18 | Proyecto: Autómatas celulares bidimensionales

Autor original: José Galaviz Casas
El texto se ha pasado en limpio con adaptaciones.

Prerrequisitos

- Arreglos
- Listas y genéricos
- Herencia

Meta

Que el alumno desarrolle una aplicación completa aplicada a un tema real haciendo uso de orientación a objetos y arreglos.

Objetivos

Al finalizar la práctica el alumno será capaz de:

- Crear una animación en `JavaFX` gobernada por una simulación matemática.

Antecedentes

En términos generales un sistema dinámico es un modelo que describe el comportamiento, a lo largo del tiempo, de un sistema en función de sus estados previos. Un caso

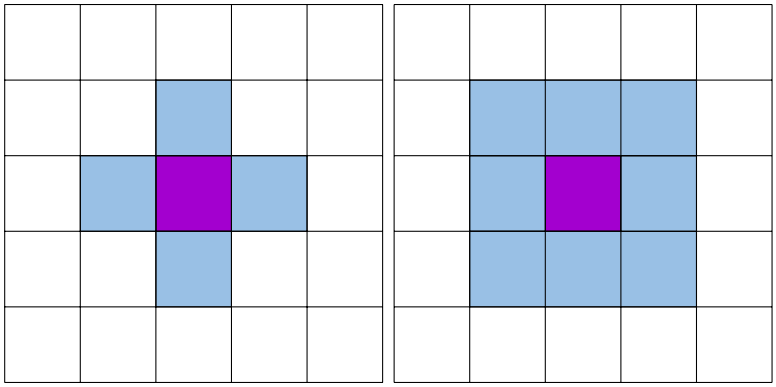

Figura 18.1 Izquierda: vecindad de Von Neumann. Derecha: vecindad de Moore. Los dos tipos de vecindades más usuales en autómatas celulares bidimensionales. El estado de la celda central es determinado, en el siguiente paso temporal, por su valor actual y el de las celdas coloreadas a su alrededor.

particular de un sistema dinámico son los sistemas dinámicos discretos, en lo que el tiempo no es una variable continua sino que avanza a "pasos". En general un sistema dinámico discreto está descrito completamente por la función $f : \mathbb{X} \to \mathbb{X}$ que permite determinar el estado actual del sistema x_t en función de su estado previo x_{t-1}, es decir $x_t = f(x_{t-1})$, donde x_{t-1} y x_t son dos estados de un conjunto de posibles estados del sistema $\mathcal{X}$, en notación: $x_{t-1}, x_t \in \mathcal{X}$.

Un autómata celular es un sistema dinámico discreto, en el que de hecho tanto el espacio como el tiempo y los posibles estados del sistema son discretos. El espacio se divide en *celdas* regulares de igual tamaño y forma, lo que constituye una *malla*. En cualquier instante del tiempo t cada celda de la malla posee un valor determinado de un conjunto de posibles valores, lo que constituye un *estado*. Para determinar el estado de la celda en el siguiente paso temporal (en $t + 1$), se aplica una *regla local*, una función que determina, con base en el valor de la celda misma y de algunas de sus celdas vecinas al tiempo t, el nuevo valor del estado de la celda.

Por supuesto, dado que el estado de una celda está en función de su estado previo y del estado previo de sus celdas vecinas conviene especificar quienes son sus celdas vecinas. Hay una infinidad de opciones, pero las más usuales en los autómatas celulares bidimensionales son dos: la vecindad de Von Neumann y la de Moore. La vecindad de Von Neumann de una celda está constituida por aquellas celdas que se encuentran arriba, abajo, a la derecha y a la izquierda de la celda en cuestión. La de Moore considera además las que están arriba a la derecha, arriba a la izquierda, abajo a la derecha y abajo a la izquierda. En la Figura 18.1 se ilustran los dos tipos de vecindad.

Los autómatas celulares han recibido mucha atención porque han resultado ser modelos muy adecuados para describir el comportamiento de algunos fenómenos naturales muy complejos, con unas cuantas reglas locales muy simples. Desde el punto de vista teórico resultan ser también muy interesantes dado que son un ejemplo de

sistemas con interacciones no lineales: el conocer cómo operan locamente no permite muchas veces conocer el commportamiento general del autómata a largo plazo: reglas simples que engendran comportamientos complejos, algo que suele calificarse como *propiedades emergentes*. Los autómatas celulares resultan estar muchas veces emparentados con los procesos caóticos o en la frontera del caos, con fenómenos de criticalidad auto-organizada y en general con lo que se denominan sistemas complejos.

JavaFX

En este proyecto tendrás que realizar el programa completo. Para que tengas una idea de cómo puedes empezar, se te entrega un pequeño demo.

Actividad 18.1

Si tienes `Java` 10, puedes usar `ant` para ejecutarlo. Si tienes `Java` 11, usa los comandos siguientes:

```
$ export PATH_TO_FX=<where_you_placed_it>/javafx-sdk-11.0.1/
    ↪ lib
$ javac --module-path $PATH_TO_FX --add-modules=javafx.
    ↪ controls -d ./classes --source-path ./src src/automatas/
    ↪ Demo.java
$ java --module-path $PATH_TO_FX --add-modules=javafx.controls
    ↪  -classpath classes automatas.Demo
```

Ojo, `ant clean` aún te servirá para entregar tu código sin archivos compilados.

EJERCICIOS

Hay varios modelos basados en autómatas celulares que podemos programar, todos requieren la elaboración previa de las mismas clases.

Opción 1: Actividad sísmica

Se utilizan vecindades de Von Neumann. Cada celda puede tomar valores en el conjunto $\mathbb{X} = \{0, ..., M\}$. A M se le llama *valor umbral*.

1. En $t = 0$ la malla se inicializa con valores aleatorios de entre los valores válidos inferiores a M.

2. En cada paso temporal t se elige aleatoriamente un sitio de la malla y su valor se incrementa en una unidad.

   Otra opción es incrementar los valores de todas las celdas, esta última opción evoluciona más rápido.

3. Si alguna celda alcanzó el valor umbral en $t-1$, para t decrementa su valor en cuatro unidades.

4. Si un sitio tiene vecinos en valor umbral en $t-1$, para t incrementa su propio valor en una unidad por cada vecino en valor umbral (sin rebasar el valor umbral).

Los sitios (celdas) de la malla con valor umbral modelan aquellos lugares de una falla tectónica que han acumulado mucha energía y ya no pueden más. Cuando un sitio de la falla alcanza el umbral lo único que le queda por hacer es romperse liberando energía que deben absorber, si pueden, sus sitios vecinos. Cuando muchos sitios se rompen al mismo tiempo se genera un sismo, la magnitud de éste está en función del número de celdas que se rompen.

Opción 2: Un modelo de propagación de epidemias

Se utilizan también vecindades de Von Neumann. Los valores de las celdas están en el conjunto $\mathbb{X} = \{0, ..., a+g\}$, dividido en tres subconjuntos: $A = \{0\}$, $B = \{1, ..., a\}$ y $C = \{a+1, ..., a+g\}$, el conjunto A es *susceptible*, el B es *infeccioso* y el C es *inmune*.

1. Un individuo susceptible se vuelve infeccioso si al menos un vecino es infeccioso.
2. Un individuo inmune después de g pasos se vuelve susceptible.
3. Un individuo que ha sido infeccioso o inmune por menos de g pasos sólo incrementa su valor.
4. Un individuo infeccioso después de a pasos se vuelve inmune.
5. Un individuo susceptible sin vecinos infecciosos permanece susceptible.

Opción 3: Un modelo de incendios forestales

El modelo que se describe a continuación se suele llamar incendio forestal estocástico.

Los valores de las celdas son 0 (*vacío*), 1 (*árbol*) y 2 (*árbol incendiado*), se utilizan vecindades de Von Neumann. Este modelo es el más complicado porque es estocástico,

esto es, se utilizan tres diferentes parámetros probabilísticos que controlan el proceso: p probabilidad de que cresta un árbol $(0 \to 1)$, f probabilidad de que un árbol se incendie espontáneamente $(1 \to 2)$ y g probabilidad de que un árbol sea inmune al fuego. Las reglas son las siguientes:

1. Un sitio 0 se vuelve 1 con probabilidad p.
2. Un sitio 1 se vuelve 2 con probabilidad $(1 - g)$ si al menos un vecino está en estado 2.
3. Un sitio 1 se vuelve 2 con probabilidad $f * (1 - g)$ si no hay vecinos en estado 2.
4. Un sitio 2 se vuelve 0 al siguiente paso temporal.

Generalidades

El proyecto debe implementar, al menos, dos de los tres modelos planteados aquí, se dará máximo un punto extra a quien implemente correctamente los tres.

1. Hacer una clase abstracta de autómatas celulares que implemente las cosas que hay que hacer en todos los casos y que permita fijar la firma de algunos métodos; por ejemplo, uno que actualice el estado de una celda dado su estado actual y el de sus vecinos.
2. Debe presentarse gráficamente la evolución temporal del autómata. Las celdas en diferente estado deben distinguirse por su color. Los colores se dejan a elección del programador.
3. Cuando la simulación termine debe también mostrar una gráfica del número de celdas en estado crítico contra el número de pasos temporales (i.e. abscisas=paso temporal, ordenadas=número de celdas en estado crítico en ese paso temporal). Los estados criticos son:
   - Para el modelo de sismos: las celdas con energía en umbral.
   - Para el modelo de epidemias: las celdas infecciosas.
   - Para el modelo de incendio forestal: las celdas incendiadas.
4. Implementar un menú (puede ser en la consola), que pida al usuario qué modelo desea correr y con qué parámetros, antes de lanzar la simulación.

Entrada

- El número de pasos temporales que el usuario desea dejar evolucionar el sistema.

- El tamaño de la malla.
- Parámetros de control para el autómata (umbrales, valores de cambio de estado, etc.)

Salida

- El desplegado gráfico de la evolución del autómata.
- El desplegado de la gráfica de puntos críticos vs. tiempo.

REFERENCIAS

Bak, P. y K. Chen (1991). «Self-Organized Criticality». En: *Scientific American* 264. Una de las fuentes primigenias del tema de la criticalidad auto-organizada. En la hemeroteca de la Facultad está la colección de *Scientific American* desde hace unos 50 años a la fecha (nota del 2003)., págs. 46-53.

Gaylord, Richar J. y Kazume Nishidate (1996). «Cellular Automata Simulations with Mathematica». En: *Modelling Nature*. La colocación en la biblioteca es QA267.5,C45,G39. Incluye muchos ejemplos más de autómatas celulares como modelos de fenómenos naturales. TELOS-Springer Verlag.

Nakanishi, Hiizu (jun. de 1990). «Cellular-automaton model of earthquakes with deterministic dynamics». En: *Physical Review A* 41.12. Una generalización del modelo presentado aquí. Además se muestra que los resultados arrojados por éste son consistentes con la ley de Gutenberg-Richter, el modelo continuo más robusto en terremotos., págs. 7086-7089.

Schönfisch, Birgitt (1995). «Propagation of Fronts in Cellular Automata». En: *Physica D: Nonlinear Phenomena* 80. Incluye un par de modelos de propagación de epidemias., págs. 433-450.

Bibliografía

Aho, Alfred V., Ravi Sethi y Jeffrey D. Ullman (1998). *Compiladores. Principios, técnicas y herramientas.* Trad. por Pedro Flores Suárez y Pere Botella i López. Pearson, Addison Wesley Longman.

Fish (sep. de 2005). *Float*. Inglés. Apple. URL: http://ridiculousfish.com/blog/posts/float.html.

I/O Streams (2016). URL: https://docs.oracle.com/javase/tutorial/essential/io/streams.html.

López Gaona, Amparo (2012). *Introducción al desarrollo de programas con Java. Prácticas.* Las prensas de Ciencias.

Viso, Elisa y Canek Peláez (2007). *Introducción a las Ciencias de la Computación con Java*. Las prensas de ciencias.

– (jun. de 2012). *Introducción a las Ciencias de la Computación con Java*. 2a. Temas de computación. Las prensas de ciencias.