Escuela Técnica Superior de Ingeniería Informática
etsinf
DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

# Computabilidad y Complejidad

## Práctica 2: Autómatas Celulares (I - Fundamentos básicos)

# Autómatas Celulares

**Índice**:



## Bibliografía Básica

- Cellular Automata. A Discrete Universe. A. Ilachinski. World Scientific. 2001.
- Cellular Automata. A Parallel Model. M. Delorme, J. Mazoyer (Eds.) Kluwer. 1999.
- Game of Life Cellular Automata. A. Adamatzky (Ed.) Springer. 2010.
- Handbook of Natural Computing (Vol. 1). G. Rozenberg, T. Bäck, J.N. Kok (Eds.) Springer. 2012.

# Autómatas Celulares

## Un poco de historia

John von Neumann

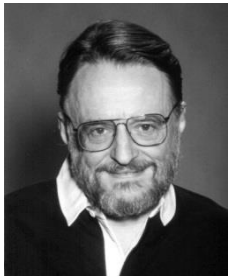
El concepto de autómata celular (AC) fue propuesto a finales de los años 40 y principios de los 50 por John von Neumann y Stanislaw Ulam. En el caso de von Neumann su principal motivación era el diseño y la propuesta de máquinas que emularan a las redes neuronales naturales con capacidad de auto-reproducción.

Stanislaw Ulam

En los años 70 John H. Conway propuso un autómata celular bidimensional cuyo interés superó a la comunidad académica: *El Juego de la Vida*. Hoy en día el Juego de la Vida de Conway se ha tomado como un referente para los estudios de simulación de sistemas dinámicos cuya evolución permite observar fenómenos como la emergencia de patrones complejos y la auto-organización.

Stephen Wolfram

En los años 80 Stephen Wolfram propuso un estudio sistemático de los AC unidimensionales. Su principal aportación en este campo fue la propuesta de una taxonomía que permitía clasificarlos en cuatro tipos según su comportamiento. Además, propuso un método de codificación de las reglas de comportamiento de los AC y, a partir de sus trabajos, se pudo demostrar que algunas de las reglas de comportamiento propuestas permitían la completitud del modelo (es decir, eran equivalentes a la máquina de Turing).



John Conway

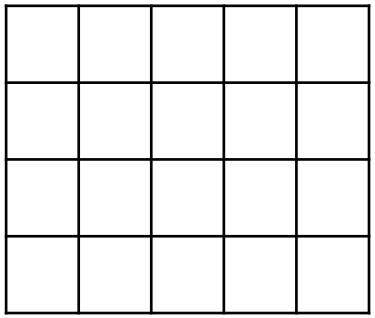
En la actualidad, los AC tiene aplicación en multitud de campos científicos tales como la física, la criptografía, la computación y la biología.

# Autómatas Celulares

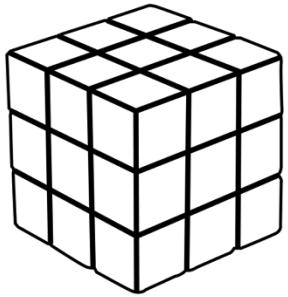
## Definición intuitiva

Un autómata celular lo podemos concebir como un espacio n-dimensional, en principio infinito, compuesto por células ubicadas en ese espacio. Cada célula, en cada momento de tiempo, se encuentra en un estado (de entre un número finito de estados) que queda determinado (a partir de una regla predeterminada) por el estado en el que se encontraban sus "vecinos" en el instante anterior.

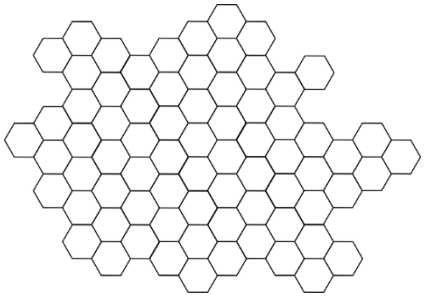
**unidimensional**



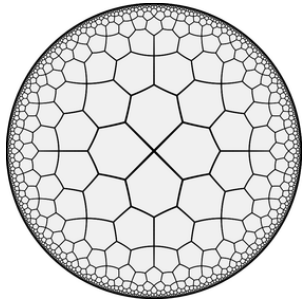
**bidimensional**



**tridimensional**



**bidimensional**
**vecindario hexagonal**



**en espacio hiperbólico**

# Autómatas Celulares

## Definición formal

Un autómata celular d-dimensional A se define mediante la tupla

$$A=(Z^d,S,N,f)$$

donde

**$Z^d$** es el conjunto (infinito) de vectores de dimensión d sobre los números enteros

**S** es un conjunto (finito) de estados

**N** es un subconjunto finito de $Z^d$ denominado el vecindario de A y definido como el conjunto

$$N = \{n_j : n_j = (x_{1j},\ldots, x_{dj}),\ j \in \{1,..,m\}\}$$

**$f : S^{n+1} \rightarrow S$** es la función de transición local (o regla local) de A

# Autómatas Celulares

## Vecindarios

El vecindario de una célula c es el conjunto de células que determinan su evolución. Es un conjunto finito. Consideraremos que cualquier célula forma parte de su vecindario.

**vecindario Moore**

$$NM(z) = \{x : x \in Z^d, d_\infty(z,x) \leq 1\}$$

$$\|z\|_\infty = Max\{|z_i| : i \in \{1,\ldots,d\}\}$$

**vecindario von Neumann**

$$NVN(z) = \{x : x \in Z^d, d_1(z,x) \leq 1\}$$

$$\|z\|_1 = \sum_{i=1}^{d} |z_i|$$

# Autómatas Celulares

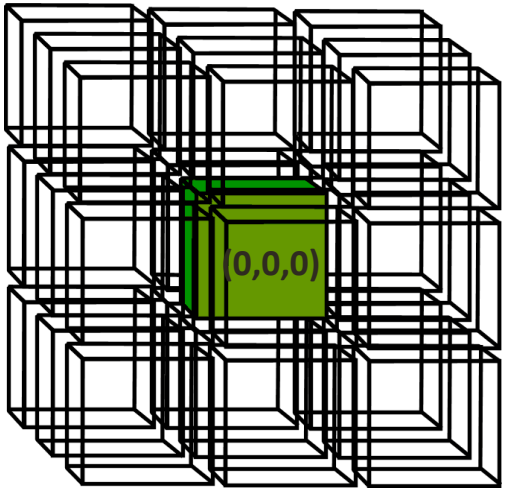
## Vecindarios

| | | | | |
|---|---|---|---|---|
| | (-1) | (0) | (1) | |

vecindario Moore/von Neumann 1d

| | | | | |
|---|---|---|---|---|
| | | | | |
| | (-1,1) | (0,1) | (1,1) | |
| | (-1,0) | (0,0) | (1,0) | |
| | (-1,-1) | (0,-1) | (1,-1) | |
| | | | | |

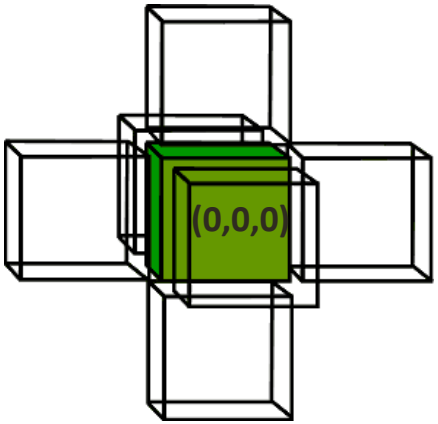
vecindario Moore 2d




vecindario Moore 3d

| | | | | |
|---|---|---|---|---|
| | | | | |
| | (-1,1) | (0,1) | (1,1) | |
| | (-1,0) | (0,0) | (1,0) | |
| | (-1,-1) | (0,-1) | (1,-1) | |
| | | | | |

vecindario von Neumann 2d




vecindario von Neumann 3d

# Autómatas Celulares

## Funciones de transición (reglas locales)

Consideraremos que el conjunto de estados es binario {0,1}.
La función de transición toma en cuenta los estados de una célula y de sus vecinas en el instante de tiempo t y establece el estado de la célula en el instante t+1.

### Caso 1-dimensional

Tomando en cuenta el vecindario Moore/von Neumann, hay que considerar tres células para calcular el estado en el siguiente instante de tiempo.

Consideraremos dos tipos de representación de las reglas:

(a) Números de Wolfram

Las tres células a considerar, forman un número binario de tres dígitos. Existen 8 configuraciones distintas y cada configuración puede dar un valor de 0 ó 1. Las combinaciones posibles de cada configuración originan $2^8$ posibles reglas que se pueden enumerar de la 0 a la 255.

Ejemplo: La regla 54

| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| $2^0$ | $2^1$ | $2^2$ | $2^3$ | $2^4$ | $2^5$ | $2^6$ | $2^7$ |

$2^1 + 2^2 + 2^4 + 2^5 = 54$

# Autómatas Celulares

Funciones de transición (reglas locales)

Caso 1-dimensional

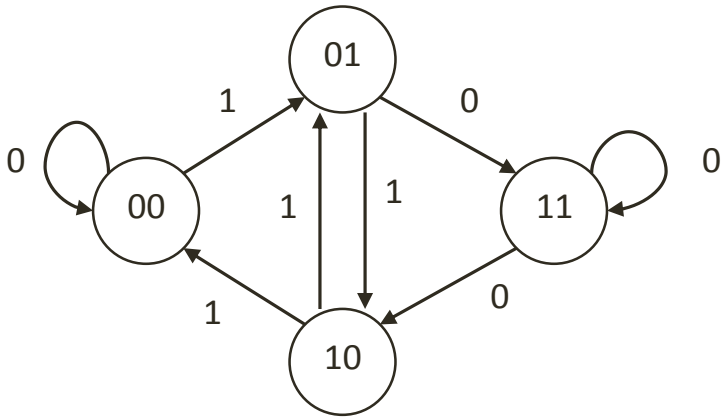
(b) Representacion mediante un grafo de De Bruijn

Un grafo de De Bruijn se define mediante la tupla $(S^n,E)$ donde S es el conjunto de estados y n el número de vecinos de una célula. El conjunto de aristas E se define de forma que a la arista (xy,yz) le corresponde la etiqueta f(xyz) (obsérvese que $y \in S^{n-1}$ mientras que $x,z \in S$)

Ejemplo: La regla 54

000 001 010 011 100 101 110 111

0 1 1 0 1 1 0 0



# Autómatas Celulares

Funciones de transición (reglas locales)

## Caso 2-dimensional

Para el caso 2-dimensional debemos tener en cuenta en primer lugar qué tipo de espacio estamos utilizando (euclídeo, hiperbólico, etc.), qué tipo de rejillas (cuadradas, hexagonales, triangulares, etc.), qué conjunto de estados (por lo general emplearemos un conjunto binario) y qué tipo de vecindario (Von Neumann, Moore, etc.). Por último, se especifica la regla local.

| **Algunos tipos de regla** | **#Reglas (Von Neumann)** | **#Reglas (Moore)** |
|---|---|---|
| General | $2^{32} \cong 4 \times 10^{9}$ | $2^{512} \cong 10^{154}$ |
| Con simetría reflectiva | $2^{24} \cong 2 \times 10^{7}$ | $2^{288} \cong 5 \times 10^{86}$ |
| Con simetría completa | $2^{12} = 4096$ | $2^{102} \cong 5 \times 10^{30}$ |

# Autómatas Celulares

Funciones de transición (reglas locales)

## Caso 2-dimensional

Las reglas totales se pueden codificar de acuerdo a un código predeterminado, de forma similar al caso unidimensional.

Wolfram (1984) propuso una clasificación de los AC unidimensionales de acuerdo a su comportamiento observado en una rejilla bidmiensional:

- Clase I El AC evoluciona a puntos fijos homogéneos
- Clase II El AC evoluciona a configuraciones periódicas
- Clase III El AC evoluciona a patrones caóticos no periódicos
- Clase IV El AC evoluciona a patrones complejos con estructuras locales que se propagan

# Autómatas Celulares

## Condiciones de frontera

Dado que el espacio del conjunto de células es infinito, nos encontramos con la dificultad de establecer en un espacio finito qué sucede con las células que ocupan la frontera. Se han adoptado diversas soluciones, algunas de las cuales pasamos a describir

**Frontera abierta**. Se considera que fuera de la rejilla residen células, todas con un valor fijo.

**Frontera periódica**. Se considera a la rejilla como si sus extremos se tocaran. En una rejilla de dimensión 1, esto puede visualizarse en dos dimensiones como una circunferencia. En dimensión 2, la rejilla podría visualizarse en tres dimensiones como un toroide.

**Frontera reflectora**. Se considera que las células fuera de la rejilla "reflejan" los valores de aquellas de dentro. Así, una célula que estuviera junto al borde de la rejilla y fuera de ella tomaría como valor el de la célula que esté junto al borde de la rejilla y dentro de ella.

**Sin frontera**. Haciendo uso de implementaciones que hagan crecer dinámicamente el uso de memoria de la rejilla, se puede asumir que cada vez que las células deben interactuar con células fuera de la rejilla, ésta se hace más grande para dar cabida a estas interacciones.

# Autómatas Celulares

## Implementación en Mathematica

Las reglas se pueden implementar como una lista de listas, donde cada lista se compone del conjunto de estados de las células que participan en la regla, de acuerdo con la vecindad, y en la última posición se añade el estado que alcanzará la célula tras la aplicación de la regla.

Ejemplo: La regla 54

| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |

```
regla54={{0,0,0,0}, {0,0,1,1}, {0,1,0,1}, {0,1,1,0}, {1,0,0,1}, {1,0,1,1}, {1,1,0,0}, {1,1,1,0}}
```

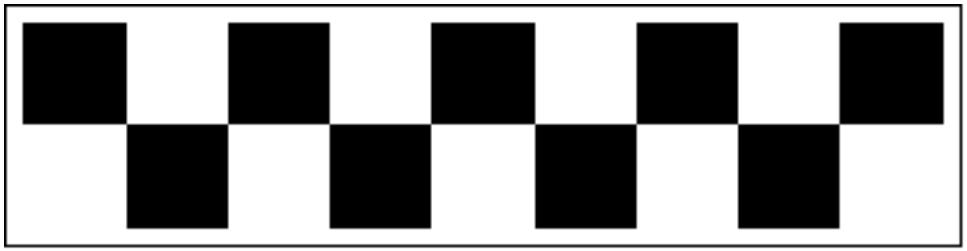
Con la anterior representación de las reglas es sencillo calcular los estados del AC en cada momento de tiempo. Así, si utilizamos la función Cases de *Mathematica* podemos obtener los estados de cada configuración.

`Cases[regla54,{0,1,0,_}]` nos devuelve el valor {{0,1,0,1}}

# Autómatas Celulares

## Implementación en Mathematica

```
ArrayPlot[{{1, 0, 1, 0, 1, 0, 1, 0, 1}, {0, 1, 0, 1, 0, 1, 0, 1, 0}}]
```




ArrayPlot puede tomar como parámetro una lista de listas de 0s y 1s y muestra como salida un gráfico en forma de matriz donde el valor 1 se visualiza en negro y el valor 0 en blanco.

En un autómata unidimensional, cada configuración es una lista de 0s y 1s. Si almacenamos n configuraciones en una lista de listas, las podremos visualizar mediante ArrayPlot situando en la base del gráfico la configuración inicial.

# Autómatas Celulares

## Implementación en Mathematica

ListAnimate[{expr1,expr2,…}]

ListAnimate genera una animación donde el frame i-esimo es la componente i-ésima de la lista que toma como argumento de entrada

Podemos visualizar la evolución del AC si vamos almacenando de forma incremental las listas de sus configuraciones.

# Autómatas Celulares

Actividades propuestas

Diseñe un módulo *Mathematica* que, tomando como entrada los siguientes parámetros, nos proporcione como salida la simulación del AC unidimensional correspondiente

Inicial: Lista con la configuración inicial del AC
Regla: Valor entero que codifica la regla como un número de Wolfram
t: Número de configuraciones a calcular a partir de la configuración inicial

```
AC[Inicial_List,Regla_Integer,t_Integer]:=Module[{  },
      …
]
```

Considere un conjunto de estados binario {0,1} y condición de frontera periódica.

Se recomienda implementar módulos para cada acción significativa de la simulación (por ejemplo, un módulo que convierta el número de regla en una lista de listas, otro módulo que, a partir de una configuración calcule la siguiente, etc.)