Distributed cellular automata and dependencies

Julien Herrmann, Mohamed Sid-Lakhdar
julien.herrmann@ens-lyon.fr, mohamed.sid_lakhdar@ens-lyon.fr

Disclaimer

Use the C programming language. Using any other language including C++ is likely to halve your mark.

1 One-dimensional cellular automata

Consider a one-dimensional cellular automaton of radius 1 with two states in $\{0,1\} = \mathbb{B}$. It can be characterized by a function $f : \mathbb{B} \times \mathbb{B} \times \mathbb{B} \to \mathbb{B}$ or as a sequence of 8 bits where the first is $f(1,1,1)$, the second is $f(1,1,0)$, etc. For example the automaton below can be written $(0,1,1,1,1,1,0,0)$ which is 124 in binary. This automaton is thus called the Rule 124 cellular automaton. The rule is applied on an unbounded array of cells (a_i) to get a new array (b_i) such that $b_i = f(a_{i-1}, a_i, a_{i+1})$. Getting (b_i) from (a_i) is called a step. Also, always suppose that f is death-preserving, i.e. $f(0,0,0) = 0$.

x	y	z	$f(x,y,z)$
1	1	1	0
1	1	0	1
1	0	1	1
1	0	0	1

x	y	z	$f(x,y,z)$
0	1	1	1
0	1	0	1
0	0	1	0
0	0	0	0

Question 1. Write a sequential program running a cellular automaton a certain number of steps on a given array, such that:

```
./run-ca-1d-cycle rule nbsteps input
```

will run `nbsteps` steps of the cellular automaton corresponding to `rule` (e.g. `rule` can be 124) on the input `input` in this format: `00100010110`. At the end the program shall print the resulting array. Considering the array is of constant size and is cyclic, i.e. the new value of the first (and the last cell) depends on the old value of the last (and first cell, respectively). You do not need to use MPI for this question.

Now consider that the array is not cyclic anymore and consider instead an infinite number of cells filled with zeroes except for some values some input `input` where `input` is small enough to fit in the memory of one process. However the size of the non-zero part of the array will change over time, and may eventually exceeds the available memory in one process. You know in advance the number of iterations n your program will have to make.

Question 2. Write a parallel algorithm using MPI that grossly bounds the maximum size of the non-zero part knowing n, then allocates the corresponding memory all over the processes and finally run the cellular automaton steps on it. The program shall then output the number of non-zero cells. Consider that the topology of your network is a line.

```
mpirun -np 10 ./run-ca-1d-unbounded 124 200000 10001011
```

For example if $n = 2$ and `input=1001` you need only 8 memory cells:

```
0001001000
00??????00
0????????0
```

2 Game of life

Consider Conway's game of life on a square torus.

Question 3. How would you distribute this 2D celullar automaton on a network with a 2D toric square grid topology? Write an algorithm executing one step of Conway's game of life on such a distributed square grid. Explain carefully on which processes is distributed each cell, and where will happen the computation of its next state. (Drawings are welcome)

Question 4. Implement this algorithm in `life.c` using MPI. The program will have several arguments: (1) the size of the square matrix, (2) the number of execution steps to be applied, (3) a floating point number d such that $0 \leq d \leq 1$. Randomly fill the grid such that the expected density of non-zero cells is d. Initialization should be a simple random filling of the grid with a seed depending on the rank of the process. The program should output the density of non-zero cells after the indicated number of steps.

3 Causality violations

To simplify questions and answers, in the next two questions (5 and 6) consider that the new state of a cell depends only on the cells above, below, on the left, on the right (respectively N, S, W, E) and not on all the other neighbors (NW, NE, SW, SE, and the cell itself). Questions 5 and 6 are *not* about an adaptation of Conway's game of life – consider the automaton has arbitrary rules.

In Conway's game of life the new state of every cell depends on its old state and the old states of all its neighbors. Consider the case where the new state of every cell depends on the old state of some of its neighbors but also on the **updated** state of the other neighbors. Note that this is not called a cellular automaton anymore, because in one step a cell can depend on cells that are very far away.

Consider a plain 2D grid without last/first rows or last/first columns neighborhood. If a cell depends on the updated value of N and S, then they will be a circularity: a cell will depend on the cell below that will depend on the first one, so this is not well-defined.

Question 5. We write $\mathsf{DEP} \subseteq \{N, S, W, E\}$ the set of the directions of these dependencies. On which conditions on DEP the automaton is still well-defined? Same question in a grid for which last and first rows *are* neighbors? What about when last and first columns are, too?

Now in the case where this is not well-defined, we now impose the order of computation so that we now which cell has to be updated first: the first line should be computed first, from left to right, then the second line is computed the same way, etc. This is a sequential way to uniquely define the next step.

Question 6. On which conditions on DEP is it possible to find an efficient parallel algorithm on a 2D grid (toric) topology network that gives exactly the same result as the sequential ordering of the computations? What if the last and first rows are neighbors? What if the last and first columns are neighbors? What if both are? Same questions for a ring topology.

Question 7. Implement in `deplife.c` Conway's game of life with dependencies $\mathsf{DEP} = \{NW, N, NE\}$ i.e. where instead of considering the row above to update a cell, we consider the updated row above. Use MPI and consider that the topology of your network is a 2D-grid. This time consider the neighbors outside the grid are dead. Input/output should be done as in Question 4.

Question 8. Can you apply a space-time transformation on the previous modified automaton so that you get back a celullar automaton without dependencies?

Question 9. Implement in `xlife.c` the game of life – without dependencies – such that as the non-zero zone expands, you extend and reallocate the grid on the processes.

Input: the first argument is the number of steps, the second argument is the name of the file containing the initial grid, in the following format:

```
3
7
0011100
0010000
0001000
```

(First two lines: number of rows, number of columns, followed by one row per line and one cell per byte.)

Output: the number of living cells after the specified number of steps.