TP 7 : Parity rule and *Game of life*

Cours de modélisation numérique

21 April 2023

Introduction

The goal of this series is to implement a cellular automaton evolving according to the *parity rule* as well as the *Game of Life*. In the case of the first one, we will try to update some more theoretical properties.

The parity rule

The parity rule, introduced by E. Fredkin in the early 1970s, is a dynamic which, despite its simplicity, offers a very rich behavior. With this rule, cells can only have two states : 0 or 1. The state of a cell at time $t+1$ is defined by the state of its four neighbors at time t according to the following function :

$$\psi_{t+1}(i,j) = \psi_t(i-1,j) \oplus \psi_t(i+1,j) \oplus \psi_t(i,j-1) \oplus \psi_t(i,j+1)$$

where the operator $\oplus$ represents the sum modulo 2. In the case of this rule the neighborhood considered is the Von Neumann neighborhood (fig. 1). We will consider here periodic boundary conditions, that is the grid can be considered as a torus.

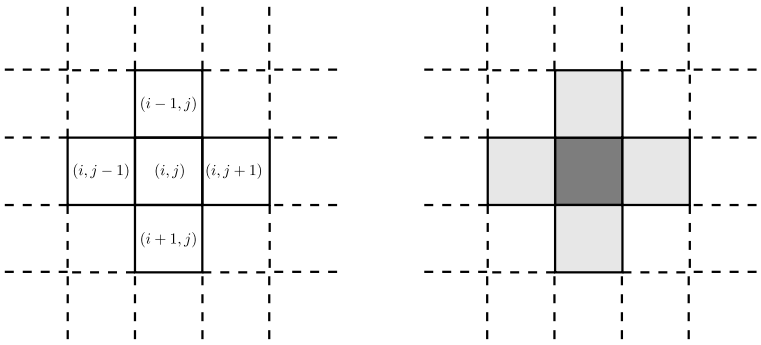

FIGURE 1 – *Illustration of the Von Neumann neighborhood used for the parity rule*

The Game of Life

The Game of Life is a cellular automaton proposed by John H. Conway in 1970, which owes its undeniable popularity to its supposedly realistic character. As in the parity rule, the cells can only take binary values 0 (dead) or 1 (alive), but contrary to this one we work here with the Moore neighborhood, i.e. each cell interacts with its eight neighbors (figure 2).
The dynamic is defined as follows :

— a living cell with less than two living neighbors dies
— a living cell with two or three living neighbors lives
— a living cell with more than three living neighbors dies
— a dead cell with exactly three living neighbors lives

This rule is applied in a synchronous way to all the cells of the automation. As before, the topology is periodic.

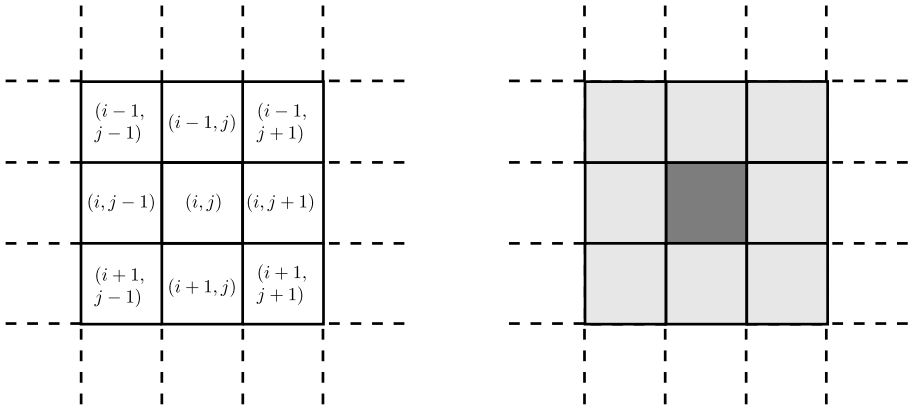

FIGURE 2 – *Illustration of the Moore neighborhood used for the Game of Life*

Work to do

Implementation

A start of implementation for each of these two automata is available on Moodle. You are asked to complete them, taking into account the following constraints :

— The automaton must have the form of a square matrix. You will need to use the structures of `numpy`. As in series 1, except for the time loop, avoid using `for` loops to access the state of a cell's neighbors.

— You will need to use a *lookup table* (LUT). A LUT is a data structure stored in memory, used to replace a calculation by a simpler lookup operation. In the case of this series the calculation to be replaced is that of the state of a cell at time t according to its neighborhood at time $t-1$. The simplest way is to use a table where each index corresponds to a possible state of the neighborhood. So you have to find a way to encode the different states into an integer which will be used as an index to find the value corresponding to state in an array.

— Your program must take as input the name of a file describing the initial state of your automation [1], and produce as output a file of the same format representing the state of the automation at time t_{max}. The number of iterations t_{max} must be left to the user.

Questions about the Parity rule

Evolve the parity rule for a number of iterations being a power of 2 ($2^k, k = 1, 2, 3, \ldots$). If your implementation is correct, you should observe that the initial pattern is translated four times by 2^k sites, once in each direction. Demonstrate mathematically that an initial pattern in a cell is duplicated four times (up, down, left, right) after 2 iterations.

Then evolve an automaton with the parity rule whose length L of one side is a power of 2, for a number of iterations equal to $L/2$. Describe what you observe and try to explain it using the answer to the previous question.

With these results, can you propose a clever way to know the state of a CA evolving according to this rule at time t without having to compute all the intermediate iterations ?

Questions about the Game of Life

After completing the file `jdv_template.py`, test the code for different configuration files, located in the folder `examples`.

1. The format of this file must be the following :
— each line of the file corresponds to a row of the matrix representing the automation
— each element of the matrix is separated by a space
— the elements of the matrix take the value 1 or 0