COMPLEXITY INTERNATIONAL

Volume 7
http://www.complexity.org.au/

Representation of some cellular automata by means of equivalent L Systems

Manuel Alfonseca, Alfonso Ortega

Universidad Autónoma de Madrid,
Campus de Cantoblanco, 28049
Madrid, Spain
Email: {Manuel.Alfonseca;Alfonso.Ortega}@ii.uam.es

Abstract

This paper presents some facts that make cellular automata parallel to L Systems. Several examples of cellular automata are considered and equivalent L Systems are constructed. A general procedure should be able to manage n-dimensional cellular automata, thus one-dimensional, two-dimensional and three-dimensional cellular automata are studied here. Some recommendations to generalise the techniques to n-dimensional automata are given. The paper comments on the difficulties to develop a general procedure and the reasons to give several recommendations instead of an algorithm.

1. Introduction

1.1 L Systems

In 1968, Aristid Lindenmayer [1] defined a new type of grammar (a parallel derivation grammar), which differs from the normal Chomsky grammars (sequential derivation grammars) because the rules are applied simultaneously, rather than one at a time.

Lindenmayer derivation grammars, also called L systems, can be classified in different ways:

- Context sensitive (IL systems) versus context free (0L systems).
- Deterministic (DL systems) versus non-deterministic.
- Propagative (PL systems) versus non-propagative.
- EL systems, with extensions.
- TL systems, with tables, where the set of production rules includes two or more complete subsets of production rules, that will be applied alternatively in each derivation.

Received: 22/06/1999
Accepted: 11/02/2000

These types may be combined: A D0L system is a deterministic context free system; a PD0L system is propagative, deterministic and context free; an EIL system is context sensitive with extensions...

A D0L system is the three-fold (Σ, P, ω), where Σ is an alphabet (a finite non-empty set of symbols); P is a set of production rules of the form A::=x (where A∈ Σ is a symbol in the alphabet and $x \in \Sigma^*$ is a word or string of symbols in the alphabet); and $\omega \in \Sigma^*$ is the starting word or axiom. Every symbol appears exactly once at the left of a production rule (this makes the system deterministic). This is the basic and simplest case of an L System. In the paper, examples of other types of L Systems are also shown.

An example of a D0L system is:

({F,+,-}, P, F++F++F)

where P is the following set of production rules:

F ::= F-F++F-F

+ ::= +

- ::= -

A derivation of a word in a D0L system is the new word obtained when each symbol in the first word is replaced by the right part of the production rule whose left part is that symbol. In the previous example, we can get the following derivation from the axiom:

$$F++F++F \rightarrow F\text{-}F++F\text{-}F++F\text{-}F++F\text{-}F++F\text{-}F++F\text{-}F$$

The word obtained becomes the starting point of a new derivation, and so on.

L systems have been successfully applied to the simulation of biologic processes, such as plant growth, leaf development, pigmentation of snail shells, etc. They are also appropriate to represent fractal objects obtained by means of recursive transformations [2-6]. The initiator maps to the axiom of the L system, the generator to the production rules, and the recursive applications of the generator to the initiator correspond to the successive derivations of the axiom.

N dimensional L Systems are a subset of IL Systems:

- Strings derived by n dimensional L Systems are n dimensional arrays.
- The size of the axiom is the number of elements in the n dimensional array, i.e. the product of the maximum number of elements in each dimension.
- The size of every word derived from an n dimensional L System is equal to the size of the axiom.
- In order to define the context, a neighbourhood rule is needed, such as:
- The von Neumann neighbourhood of the node at position (i,j) is the set {(i,j-1), (i,j+1), (i-1,j), (i+1,j)}
- The Moore Neighbourhood of the node at position (i,j) is the set {(i-1,j-1), (i-1,j), (i-1,j+1), (i,j-1), (i,j+1), (i+1,j-1), (i+1,j), (i+1,j+1))}

1.2 Cellular Automata

A *cellular automaton* [7-10] is defined as the six-fold (G,G_0,N,Q,f,T), where G is a matrix of automata, G_0 is the initial state of the grid (a mapping from G in Q, an injective function that assigns an initial state to each automaton in the grid), N (neighbourhood) is a function that assigns to each automaton in the grid the set of its neighbours, Q is the set of possible states of every automaton in the grid, f is the transition mapping from QxQ^n in Q which defines the next state of each automaton in the grid given its current state and the states of its n neighbours, and T is the set of final or target states.

Every automaton in the grid has the same number of neighbours, transition mapping and set of possible and final states [11], but they may differ from other automata in the grid only in their initial states. The input to the automaton associated to a given point in the grid is the set of states of the automata associated to its neighbours.

Cellular automata may differ in the following:

- The shape and size of the grid, usually square, rectangular or triangular, which may be infinite.
- The definition of the set of neighbours to a given grid point.
- The actual finite automaton associated to each point in the grid. If this automaton is deterministic/probabilistic, the cellular automaton is deterministic/probabilistic.
- The set of initial states of all the automata.

In cellular automata, specially those that use an infinite grid, the set of states of the finite automaton associated to the grid points usually includes a special symbol (the *empty* state). The number of automata not initially in the empty state is assumed to be finite.

2. L Systems and cellular automata

The possibility of generating L Systems equivalent to given cellular automata is suggested by the identification of several similarities between both systems:

- Both have initial information that can be considered as their starting state.
  - For an L system, the initial string (the axiom).
  - For a cellular automaton, the set of all the initial states of its finite automata, which can be considered the initial state of the cellular automaton.
- Both have components that record the way the system changes:
  - For an L system, the set of production rules.
  - For a cellular automaton, the transition function of its finite automata.
- Both architectures generate the next state by applying the transformation to every component in parallel. The L System changes each symbol of the current string, the cellular automaton changes the state of each automaton in the grid.

These similarities indicate that it should be possible to find a convergence between cellular automata and L Systems.

This is not the first attempt in this direction. In reference [12], the same computing procedure (genetic programming) is applied to both L systems and cellular automata, showing that there must be a structural relationship between both, although no attempt is done to

convert from one representation form to the other, and the examples in the two domains are different. The equivalence problem we are describing here is not raised.

In more recent references [13-15], Stauffer and Sipper do tackle the question of L systems and cellular automata equivalence, although in a different context to the one we are presenting here. On the first hand, they are mainly interested in self-replicating automata. Secondly, they convert L systems into equivalent cellular automata, a movement in the opposite direction to ours. Finally, their cellular automata are not really equivalent to an L system, but to the graphical interpretation of the latter using a turtle graphics formalism.

This paper shows several applications of a procedure that makes it possible to generate L systems equivalent to given cellular automata without the need to introduce a graphical representation. As examples, we have chosen three cellular automata with different dimensionality, to show that our procedure is applicable in quite different situations.

3. One-dimensional cellular automata

A one-dimensional cellular automaton is a linear chain of automata. The neighbourhood relationship in one-dimensional automata consists of predecessors and successors. One of the better-studied neighbourhoods for one-dimensional automata consists of the automaton itself and its two nearest neighbours.

3.1 One-dimensional cellular automaton with three inputs that generates the Sierpinski gasket

Let us consider finite automata whose state is a member of the {0, 1} set. The new state of each automaton is a function of its own state and that of its two immediate neighbours, left and right, which we will call its predecessor and successor, respectively. This family of automata can be defined by means of three bits that represent the state of the neighbour to the left, the state of the automaton itself, and the state of the neighbour to the right. Thus there are 2^3=8 different input values and 2^8=256 possible state change rules. The transition function of the automata can thus be encoded in decimal notation by a number from 0 to 255, which represents the eight new state bits corresponding to the eight input configurations. For example, function 90, with the binary notation 01011010, has the following outputs:

Table 1.

State of previous automaton	State of this automaton	State of following automaton	New state of this automaton
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	0

It is easy to devise an (1,1) DIL System whose derived words correspond to the consecutive generations of this automaton:

The alphabet is the set $V_0=\{0,1\}$.

The set of production rules P can be directly obtained from the tables as above. For the example of function 90, the following set will be used:

$$P_{90}=\{111::=0,\ 110::=1,\ 101::=0,\ 100::=1,\ 011::=1,\ 010::=0,\ 001::=1,\ 000::=0\}$$

The L system for this example is $S_{90}=\{V_0, P_{90}, \alpha_0\}$, where the axiom α_0 is the binary string that represents the initial configuration of the cellular automata. Let $\alpha_0 = 0^{23}10^{23}$ be the axiom. Its first 24 derivations are shown below for the above system:

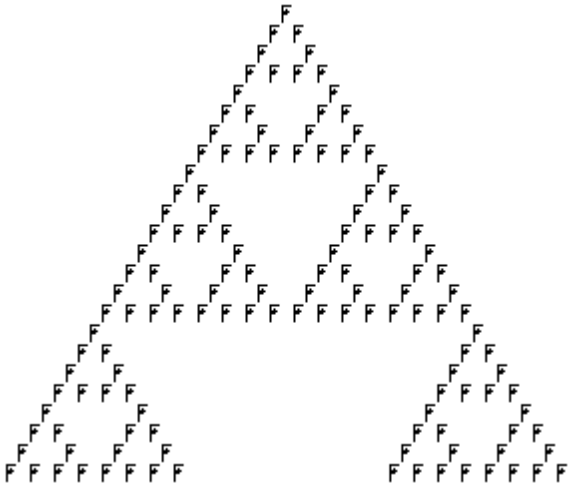

Figure 1.

4. Two-dimensional cellular automata

Two-dimensional cellular automata use two-dimensional grids. The shape and the size are not fixed, but they are usually rectangular and possibly infinite. One of the best known examples is John Conway's game of life, which is based on a very simple cellular automaton that uses a rectangular potentially infinite grid and a set of two possible states for the automata. Its conversion to an equivalent IL system is trivial, and we will not consider it here.

4.1 A cellular automaton simulating an ecosystem

The cellular automaton we shall consider shows the following characteristics: the grid is rectangular and possibly infinite; the only neighbour of each cell is the cell itself; a finite automaton is associated to each cell of the grid. The state of these automata represents a combination of individuals for which several conditions hold:

- There are two kinds of individuals: prey and predator.
- Each cell can contain up to four prey individuals.
- There are two possible states for every predator, a and b.
- Each cell can contain up to four a predators and up to four b predators.
- Thus, the maximum number of individuals allowed in a cell is twelve.

The state of each cell changes in two alternative steps:

1. Predation and reproduction takes effect according to the following rules:
   a) A predator in the a state dies if there is no prey in the same cell.
   b) A predator in the a state goes into the b state if there are at least two prey individuals in the same cell and there is room for a predator in the state b in the cell. If this is the case, one of the prey individuals dies (is eaten).
   c) A predator in the b state goes into the a state if there is no prey in the same cell.
   d) A predator in the b state becomes two predators in the a state (reproduces) if there are at least two prey individuals in the same cell and there is room for the two predators in the a state in the cell. If this is the case, one of the prey individuals dies (is eaten).
   e) The prey reproduces if there are at least two and at most three individuals in the same cell.

Rules are applied successively in the order shown above. The condition for application is tested on the initial state. The available space and the number of individuals are tested on the current situation.

2. Movement takes effect using a Von Neumann neighbourhood. The goal is to simulate some kind of non-deterministic movement of each individual. The following rule is stated:
   a) Each individual can change its direction by choosing at random one of the four possible ways: north, south, east and west. Only one individual per species and state can go in the same direction.

We shall represent both steps by means of two different cellular automata that execute alternatively, and convert each of them into an equivalent L system. The combined automaton will be represented by an L system with tables that combines both. The behaviour of this cellular automaton displays many of the properties of real ecological systems, such as Volterra-type oscillations [16-17].

4.1.1 Predation and Reproduction

We shall represent the two predator states by the letters a and b, the prey by an x. The state of a cell will be $a^{\#a}\ b^{\#b}\ x^{\#x}$ where

- #a is the number of predators in the a state in the cell.
- #b is the number of predators in the b state in the cell.
- #x is the number of prey in the cell.

The transition rules can be expressed by two strings of symbols: the individual states and the result of the change:

a) Rule 1: $ax^0 ::= a^0$ (x^0 and a^0 mean that there are no x or a individuals in the cell).

b) Rule 2: $ax^2 ::= bx$

c) Rule 3: $bx^0 ::= a$

d) Rule 4: $bx^2 ::= a^2x$

e) Rule 5: $x^2 ::= x^3$

As a first example, let us consider the state $a^4b^4x^4$.

- Rule 2 (ax^2 ::= bx) is applicable four times because there are four individuals of type a and four (at least two) individuals of type x. Nevertheless, there is no room for four new b individuals. So rule 2 is not applied and the number of a and b individuals remains unchanged.
- The same happens with rules 4 and 5.

Second example: A cell with three preys, one predator in the a state and one predator in the b state: $a^1b^1x^3$.The following rules will be applied:

- Rule 2 (there are at least two x and one a in the cell.) In the initial state, this rule is applicable once, because there is only one a individual. Rule 2 needs room for a new b individual. Since there is currently only one, this modification is possible. Rule 2 removes one a and one x, resulting in $a^0b^2x^2$.
- Rule 4 (there is one b predator and more than one prey in the cell.) This rule is applicable once, because in the initial state there is only one b individual. It needs space for two new a individuals. Currently there are no individuals of type a. The change is therefore possible. Rule 4 removes one b and one x, resulting in $a^2b^1x^1$.
- Rule 5 (prey reproduction.) This rule is applicable once because in the initial state there is a pair of x. Space for a new x is required. In the current situation there is only one individual of type x. The change is then possible, resulting in $a^2b^1x^2$.

Third example: A cell with three preys and three predators in the a state: $a^3b^0x^3$.The following rules will be applied:

- Rule 2 (there are at least two x and one a in the cell.) This rule is applicable three times because there are 3 a individuals. The first application of rule 2 needs room for a new b. There is none, so this modification is possible. Rule 2 removes one a and one x, resulting in $a^2b^1x^2$. The second application of rule 2 is also possible, resulting in $a^1b^2x^1$. Finally, the last application of rule 2 is also possible, resulting in $a^0b^3x^0$.
- Rule 5 (prey reproduction). This rule is applicable once in the initial state, because there is a pair of prey individuals. In the current situation there are no x, so the space requirements apply. The change is then possible, resulting in $a^0b^3x^1$.

4.1.2 An equivalent L System

It is possible to devise a two-dimensional L System whose derived words correspond to the consecutive generations of this automaton. A single symbol will be associated to each cell, rather than a string with a symbol per individual. The rules are coded with a higher degree of abstraction: instead of representing each rule as it is applied, the cell transition is treated as a whole. A production rule is used to explicitly record each possible change.

The state of a cell will be represented by the letter s with the exponents of the string used in the cellular automaton to represent the state as a sub-index. Thus:

- the state whose string is $a^1b^1x^3$ is represented as s_{113}
- the state whose string is $a^2b^1x^2$ is represented as s_{212}

Conversely

- Symbol s_{031} stands for the state with string $a^0b^3x^1$
- Symbols s_{215}, s_{714}, s_{298} represent no valid state.

In the previous examples we saw that we go from state $a^1b^1x^3$ into state $a^2b^1x^2$, so the following rule must belong to the set of rules of the 2 dimensional L system:

$$s_{113} ::= s_{212}$$

The total number of symbols equals the number of variations with repetitions of five elements ({0,1,2,3,4}) taken 3 at a time, that is, $5^3 = 125$.

The complete set of production rules is shown below:

```
P={         s314::=s232, s444::=s444, s434::=s344, s424::=s433, s414::=s332, s404::=s042, s344::=s344, s334::=s434,
s324::=s333, s314::=s232, s304::=s033, s244::=s434, s234::=s334, s224::=s422, s214::=s223, s204::=s024, s144::=s334,
s134::=s423, s124::=s413, s114::=s214, s104::=s014, s044::=s424, s034::=s414, s024::=s404, s014::=s204, s004::=s004,
s443::=s444, s433::=s343, s423::=s431, s413::=s141, s403::=s131, s343::=s344, s333::=s432, s323::=s331, s313::=s041,
s303::=s031, s243::=s433, s233::=s332, s223::=s231, s213::=s221, s203::=s022, s143::=s333, s133::=s421, s123::=s411,
s113::=s212, s103::=s013, s043::=s422, s033::=s412, s023::=s402, s013::=s203, s003::=s004, s442::=s443, s432::=s342,
s422::=s241, s412::=s231, s402::=s221, s342::=s343, s332::=s431, s322::=s141, s312::=s131, s302::=s121, s242::=s432,
s232::=s331, s222::=s041, s212::=s031, s202::=s021, s142::=s332, s132::=s231, s122::=s221, s112::=s211, s102::=s012,
s042::=s421, s032::=s411, s022::=s401, s012::=s202, s002::=s003, s441::=s441, s431::=s431, s421::=s421, s411::=s411,
s401::=s401, s341::=s341, s331::=s331, s321::=s321, s311::=s311, s301::=s301, s241::=s241, s231::=s231, s221::=s221,
s211::=s211, s201::=s201, s141::=s141, s131::=s131, s121::=s121, s111::=s111, s101::=s101, s041::=s041, s031::=s031,
s021::=s021, s011::=s011, s001::=s001, s440::=s400, s430::=s300, s420::=s200, s410::=s100, s400::=s000, s340::=s400,
s330::=s300, s320::=s200, s310::=s100, s300::=s000, s240::=s400, s230::=s300, s220::=s200, s210::=s100, s200::=s000,
s140::=s400, s130::=s300, s120::=s200, s110::=s100, s100::=s000, s040::=s400, s030::=s300, s020::=s200, s010::=s100,
s000::=s000 }
```

The equivalent L system is $(\{S_i\}^{125}_{i=1}, P, \alpha_l)$. The axiom ($\alpha_l$) is the matrix of the initial states of all the automata in the grid, translated by means of the following function, which converts a state of the finite automata into a symbol of the L system:

$$f(a^{\#a}\ b^{\#b}\ x^{\#x}) = s_{\#a\ \#b\ \#x}$$

This equivalent L system has been used to improve significantly the performance of our simulation of the cellular automaton described here.

4.1.3 Movement

In this step, each individual will choose a destination at random. To distinguish each individual by the destination it has decided to follow, we shall use the symbols in the set { ↑ , → , ↓ , ← }. The population of individuals in a cell is described by means of their symbol followed by a string of symbols taken from the set { ↑ , → , ↓ , ← }, showing the direction that the individuals will follow. The symbols are always written in the order shown in the set. For example, if there are two a individuals, one of them going to the south and the other to the west; three b individuals, none of them going north, and four x individuals, the situation will be represented thus:

$$a\ \downarrow\ \leftarrow\ b\ \rightarrow\ \downarrow\ \leftarrow\ x\ \uparrow\ \rightarrow\ \downarrow\ \leftarrow$$

Two phases are needed to simulate the movement rule: first each individual chooses at random the direction it will follow, then each individual reaches its destination. In the preceding example, we look at the two a individuals first. They can choose at random one between the following options:

$$\{\ a\uparrow a\leftarrow,\ a\rightarrow a\leftarrow,\ a\downarrow a\leftarrow,\ a\uparrow a\rightarrow,\ a\uparrow a\downarrow,\ a\rightarrow a\downarrow\ \}$$

The six options listed above are the only possible options, i.e. any other combination of arrows is forbidden. The same could be done with b and x. After choosing their destination, the following string could be obtained:

$$a \uparrow \leftarrow b \uparrow \rightarrow \downarrow x \uparrow \rightarrow \downarrow \leftarrow$$

Once the direction of movement is chosen, each individual follows it. Let us look at an example. The following initial situation is considered:

Table 2.

	a → ↓ b → ↓ x ↓	
a → ↓ b →	a ↑ ← b ↑ → ↓ x ↑ → ↓ ←	a ↓ ← x→ ↓
	a ↑ → b → x →	

We compute the next state of the central cell, which does not depend on the previous state of the same cell, but only on the direction of the individuals in the neighbouring cells that point to the central cell. In the example, there are four a, two b and one x in this situation. The maximum number of individuals in the same state allowed in a cell is four, and four is also the maximum number of different destinations. This avoids collisions of too many individuals in the same destination. The following string represents the next state for the central cell:

$$a \uparrow \rightarrow \downarrow \leftarrow b \rightarrow \downarrow x \downarrow$$

4.1.4 An equivalent L System

It is possible to devise a two-dimensional L System whose derived words correspond to the consecutive generations of this automaton. It is advisable to use a more readable way of naming each possibility. Assume that the set of directions are ordered as follows:

$$\{ \uparrow , \rightarrow , \downarrow , \leftarrow \}$$

The set of directions for a given symbol in a cell can be represented by a four digit binary number as in the following examples:

- 1111 means that there is one individual per direction.
- 0101 means that there is one individual going east and one going west.
- 1000 means that there is only one individual that has chosen north.

The different types of individuals will be described in the following order: {a, b, x}. Thus, $s_{1010,1111,0001}$ stands for $a \uparrow \downarrow b \uparrow \rightarrow \downarrow \leftarrow x \leftarrow$.

As in the cellular automaton, there are two phases:

- First, each individual chooses at random its next destination.
- Second, each individual moves to its destination.

These two phases can be expressed by means of two different tables of production rules. The first has an enormous number of possible combinations, which we will simplify by giving an algorithm to generate the right-hand side of each rule as a function of its left-hand side. This phase is independent of the von Neumann neighbourhood, which is used in the second phase.

Example: Let $s_{1010,1111,0001}$ be the symbol studied. The associated set of production rules is:

$$\{ s_{1010,1111,0001} ::= s_{n1,1111,n3} \mid \text{where n1 is any permutation of 1010 and n3 is any permutation of 0001} \}$$

This set contains 24 rules.

In general:

$$P_2=\{ s_{na,nb,nx} ::= s_{na',nb',nx'} \; \forall \; s_{na,nb,nx} \in V, \text{ where na' is a random permutation of na, nb' is a random permutation of nb, and nx' is a random permutation of nx} \}$$

Obviously, P_2 is non-deterministic because there are a lot of possible right hand sides for each symbol.

Once each individual knows where it will go, the second table of production rules can be applied.

This table formalises the movement of the individuals. Each individual follows its arrow. Provided that there are four allowed directions ($\{ \uparrow , \rightarrow , \downarrow , \leftarrow \}$), four neighbours are needed to calculate the next content of any cell. That means that the two-dimensional L System built has interactions and the von Neumann neighbourhood is the context. Let N(x) be the von Neumann neighbourhood of x.

Once again there are many rules but the system is deterministic. The new symbol at a position content can be calculated by means of the following algorithm:

Example: Assume that we have the following situation:

$$\begin{array}{ccc} & s_{0110,0110,\ 0010} & \\ s_{0110,0100,0000} & & s_{0011,0000,0110} \\ & s_{1100,0100,0100} & \end{array}$$

We take the first digit from the bottom neighbour, the second from the left neighbour, the third from the top neighbour and the fourth from the right neighbour, and move these digits to the new central symbol. In the example we get

$$s_{\ 1111,\ 0110,\ 0010}$$

In general, the set of production rules is defined as follows:

$$P_3=\{ N(s_{na,nb,nx}) ::= s_{na',nb',nx'} \; \forall \; s_{na,nb,nx} \in V, \text{ where N(x) is the von Neumann neighbourhood of the cell and na',nb',nx' are calculated as indicated in the example} \}$$

The axiom is obtained by expressing the state in every node of the grid with the convention presented in the previous paragraphs. The two dimensional L System associated to this automaton is $(V, \{P_2, P_3\}, \alpha_2)$. Where P_2, P_3 and α_2 were defined above. Each table of production rules is used once. The table used first is P_2.

4.1.5 The combined cellular automaton represented by an L system

In order to be able to combine the L systems defined in the previous sections, we will introduce some redundant terms in the symbols used with the second L system, containing the number of ones in each section of the sub-index. This number is the same used as the index in the first L system.

For instance:

$$s_{0111,\ 1111,\ 0010} \text{ becomes } s_{0111,\ 1111,\ 0010,3,4,1}$$

The alphabet of the L System is

$$V_2=\{\ s_{na,nb,nx,ea,eb,ex} \mid ei \text{ is the number of ones in } ni\ \forall\ i\in\ \{a,b,x\}\}$$

This L System needs three different tables of production rules that are applied once in order before alternating. The tables are the sets of production rules P_1, P_2, P_3. The only difference is that the indexes used in the symbols of P_1 are now the last three sub-indexes. Since P_3 uses the von Neumann neighbourhood, the whole system is considered to use it. The axiom α_3 is calculated as described above. So the complete L System is as follows: $\{V_2,\{P_1,P_2,P_3\},\alpha_3\}$

5. Three-dimensional cellular automata

Three-dimensional cellular automata distribute their cells and the associated finite automata over a section of $\mathbb{R}^3$. The shape and size of the grid is not fixed, but the automata are usually located at the vertexes of cubes that share their faces. An example of such automata is presented in the following paragraphs.

5.1 A three-dimensional cellular automata that generates and propagates pulses.

This cellular automata uses the grid drawn below:

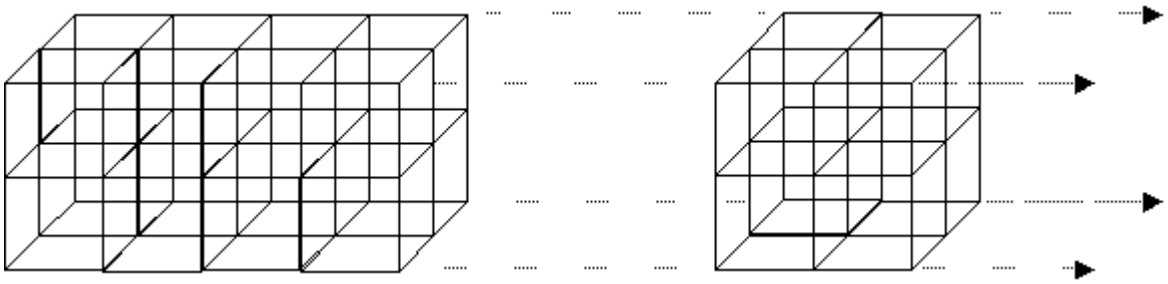

Figure 2.

This is a square prism potentially infinite to the right and made up of small cubes of d units per side.

The neighbourhood used is drawn in the next figure.

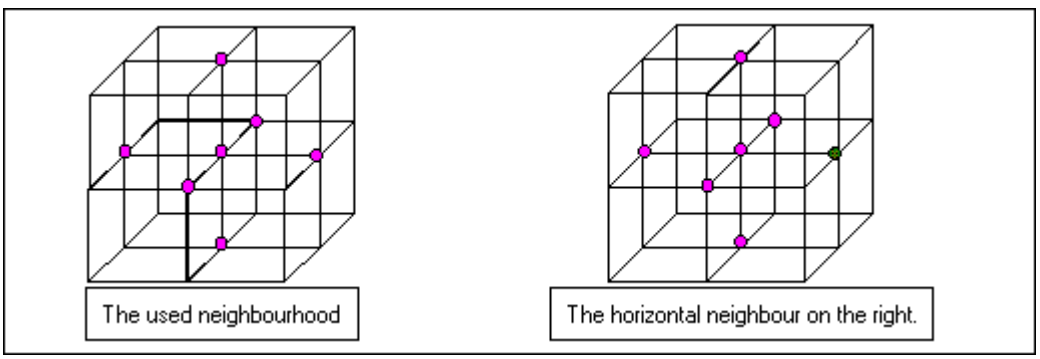

Figure 3. The neighbourhood

The neighbourhood of a cell is the cell itself and its six nearest neighbours, the six cells that surround it at d units of distance. Each automaton takes as state an element of the set {0,1}. The initial state is shown on the following page. Dark points mean a state of 1, blue points a state of 0.

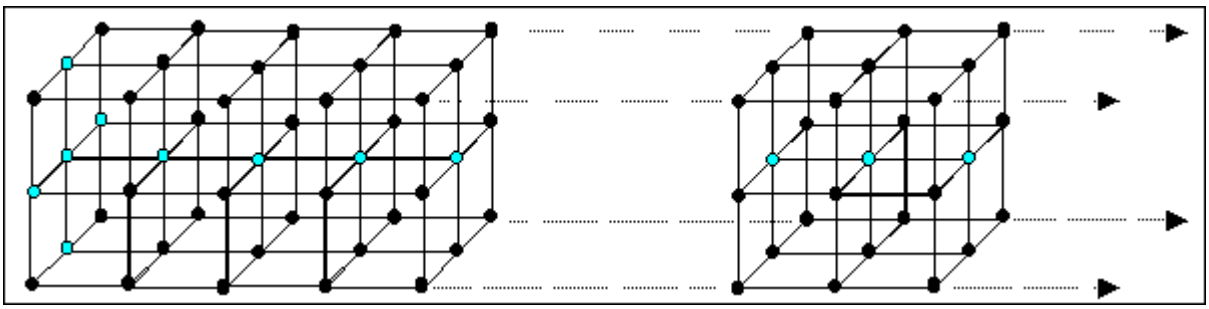

Figure 4. The initial state

To calculate the next state, each automaton follows the following rules:

- It does not take into account its horizontal neighbour to the right (the dark one in the neighbourhood figure).
- If the four neighbours in the vertical plane have a state of 0, the next state of the considered automaton changes no matter the value of the neighbour to the left.

This rule is shown in the following table and figure:

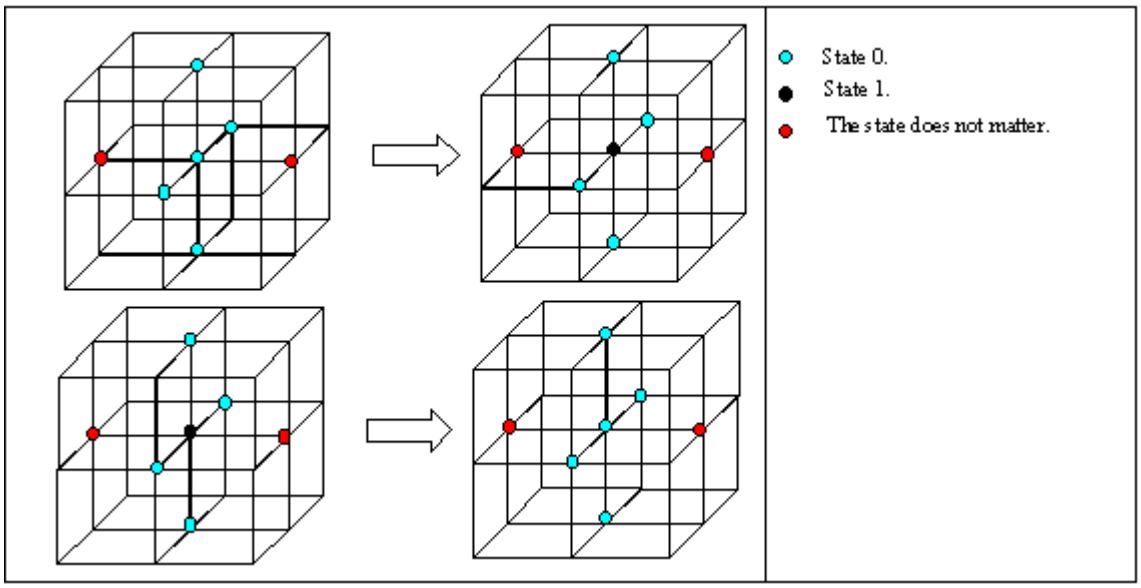

Figure 5.

Table 5.

Neighbour						Automaton's	Automaton'
up	down	front	back	right	left	state i	s state i+1.
0	0	0	0	x	y	0	1
0	0	0	0	x	y	1	0

Where x and y are any value from the set {0, 1}.

- If the four neighbours in the vertical plane have a state of 1, the next state of the current automaton depends on the state of its neighbour to the left and the state of the current automaton. The state changes when both values are not the same, otherwise the state remains unchanged. This rule is shown in the following table and figure:

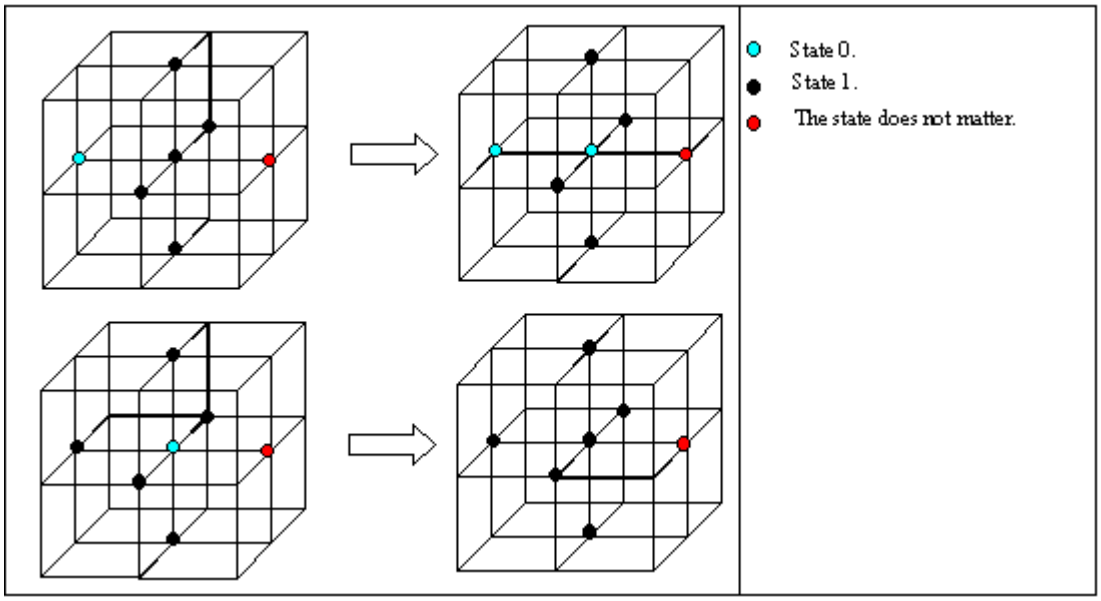

Figure 6.

Table 6.

Neighbour						Automaton's	Automaton'
Up	down	front	back	right	left	state i	s state i+1.
1	1	1	1	x	0	0	0
1	1	1	1	x	0	1	0
1	1	1	1	x	1	0	1
1	1	1	1	x	1	1	1

Where x is any value from the set {0, 1}.

- The state of the automaton remains unchanged otherwise.

The first steps after the initial state of the whole automaton are shown on the following page.

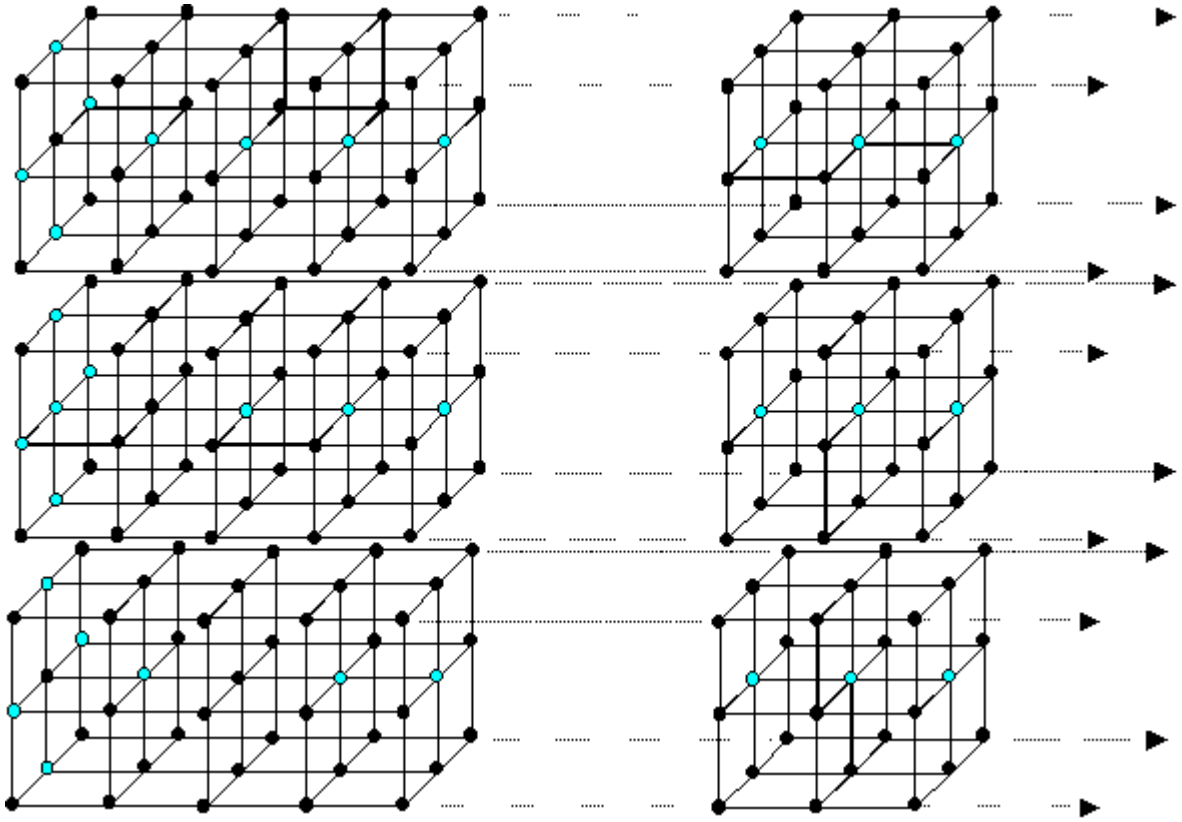

Figure 7.

The left face of the prism is generating pulses at the odd steps, and these pulses are propagated across the grid.

The initial states of the automata at the vertexes of the leftmost face are particularly relevant. If they are 0, several automata in this face will change their value generating spurious and noise pulses. The aim of this cellular automaton is to generate and propagate pulses across its central axis. The state of every automata on the boundaries remains unchanged because their up, down, front and back neighbours never have the same states.

It is possible to devise a three dimensional ILSystem whose derived words correspond to consecutive generations of this automaton. The neighbourhood used was presented above. The alphabet of this system is $V_4 = \{0,1\}$.

In order to express the context in a readable way the following order will be imposed to the symbols at the left-hand side of the production rules.

Table 7.

Order	Automaton
1	The upper neighbour
2	The lower neighbour
3	The neighbour in front
4	The neighbour at the back
5	The neighbour to the right
6	The neighbour to the left
7	The current automaton

A binary vector whose digits are the states of these seven automata is used as the left-hand side in every production rule. The rules could be directly copied from the previous tables, so the set of production rules P_4 can be expressed as follows:

P_4={ 0000xy0::=1 ∀ x, y ∈ V_4; 0000xy1::=0 ∀ x, y ∈ V_4; 1111x00::=0 ∀ x ∈ V_4;
1111x01::=0 ∀ x ∈ V_4; 1111x10::=1 ∀ x ∈ V_4; 1111x11::=1 ∀ x ∈ V_4;
otherwise, the following rule is used: xyzuvws::=s ∀ x, y, z, u, v, w, s ∈ V_4 }

The axiom α_4 is the three dimensional binary array obtained from the initial state figure, where the value is obtained from the colour of the points by applying the rule below:

$$\begin{cases} 1 \; if \quad dark \\ 0 \quad otherwise \end{cases}$$

The equivalent L system is { V_4, P_4, α_4 }.

6. Conclusions

We have shown that it is possible to obtain equivalent L systems of quite different, multidimensional and complicated cellular automata. We are not offering at this point a general conversion procedure, but provide the following set of suggestions:

- The dimension of the cellular automata is not a limitation to the possibility of obtaining an equivalent L System. There is a lot of literature about one-dimensional and two-dimensional cellular automata. Examples of three-dimensional cellular automata are unusual. There is practically nothing about n-dimensional cellular automata for n>3. A possible reason is the impossibility of obtaining graphic representations of their behaviour.
- As observed in the previous examples, the bottleneck is the formalisation of the function that generates the next state of each automaton in the grid. A general technique should follow this recommendation:
  - The first step is identifying the possibly independent and radically different behaviours of the automata. The next step is building a set of production rules to express each behaviour, and putting them together in an L System with tables, after identifying the conditions where each table should be used. A good starting point to identify the set of production rules is the use of different neighbourhoods in the cellular automaton.
- In all the examples, the behaviour could be coded into one or more sets of production rules joined together in an L System with tables. We have not found examples that made this approach fail. It is possible to build each set of production rules because the following conditions hold:
  - Every cellular automaton considers a finite number of nearest points in the grid as their neighbourhood.
  - Every cellular automaton has a finite set of possible states for their individual automata.

These two facts are the key that makes the process possible. Given these two conditions, it is easy to express all the possible change rules, provided that the above sets are finite. The grid, however, can be infinite.

In the examples, the production rules have been completely listed whenever it was reasonable. Some examples show an enormous (though finite) number of combinations. In that case, a generation algorithm has been given.

Would it be possible to design a general algorithm to build the L System associated to any cellular automaton? The answer must be found in the nature of the description of the cellular automata, rather than in the expressive power of the observations made above. The behaviour of a cellular automaton can be expressed in almost any way. A more formal description is needed in order to build general equivalent L Systems.

7. References

[1] Lindenmayer, "Mathematical Models for Cellular Interactions in Development" (two parts), J. Theor. Biol. 18, 280-315 (1968).

[2] Mandelbrot, The Fractal Geometry of Nature, W. H. Freeman, San Francisco, 1982.

[3] M. F. Barnsley, Fractals Everywhere, Academic Press, Inc., Boston, 1988.

[4] S. Papert, Mindstorms: Children, Computers, and Powerful Ideas, Basic Books, New York, 1980.

[5] G. Weisbuch: Complex systems dynamics. A lecture notes volume in the Santa Fe Institute studies in the sciences of complexity. Addison-Wesley Publishing Company. 1991

[6] S. D. Casey and N. F. Reingold, "Self-Similar Fractal Sets: Theory and Procedure," IEEE Computer Graph. & Appl. 14, 73-82 (May 1994).

[7] J. Von Neumann, J.: "Theory of Self-Reproducing Automata", Univ. of Illinois Press, Urbana, 1966.

[8] A.W. Burks, "Essays on Cellular Automata", Univ. of Illinois Press,Urbana, 1970.

[9] S. Wolfram, "Theory and Application of Cellular Automata", World Sci. Publ., Singapore, 1986.

[10] Kari, J.: "Cellular Automata. An Introduction", in "Artificial Life: Grammatical Models", ed. by G. Paun, Black Sea Univ. Press, Bucharest, 1995.

[11] P. Linz, "An introduction to Formal Languages and Automata", D.C. Heath and Co., Lexington, 1990.

[12] J.R. Koza: "Discovery of Rewrite Rules in Lindenmayer Systems and State Transition Rules in Cellular Automata via Genetic Programming", Symposium on Pattern Formation (SPF-93), 1993.

[13] M.Sipper, D.Mange, A.Stauffer: "Ontogenetic hardware", BioSystems 44, p.193-207, 1997.

[14] A.Stauffer, M.Sipper: "On the relationship between cellular automata and L-systems: The self-replication case", Physica D 116, p.71-80, 1998.

[15] A.Stauffer, M.Sipper: "L-hardware: Modeling and implementing cellular development using L-systems". In D. Mange and M. Tomassini, editors, "Bio-inspired Computing Machines: Toward Novel Computational Architectures", Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, p. 269-287, 1998.

[16] M.Alfonseca, J.de Lara, E. Pulido: "Educational simulation of complex ecosystems in the World-Wide Web", Proc. ESS'98, SCS Int., p.248-252, 1998.

[17] Volterra, V.: "Leçons sur la Théorie Mathématique de la Lutte pour la Vie", Gauthier-Villars, Paris, 1931.