MCS 320 Project Three: Cellular Automata

The goal of the project is to use MATLAB or Octave to study cellular automata. A cellular automaton is a discrete model of bacterial growth. Imagine a grid of cells. Each cell can contain exactly one instance of an organism. The content of the cell is one or zero, depending whether there is life in the cell or not. In MATLAB or Octave, we represent the grid by a 0/1 matrix and visualize it via the command `spy`:

\[ \text{spy(round(rand(30,30)))} \]

To simulate bacterial growth we use three ingredients:

1. visualization of the state matrix
   The command `spy` takes on input a matrix and makes a picture marking every nonzero element in the matrix with a dot. Nothing is marked for zero elements. For example: `spy(round(rand(100,100)))` will show an image of a random 0/1 matrix. Put this example command in a loop and apply `sleep(1)` to pause for a second, and we have a simple animation.

2. computation of the 9 bits at the \((i,j)\)-th cell
   If the state of one cell depends only on its eight closest neighbors, we store the state of one cell and its neighbors in a sequence of 9 bits. The state of the \((i,j)\)-th element are its eight neighbors and the value of the \((i,j)\)-th element itself. With the \((i,j)\)-th element at its center, enumerate the states of the 9 elements lexicographically, i.e.: values at the positions \((i-1,j-1)\), \((i-1,j)\), \((i-1,j+1)\), \((i,j-1)\), \((i,j)\), \((i,j+1)\), \((i+1,j-1)\), \((i+1,j)\), \((i+1,j+1)\) give 9 bits.

3. use the 9 bits in the application of the rule
   With the 9 bit sequence, we can determine the transition from one state to the next by a a vector of 512 bits. Observe that 512 equals \(2^9\). The value of these 9 bits determines a number, used as the index in the 512-bit vector. This 512-bit vector determines the rule. To generate a random rule, do `round(rand(1,512))`. The command `bin2dec` turns a bit string into a decimal number. For example, `bin2dec('100110100')` returns the value 308. The value of the \((i,j)\)-th element is then determined by the value of the rule vector of position 308. We apply the rule to all inner cells, i.e.: elements which are not on the boundary, so the boundary elements remain unchanged.
Assignment One: build a simulation

A couple of short scripts will simulate a cellular automaton. The input parameters are the initial 0/1 matrix, a vector of 512 bits to define the rule, and the number of stages in the simulation. As the simulation runs, we see the patterns displayed in a separate window. When the simulation ends we get a vector with the number of living cells for every stage.

**Assignment One.** Write the script that starts with the line `function e = showca(A,r,n)`. The input arguments are (1) `A` the initial 0/1 matrix; (2) `r` the rule; and (3) `n` the number of stages. The vector `e` on return has range `1:n+1` with the number of ones at every stage. To implement `showca` it is recommended to define other helper functions to compute the 9 bits index to the rule and then to apply the rule to every inner cell in the grid.

Assignment Two: experiment with the model

A random rule is defined by a random vector of 512 bits long. What we call a half rule has 256 ones in either the upper or lower half of the vector and elsewhere zero. Starting at an initial random 0/1 matrix, for a random rule we expect the number of living cells to remain stationary and in a random pattern. For special rules we expect special patterns.

**Assignment Two.** Do several experiments with random and half rules, starting each time at different initial random matrices. Each run should have at least as many stages as the dimension of the matrix. Report for each experiment the final number of living cells. Describe the evolution of the patterns you see plotted. Explain the observed patterns.

Assignment Three: Conway’s game of life

Conway’s game of life is one of the $2^{256}$ cellular automata we can simulate. The rules are
1. An empty cell becomes alive when it has three neighbors.
2. A living cell can either die or survive, as follows:
   2.1 die by loneliness, if the cell has one or no neighbors;
   2.2 die by overpopulation, if the cell has four or more neighbors;
   2.3 survive, if the cell has two or three neighbors.

**Assignment Three.** What is the rule vector for Conway’s game of life?

The deadline is Wednesday 30 April 2008 at 10AM

Bring *your* solution to the project to class. *The your* is emphasized to stress that your solution is the result of an *individual* effort. Collaborations are *not* permitted.

The solution to this project consists in two parts:
1. A print out of all the `.m` files that you bring to class.
2. Email your scripts as an attachment to me so I can verify your runs.

If you have questions or difficulties, feel free to come to my office for help.