliders que produce en un período, y el período se define de la misma manera que para los gliders.

Tabla 3: Características de glider-guns.

Glider Gun	Glider que produce	período	Frecuencia
G_1	g_1	6	6
G_2	g_2	22	6

Otra partícula importante es aquella capaz de eliminar los gliders, se le denomina eater. Existen dos tipos básicos de eaters en la regla Spiral, estos se presentan en la figura 7. Sin importar cuantas veces y en que ángulo

choque un glider con un eater, este será destruido.

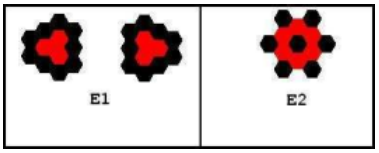

Figura 7: Tipos de eaters.

4 Computación en la regla Spiral

Mediante la manipulación de las partículas básicas que presenta la regla Spiral es posible implementar computación, y se demuestra mediante compuertas lógicas.

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 E1. Un glider-gun limitado en 5 de sus 6 flujos se puede apreciar en la figura 8. Esto mismo se puede extender para el glider-gun G2.

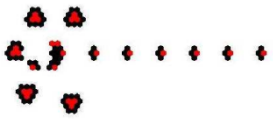

Figura 8: Glider-gun G1 con 5 flujos eliminados.

De la misma manera que en Life [8], en la regla Spiral 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 9.

La sincronización entre glider-guns es una de las bases primordiales para la creación de compuertas lógicas, no solo en la regla Spiral, sino en cualquier autómata, pues la computación está basada en las colisiones entre las partículas y la reacción que estas colisiones producen [2],[9],[11], [12]. Después de observar la regla Spiral se notó una similitud entre las colisiones existentes entre gliders, dichas colisiones se muestran en la figura 10. En la imagen se observan tres colisiones diferentes, la colisión del inciso A tiene como reacción el cambiar de dirección del 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 para la construcción de las compuertas lógicas.

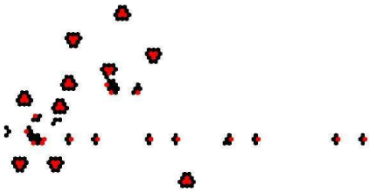

Figura 9: Flujo de gliders modificado.

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 necesarios 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; esto con ayuda de un simulador que permitía la manipulación de los estados mediante una interfaz gráfica que fue desarrollado para este fin.

Las compuertas implementadas mediante la regla Spiral son: AND, OR y NOT así, con estas compuertas la regla Spiral posee una lógica universal.

A lo largo de la búsqueda se logró encontrar otras compuertas más, estas son: NOR, XOR y XNOR.

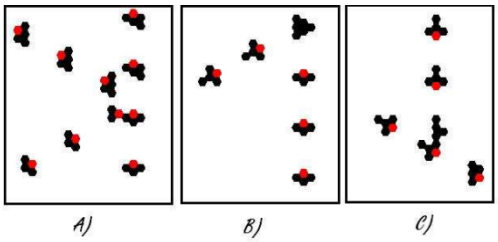

Figura 10: Tipos de colisiones entre gliders.

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 XNOR.

Tabla 4: Tabla de verdad de la compuerta NOT.

A	S
0	1
1	0

Tabla 5: Tablas de verdad para las compuertas AND, OR, NOR, XOR y XNOR.

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

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 11, 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 12, 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 partió 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 13 se puede observar la compuerta OR.

5 Discusión, resultados y trabajo a futuro

Se ha analizado la regla Spiral y las colisiones entre partículas, así como sus reacciones. Gracias a estas construcciones se ha demostrado que es posible implementar computación dentro de la regla Spiral para autómatas celulares hexagonales; 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 serán de ayuda en la búsqueda de construcciones más complejas. Por ejemplo, un objetivo siguiente puede ser la construcción de un medio sumador utilizando la compuerta XOR y AND que se tienen actualmente. Además de utilizarlas para cons-

truir dispositivos más complejos, simular una función computable completa o algún otro sistema activador, inhibidor y refractario; incluso cualquier otro sistema no-lineal con dicha dinámica. Dentro de la regla Spiral existen partículas que aún no han sido utilizadas con fines computacionales, en este sentido queda abierta la posibilidad de encontrar más partículas, o tomar algunas de las ya encontradas para utilizarlas en nuevas construcciones, o alguna construcción derivada de las ya existentes.

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, será necesario utilizar espacios de evoluciones más amplios, lo que conlleva un mayor procesamiento y una visualización menos agradable; es por eso, que se propone como trabajo a futuro utilizar un clúster de visualización. Finalmente, resta pensar en los diferentes tipos de análisis estadísticos que pueden ser mostrados por el simulador; por ejemplo, un análisis de densidad de estados, o de partículas "vivas" dentro del espacio de evoluciones.

Referencias

1. Andrew Adamatzky (Ed.) (2002), *Collision-Based Computing*, Springer.
2. Andrew Adamatzky, Andrew Wuensche and B. De Lacy Costello (2005), "Glider-based computing in reaction diffusion hexagonal cellular automata", *Journal Chaos, Solutions & Fractals*.
3. Andrew Adamatzky and Andrew Wuensche (2006), "Computing in Spiral Rule Reaction-Diffusion Hexagonal Cellular Automaton," *Complex Systems* 16 (4).
4. Andrew Adamatzky and Christof Teuscher, *From Utopian to Genuine Unconventional Computers*, Luniver Press, 2006.
5. Andrew Ilachinski, *Cellular Automata: A Discrete Universe*, World Scientific Press, 2001.
6. Andrew Wuensche (2004), "Self-reproduction by glider collisions; the beehive rule," *International Journal Pollack et. al*, editor, Alife9 Proceedings, MIT Press.
7. Andrew Wuensche (2005), "Glider Dynamics in 3-Value Hexagonal Cellular Automata: The Beehive Rule," *International Journal of Un- conventional Computing*.
8. Elwyn R. Berlekamp, John H. Conway y Richard K. Guy, *"Winning Ways for Your Mathematical Plays"* Volume 4, Second Edition, 927- 961, A K Peters.
9. Genaro J. Martinez, Andrew Adamatzky, Harold V. McIntosh, and Ben De Lacy Costello, "Computation by competing patterns: Life rule B2/S2345678" desde *Automata 2008: Theory and Applications of Cellular Automata* páginas 356–366, Luniver Press, 2008.
10. Genaro J. Martínez, Adriana M. Méndez, y Miriam M. Zambrano,Un subconjunto de autómata celular con comportamiento complejo en dos dimensiones, Escuela Superior de Cómputo, Instituto Politécnico Nacional, México, 2005, http://uncomp.uwe.ac.uk/genaro/Papers/Papers on CA.html.
11. Genaro J. Martinez, Andrew Adamatzky, and Ben De Lacy Costello,(2008) "On logical gates in precipitating medium: cellular automaton model" Physics Letters A 1 (48), 1-5.

12. Genaro J. Martinez, Andrew Adamatzky, and Harold V. McIntosh (2006), "Phenomenology of glider collisions in cellular automaton Rule 54 and sso-ciated logical gates", Chaos, Fractals and Solutions 28, 100-111.
13. Harold V. McIntosh, *One Dimensional Cellular Automata*, Luniver Press, 2009.
14. Howard Gutowitz (1991), *Cellular Automata, Theory and Experi- ment*, The MIT Press.
15. Stephen Wolfram (2002), *A New Kind of Science*, Champaign, Illinois, Wol-fram Media Inc.
16. Tommaso Toffoli and Norman Margolus (1987), *Cellular Automata Machines*, The MIT Press, Cambridge, Massachusetts.
17. Von Neumann, John (1966), *"Theory of self-reproducting automata"*, Urbana and London, University of Illinois.
18. Simulador de la regla Spiral para Windows; http://uncomp.uwe.ac.uk/genaro/Papers/Thesis.html

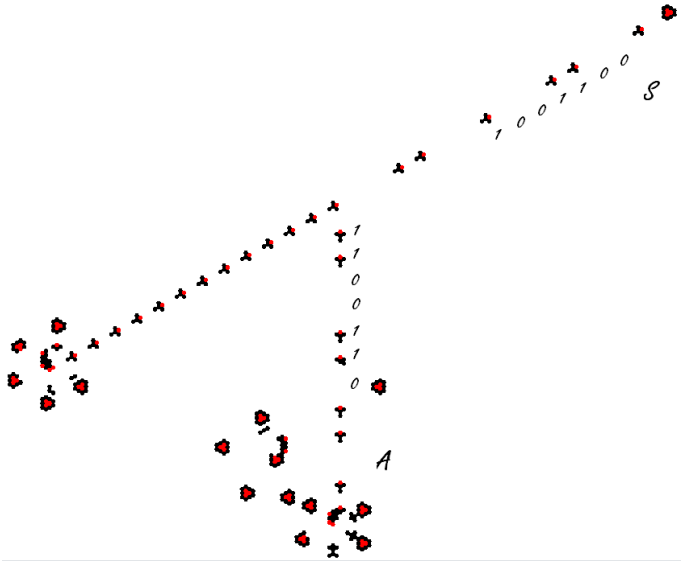

Figura 11: Compuerta NOT en la regla Spiral.

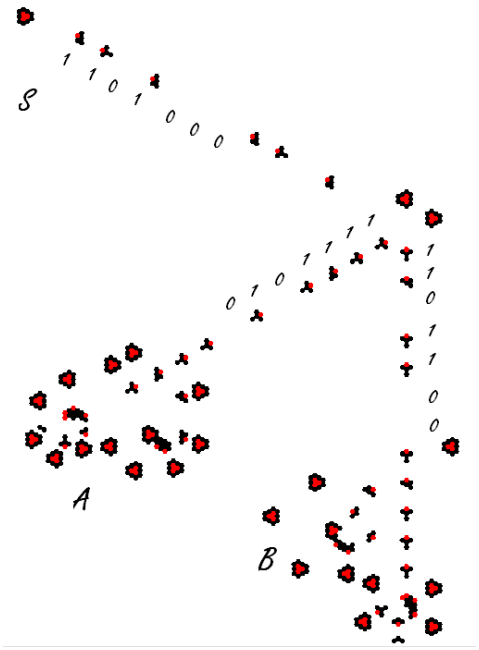

Figura 12: Compuerta AND en la regla Spiral.

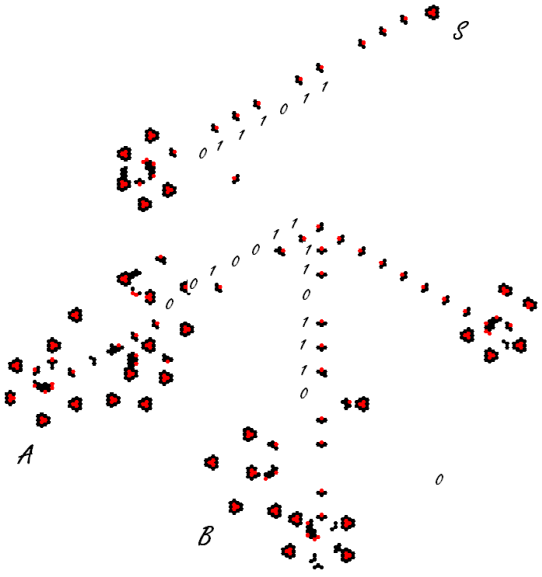

Figura 13: Compuerta OR en la regla Spiral.