Chapter 12

Cellular Automata for Modeling Spatial Systems

12.1. The concept of the automaton and its modeling

The evolution of computer power in the past few years has facilitated the emergence of simulation methods at the expense of the analytical resolution of mathematical models. Indeed, cellular automaton simulation allows us to free ourselves from the resolution of partial differential equations, by explaining these equations in discrete terms of time, space and condition. Thus, the performance of office computers, and the development of theories and techniques of simulation like cellular automata or multi-agent systems allow us to attack these increasingly complex systems, with a quite fine discretization of space and time.

Moreover, the difficulty, or the impossibility even, of performing experiments in the social or environmental field to test hypothetical theories, adds to the interest of simulation, which allows the rapid realization of numerous tests, supported by graphical results, often connected to a geographic information system (GIS) which allows the easy comparison of the result with the observed terrain.

Nonetheless two paths seem to diverge quite substantially in this domain. The quest to complexify the models risks losing the essential objective of the research, that is, to explain.

Chapter written by Patrice LANGLOIS.

To explain is to articulate, in the easiest possible way, a phenomenon in the framework of a rational theory. However, the eagerness to continually simulate reality more precisely through models is also the wish of practitioners who can also make weaker predictions such as in city planning, meteorology, weather forecasts, etc. Unfortunately these two approaches are not always compatible, as often the closer a model resembles reality, the less it can actually explain.

12.2. A little bit of history

World War II acted as a stimulant for many new scientific developments. It led to the birth of the computer, made necessary due to the huge number of calculations needed for the creation of the atomic bomb. It is also the period of the development for cryptographic methods used to decode the German's secret messages. In this environment, mathematicians like the American John Von Neumann (1903-1957) and the Englishman Alan Turing (1912-1954) became pioneers. Von Neumann worked on the design of the first computers at the Los Alamos National Laboratory. There he invented cellular automata in the late 1940s. It was also Von Neumann who developed game theory in 1944 [VNM 44]. Turing, for his part, invented automatic decoding systems to decode German encrypted messages. He also participated in the development of ideas for what became computing, and most importantly, through his theoretical work, he invented the concept of the virtual machine, the Turing machine. He participated in the mathematical revolution of the 20^{th} century following Gödel's results concerning incompleteness where he showed in particular that numbers and functions exist that are incalculable and possess unsolvable problems. All of these developments question the grand theoretical program imagined by Hilbert during the previous century, who ambitiously set out to codify mathematical reason in a general system of axioms and rules of inference.

It is in this context that the concept of the cellular automaton emerged [FAT 01]. Von Neumann tried to invent an electromechanical machine with this capacity but the level of technology at the time was insufficient. One of his colleagues, Stanislaw Ulam (1909-1984) who worked on recursive geometrical objects, gave him the idea of a formal construction, using the computers in the Los Alamos laboratory to operate a cellular system subject to simple rules. After this the cellular automaton was born.

Von Neumann developed a virtual auto-reproductive machine which had the properties of a universal calculator, although he did not publish it in his lifetime, perhaps thinking that it was too complex (29 states) and that it did not follow the "natural" rules of physics concerning invariance by rotation and by symmetry.

It was not until 1970 that a much simpler cellular automaton was made public, John Conway's game of life. It was publicized by Martin Gardner in the American Scientist. In 1982, Conway and other researchers proved that the game of life also possessed the properties of a universal calculator. It is thus far the simplest cellular automaton constructed with this property, since there are only two states and moreover it verifies the properties of invariance by isometric transformation. We will not here develop these mathematical properties, as they are quite difficult. For more details see [BER 82] or [POU 85].

In order to present the concept of the cellular automaton in a didactic fashion we will not be following the historical development of this concept. We will begin with the most elementary concept of the automaton in its finished state before formally constructing that of the cellular automaton and seeing examples applied in geography.

We will address here only the automata in discrete time and state, even if the continuous automata associated with mathematical techniques such as the Laplace transformation can play an important role in certain areas of geography such as hydrology.

12.3. The concept of the finite state automaton

A finite state automaton is a mathematical object, and we will first present it intuitively to better understand its formalization after that. We must imagine a device which has at least an input channel, an output channel, connected to a box containing a self-powered mechanism. An input channel receives one by one (sequentially), coded information with symbols that constitute the input alphabet. Likewise the output channel produces symbols written in the output alphabet. In short, the box contains the means of internal representation, a memory capable of containing a symbol called the *state* of the automaton traced in the alphabet of states. The three alphabets, input, output and state contain only a finite number of symbols. The value of an input or an output can be logical (binary), quantitative (integer, real), qualitative or purely symbolic (encryption according to a discrete alphabet like before). It can also constitute a vector of elementary inputs (or outputs) when there are several input or output channels. Once again we give these more or less complex values entering or exiting the automaton the name symbols, without specifying their nature.

The internal mechanism breaks down into two functions. The first function is the capacity to read the symbol at each input (which we call the vector or more simply the input) and to modify the state of the automaton contingent on the input and the previous state. This is the *transition function*. The second function enables a symbol

to be present outside in the output channel, calculated in contingence with the input and the state of the automaton. This is the *output function*.

The input, output and state symbols can, in the most general cases belong to different alphabets. However, in a simplified version, used in particular with cellular automata, the same alphabet is used for the three and the output mechanism is reduced to its simplest form and consists only of producing the state of the automaton on output. Therefore the mechanism is reduced to a single transition function.

Consequently it is apparent that a finite state automaton is an elementary system which can serve the construction of a complex system by a series of connections or in parallel with several automata. The outputs of some are connected to the inputs of others. To be able to connect several automata, it is necessary to synchronize them through the definition of a common time and a control mechanism synchronized between the automata and their connections. We then obtain a network of automata [WEI 89].

12.3.1. *Mealy and Moore automata*

We are now able to formally define the idea of an finite state automaton, or Mealy's automaton, as a structure $M = (S, A, B, T, H)$ where S is the state alphabet, A the input alphabet, B that of the output, T the transition function which is the application of $S \times A$ to S, and finally H the output function, which is the application of $S \times A$ to B.

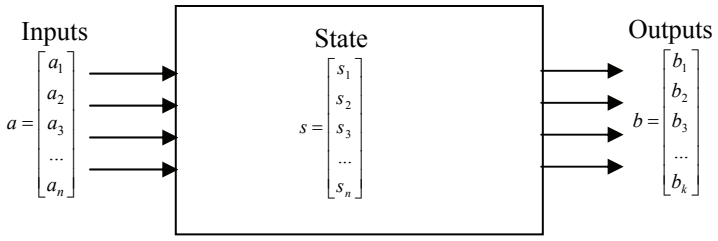


Figure 12.1. *General outline of an automaton*

Mechanism: at the discrete point *t,* for the automaton in the state *s(t)*, the arrival of an input value *a(t)* makes the automaton pass into an other state *s(t+1)* by the

application of the transition function *T,* and calculates the output *b(t)* by applying the function *H*. The automaton will be outlined by these two equations:

$$s(t+1) = T(s(t), a(t))$$
$$b(t+1) = H(s(t), a(t))$$

which are those of a *dynamic deterministic system in discrete time.* If the output *b* does not depend on the input *a* in *H*, then it is Moore's automaton.

12.3.2. *An example of Moore's automaton*

Now a simple example of an automaton is presented to show how such an object can be manipulated. The following Moore's automaton is called an adder. It is an automaton that takes two input numbers in binary code and sends back their total in output, also in binary code. Each number *n* is composed of *k* bits and is noted $n = n_{k-1} \ldots n_i \ldots n_3 n_2 n_1 n_0$ which symbolizes its binary spelling, the succession of 0 and 1. In total *n* is considered as a word of *k* letters written with the alphabet {0, 1}. *m* and *n* represent the two inputs and *r* the output of the automaton containing the total of *m* and *n.* The automaton reads the two numbers sequentially, starting from the right, in other words at the beginning with the least heavy bits. At each stage *i* it processes the bits m_i and n_i and calculates their total, r_i. The automaton also needs a state, *s* to memorize the carry digit (0 or 1). The alphabet of input, output and state is therefore the same, $A = B = S = \{0,1\}$. The two functions of transition *T* and of output *H* are defined according to the binary addition table: 0+0 = 0; 0+1 = 1; 1+0 = 1, these three additions are made without a carry digit (in other words a carry digit of 0) and 1+1 = 0 with a carry digit of 1. The transition function *T* therefore combines a carry digit s_i and an input m_i+n_i, a new state s_{i+1} which is the new carry digit after the addition of m_i and n_i. This is written $s_{i+1} = T(s_i ; m_i n_i)$. They are presented as follows: $T(0; 00)=0$; $T(0; 01)=0$; $T(0; 10)=0$; $T(0; 11)=1$; $T(1; 00)=0$; $T(1; 01)=1$; $T(1; 10)=1$; $T(1; 11)=1$. Also the output function *H* is defined by $r_i = H(s_i; m_i n_i)$ with: $H(0; 00)=0$; $H(0; 01)=1$; $H(0; 10)=1$; $H(0; 11)=0$; $H(1; 00)=1$; $H(1; 01)=0$; $H(1; 10)=0$; $H(1; 11)=1$. This is summarized in the two following tables.

T				
$m_i n_i$ / s_i	00	01	10	11
0	0	0	0	1
1	0	1	1	1

H				
$m_i n_i$ / s_i	00	01	10	11
0	0	1	1	0
1	1	0	0	1

Table 12.1. *Transition function and output function*

The machine operates in the following fashion: in the first instant, the state (carry digit) s_0 is at 0 so the automaton reads the first two bits $m_0+n_0 = 1+1$. The table of function T gives the following state $s_1 = 1$, and table H gives the output $r_0 = 0$ which corresponds to "1+1=0 keep 1". Then we move on to the second bit $m_1+n_1 = 0+1$; with a carry digit of 1 which again gives $r_1=0$ and we keep $s_2=1$ and so on. The final result is $r = 01101000$ and the carry digit is zero. If the carry digit is not zero at the end of the k bits calculation, there is an overflow in capacity, and the result cannot be carried in k bits.

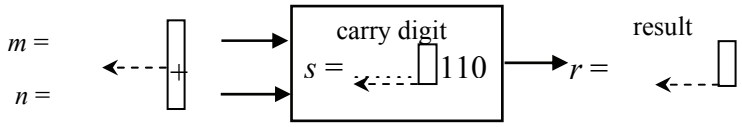


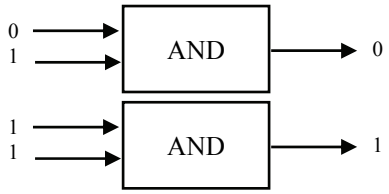
Figure 12.2. *The adder*

12.3.3. *Moore's automaton simplified*

Very often, and this will be the case for cellular automaton, the input function is reduced to the simple communication of the state towards the exterior (this is the identity function). The function H in this model has been omitted, it becomes apparent. In this case there is also $B = S$ as the output symbols are the states. Moreover, as the outputs of an automaton are often the input of another automaton, $A = S$ is also used, resulting in there being only one set of symbols for the inputs, states and outputs at the same time. A simplified automaton M is therefore limited to the data set S of states and of the transition mechanism T, therefore $M = (S,T)$.

12.3.4. *Logic gate AND: an example*

A logic gate can be considered like a Moore's automaton simplified to two inputs and one binary output (or Boolean). Here the transition function does not depend on the inputs or the state. For example, the Boolean operator AND takes state 1 (and sends it on output) if its tow inputs are worth 1; if not it takes the value 0.



To define the transition function *AND* it suffices to give for each pair of inputs possible, the value of the associated output: therefore *AND*(0,0)=0, *AND*(0,1)=0, *AND*(1,0)=0 and *AND*(1,1)=1. These values can be summarized in a matrix of 4 columns and 2 lines. In the first line of *T* all of the possible inputs are placed and in the second line the associated outputs. This is therefore equivalent to writing the truth table of the logic operation *AND*.

$$T = \begin{pmatrix} 00 & 01 & 10 & 11 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

By the interconnection of automata which carry out the logic operation of basic Boolean algebra (AND operator, OR operator, NO operator) complex logical functions can be constructed and the arithmetical calculation of the binary numbers are deducted, which are at the root of the function of microprocessors. Therefore the adder can construct itself like a combination of logic gates.

12.3.5. *Threshold automata, window automata*

Threshold automata are very widely used, especially in neuron networks. This represents a simplified automaton with binary values. It has n inputs where $a_i(t)$ is the value of the input I associated with a weight p (a real number, called synaptic weight in the neuron network), p_0 being the weight of the state of the automaton and Θ a threshold of excitability. This excitation level is defined by a linear combination of inputs and state. If the excitation level is lower than the threshold Θ then the state remains equal to 0 (not excited). If not it passes to 1 (excited). Therefore, the new state (which is also the output) is calculated by:

$$a_0(t+1) = \begin{cases} 1 & \text{if} \quad \Theta \le \sum_{i=0}^{n} p_i a_i(t) \\ 0 & \text{else} \end{cases}$$

The window automata are also often widely used. The state becomes excited when the value of a linear combination of states belongs to an interval between a minimal Θ_{min} threshold and a maximum Θ_{max} threshold:

$$a_0(t+1) = \begin{cases} 1 & \text{if} \quad \Theta_{\min} \le \sum_{i=0}^{n} p_i a_i(t) \le \Theta_{\max} \\ 0 & \text{else} \end{cases}$$

The simplest example of a window automaton, used for example in the game of life, is where the weight p is worth 1, which means that the excitation level is simply the number of its excited neighbors.

12.3.6. *The automaton and the stochastic process*

In cases when the transition mechanism is no longer functional but random (which is frequent in social sciences) a generalization of the function $y = f(x)$ is used, which is called *transition probability:* instead of associating a single value y with each value of x, as the function does, a transition probability combines the total worth of several x values of y but these values only appear, given that x, in accordance with a certain probability $\pi(x, y)$. Thus, if for a given value of x the probability $\pi(x, y)$ is zero for all the value of y except one (which is therefore of certain probability) and we find the usual function concept.

For example, take a network of n automata that model the flow of transport (counting, for example, the number of vehicles) where the nodes each contain a stock, the overall stock of the system staying unchanged. Each automaton is connected to the others. Each possesses a state $s_i(t)$ which represents its stock at the time t. The probability $\pi(i, j)$ is the probability that an element of the site i passes into j. We can then proceed to progress this process using the Monte Carlo method. If the sizes are very big it is also possible to treat the model in a deterministic manner, the new stock is equal to the earlier stock with less outputs and more inputs, which is written simply by:

$$s_j(t+1) = \sum_{i=1}^{n} \pi(i, j) s_i(t)$$

if we call it $s(t)$, the vector line of n states at the moment t and T the matrix of the transition containing the $\pi(i, j)$, the calculation of $s(t+1)$ is made with the following matrix product:

$$s(t+1) = s(t).T$$

We will examine in a little more detail a probability diffusion model, Hägerstrand's model.

12.4. The concept of the cellular automaton

After having examined the concept of the automaton we can now move on to examine the concept of the cellular automaton as a network of automata in a finished state, all identical and dispersed regularly in space. The automata here are called cells, and the input-output connections between cells are the links between the automata in this space. Immediately it is apparent that this concept can be used in geography to model a spatial dynamic. The cells are like the pixels of an image but which also possess an evolution mechanism of their value.

12.4.1. *Level of formalization*

The concept of cellular automaton can be defined on at least two levels, which we will identify as "concrete" and "abstract".

The "concrete" level is the computing model (graphic, conceptual or algorithmic) which will be programmed. This model therefore has the objective of making a program work in a computer and producing results on a screen or in a file using the information we give it.

The "abstract" level is a purely mathematical definition, very simple in its structure; its properties are simplified in comparison to the "concrete" level. This allows the fundamental properties to be studied more easily. Nonetheless, in this simplification, certain characteristics are generalized distancing themselves from their concrete form. For example, in order to avoid the effects of borders which modify configurations during functioning, we consider, in its most abstract form, that the cellular space is an infinite network of cells. This makes it impossible to concretize in a computer.

These two definitions of levels are obviously useful but can be misinterpreted if the reader does not find their context in the description. Our objective here is not to advance the mathematical theory of cellular automata but to show applications that can be used in the particular field of geography. Nevertheless, this does not prevent us from profiting from the theory to properly define the "concrete" automata in concern with the rationality of the model.

We will use a formalized definition in the presentation of the concept of the CA to remain general and didactic. In the applications section the models used are much too complex to be able to formalize completely. They will therefore be described in a more intuitive manner so as not to forget the objective, which is here the application and not the theory. For a mathematical approach to the theory of CAs Nicholas Ollinger's thesis [OLL 02] "Cellular automata: structures" can be consulted.

12.4.2. *Presentation of the concept*

A *cellular automaton (CA)* is a network of Moore's automata simplified, interconnected and (in general) of identical types. Each automaton is called a *cell*[1]. These cells are organized within a *network* (of one, two or three dimensions, rarely more) where they occupy the nodes. They are connected to each other by a *neighborhood graph***,** which makes up the network links. Each cell, at each moment, is in a certain state (a whole, a color, etc.), which belongs to a *set of finished states* common to all cells. The connections between the cell and its neighborhood allow the cell to "know" the state of its neighbors. Thus, the motif constituted by its own state surrounded by the states of its neighboring cells allows each cell, with the help of its *transition mechanism* to evolve its state.

The cellular network possesses a *structure* which simultaneously defines its global and local characteristics: global form and area size, network geometry, the topology of the linking edges: an infinite area, or a limited area without joining, or a finite area but unlimited due to a total or partial linking which can be looped in one dimension or for two dimensional in cylinder, sphere, torus, etc.)

Moreover, the cells are located and "drawn" in a geometric *space*; they have a form (2D: squared, rhombus, triangle, etc.). The joining of this group of forms makes up the *spatial domain* of the cellular automaton that must be connected (most often a rectangle for the squared cells). The functioning of the cells is linked to a common *time* for all cells. This time is discrete, it is represented by a variable integer ***t*** that is worth 0 at the start of the simulation and rises by 1 at each stage of the transition of the automaton.

Finally, it is necessary to define the *cellular model***,** which understands the definition of the states and the transition mechanism.

We can now give the formal definition of a CA.

---

1 Von Neumann worked on the modeling of the auto-reproduction of life, using biological analogy.

12.4.3. *The formal definition of a cellular automaton*

A cellular automaton is a quadruplet ($\boldsymbol{Z}^d$, *S, V, T*) where the integer *d* is the *dimension* of the CA, the finished group *S* is the *set of states*, *V* a series of *n* elements of Z^d is the *neighborhood operator* (or more simply the neighborhood) and the function *T* of S^{n+1} in *S* is the *local transition rule* (or more simply the transition).

Given certain cellular automaton *A*, we denote by S_A, V_A and T_A respectively the group of states, the neighborhood and the transition of the cellular automaton *A*.

This definition, a little abstract, needs a few details.

12.4.4. *The cellular network*

In the definition, the cellular network is identified as a direct product (Cartesian) $\mathbf{Z}^d$. It represents the indexation of cells forming a regular network immersed in the geometrical space at *d* dimensions $\boldsymbol{R}^d$. Thus, in one dimension, a line of cells forms the network, each indexed by an integer *i.* In two dimensions (d=2) the cells are organized in the nodes of a gridline, and $\mathbf{Z}^d$ is the group of indexes (i_1, i_2) of integers, representing the number of line and column of each node of the network.) We will frequently call the index an element $i = (i_1, i_2, \ldots i_d)$ of $\mathbf{Z}^d$.

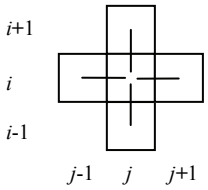
In practice, the number of cells remains complete, it is limited to a connection area $D = [1, n_1]\times[1, n_2]\times\ldots\times[1, n_d]$. For example, for $d = 1$, $D = \{1, 2, \ldots, n_1\}$, for $d = 2$, *D* is formed by couples of integers (i_1, i_2) with $i_1 \in [1, n_1]$ and $i_2 \in [1, n_2]$.

12.4.5. *The neighborhood operator and cell neighborhoods*

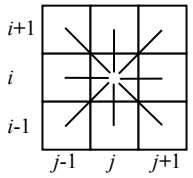
The neighborhood operator is an application *V* which allows the construction of all the cell neighborhoods by the same method. It is formalized by a series of *n* translation vectors (the relative offsets of indexes) allowing it, as long as it is applied to a cell *I*, to make all the cells of its neighborhood. For example in one dimension, the neighborhood operator $V = (-1, 0, 1)$ allows us to obtain the neighbors of cell 72 by 3 offsets, towards the left, the centre and right by: $i \rightarrow i-1$, $i \rightarrow i$ and $i \rightarrow i+1$ therefore the neighborhood: $V(72) = (71, 72, 73)$.

In two dimensions, the neighborhood type $V_4 = ((0, 1), (0, -1), (-1, 0), (1, 0))$ called the Von Neumann neighborhood (see Figure 7.3) makes it possible to access the four cells situated above, below, left and right of the cell (i, j) in question:

$$V_4(i, j) = ((i, j+1), (i, j-1), (i-1, j), (i+1, j)).$$



The neighborhood type $V_8 = ((-1, -1), (0, -1), (1, -1), (-1, 0), (1, 0), (-1, 1), (0, 1), (1, 1))$, known as Moore's neighborhood (see Figure 12.3) enables access to the eight cells situated around the cell (i, j) of reference:



$$V_8(i, j) = \big((i\text{-}1, j\text{-}1), (i, j\text{-}1), (i\text{+}1, j\text{-}1), (i\text{-}1, j), (i\text{+}1, j), (i\text{-}1, j\text{+}1), (i, j\text{+}1), (i\text{+}1, j\text{+}1)\big).$$

12.4.6. *Input pattern*

The input pattern a_i of a cell i is the vector of n states of cells of its neighborhood $V(i)$, therefore $a_i = \left(s_j\right)_{j \in V(i)}$. Thus, for the neighborhood $V(i) = (i\text{-}1,\ i,\ i\text{+}1)$, the pattern of its states is therefore the sequence $a_i = \left(s_{i-1}, s_i, s_{i+1}\right)$.

The neighborhood of a cell may or may not include the cell itself. In the case where it is contained, a more concise notation of the transition mechanism T is allowed which only takes on input a_i instead of s_i and a_i. However, it can happen that the treatment of the cell state can be different from that of the neighborhood, and it is in this instance that they are differentiated.

12.4.7. *The local rule of the transition of the cell*

The automaton of each cell i is of the form $M_i = (S, T)$ where S is the group of states and T the transition mechanism which is written $s_i(t\text{+}1) = T(s_i(t), a_i(t))$ if the neighborhood does not contain the cell, or more simply $s_i(t\text{+}1) = T(a_i(t))$, if the neighborhood contains the central cell.

12.4.8. *Configuration and global transition mechanism*

The *configuration* of the CA occurs at the moment *t,* the application associates a state $s_i(t)$ with each cell *i* of the network. When there are *n* cells in one-dimension it is a vector of states $s(t) = (s_1(t), s_2(t), \ldots, s_n(t))$. The *global transition mechanism G* occurs when the application is dealing with an ordinary configuration *C* the configuration $C' = G(C)$ obtained by applying the local rule of transition to each network automaton.

12.4.9. *Configuration space: attractor, attraction basin, Garden of Eden*

With deterministic automata, if at time t_1 we again come across a configuration *C* already found at the time t_0 the series of configurations will repeat itself after t_1 in the same way it did after t_0 until it returns to *C*. The system between is therefore in a loop called an *attractor*. If the period of the loop (its length) is equal to 1, it is a *fixed point*, if not it is a *limited cycle*.

The group of configurations, which reach a given attractor, after an unknown number of iterations, is called the *attraction basin*.

In addition, it is interesting to know which configurations reach the same attractor or which bond more or less with each other. The configuration space is equipped with a distance that counts the number of states which differ in the two configurations (Hamming distance). If the network contains a finite number of automata, the number of configurations is also finite and in this way, the number of attractors and of basins is also finite and every configuration reaches an attractor after a finite number of iterations. The group of attractor basins is therefore a partition of the configuration space. However, if the network is infinite (general definition) a series of never converging configurations can exist.

We can also investigate configurations which can never be reached, that's to say those which can only be taken as initial configurations. These are called *Gardens of Eden*. Moore posed the question of the existence of the Gardens of Eden in 1962 in relation to auto-reproducing automata. Alvy Smith demonstrated the existence of these Gardens of Eden in the game of life in 1970.

It is obvious that the combinations of configurations are enormous, and this is why the theoretical study of the behavior of the CA is so complex, behavior which depends on both the initial configuration and the transition mechanism. It is quite an active field of research and the theory of cellular automata is beginning to take shape. Moreover, CAs are very handy tools for the study of discrete dynamic systems. One of the important theoretical questions being posed is the classification

of CAs. After an exhaustive study of the 256 binary CAs in one-dimensional Wolfram proposed a classification of the CAs in four categories, inspired by the theory of dynamic systems [WOL 83] [WOL 86]. This classification was criticized, but injected enthusiasm into the area of study and has even been the subject of a recent PhD thesis [OLL 02] which address other problematic relative to the calculability of indecisiveness and in particular universal calculability (which confers with a CA the attribute of power to simulate any cellular automaton). Problems linked to chaos, instability, sensibility of initial conditions are also important questions concerning dynamic systems and information theory [MAR 01].

12.4.10. *2D cellular automata*

1D cellular automata will not be developed here (see [WOL 02], [WEI 89]). 2D, surface automata, principally used in geographical simulation, will be examined here. The cell space is most often a rectangular area, associated with a network of squared mesh. However, automata with triangular or hexagonal meshes can be found, to see more complexity (Delaunay's triangulation, Voronoï diagrams) in irregular networks.

The network geometry infers a type of neighborhood between the cells.

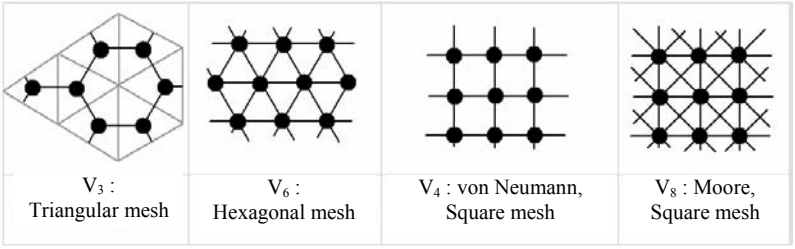


Figure 12.3. *Common types of neighborhood*

Figure 12.3 shows the most common types of neighborhood. For squared cells, two types of neighborhood are commonly used, V_4 and V_8. Types V_3 and V_6 are a little more complex concerning the level of indexation of points and of the definition of neighborhoods. The neighborhoods can also be defined more generally, from a particular metric space. A neighborhood is therefore formed from cells present in a disc of a certain radius centered on the cell.

For example, the *Manhattan metric* defined by $d(P_{ij}, P_{i'j'}) = |i-i'|+|j-j'|$ defines the disk in a diamond form, which is equal to V_4 for radius 1.

The *maximum metric* $d(P_{ij}, P_{i'j'}) = Max(|i-i'|, |j-j'|)$ also gives squared disks. It coincides with V_8 for a radius of 1.

For irregular meshing a neighborhood operator that would apply to all cells cannot be defined. The links of each cell with its neighborhood are specific; therefore each cell contains a list of these links in its structure.

To respect invariance by rotation and by symmetry, properties of great use in the world of physics, many cellular automata, instead of calculating their transition from the state of each neighboring cell individually (as generally happens) only the number of neighbors in a certain state (excited, for example, for binary states). Consequently threshold or window automata are often used. This is the case for the game of life.

12.4.11. *The game of life: an example*

The game of life [CON 70] is an emblematic automaton, it is very simple as regards its rules and yet at the same time complex regarding its dynamic. For this reason it has been widely studied (Conway, Gosper, Ray-Smith, etc.). In particular, the game of life possesses the universal calculator function and it is also the most simple, 2D CA with the auto-reproduction property (knowing that there are only three known CAs of this type, Von Neumann's 19 states and Codd's 8 states).

Description

Each cell is a binary, window automaton. Moore's neighborhood (V_8) is used. State 1 represents a living cell, state 0 a dead cell.

The transition $T(s, n)$ is the function of the state s of the cell i and of the number n of surrounding living cells, with $n = \sum_{j \in V(i)} s_j$. The following windows define the local mechanism:

$$T(0,n) = \begin{cases} 1 & if \quad n = 3 \\ 0 & otherwise \end{cases}$$
$$T(1,n) = \begin{cases} 1 & if \quad 2 \le n \le 3 \\ 0 & otherwise \end{cases}$$

[12.1]

which is set out by: if a cell is inactive and three of its neighbors are active it will become active, and it will only stay active if 2 or 3 of its neighbors are active.

All automata have this type of transition, that is to say whatever the function of the previous state *s* and the number *n* of living neighboring cells (to 1) can be defined using a table like the one below.

		n								
T		0	1	2	3	4	5	6	7	8
s	0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
	1	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**

The transitions can also be represented by a *transition graph* like the one below.

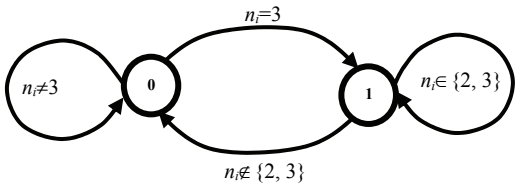


Figure 12.4. *Transition graph*

If we examine the number of automata of this type, each transition function possesses 2 x 9 = 18 possible couple values (*s, n*) to which 2 results can be attributed (0 or 1) which represents $2^{18} = 262{,}144$ possible transition functions. This is considerably more important than the 256 1D binary automata. Moreover, the behavior also depends on the initial configuration. If a quite small area is used, for example 10 x 10 = 100 cells, there are 2^{100} possible initial configurations, which represents a number of the command one thousand billion of billions of billions (10^{30}). By multiplying the number of possible transition functions the number of possible games to the command of 10^{65} are obtained.

The behavior of the game of life

Although totally deterministic, the long term configurations of the game of life, obtained according to the rules (1) are practically impossible to predict. Nonetheless some simple forms can be noticed, those that are stable when they appear isolated, like a 2 x 2 square or others that reappear according to a very short cycle, like a line of 3 cells which oscillate between a vertical and horizontal position. However, these forms are not absolutely stable. For example, they can collide with other mobile forms (named gliders) that have a quite short transformation cycle that is always

moving, like that in Figure 12.5. The collisions produce other forms, making overall behavior more complex.

Figure 12.5. *Glider: here a cyclic figure moves towards the south-east. Using symmetry 3 others can be constructed, moving in the 3 other directions*

12.5. CAs used for geographical modeling

Now a few examples of the use of cellular automata in geography will be shown. Our objective is not to make a new discovery in this area as there are, at the moment, numerous teams working on this subject. There is an abundance of bibliographies, for example the CASA website (see websites in the bibliography) or in the French speaking community the work on cellular automata applied to urban simulation [LP 97]. For educational purposes we prefer to describe more precisely a few realizations rather than citing a myriad of works without really unveiling their content. We have chosen 3 examples from varied fields, 2 of which were carried out in our laboratory. The first is inspired by the Hägerstrand model on diffusion, the major geographical process that will provide the opportunity to present a probability model and its deterministic correspondent. The second (SpaCelle) is a mini "platform", that is, software which contains no programmed model but in which the user interface allows the user to describe the behavior of the automaton through a series of simple rules, explained in a spatial representation language. This model was initially developed to be applied to urban geography but its field of use is much more general. The last example is applied to physical geography (RuiCells). It is a much more complex model than the previous one, applied to hydrological risk management. It presses layers of geographical information (DTM, digital terrain model). The automaton is constructed on a mesh of the surface in heterogenous cells (punctual, linear and surface). It models the surface runoff according to precipitation, terrain morphology and land use.

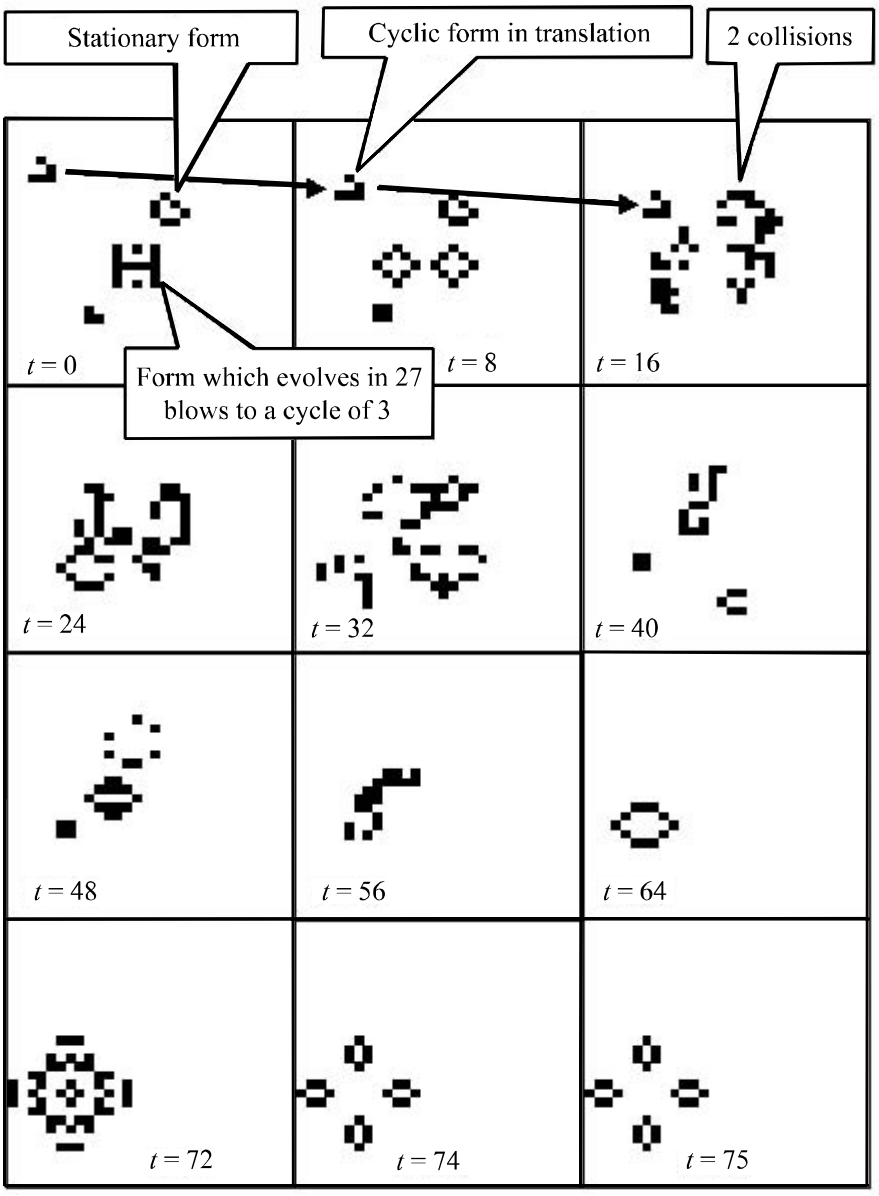


Figure 12.6. *Evolution (every 8 steps) according to the rules defined in (1) of an initial configuration of stationary forms and moving forms which collide (t = 16) to create a chaotic situation (t=16 to 56) and then restructured (t = 64) into a simple form which converges towards a fixed form in t = 74*

12.5.1. *Diffusion simulation*

12.5.1.1. *The Hägerstrand probability model*

It can be said that the interest of geographers in cellular automata begins with Hägerstrand [HAG 67] as he models the diffusion of an innovative process (agricultural grants for the transformation of wooded areas in pastures in the Asby area, south central Sweden 1929-1932). Although we can improve the Hägerstrand model by using a multi-agent system, as Eric Daudé did [DAU 04], this model equally conforms to the paradigm of the cellular automaton which behaves according to the regular cutting of time and space. The process is managed by a local transition rule applied to a neighborhood.

The big difference with the formal definition of a cellular automaton lies in the fact that this is non-deterministic. Its special area is a rectangle 70 km x 60 km dissected according to a 5 km sided grid which gives 168 cells. Each cell i contains a certain number of individuals e_i (these are the agricultural operators liable to be funded; this number remains the same during the simulation). An individual can be in one of the four following states: *innovator* (having adopted the innovation at the time t = 0), *having adopted* (in the past), *adapters* (at the present moment), and *potential adopters* (that will perhaps be adopted in the future). We seek to model the number of operators of each cell that has adopted the innovation over a course of time. The state of one cell I is therefore represented by the number $x_i(t)$ of those having adopted the innovation. The cell also contains the number of operators, $e_{i,}$ which remains constant. From these two values, we deduct $y_i(t) = e_i - x_i(t)$, the number of potential adopters. The diffusion process depends on the frequency of communication between the operator who has adopted and the potential adopters; it is a typical logistic model. Instead of using this model directly at a cellular level, we can simulate it at an individual level, that is to say, at the level of the operator itself by the random generation of an "adoption message" according to a certain probability model. This probability contact declines rapidly with distance and therefore proceeds essentially by neighborhood. For this we define a *contact field* from the neighborhood operator given in the definition of the CA. At each of these n shifts $V = (v_1,\ldots, v_k,\ldots, v_n)$ of the neighborhood operator we associate a probability of achieving a contact $P = (p_1, p_2,\ldots, p_k,\ldots, p_n)$. The vector P is the contact field whose sum of its elements is equal to 1 (like all laws of probability). The field of contact allows the random selection of a message which will be sent from the active cell i if it possesses at least a *having adopted*, to the cell j chosen at random. With a computer, random selection is carried out by a standard pseudo-random function giving a real number p in the interval [0, 1]. Such a selection allows us to choose the affected cell using the Monte Carlo method.

The local transition process takes place in the following manner: every time there is a *having adopted* (that is to say $x_i(t)$ times) in the cell i we proceed to send an adoption message. If the k[th] cell is chosen, the message is "sent" to the cell $j = i + v_k$. (2). This is followed by a new random selection q uniform between 1 and e_j (number of operators in j) if $q \leq x_j$ the message "falls" on an operator that has already adopted it and is therefore lost, if not it arrives on a potential adopter and the number of adopters is raised to 1. The same process occurs for all the cells. At the end of the iteration we update the state x_i , of all the cells i in adding to x_i the number of adopters calculated during the iteration.

The contact field can be defined homogenously, and is therefore constant throughout the area. Certain natural barriers (lakes and forests), which limit contact between the individual cells, are also taken into account. In this case the contact field is variable and must be defined for each cell or a balance of contiguity lines between cells must be taken into account.

12.5.1.2. *Deterministic diffusion model*

This diffusion model can also be treated on a cellular level if the number of individuals of each case is big enough that the large number law applies. The model therefore becomes deterministic.

The messages received in the cell i from its neighboring cells have been sent by each x_k having adopted the k[th] cell towards the $e_i - x_i$ potential adopters of the cell i. Their total is therefore $x_k(e_i - x_i)$. These messages come from boxes more or less unconnected to I, of which we know only a proportion r_k (lessening with distance) are fulfilled. We arrive therefore at a logistic formula of the number of adoptions in i achieved by messages coming from the k[th] cell of the neighborhood:

$$a_{ik} = r_k x_k (e_i - x_i)$$

The number of adopters $a_i(t)$ in i at the time t is given by the summation of the neighborhood terms:

$$a_i(t) = \left(e_i - x_i(t)\right) \sum_{k \in V(i)} r_k x_k(t)$$

and the transition function of the model is therefore:

$$x_i(t+1) = x_i(t) + a_i(t)$$

2 The sign "+" represents the translation applied to i of a vector v_k, which is a sum of two vectors. For example, if $i = (2, 5)$ and $v_k = (-1, 1)$, we will have $j = (2\text{-}1, 5\text{+}1) = (1, 4)$.

The probability contact field P is here replaced by the deterministic contact field $R = (r_1, \ldots r_k, \ldots r_n)$ of the realization rate of a message between a having adopted and an adopter at each step in time.

We present in Figure 12.7 a few results of simple simulations according to the principles defined above, initialized with one focus of innovators in the middle of the area. The first two images concern the diffusion probability: in (A) with the homogenous distribution of individuals (operators), in (B) with the random distribution of individuals. The two following images concern the deterministic diffusion: (C) with the homogenous distribution of individuals and (D) with the random distribution of individuals.

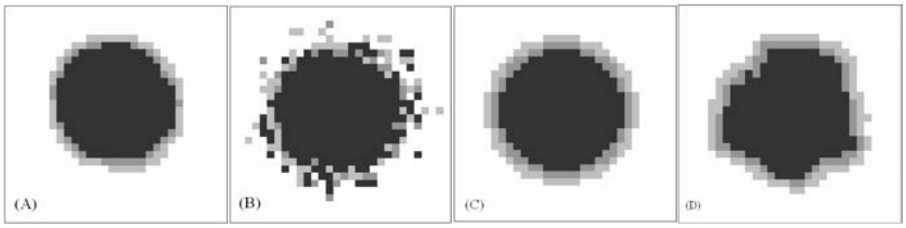


Figure 12.7. *The level of gray shows the rate of adopters at the end of several iterations (in four threshold classes 0%, 25%, 50%, 75%, 100%)*

The diffusion models are not exhausted with these few elementary examples.

12.5.2. *The SpaCelle model*

The SpaCelle model (the Libergeo base of models can be consulted) is a small platform of cellular automaton modeling. Thus the model is not defined in the software itself, nonetheless it functions on a certain number of general principles that we will call a meta-model; it is these principles that we will explain. The user must define the initial configuration, by input or importation of the states of cells along with the transition rules of the model it takes from a knowledge base. It can also define the form of an area, the cell geometry (squared or hexagonal) and the synchronization mode (synchronous or asynchronous or completely random). The state of each cell is qualitative (like a type of land usage) and it is defined by a key word and a representative color.

The performance is based on the *principle of competition.* It manifests itself between the “life force” of a cell and the “environmental forces” emanating from the other cells. When a cell is affected by a new state, we witness the birth of an

individual (cell). It is therefore affected by a maximal life duration (in this state) depending on its class (ID: infinite duration, FD: fixed duration, RD: random duration according to a life expectancy and a standard deviation). Upon its natural death, an individual changes state and takes the definite dead state according to the rule of life of its class. The individual also possesses a life force worth 1 at its birth and decreases linearly to 0 at its natural death. However, an individual can die prematurely if one of the environmental forces affecting it is stronger than its own life force.

For example the rule of life "Pav > Fri = DA(100; 25)" signifies that the class "Pav" (house type) becomes "Fri" (fallow land) after its death and possesses a random life span (RD) according to a life expectancy of 100 and a standard deviation of 25 years.

The environmental forces are defined by transition rules built on the following syntactic model: "$State_1 > State_2 = Expression$". The term "expression" represents a *spatial interaction function* or a combination of these functions. A spatial interaction function is most commonly written in the form F(X; R) and makes it possible to evaluate, for each cell, the "environmental force" owed to the individuals type X in a radius R around the cell. For example, if X is "Ind+Com" this represents the under population of the cell type "industry" or "commerce". R is the radius of the disk defining the neighboring action of X on the cell. The function F represents the type of interaction calculated. There are 20 predefined functions. For example, the function "EV(Ind+Com; 5)" signifies "there exists, at least one individual type 'industry' or 'commerce' in the neighborhood of radius 5" and the function "ZN(Ind; 5)" ZN for zero in the neighborhood (N)) signifies that there is no industry in the neighborhood radius 5.

The phrase: "wild land can become a housing estate if there is already an estate or business area within a radius of 3 and if there is no industry within a radius of 5" translates itself by the following transition rule:

Fri > Pav = EV(Pav + Com; 3) * ZV(Ind; 5)

The conjunction "and" in the phrase is represented by a multiplication sign "*" while an "or" is translated by an addition sign "+".

The knowledge base consists of 3 parts: the definition of the states, the definition of the rules of life (if necessary) and the definition of the transition rules.

Whatever the basis of the defined rule, the mechanism of the model is as follows. For each cell on state s the system executes all the transition rules $R_1, \ldots R_k$ of which the first member $State_1$ equals s. It therefore evaluates the life force f_0 of the cell and

the resulting forces $f_1,\ldots,f_k$ associated respectively with the rules $R_1,\ldots R_k$. It is always the maximum force that takes it. If several rules give the same maximum force, a random uniform selection is performed to choose the transition that will be kept among the cells of maximum force. If it is the life force f_0 that is retained, the cell remains in the state if it is one of the fixes f_i, (for i>0), the cell dies prematurely and its state becomes $state_2$, which is recorded in the right member of the rule R_i.

It can therefore be seen that the transition mechanism is deterministic. Nonetheless a random quantity exists in the case of an equality of maximum force between several rules or in the cells in which the life span is random. There also exists special interaction functions which trigger events, be it a moment selected at random or on a precise date. These chronological functions, by combining with the other functions, allow us to modify the system behavior randomly or from a precise date.

12.5.2.1. *Examples of modeling with SpaCelle*

12.5.2.1.1. The game of life

In the case of a very simple model like the game of life, it is sufficient to define two states (L= life, D= death) no life rule is necessary here and two transition rules are defined according to the formula (1):

D > L = NV(V;1;3)
L > D = SV(V;1;2;3)

The first rule signifies that each dead cell (D) takes life (L) when there are 3 active cells in its neighborhood radius 1 (the function (NV) stands for the number of neighbors). The second rule states that cells stay alive (L) only if the number of its living (L) neighbors is in the interval [2:3]. It is also necessary to specify certain choices not in the rule base like the neighborhood type (here Moore's type is used, 8 neighbors, induced by the max distance), the synchronized performance mode and the form of the squared mesh.

12.5.2.1.2. The Schelling segregation model

This model, typical of the sociology method of *methodological individualism* which uses three fundamental concepts: the concept of emergence, associated most often with aggregation, the concept of modeling, allowing the simplification of individual behavior, which in reality are all different, and finally the rationality concept of the actors who translate a hypothesis of behavioral intelligibility.

In a town made up of several social groups or communities (ethnic, religious, economic, etc.) the Schelling model shows how spatial segregation can appear

without segregational behaviors at the individual level. Indeed, it shows that even if each individual has an elevated level of tolerance concerning the presence of "foreigners to their group" in their neighborhood we nonetheless see a separation or socio-spatial segregation emerging over time which transforms by the appearance of considerably more homogenous areas than individual tolerance may have lead us to believe. It is therefore a simple yet significant example of the emergence concept in a complex system. Even if the reality is very different, this model still shows that the whole, that is to say the collective behavior must not be directly interpreted as if the rules of individual behavior applied directly to collective behavior: the individual is not segregationist therefore the group is not either. It can be seen, through this example that an individual rule can produce, if certain conditions are brought together (in this case for example a sufficiently high population density and rules given for the highest level of toleration) an overall organized behavior of which the occurrence is certain at the end of the given time but of which the form is totally random.

We have developed this model in SpaCelle as follows. The cellular area represents a town where the cells (10,000 in number) represent the habitations. The town is composed here of three communities noted A (in black), B (in dark gray) and C (in light gray). When a house is not inhabited, the cell is in the state F (free) and left in white. There are consequently four possible states for a cell: A, B, C or F.

The initial configuration results in a random selection of a state for each cell among the four possible states. This gives a quasi-equal weight between the three communities.

The rules of transition are very simple, there are no life span rules and there are two transition rules, identical for each community:

– The moving-in rule; if a cell is free, a family from any one of the three communities A, B or C has an equal chance of moving in. The installation of a family is not linked to the freeing of another cell in such a way that it lowers the overall population density (function DE) as the model possesses a maximum overall population density (99% for example) over which moving in is no longer possible. There are therefore three identical moving in rules, one for each population. As a result, the probability of the installation of an individual is the same whatever the group:

```
L>A=DE(A+B+C; 0; 0.99)
L>B=DE(A+B+C; 0; 0.99)
L>C=DE(A+B+C; 0; 0.99)
```

– The moving-out rule; if a family living in a given cell is surrounded by too many "foreigners to its group" it moves, freeing the cell (the PN function is used for

this which calculates the proportion of foreigners in the neighborhood radius 5. It is worth 1 if it is in the interval (70%–100%) and if not sends back 0). The move out is not directly linked to a move in but decreases the overall density and eventually allows a move in elsewhere. Therefore, move out is explained by a rule which has the same form for each population:

A>L=PN(B+C; 3; 0.7; 1)
B>L=PN(A+C; 3; 0.7; 1)
C>L=PN(A+B; 3; 0.7; 1)

This model is slightly different to the original Schelling model [SHE 80]. In fact that model dealt with only two different types of individuals (white and black), moreover it processed in a synchronous manner (all cells changing at the same time) and finally each move-out was immediately followed by a relocation elsewhere. Here we have taken three different populations (but this changes nothing in principle), the order of simulation is random, that is to say at each moment in time a cell is chosen at random to be independently treated from the cells already done. Finally, in our simulation the two types of action (moving-in and moving-out) are independent and are chosen uniquely in relation to the state of the cell which is chosen. According to the evaluation of the rules for this cell, if it is a free cell moving-in can occur. The behavior of the model is therefore subject to a minimum number of rules to create the dynamic of the system.

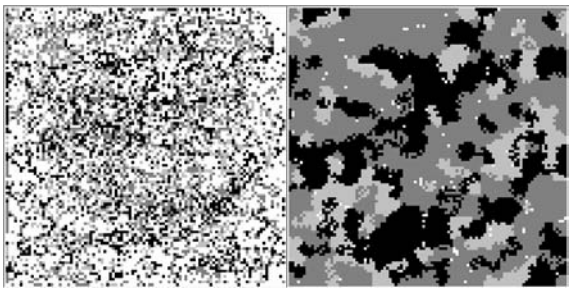


Figure 12.8. *Simulation of the Schelling model: (1) initial configuration, (2) after 50 steps in time*

12.5.2.1.3. Interpretation

We soon notice the emergence of an organization stabilizing after 50 time steps (each step corresponds to the processing of 10,000 cells chosen at random). The tolerance percentage 70% corresponds to a tolerance percentage more important than the average as if the 3 populations were equal in proportion ($^1/_3$ each) there are $^2/_3$ of 99% in the mean, 66% of cells which are made up of "foreigners". The random

situation of departing positions means that locally (within a neighborhood radius of 5) the probability of reaching or surpassing 70% of foreigners is quite high. The moving-out occurs quite often. The box left free will only be able to stabilize itself with one of the other two populations which will progressively reinforce the homogeneity. We notice then a quite complex final result, where homogenous "areas" appear, composed of one single group whereas others are composed of a mix of two groups, but none appear perfectly mixed, a situation which does not however seem forbidden by the rules.

12.5.2.1.4. Urban development in Rouen over 50 years

In [DGL 03] we used this system to simulate the urban development of the agglomeration of Rouen over a 50-year period. This work is more realistic, it begins with a real observation situation in 1954 and with a set of 15 rules it reaches a simulated situation in 1994, which is compared to the present day. Overall the configuration is very close to the observed situation. The analysis of local differences underlines the behavior which is spatially coherent, others underlining the logical evidence outside the space.

12.5.2.2. *The limits and originality of the SpaCelle model*

This model only allows dynamic modeling whose rules of evolution are spatial (interactions with the neighborhood). It is therefore simplistic but it does allow a complex modeling from a not necessarily mathematical knowledge. For example, the rules set can be constructed from the analysis of a text. However, quantitative data cannot be introduced except in declaring these values in qualitative classes (this is a finite state automaton). Moreover an economic variable like the price of land cannot be taken into consideration at the same time as land use became the model takes only one type of information. For example, if the price of land is separated into different classes the land use types can no longer be used, unless the two are mixed in a quite complex way. Nonetheless the model allows us to realize experiments by simulation, which gives quite good results that will not be discussed here. The Rouen model tends to show for example that the cost of land results from localization and spatial interaction between these localizations since it is not necessary for modeling urban evolution. However, other exogenous factors, economic or social, cannot be taken into account. We can therefore more or less analyze the influence by the analysis of the differences between the model and reality.

This system, which allows the formalization of a dynamic by *sentences* (the rules), explained in a *knowledge-representation language,* quite close to natural language, is an original alternative to classic modeling which explains a system dynamic by equations.

12.5.2.3. *Recent evolutions of the SpaCelle model*

We have developed more general versions of this model where the cells can be of any polygonal form and thus can adapt directly to the cuttings originating from a geographical database. We have also generalized the types of cell state. Instead of defining the state of one cell with a unique exclusive quality, it is more realistic to define a state of multiple behavior. For example, land use: housing but with some business and a small amount of industry and roads.

In this extension, the cell state is represented by a series of n real numbers $s = (s_1, s_2, ..., s_i, ...s_n)$, which can have very different meanings:

1) The state s can be a vector of dimension n where each dimension i of the state associated with a modality (for example 1: housing, 2: industry, 3: business, 4: road, etc.) of the same qualitative variable (here the land use) and s_i represents the proportion or probability of presence (depending on whether it is in a deterministic or probability situation) of the modality i in the cell.

2) The state s can be composed of n different quantitative variables (for example 1: population, 2: GDP, 3: surface, etc.) and s_i is the value of the i^{th} variable. We have used as part of the modeling of the development of the standard of regional life in the European Union (represented in a simplified manner by the GDP per inhabitant in purchasing power parity.) In using the rules of intrinsic rural growth but also of diffusion by neighborhood (known as horizontal interaction), we are thus underlining a competition between the two processes, the first pushing for divergence the second for convergence, of the standards of living between the regions. We search then to understand the influence of national and European aid, as well as the role of taxation (vertical interactions, rising for taxes and falling for help). This introduces a supplementary level of complexity to the studied system, both in the structure, through the necessity to use a multi-layered hierarchical automaton (regions, states, Europe) and through the dynamics, combining the horizontal and the vertical interactions.

The state s can also be a real matrix $p \times p$. We have modeled within the framework of diffusion in the behavior of French voters, using cantonal cutting as cellular meshing [BL 04] that state of the call is therefore defined by a behavior matrix $S = [s_{ij}]$, of $p \times p$ dimension where p is the number of candidates. The behavior matrix of a canton-cell evolves over time (between two elections) by diffusion of voting behavior from the polls and ultimately makes it possible to calculate a new vote vector V' (percentage of votes for different candidates) from the initial vector (observed) V by matrix multiplication $V' = SV$. A rule bearing, for example, a positive relative influence to a candidate i modifies the behavior matrix, in raising the term s_{ii} of the matrix and lowering the other terms of the column i so that the total remains constant (equal to 1).

12.5.3. *Simulation of surface runoff: RuiCells model*

This model, much more complex than the previous, was developed as a response to a concern over the understandings of intense phenomena of surface runoff which regularly provoke catastrophic damage in the form of mudslides in normally dry drainage basins. A more precise description can be found in [LAN 02] (the base of Libergo models can also be consulted).

12.5.3.1. *Inputs*

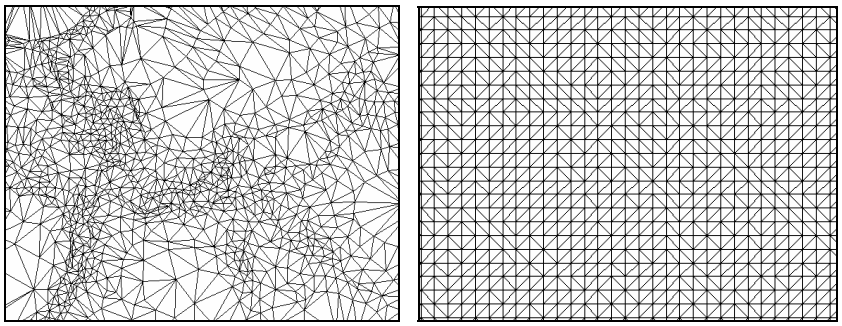
A certain amount of data coming from a geographical information system (GIS) is taken on input to the software to construct a part of the automaton structure, essentially the digital elevation model (DEM). The others are used for performance: precipitation table, vector card of land use, images, etc. These different inputs can be geometrically harmonized.

12.5.3.2. *Structure*

The model of this cellular automaton differs from the strict definition given earlier in the first part. In fact, the cells here are "drawn" on the DEM which represents an elevated surface. This surface is first meshed according to triangulation. If the DEM is composed of irregular random points (originating from, for example, a digitization of level curves) we construct a Delaunay[3] triangulation associated with these points (Figure 12.9a). If the DEM is a regular grid of points, we cut each square into two triangles, by choosing the lowest diagonal (in altitude) so as not to introduce artificial barriers to the flow (Figure 12.9b).

The cells are constructed from the surface triangle mesh. The triangles constitute the first type of cell, the surface cell. However, this is not sufficient to be able to suitably model the flow that naturally concentrates along the lines of the bottom of the valley (the thalweg). It is also therefore necessary to introduce linear cells which are the sides of triangles as they actively participate in the flow process. Finally, in diverse areas, a summit, a side, a triangle, even a group of these objects constitutes a minimal local altitude, a basin. We define therefore a punctual cell, which represents this local minimum. A specific punctual cell recovers also all that runs outside of the domain, in order to conserve the total volume of water of the simulation. These different cells are structured geometrically and topologically in such a way that each "knows" its surface neighbors.

3 The Delaunay triangulation is that of the interior of the circle circumscribed to each of its triangles contains no summit of triangulation.



(a) *(b)*

Figure 12.9. *Two modes of triangulation*

So that the surface runoff process can be modeled it is necessary to define within the cells a directed *flow graph* which indicates for each, in which other cell(s) it flows into and if the cell flows into several others, a sharing co-efficient of flow between the downstream cells needs to be defined, which is calculated through the cell geometry (form and incline).

When the flow arrives in the local minimum it is necessary to calculate the replenishment of the basin and define the spillway point and the receptor cell so that the flow graph is not interrupted. The graph must also be correctly directed in the horizontal zones towards the output zones without making loops. The algorithmic definition of this graph is a delicate part of the model. The cellular structure is routinely composed of many thousands or hundreds of thousand cells.

12.5.3.3. *Functioning*

The local functioning of each cell is operated by a "cellular motor" which is a hydrological model of surface runoff based on the discretization of differential equations in finite-difference equations (the user interface allows us to choose between several motors). To calculate the runoff it takes into account, for each period of time Δt, the volume of rainfall, the volume of water already present and the volume arriving from the upstream water cells, as well as possibly the water loss and infiltration. These variables makes it possible to calculate the volume leaving during functioning compared to the flow speed which depends itself on the slope and height of the water present.

The overall functioning of the automaton manages synchronously the circulation flow of water between the cells; the structure of the neighborhood used here being defined by the flow graph. At each iteration, which corresponds to a moment in time Δt, the automaton proceeds in two phases:

– the communication phases where the outputs (calculated before) are communicated to the inputs of the cells downstream,

– the transition phase where each cell calculates its new state $x(t+\Delta t)$ which is the new volume of water in stock and its new output $b(t+\Delta t)$ which is the volume running off downstream in function to its previous state $x(t)$ and its input $a(t)$ which is the volume coming from upstream and precipitation.

12.5.3.4. *Outputs*

Software can chart the evolution of variables in time, to draw out diverse charts and graphs like hydrographs of measured points defined by the user and precipitation curves. It also allows the calculation of the shape and surface of a basin, helps us to draw level curves and bigger trends, create charts with shadowing, or represent a basin in 3D.

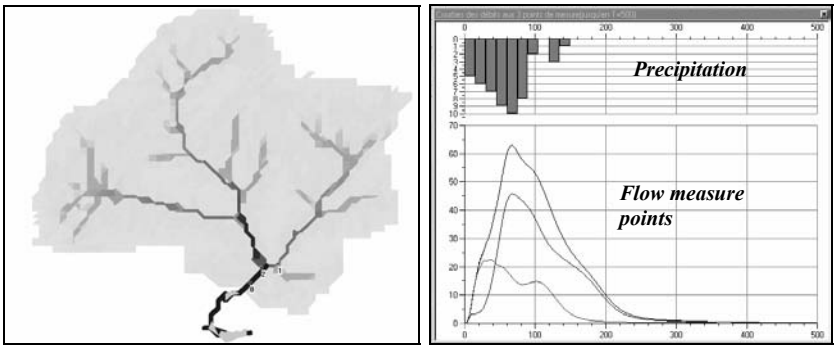


Figure 12.10. *Chart and hydrographs in different measuring points of a drainage basin*

12.6. Bibliography

[BER 82] E. BERLEKAMP, J. CONWAY, G. RICHARD, *Winning Ways*, vol. 2, New York: Academic Press, chap. 25, 1982.

[BUS 04] M. BUSSI, P. LANGLOIS, E. DAUDÉ, "Modéliser la diffusion spatiale de l'extrème droite: une experimentation sur le front national en France", *Colloque de l'AFSP*, Paris, 19 p., 2004, (see http://www.afsp.msh-paris.fr/activite/diversafsp/collassp04/collafspassp0904.html).

[CON 70] M. GARDNER, "Mathematical Games: The fantastic combinations of John Conway's new solitaire game 'life'", *Scientific American*, pp. 120-123, October 1970.

[DAU 04] E. DAUDÉ, "Apports de la simulation multi-agents à l'étude des processus de diffusion", *Cybergeo*, no. 255, 12 p., 2004.

[DUB 03] E. DUBOS-PAILLARD, Y. GUERMOND, P. LANGLOIS, "Analyse de l'évolution urbaine par automate cellulaire. Le modèle SpaCelle", *L'espace géographique*, pp. 357-380, vol. 32, no. 4, 2003.

[FAT 01] N. FATES, *Les automates cellulaires : vers une nouvelle épistémologie?*, mémoire de DEA, Paris I- Sorbonne, 2001.

[HÄG 67] T. HÄGERSTRAND, *Innovation Diffusion as a Spatial Process*, University of Chicago Press, 1967, Chicago and London.

[HEB 02] J. HEBENSTREIT, Principe de la cybernétique, in *Encyclopaedia Universalis* version 8, 2002.

[LAN 97] A. LANGLOIS, M. PHIPPS, *Automates cellulaires, application à la simulation urbaine*, 197 p., Paris, Hermes, 1997.

[LAN 02] P. LANGLOIS, D. DELAHAYE, "RuiCells, automate cellulaire pour la simulation du ruissellement de surface", *Revue Internationale de Géomatique*, pp. 461-487, vol. 12, no. 4, 2002.

[MAR 01] B. MARTIN, II, Automates cellulaires, information et chaos, PhD thesis, École Normale Supérieure de Lyon, 2001.

[OLL 02] N. OLLINGER, Automates cellulaires: structures, thesis, Lyon, 2002.

[POU 85] W. POUNDSTONE, *The Recursive Universe*, Oxford University Press, 1985.

[SHE 80] T. SCHELLING, "Micromotives and macrobehavior", Toronto, Norton, 1978. French edition in *La tyrannie des petites décisions*, Paris, PUF, 1980.

[VNM 44] J. VON NEUMANN, O. MORGENSTERN, *Game Theory and Economic Behavior*, Princeton University Press, 1944.

[WOL 83] S. WOLFRAM, "Statistical Mechanics of Cellular Automata", *Review of Modern Physics* 55, pp. 601-644, 1983.

[WOL 86] S. WOLFRAM, *Theory and Applications of Cellular Automata,* World Scientific, 1986.

[WOL 02] S. WOLFRAM, *A New Kind of Science*, Wolfram Media, 2002.

12.7. Websites

GDR Libergéo, Groupe modélisation, base de modèles: http://www.spatial-modelling.info.

CASA: Centre For Advanced Spatial Analysis (UCL): http://www.casa.ucl.ac.uk.

University of Utah (Paul Torrens): http://www.geosimulation.org/geosim.

Santa Fe institute: http://www.santafe.edu/projects/evca/.