Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

Grado en Ingeniería Informática
Algoritmos y Estructuras de Datos Avanzadas

Curso 2023-2024

Práctica 1: Autómata celular elemental

---

1. Objetivo

En esta práctica se implementan tipos de datos definidos por el usuario en lenguaje C++.

2. Entrega

Se realizará en dos sesiones de laboratorio en las siguientes fechas:

Sesión tutorada: del 5 al 9 de febrero
Sesión de entrega: del 19 al 23 de febrero

3. Enunciado

Un autómata celular [1] es un modelo matemático y computacional para un sistema dinámico que evoluciona en pasos discretos. Es adecuado para modelar sistemas naturales que puedan ser descritos como una colección masiva de objetos simples que interactúan localmente.

Los elementos básicos para describir un autómata celular [2] son:

- Un **espacio celular**, consiste en un retículo [4] que se define sobre un espacio infinito regular n-dimensional. Puede ser una línea, espacio unidimensional; un plano, espacio bidimensional, etc. Cada división homogénea del espacio se denomina celda o *célula*.
- Un **conjunto de estados**, también denominado *alfabeto*. Cada célula toma un *estado* de este conjunto finito. Los estados se pueden representar mediante valores o colores. Se denomina *configuración inicial* a la asignación de estado para cada una de las células en el momento de iniciar la evolución.
- Una **vecindad** define al conjunto finito de células consideradas adyacentes a una dada, así como la posición relativa de cada célula vecina respecto a ella.
- Una **función de transición local**, define cómo debe cambiar el estado de cada célula dependiendo de su estado anterior y de los estados anteriores de las células de su vecindad. En cada paso discreto de tiempo, que denominamos *generación*, se aplica la función de transición local a cada una de las células del espacio. Por tanto, es la regla de evolución que determina el comportamiento del autómata celular.
- **Condiciones de frontera**, en las implementaciones prácticas no es posible definir un retículo infinito, sino que se utilizan retículos finitos. Por tanto habrá células ubicadas en el borde del retículo, que requieren unas consideraciones específicas para manejar su vecindad. Se denominan condiciones de frontera a las consideraciones que permiten manejar las células en los bordes del retículo. Algunas son:
    - Frontera abierta. Se considera que todas las células fuera del retículo tiene un valor fijo.
    - Frontera periódica. Se considera que los extremos del retículo son adyacentes.
    - Frontera reflectora. Se considera que las células fuera del retículo reflejan a las células adyacentes al borde dentro del retículo.

Un autómata celular elemental [3], el autómata celular no trivial más simple, es uno de los modelos más sencillos de computación que se define a partir de los siguientes elementos:

- El espacio celular consiste en un retículo unidimensional de células. Aunque en teoría esta retícula se extienda de forma infinita, en la práctica es necesario definir unos límites. Para tratar las células ubicadas en los límites del retículo se definen las denominadas *condiciones de frontera*.

- El alfabeto contiene dos estados posibles, representados por los valores «0» y «1».

- La vecindad está constituida, para cada célula, por ella misma C, la célula adyacente por la izquierda L y la célula adyacente por la derecha R. Si tomamos el valor binario del estado de las tres células de una vecindad, LCR, tenemos 2^3=8 configuraciones posibles.

- La función de transición calcula el valor del estado para la célula $C^{(G+1)}$ en la siguiente generación (G+1), a partir de la configuración de la vecindad $L^{(G)}C^{(G)}R^{(G)}$ en la generación actual (G). Por tanto, existen 2^8=256 posibles reglas para definir el estado de una célula en la generación siguiente. Stephen Wolfram [5] propuso un esquema, conocido como el código Wolfram, para asignar a cada regla un número de 0 a 255. Este número corresponde a la representación binaria de las ocho configuraciones posibles de estados en una vecindad. Cada una de estas 256 reglas define un autómata celular elemental diferente.

El autómata celular definido por la regla 110, indicada en la siguiente tabla, es equivalente a la Máquina de Turing Universal:

111	110	101	100	011	010	001	000	$L^{(G)}C^{(G)}R^{(G)}$
0	1	1	0	1	1	1	0	$C^{(G+1)}=(C^{(G)}+R^{(G)}+C^{(G)}*R^{(G)}+L^{(G)}*C^{(G)}*R^{(G)})\%2$

Regla 110_d = 01101110_b

4. Notas de implementación

El objetivo de esta práctica es implementar las clases, tipos de datos definidos por el usuario, que representan los componentes de un autómata celular elemental. Para cada clase se definen sus responsabilidades, entendidas como la información que debe almacenar y las operaciones que debe realizar. Se propone las siguientes clases:

1. La célula, `Cell`, es responsable de encapsular su estado binario y su posición dentro del retículo unidimensional que representa al espacio celular. También es responsable de conocer su vecindad y su función de transición.

    a. El constructor de la célula recibe como parámetro su posición dentro del retículo, y de forma opcional su estado en la configuración inicial. Por defecto, la célula se crea con estado «0».

    ```
    Cell::Cell(const Position&, const State&);
    ```

    b. Se dispone de un método para acceder al estado de la célula.

    ```
    State Cell::getState() const;
    ```

c. Aunque el estado de una célula evoluciona sin necesidad de acceso exterior, se implementa un método modificador para poder establecer la configuración inicial.

```
State Cell::setState(State);
```

d. La célula dispone de un método que aplica la función de transición para obtener su estado en la siguiente generación. Para ello, además de su posición, la célula necesita conocer el estado de las células de su vecindad en la generación actual. Este método recibe el retículo por parámetro y calcula el siguiente estado sin evolucionar.

```
int Cell::nextState(const Lattice&);
```

e. Después de que cada célula haya calculado su siguiente estado, la evolución del autómata celular consiste en hacer que cada célula actualice su valor de estado.

```
void Cell::updateState();
```

f. La célula es responsable de su visualización en pantalla, que realiza mediante la sobrecarga del operador de inserción en flujo. Para facilitar la visualización se utilizará el carácter `'X'` para representar el valor de estado «1», y el carácter espacio `' '` para el valor de estado «0».

```
ostream& operator<<(ostream&, const Cell&);
```

2. El retículo, `Lattice`, contiene un array de `N` células. Este objeto es responsable de crear y almacenar las células que representan el espacio celular. También es responsable de controlar la evolución y llevar la cuenta de las generaciones.

   a. El constructor del retículo crea las células en memoria dinámica.

   b. Método para cargar la configuración inicial del autómata celular. Esto es, inicializa el estado de cada célula en la generación G=0. Por defecto, la configuración inicial consiste en colocar el valor de estado «0» en todas las células, salvo en la célula central del retículo que tendrá el valor de estado «1».

   c. El retículo dispone de un método para dar acceso de lectura a las células.

```
const Cell& Lattice::getCell(const Position&) const;
```

   d. El retículo es responsable de controlar la evolución del autómata, asegurando que todas las células calculan su estado en la generación G+1 a partir de los valores de estado en la generación G. Esta operación la realiza el siguiente método:

```
void Lattice::nextGeneration();
```

      Para ello se realizan dos recorridos sobre las células del retículo:

      i. En el primer recorrido cada célula accede a su vecindad y aplica la función de transición para calcular su estado siguiente.

      ii. En el segundo recorrido cada célula actualiza su estado.

   e. Para la visualización del retículo se sobrecarga el operador de inserción en flujo. Un retículo se muestra como una línea en pantalla.

```
ostream& operator<<(ostream&, const Lattice&);
```

f. El retículo debe manejar las condiciones de frontera, que tal y como se ha indicado previamente, se corresponden con el tratamiento de las células que se encuentren en los bordes del mismo. En esta práctica se implementarán las siguientes técnicas:

   i. Frontera abierta. Se añade una célula en cada borde del retículo con un valor de estado fijo e inalterable. Una frontera se dice **fría** si las células fuera de la frontera tiene estado «0», y **caliente** si tiene estado «1».

   ii. Frontera periódica. En un array de N posiciones, se considerarán adyacentes las celdas ubicadas en la posición cero y N-1.

3. El programa principal tiene el siguiente comportamiento.

   a. Recibe por línea de comandos los siguientes argumentos:

      `-size <n>`, `n` es el tamaño del retículo. Número de células.

      `-border <b [v]>`, `b=open, v=[0|1]`. Frontera abierta, fría o caliente. `b=periodic`

      `[-init <file>]`, (opcional) `file` es un nombre del fichero que contiene un array de estados con la configuración inicial del autómata celular. Si no se especifica se utilizará la configuración inicial por defecto, esto es, un «1» en la célula central del retículo.

   b. Crea los componentes del autómata celular a partir de los argumentos recibidos y simula la evolución del autómata celular, mostrando por pantalla la configuración de estados en cada generación. La simulación se puede detener en cualquier momento pulsando un carácter elegido como fin de ejecución.

Durante las sesiones de laboratorio se podrán solicitar cambios y/o ampliaciones en la especificación de este enunciado.

5. Referencias

[1] Wikipedia: Autómata celular
https://es.m.wikipedia.org/wiki/Autómata_celular

[2] Universidad de Sevilla: Blog del Prof. Fernando Sancho Caparrini
https://www.cs.us.es/~fsancho/Blog/posts/Automatas_Celulares.md.html

[3] Wikipedia: Autómata celular elemental
https://es.wikipedia.org/wiki/Autómata_celular_elemental

[4] Wikipedia: Definición matemática de retículo.

https://es.wikipedia.org/wiki/Retículo_(matemáticas)

[5] Wikipedia: Stephen Wolfram
https://es.wikipedia.org/wiki/Stephen_Wolfram