Bio-Inspired Artificial Intelligence

Cellular Automata Laboratory Exercise

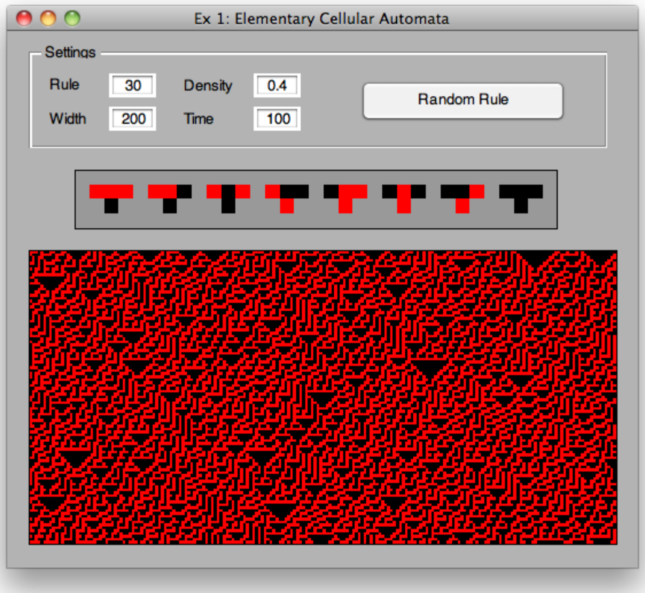
Thomas Schaffter & Sara Mitri

November 13, 2009

Exercise 1 - Elementary Cellular Automata

We will start with the most classical and simple example: the 256 Elementary CA. These are binary one-dimensional CA with neighborhoods composed by 3 cells.

Open Matlab and run the m-file *ex_1.m*. This tool allows you to generate and visualize easily the behavior of the 256 elementary CA. In the *Settings* panel, a rule code can be assigned in the range [0,255]. *Width* and *Time* define the width of the cellular space and the number of time steps computed and displayed, respectively. Each run has the automaton space initially filled with random cell states (active cells in red). The density of the cells initially active can be changed by assigning a real value in the range [0,1] to the variable *Density*.

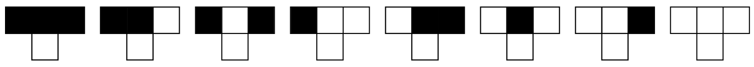


Rule 30 Density 0.4
Width 200 Time 100

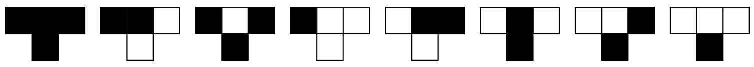
Figure 1: Matlab tool to visualize the 256 elementary CA. After each run, the *transition rule* and the *cellular space* are displayed. Active cells are represented in red.

Transition Rule

Check your understanding of the way the numeric code is assigned to the transition rule of the elementary CA. Complete below the transition rule associated to Rule 60 (here active cells are in black) without using Matlab.

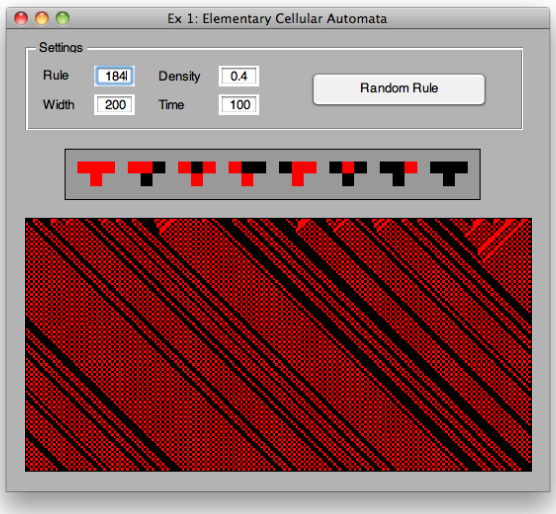


Find the rule code of the following transition rule. Check now your results with the Matlab tool and observe the behaviors determined by these two rules.



Traffic Rules

Run the *traffic* rule corresponding to Rule 184. Try different values for the initial density of the *cars* (red cells) and see what kind of *traffic flow* emerges.

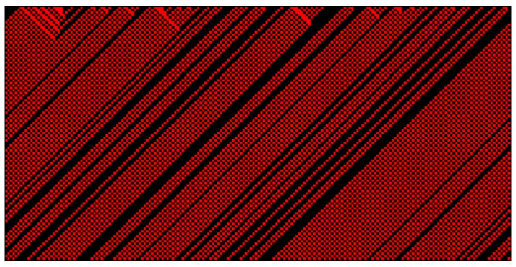

Ex 1: Elementary Cellular Automata

Settings

Rule 184 Density 0.4

Width 200 Time 100

Find the rule code that makes the cars run in the opposite direction relative to Rule 184. What kind of boundary condition is implemented ?

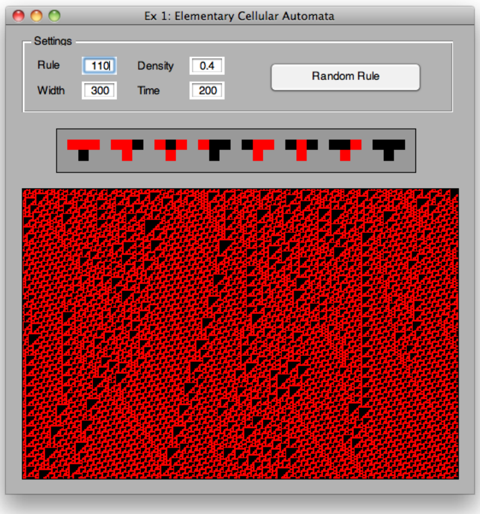


Try to define other kinds of traffic rules, e.g. one where cars do not wait for the space in front of them to become free in order to move. Notice that one must take care to ensure the *conservation of the cellular space*, i.e. cars should not be created nor destroyed by the CA rule.

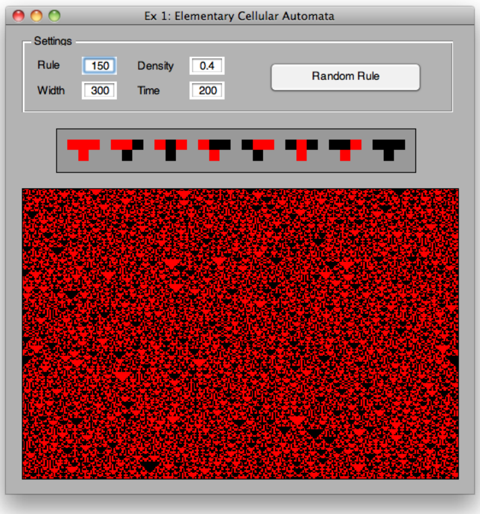
Qualitative Classification

Observe the behavior of Rule 40, Rule 56, Rule 30 (the Random Number Generator) and Rule 110 (the computationally universal rule at the heart of Wolfram's book "A New Kind of Sciences") that were used in the lecture as examples of the *four qualitative classes* of CA behavior.

Assign rule number at random using the button *Random Rule*. Observe the resulting CA behavior and try to assign the rule to one of the four classes of the qualitative classification. Try to do that for at least 5 random rules.


Ex 1: Elementary Cellular Automata
Settings
Rule 110 Density 0.4
Width 300 Time 200

(a)


Ex 1: Elementary Cellular Automata
Settings
Rule 150 Density 0.4
Width 300 Time 200

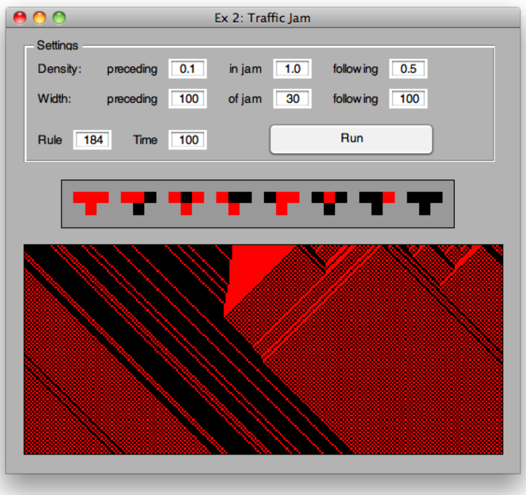
(b)

Figure 2: (a) Rule 110 features a complex, localized, propagating and interacting structure. (b) Which one of the four qualitative classes does Rule 150 belong to ?

Exercise 2 - Traffic Jam

The animation of elementary CA is not really useful for most of them, since it produces what appears as a meaningless sequence of cellular space configurations. However, in the case of the *traffic* elementary CA defined by Rule 184, the animation implemented here gives an easily interpretable and visually appealing result.

Run the m-file *ex_2.m*, that implements a single lane route containing a *traffic jam* characterized by a high density of cars, preceded and followed by two zones of lower car density. The parameters collected in the *Settings* panel allow the definition of the widths and car densities of the three zones. Every click on the button *Run* generates a different initial condition, i.e. the initial state of the traffic jam.


Ex 2: Traffic Jam

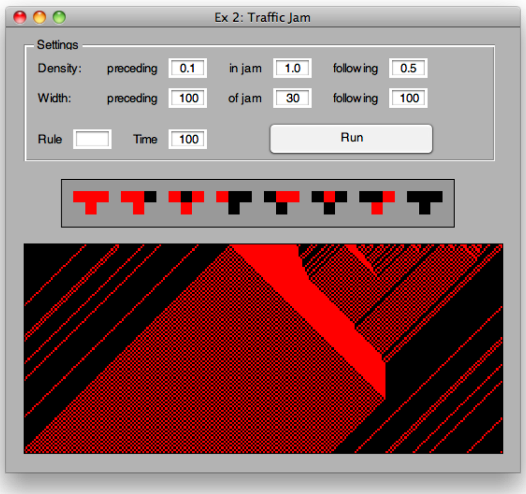
Settings
Density: preceding 0.1 in jam 1.0 following 0.5
Width: preceding 100 of jam 30 following 100
Rule 184 Time 100

Figure 3: Illustration of a *traffic jam* defined by a high density of cars, preceded and followed by zones of lower car density. Cars are represented by red cells.

Modify the value of the *Density* and *Width* parameters and click on *Run*. Observe and qualify the behavior of the traffic jam as modeled by Rule 184.

Define and observe the behavior of a *negative jam* (a clearing between two jams).

You found in Exercise 1 the rule code for the motion in the opposite direction relatively to Rule 184. Assign the rule code found in the field *Rule* of the interface and see if the motion is actually reversed.


Ex 2: Traffic Jam

Settings
Density: preceding 0.1 in jam 1.0 following 0.5
Width: preceding 100 of jam 30 following 100
Rule Time 100

Figure 4: Run the CA with the rule code that leads to the above traffic jam behavior. Is the motion reverted compared to the one displayed in Figure 3 ?

Exercise 3 - Diffusion-Limited Aggregation

Diffusion-limited aggregation (DLA) is the process whereby particles undergoing a random walk due to Brownian motion cluster together to form aggregates of such particles. This model was proposed to simulate certain types of aggregation, for instance metal ions diffusing through a fluid and sticking to a charged electrode. "Diffusion" because the particles forming the structure, also called *Brownian tree* or *cluster*, wander around randomly before attaching themselves ("aggregating") to the structure. "Diffusion-limited" because the particles are considered to be in low concentrations and therefore don't interact together. Other examples can be found in non-living and living nature, e.g. mineral deposition, snowflake growth, lightning paths or corals growth.

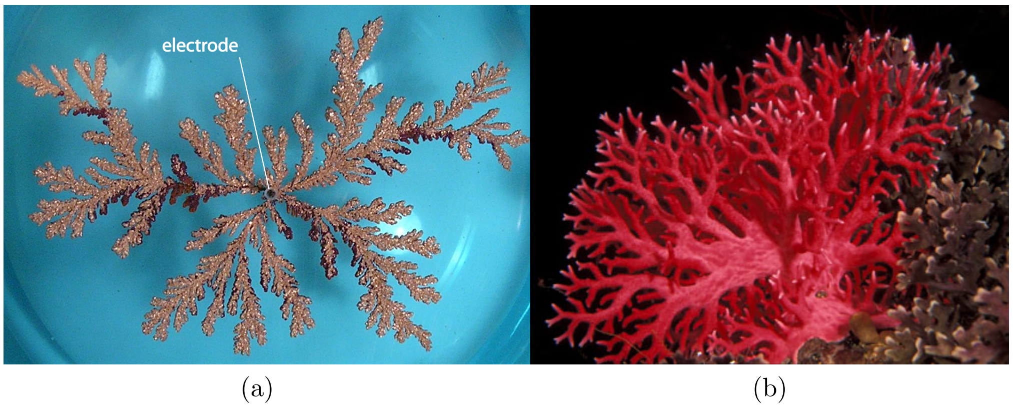


Figure 5: (a) DLA structure grown from a copper sulfate solution in an electrodeposition cell. (b) Red coral *Errina novaezelandiae* in the Te Awaatu Marine Reserve in Fiordland.

Cellular Automaton

The goal of this exercise is to implement a two-dimensional CA that mimics a DLA process. Open Matlab and run the m-file *ex_3.m*. Have a look at the content of the file to become familiar with the main variables (top part). The automaton space is formed by an array of cells, which size is given by the integer parameters *nx* and *ny*. *The x-axis represents the vertical axis and the y-axis the horizontal axis.* Motionless, non-interacting particles (blue cells) are initially present in the CA space. Their density can be changed through the variable *particlesDensity*.

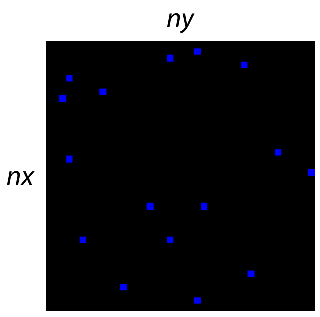


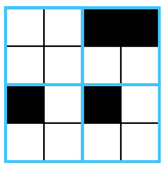
Figure 6: First run of *ex_3.m*. The above automaton space is defined by an array of 40x40 cells. Motionless particles (in blue) are initially present in the environment.

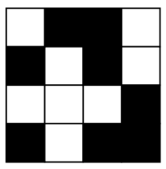
Implementation of a Pseudo Brownian Motion

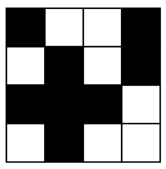
This section describes how to obtain particles that undergo a random walk. These particles are assumed to be in low concentrations, so no interaction between them are modeled.

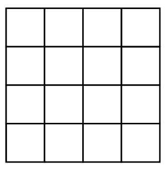
Start by setting in the code the number of time steps to T=7000. Run *ex_3.m* and observe the motion of the particles. Note that you can speed up or slow down the walk of the particles by adjusting the parameter *delay*. You can also change the size of the automaton space (nx and ny must be divisible by 2) or resize the window to get a better visibility. Try to qualify the movement of the particles. What kind of neighborhood is implemented ? Does the observed motion features some randomness ? If it's possible, find the deterministic and random components of the motion.

Try to understand what does the code between the tags "Pseudo Brownian motion" and "End". The automaton space is divided into *blocks* of 2x2 *cells* inside which the position of the particles is updated. At each time step, two random, complementary matrices containing "0" and "1" elements are generated (cw and ccw). With your understanding of the code, find and draw the updated position of the particles at time $t+1$ for the initial configuration of the following 4x4 CA. Consider vectors $xind = yind$ = [1, 3] which define the indexes of the upper-left cell of each block, and black cells below to represent particles and/or "1" matrix elements[1].


CA space (t)


cw


ccw


CA space (t+1)

You observe that the motion of a particle is actually limited to the block it originally belongs to. Which simple modification could be adopted to allow the particles to move across the entire automaton space ?

What happens if you replace in the code s=0 by s=mod(t,2) ? Run the modified m-file, observe and comment the new motion of the particles.

Growth of Brownian Trees

This section aims to simulate the growth of *Brownian trees* (or *"sticky" clusters*) using the cellular automaton developed at the end of the previous section. Beforehand, operate the following modifications where required. Then run *ex_3.m*.

- $nx = ny = 200$
- $particlesDensity = 0.1$
- $T = 7000$
- $delay = 0$

[1]Hint: only the elements in cw that are defined by $xind$ and $yind$ are used, that are here $cw(1,1)$, $cw(3,1)$, $cw(1,3)$, and $cw(3,3)$.

- *enableBrownianTree* = 1

At time t=0, the Brownian tree is composed of a single, yellow cell placed at the center of the automaton space. Observe how the tree grows from moving particles that stick to it. What is the condition required for a blue particle to stick and become part of the tree ? Figure 7 illustrates the growth of two trees resulting from two successive runs of *ex_3.m*.

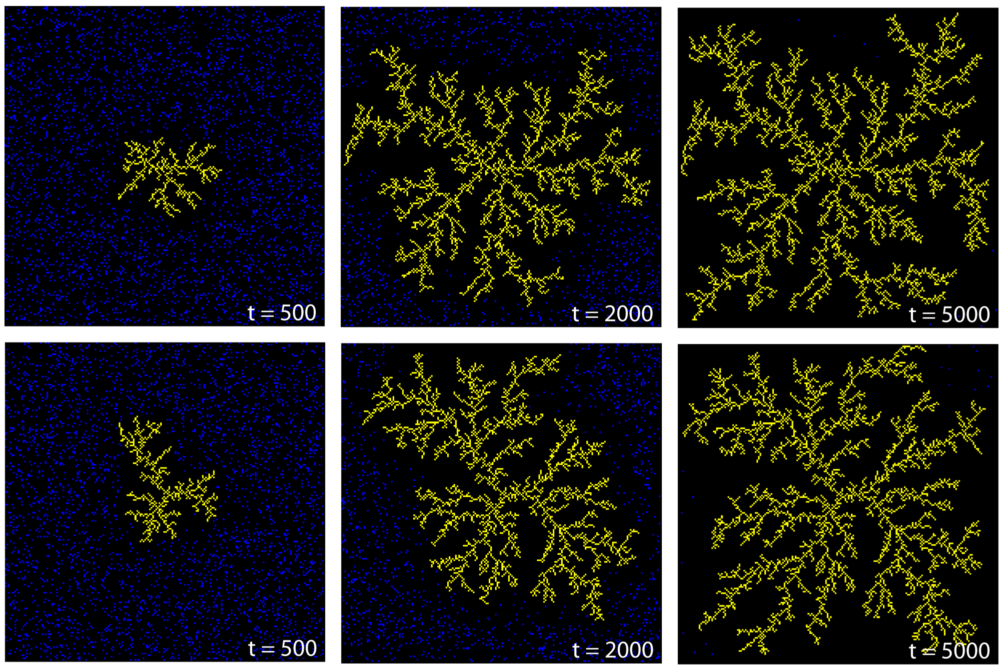
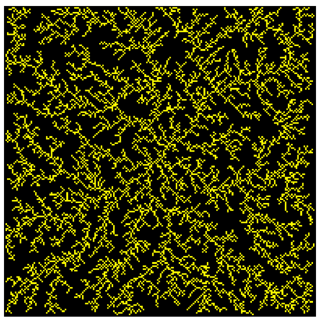


Figure 7: Growth of Brownian trees using a cellular automaton that mimics a diffusion-limited aggregation (DLA) process. The two different trees are the result of two successive runs of *ex_3.m* (with identical parameter values). Snapshots are taken at time t=500, 2000 and 5000 iterations.

Have a look at the structure of the following tree. What distinct it from the two ones displayed in Figure 7 (t=5000) ? Try different values for the parameter responsible of this change and observe how it affects the structure of the tree.



At which part of the tree does the growth occur mainly ? How do you expect the size of a Brownian tree (number of cells that compose it) to vary in function of the time ?

You can plot the size of a tree over the time by setting *plotGrowth* = 1. To speed up the process, you can comment the part of the code which updates the graphical representation of the cellular automaton. Does the obtained profile correspond to what you expected ? Comment the following growth profile.

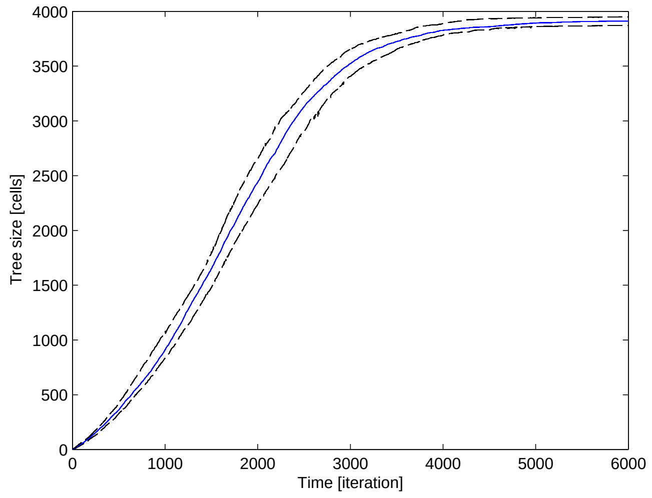


Figure 8: Size of Brownian trees (number of cells) over the time (number of iterations). The plain line is the median and the dashed lines are the upper and lower quartiles obtained from the growth of 50 trees (*particlesDensity* = 0.1).

Does the initial density of the blue particles present in the automaton space affects the general trend of the observed growth profile ?

Do diffusion-limited aggregation processes possess fractal properties ? Look at Figure 9 to convince yourself.

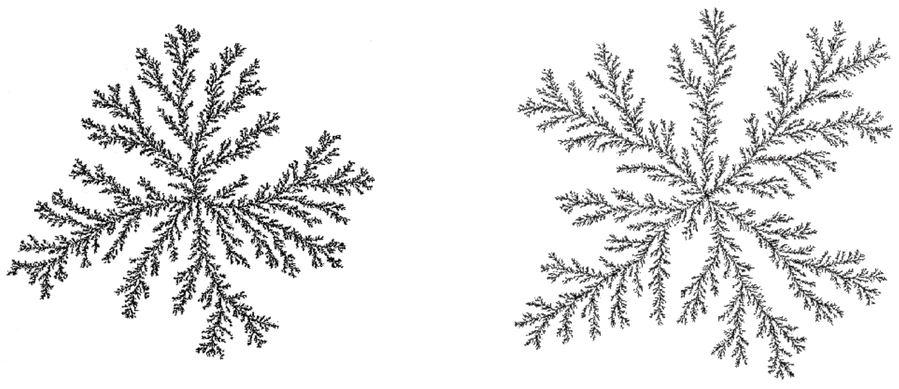


Figure 9: Diffusion-limited aggregates with (a) 1 million and (b) 100 million particles.

Compare the results obtained so far in Figure 7 with the developed cellular automaton with the red coral structure and the structure grown from a copper sulfate solution displayed in Figure 5. What is the main difference observable ? Try to find what can account for this difference. Which single parameter p would you need to add to the current implementation of the CA ? Figure 11 illustrates the result of the improved cellular automaton run with decreasing values of this new parameter.

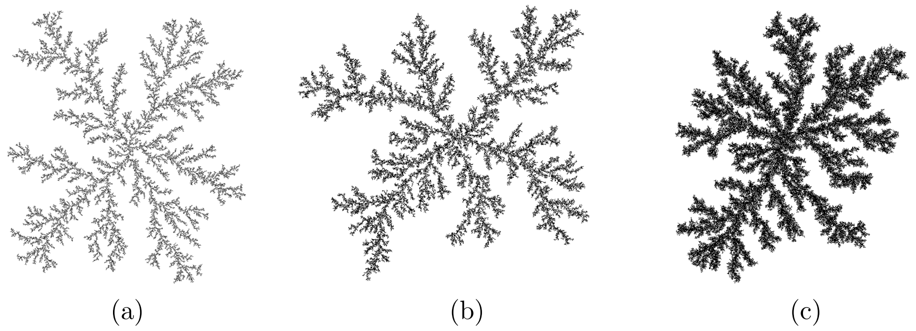


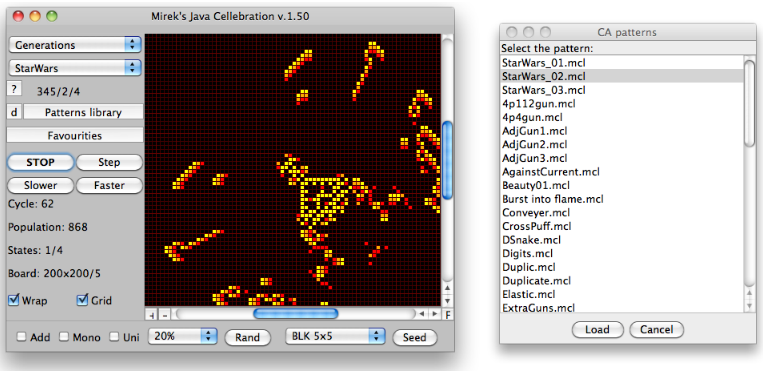
Figure 10: One single parameter p accounts for the difference between these three Brownian trees. What does it represent ? (a) $p = 0.2$ (b) $p = 0.05$ (c) $p = 0.01$

Mirek's Cellebration

A very complete Java tool for CA experiments has been implemented by Mirek Wojtowicz. It can be either downloaded as a standalone software or run as an applet from:

`http://psoup.math.wisc.edu/mcell/`

MJCell allows playing 300+ Cellular Automata rules and 1400+ patterns. It can play rules from 13 CA rules families such as Generations, Life, 1D totalistic, 1D binary, Neumann binary, General binary, Margolus neighborhood, etc. Hence, MJCell is the ideal tool to experiment the rules introduced in the lecture and observe the variety of patterns and dynamics that can occur in a CA universe.



Generations
StarWars
345/2/4
Cycle: 62
Population: 868
States: 1/4
Board: 200x200/5
☑ Wrap ☑ Grid
☐ Add ☐ Mono ☐ Uni 20% BLK 5x5

Select the pattern:
StarWars_01.mcl
StarWars_02.mcl
StarWars_03.mcl
4p112gun.mcl
4p4gun.mcl
AdjGun1.mcl
AdjGun2.mcl
AdjGun3.mcl
AgainstCurrent.mcl
Beauty01.mcl
Burst into flame.mcl
Conveyer.mcl
CrossPuff.mcl
DSnake.mcl
Digits.mcl
Duplic.mcl
Duplicate.mcl
Elastic.mcl
ExtraGuns.mcl

Figure 11: In Star Wars rule, most cells travel horizontally and vertically at light speed in all sorts of Hauler-like formations. Many formations resemble space ships, often shooting missiles, what gave Mirek Wojtowicz the idea for the rule name.