# Computer Morphogenesis

Jean-Louis Giavitto and Antoine Spicher

## 1 Explaining living matter by understanding development

### *1.1 The animal-machine*

In 1739, Jacques de Vaucanson (1709–1782) presented a celebrated automaton to the French Academy of Sciences. It was called the *Canard Digérateur* (Digesting Duck, Fig. 1), a masterpiece of anatomical simulation, with more than four hundred moving parts reproducing the main vital functions (respiration, digestion, locomotion): the animal flapped its wings, ate grain and defecated (the grain being digested by dissolution, according to the inventor).

In making these "mobile anatomies", Jacques de Vaucanson was almost certainly influenced by the biomechanistic philosophy of René Descartes (1596–1650), who reduced the organs of the human body to parts in a machine "designed by God". Indeed, Descartes believed that one can understand life by comparing it to a machine: that one can explain the main bodily functions — digestion, locomotion, respiration, but also memory and imagination — as if they were produced by an automaton, like a clock designed to show the time simply by the layout of its wheels and counterweights. But when René Descartes tried to convince Queen Christina of Sweden that animals were just another form of machine, she is said to have replied:

> Can machines reproduce?

Three centuries were to pass before her question received an answer. A hundred years later, the automata of Vaucanson were imitating the main physiological functions, but they still could not reproduce, and it was only with the publication of an article by John Von Neumann in 1951, *The General and Logical Theory of Automata*, that it was finally possible to believe that a machine could effectively build a copy of itself [34].

To meet Queen Christina's objection, it is necessary to define precisely what we mean by "machine" and what we mean by "reproduction". For Von Neumann,

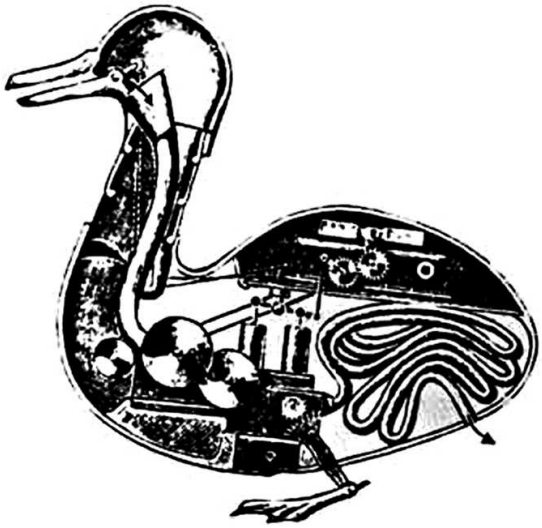

**Fig. 1** Vaucanson's duck. Voltaire described Vaucanson in these lines: *"While rival of the old Prometheus' fame, Vaucanson brings to man celestial flame. Boldly to copy nature's self aspires, And bodies animates with heavenly fires"*

who had a very functionalist approach to this question, mechanics can ultimately be reduced to a computer programme, and reproduction consists in duplicating this programme. This does not mean using a command in the computer's operating system to copy a file containing a programme, but ensuring that the functioning of the programme produces a complete and functional description of the programme itself. Fig. 2 shows an example of such a programme written in the programming language `C`: its execution produces a file containing the exact copy of its own code. This is called a *self-replicating* code.

```
#include<stdio.h>
main(){char*c="\\\"#include<stdio.h>%cmain(){char*c=%c%c%c%.
102s%cn%c;printf(c+2,c[102],c[1],*c,*c,c,*c,c[1]);exit(0);}
\n";printf(c+2,c[102],c[1],*c,*c,c,*c,c[1]);exit(0);}
```

**Fig. 2** A self-replicating code. This programme is made up of two lines of code in the programming language `C`. The second line of the programme (starting with `main`) has been arbitrarily typeset over three lines to make it more legible

Von Neumann's purpose was clearly to show that living processes can be reduced to mechanical processes, described by operations that can be performed autonomously, without the help of an "invisible mahout": to a machine, in other words. And for Von Neumann, like Queen Christina, reproduction and development are a

specific characteristic of living things. But for Von Neumann, this characteristic is just a particular property possessed by certain machines, not a quality that transcends physical processes, giving special status to biological ones. The existence of a machine, an automaton, capable of reproduction, is therefore a key factor in the age-old debate opposing the relative status of biology and physics.

This debate has not been easy to settle, reproduction being one of the most fundamental processes in the life of organisms and appearing to resist any physical explanation. Intuition suggests that if, as a result of its functioning, a machine *A* can produce a machine *B*, then *A* must contain, in one form or another, a complete description not only of *B* but also of the specific mechanisms instructing it how to use that description to actually produce (construct) *B*. This description must be internal to *A*, otherwise we would be dealing with a mechanism of copying rather than reproduction. We should therefore be able to define a certain measure of complexity and show that *A* is necessarily more complex than *B*. But in this case, our intuition leads us astray.

## *1.2 From self-reproduction to development*

Modern biologists may ask themselves the same questions as the philosophers and queens of past centuries, but today they seek to understand the mechanisms of reproduction by elucidating the *processes* leading from the germ cell to the complete organism: the aim is to understand, step by step, the *construction of an organism over the course of time*, through the multitude of local interactions of its constituent elements. In a word, *development*.

The elements that Von Neumann brought to the debate are very abstract: they are based on the description of a cellular automaton which reproduces, over the course of time, the configuration of a spatial subdomain in a neighbouring region. A cellular automaton can be described by a *predefined* network of sites, called cells, each cell possessing one of a finite set of states. The state of each cell is updated according to a predefined rule of evolution, which takes into account the state of the cell and the state of its neighbours at time $t$ to calculate the state of the cell at time $t+1$. The functioning of the automaton corresponds to the updating of the state of its cells at discrete time intervals (see Fig. 3).

We are a long way from the molecular mechanisms to which modern biologists wish to reduce biological phenomena. The existence of a self-replicating automaton suggests that there is no problem of principle in the existence of such a machine, but it tells us nothing about the "how" of biological processes. Nevertheless, the concepts of programme, code, automaton, memory and information have invaded biology and assumed an explanatory value, especially in developmental biology [20]: biologists need models and metaphors to understand (i.e. to represent, analyse and interpret) the huge mass of experimental data they have collected. For example, the concept of *genetic code* plays a similar role in the living cell as the rule governing the evolution of states does in the Von Neumann automaton.

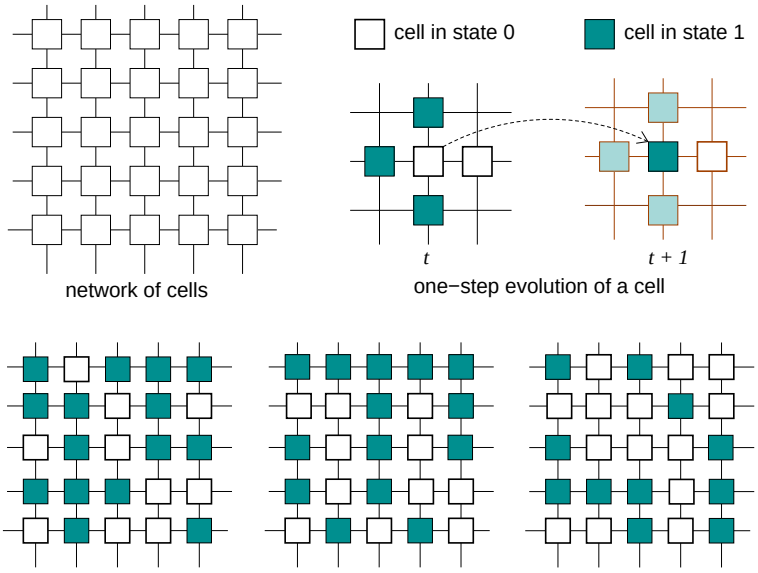


**Fig. 3** A cellular automaton is a network of cells, each joined to its neighbours by links. Here the network is a rectangular grid. Each cell possesses a state (here either 0 or 1). The rule of evolution used here is: the state of a cell is the modulo-2 addition of the states of the neighbouring cells. An example of the evolution of one cell is shown top right. The three networks below show three successive stages in the evolution of the automaton. The rule is applied simultaneously to all the cells. Von Neumann's self-replicating automaton is a model of this type, where the rules of evolution lead to the reproduction of the initial configuration of a given region in an adjacent region

## *1.3 Development as a dynamical system*

The concept of *dynamical system* allows to formalise the idea of process of development. A dynamical system (DS) is characterised by observations that evolve over time. These observations are the *variables* of the system, and they are linked by certain relations. These variables account for relevant properties of the system (whether they be biological, physical, chemical, sociological, or other). At a given moment in time, they have a certain value, and the set of these values constitutes the *state of the system*. The set of all the possible states of a system constitutes its *state space* (or configuration space). For example, a falling stone is a system characterised by the variables *position* and *velocity* of the stone. These two variables are not independent: if we consider the position of the stone as a function of time, then its velocity is the derivative of that function.

The succession of system states over time is called a *trajectory*. A DS is a formal way of specifying how the system moves from one point in the configuration space (one state) to another point (the next state). This can be done directly, by a function (the function of evolution of the system), or indirectly, by giving constraints (equations) on the possible future state (which is not necessarily unique, if the system

is not deterministic). A variety of mathematical formalisms correspond to this very general concept of dynamical system. For example, the variables can take continuous or discrete values. Likewise, the progression of time can be continuous or in discrete steps. Examples of formalisms corresponding to these cases are listed in Table 1.

In simple cases, the trajectory of a dynamical system can be expressed explicitly by an analytic function of time $t$. In the case of the falling stone, for example, the differential equations $\mathrm{d}x/\mathrm{d}t = v$ and $\mathrm{d}v/\mathrm{d}t = \mathbf{g}$ can be explicitly integrated to give the distance travelled by the stone as a function of time: $x = \mathbf{g}t^2/2$.

In more complex cases, an analytic equation giving the trajectory does not exist, and computer simulation is then a favoured approach for studying the trajectories of the system. In addition, instead of focusing on one particular trajectory, we can look at qualitative properties satisfied by all the possible trajectories, for example: "if we wait long enough, the system ends up in a well-defined state in which it then remains" or "if the trajectory passes through these states, it will never return". When there is no faster means of predicting properties than by observing or simulating them, we qualify them as *emergent properties*. Note that DS with very simple specifications can produce very complex trajectories (we sometimes speak of *chaotic behaviour*); moreover, calculating the trajectory of the system can be expensive in terms of computer time and require a vast amount of memory.

## The structure of states

Another important characteristic by means of which dynamical systems can be classified is the structure of states. In the example of the falling stone, the structure of a state is simple: it is a pair of vectors (velocity, position).

Very often, the structure of a state reflects the spatial organisation of the system. Let us take the example of the diffusion of heat in a volume. The distribution of the temperature has a structure, related to the spatial organisation of the volume. We can therefore define a scalar field assigning a temperature to each point. The evolution of this field follows a law of diffusion specified by a partial derivative equation. This links the temperature at time $t + dt$ of a point $p$ to the values of the temperature field at $p$ and in its neighbourhood at time $t$.

**Table 1** Three examples of formalism used to specify a dynamical system according to the continuous or discrete nature of the variables and of time. Iterated functions correspond to sequences $x_{n+1} = f^{n+1}(x_0) = f(x_n)$ for a given function $f$ on $\mathbf{R}$. Many other formalisms have also been studied

| C : continuous<br>D : discrete | differential equation | iterated functions | finite automaton |
|---|---|---|---|
| Time | C | D | D |
| State | C | C | D |

Very often, subsystems only interact if they are connected or physically close: we call this the property of *locality* (there is no action at a distance). The structure of a state then reflects this division into subsystems, and the function of evolution respects the property of locality. For the evolution of temperature in a volume, each state assigns a temperature to each point in the volume $V$ and the state space is therefore the set of functions of $V$ in $\mathbf{R}$. The heat diffusion equation governing the evolution of the system indicates that the temperature of a point in $V$ depends solely on the temperature of the neighbouring points.

## Development as trajectory of a dynamical system

Above, we stated that the concept of *genetic code* has much in common with the rules specifying the evolution of cell state in Von Neumann automata. This is the concept underlying the "all-genetic" paradigm, according to which the complete evolution of the organism is coded in its genetic material, and every characteristic is uniquely determined by the genes. This viewpoint has been substantially challenged [2], in favour of a more flexible approach, reconciling the genetic and epigenetic viewpoints on development. Living systems may be dynamical, but they are also open systems, interacting with their environment. Development should therefore be regarded as a co-construction, depending on interactions both within the system and outside it (with the environment). Genetic material does not constitute a complete and sufficient description of any given organism, although it is indispensable. Cell machinery, for example, also plays a central role, as has been demonstrated experimentally by the technique of cloning in which a nucleus (i.e. the genetic material of a cell) is introduced into a germ cell.

However, the processes of morphogenesis involving the movement and reorganisation of matter are also characterised by a second property: the state space and its topology can also evolve over time.

Let us illustrate this idea by comparing it to the two examples described above. In the case of the falling stone, the velocity and the position of the stone change at each moment but the system is always adequately described by a pair of vectors. In this case, we say that the dynamical system has a *stable structure*. The same is true for the evolution of the temperature in the volume $V$: $V$ is fixed in advance and each state is always an element of $V \to \mathbf{R}$. In these two examples, the state space can be described adequately at the beginning of time, before the simulation; it corresponds to the space of the measurements of the system. The value of these measurements changes over time, but the data of the state space and its topology are not variables of the system and cannot evolve over the course of time.

Quite the opposite holds true for the processes of development: biological processes form highly structured and hierarchically organised dynamical systems, the spatial structure of which varies over time and must be calculated in conjunction with the state of the system. We call this type of system a *dynamical system with dynamical structure*, which we shall abbreviate to $(\mathrm{DS})^2$.

The fact that the very structure of a biological system is dynamical has been highlighted by several authors; we can cite, in different domains: the concept of *hypercycle* introduced by Eigen and Schuster in the study of autocatalytic networks [13], the theory of *autopoietic systems* formulated by Maturana and Varela [43], *systems of variable structure* developed in control theory by Itkis [30], or the concept of *biological organisation* introduced by Fontana and Buss to formalise and study the emergence of self-maintaining functional structures in a set of chemical reactions [18]. The objective of all of these works has been to grasp and formalise the idea of change in the structure of a system, change that is coupled with the evolution of the state of the system.

$(\mathrm{DS})^2$ are widespread in models of plant growth and more generally in developmental biology, in multiscale cell models, mechanisms of protein transport and compartmentalisation, etc. But they are also relevant in other domains, such as the modelling of mobile networks, Internet and the Web, the development of cities, traffic jams, self-assembly processes, autocatalytic networks in chemistry, semantic networks in learning, social behaviour, etc.

An example

To illustrate the concept of $(\mathrm{DS})^2$, let us take the example of the development of an embryo. The initial state of the embryo is described by the state $s_0 \in \mathscr{S}$ of the germ cell (however complicated that description might be). After the first division, we have to describe the state with 2 cells, that is to say a new state $s_1 \in \mathscr{S} \times \mathscr{S}$. But when the number $n$ of embryo cells becomes large enough, the state of the system can no longer be adequately described by an element of $\mathscr{S}^n$. This set only describes the state of each cell; it does not contain the spatial information necessary to describe the network of cells (their positioning in relation to each other). And yet this network is of prime importance, because it conditions the diffusion of signals (chemical, mechanical or electrical) between cells and therefore, in the end, their functioning. With each movement, division or death of a cell, the topology of this network changes. For example, during *gastrulation*, cells initially far apart become neighbours, enabling them to interact and changing their destiny (cell differentiation).

## *1.4 What formalism for dynamical systems with dynamical structure?*

Dynamical systems with dynamical structure are difficult to study because they are difficult to formalise. Let us return to the example of the embryo to illustrate this.

We have indicated that the position of each cell changes over time, making it difficult, for example, to specify the processes of diffusion between cells. One solution that comes immediately to mind is therefore to complete the state of a cell with

information about its position, and to consider $\mathscr{T} = \mathscr{S} \times \mathbf{R}^3$ as a building block[1] allowing to construct the set:

$$\begin{aligned}\mathscr{T}^* &= \mathscr{T} \cup \mathscr{T}^2 \cup \ldots \cup \mathscr{T}^n \cup \ldots \\ &= \mathscr{T} \cup \mathscr{T} \times \mathscr{T}^* .\end{aligned}$$

It is certainly possible to characterise an embryo as a point in this phase space, but that does not get us very far: $\mathscr{T}^*$ has very little intrinsic structure and does not provide much information about the possible trajectories of the systems. For example, the function of evolution will be very difficult to define and there is little chance that it will be continuous.

### The problem of locality

The function of evolution will be difficult to define because specifying the position of each cell in terms of its coordinates $\mathbf{R}^3$ presupposes the definition of a global reference point. During the evolution of the embryo, the growth of a cell pushes away the neighbouring cells, which in turn push away their neighbours, until the position of every cell has been changed. Between two successive states, we therefore have to express the change in the position of each cell by a *global transformation* of coordinates. Because it must express globally the changes in each position, and because these changes are due to multiple concurrent local transformations, the expression of this transformation can be arbitrarily complex.

The origin of this problem lies in the *extrinsic and global* expression of the form of the system[2] and one solution is therefore to specify intrinsically the position of each cell, for example by including the distance from its neighbours in the state $s \in \mathscr{S}$ of each cell. In this case, the specification of changes in the position of a cell is local, but as the neighbourhood of each cell changes, we are again faced with the problem of a state space that changes over time.

### The problem of continuity

Let us return to the example of the falling stone. The position and the velocity of the stone vary continuously. The state of the system therefore varies continuously over time and the trajectory of the system is a *continuous function* of time in the

[1] To simplify, we only take into account the position of each cell in $\mathbf{R}^3$, but we should also specify its form, which conditions its neighbourhood and its exchanges with other cells (for example the surface exchange area between two neighbouring cells, which conditions intermembrane flow).

[2] In the approach described, the specification of the position of the cells uses a global reference point independent of the growing embryo. This reference point corresponds to the identification of points in the space surrounding the form, and not to a process intrinsic to the growing form: the laws governing the movement, division and death of cells would be the same if the embryo was developing within a toric volume (but the result could be different because the neighbourhoods of the cells would be different).

state space. This continuity allows to reason in terms of infinitesimal evolutions of the system and to write a differential equation characterising the trajectory. In more complicated cases, we obtain a partial derivative equation (when the state has a spatial structure) or a set of such equations when several different modes of functioning have to be taken into account (a finite and usually small number).

In the case of embryo development, this is no longer possible: as long as there is no movement[3], division or death of cells, the state *s* belongs to a certain $\mathscr{T}^n$ and this evolution is continuous (assuming that the electric potentials, chemical concentrations, mechanical constraints, etc. evolve continuously). But the essential morphogenetic events (for example a cell division that changes the state from $\mathscr{T}^n$ to $\mathscr{T}^{n+1}$) are by nature discontinuous[4].

Towards other solutions

The modelling and simulation of the evolution of a $(\mathrm{DS})^2$ are therefore particularly arduous: it is difficult to define the structure and the dynamics of the system at the same time, because one is dependent on the other. The example given above highlights the inadequacy of global and continuous formalisms (we want to express an evolution as a succession of discrete morphogenetic events corresponding to qualitative discontinuities and changes). However, it is still possible to describe these systems, with the laws of evolution often being informally described as a set of *local* transformations acting on an ordered set of discrete entities.

Faced with these difficulties, several researchers have suggested using *rewrite systems* to formalise this type of description.

## 2 Rewrite systems

### *2.1 Introduction*

Rewrite systems (RS) are among the formalisms that computer scientists have appropriated and developed, especially for modelling changes in the state of a process. A rewrite system is a mechanism allowing to define the replacement of one part of an object by another. The objects concerned are usually *terms* that can be represented by a tree, of which the inner nodes are operations and the leaf nodes are constants (see Fig. 4). An RS is defined by a set of rules, and a rule is a pair denoted $\alpha \to \beta$. A rule $\alpha \to \beta$ indicates how a sub-term $\alpha$ can be replaced by a term $\beta$.

[3] Cell movement is sufficient to change the topology and therefore the interaction between cells.

[4] In the example we have been using, morphogenetic events are discontinuous because the modelling is done at cell level. We could have modelled the concentration of different molecules at each point in space, which might have avoided this problem of discontinuity (the movement of each molecule being *a priori* continuous). But this raises another problem: how do these concentrations represent the biological entities that interest us: cells, tissues, organs, etc.?

## An example

Let us take the arithmetical expressions and the rule $0+x \to x$. Intuitively, this rule specifies that any expression that can take the form "0 added to something denoted by $x$" can be rewritten more simply as "the thing denoted by $x$". Thus, the expression $e = 1+(0+3)$ can be rewritten as $e' = 1+3$ by applying the above rule to the subterm $(0+3)$ of $e$. We also write $e \to e'$ to indicate that $e$ can be rewritten as $e'$ through one sole application of the rule.

The sequence $e \to e_1 \to \ldots \to e_n \to e'$ is called a *derivation* of $e$. We say that $e$ is a *normal form* if there is no $e'$ such that $e \to e'$.

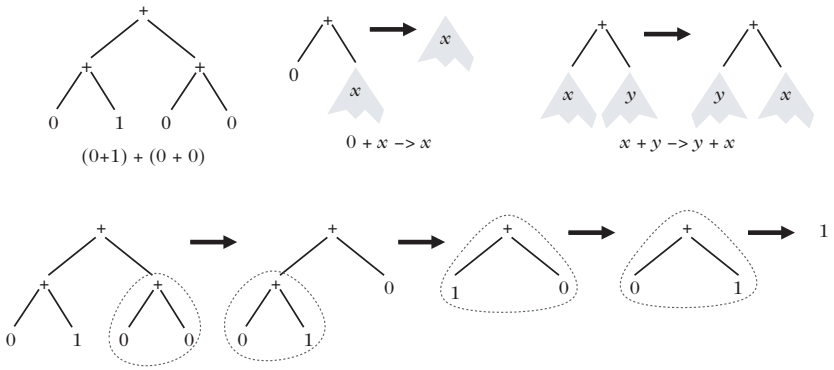


**Fig. 4** Representation of the term $(0+1)+(0+0)$ and application of the two rules $0+x \to x$ and $x+y \to y+x$. At each reduction, the strategy here is to apply one rule at a time. The subtree filtered by the left side of the rule to be applied is circled by a dashed line. The applications are non-deterministic, in the sense that we could have chosen other applications at each step. For the first reduction, for example, we could have applied the same rule $0+x \to x$ to the left subtree of the root rather than the right subtree. We could also have chosen to apply the rule $x+y \to y+x$ to any of the three inner nodes (3 possibilities). The final term obtained is the constant 1, and this is a *normal form* for the two rules

## RS and decision procedure in an equational theory

The original motivation behind RS was to provide a decision procedure in equational theories. In these theories, the aim is to prove automatically the equality of two complex terms solely by using predefined elementary equalities. The idea is to orientate the equations (for example, to orient the equality $0+x = x$ into a rule $0+x \to x$) and to use the rules obtained to derive the normal form $e'$ of a term $e$. The normal form $e'$ is equivalent to $e$ (since each substitution transforms a subterm into an equivalent term) and can be interpreted as a simplification of $e$. Two terms $e_1$ and $e_2$ are then equivalent in the theory if they reduce to the same normal form

$e$. For example, $e_1$ defined by $0+(1+3)$ is equivalent to $e_2$ defined by $1+(0+3)$, because $e_1$ and $e_2$ reduce to the same normal form $e$: $1+3$.

For this decision procedure always to succeed, there must exist a normal form for each expression (property of *normalisation*) and each expression must have one sole normal form (property of *confluence*). These two properties are not quite sufficient for the decision procedure to calculate automatically; at each step we must also choose a derivation, i.e. choose which subterm will be rewritten and by which rule: this is the *strategy* of rule application.

The theory of RS [9, 10] is mainly used in algebra and logic, but it can be applied in almost every branch of computing (from Petri networks to symbolic calculus, from the theory of demonstration to lambda calculus). One key result is that RS, considered as processes of calculation, are *Turing-complete* (any computational process, i.e. described by a Turing machine, can be formalised by an RS). The use of rules to transform a term is such a fundamental operation that several generic environments have been developed to define and apply RS (see, among others, the websites of the projects ELAN [16] and MAUDE [33]). The tools differ in the terms they take into account, the $\alpha$ patterns allowed on the left-hand side of a rule for selecting subterms, and the strategies of application that can be defined.

## *2.2 Rewrite systems and the simulation of dynamical systems*

The above presentation suggests that a rule $\alpha \to \beta$ specifies a term $\beta$ equivalent to (and simpler than) the term $\alpha$. But we can interpret this rule as the result of a calculation (the expression $\beta$ is the result of calculating the expression $\alpha$) or as the evolution of a subsystem changing from state $\alpha$ to state $\beta$. RS can therefore be used to model and simulate DS:

- a state is represented by a term and the state of a subsystem is represented by a subterm;
- the evolution function is encoded by the rules of the RS in the following manner: the left side of the rule corresponds to a subsystem in which the elements *interact*, and the right side of the rule corresponds to the result of their interaction.

Thus, the derivation of a term $s$ corresponds to a possible trajectory of a DS starting from the initial state $s$. A rewrite rule then corresponds to the specification of the evolution of a subsystem. A normal form corresponds to a fixed point in the trajectory (the system is in equilibrium and no evolution can take place).

### An example

For the development of the embryo, a rule $c \oplus i \to c'$ can be interpreted as a cell in the state $c$ which, on receiving a signal $i$, evolves to the state $c'$; a rule $c \to c' \oplus c''$ represents a cell division; a rule $c \to \emptyset$ ($c$ gives nothing) represents apoptosis; etc.

[17, 23]. The idea is that the evolution of a biosystem is specified by rewrite rules of which the left side selects an entity in the system and the messages sent to it, and the right side describes the new state of the entity. The operator $\oplus$ which appears in the rule denotes the composition of local entities in a global system (in our example, the aggregation of cells in an embryo). The capacity to represent both the changes of state and the appearance and disappearance of cells within the same formalism makes RS good candidates for the modelling of $(\mathrm{DS})^2$.

### 2.2.1 Dealing with time

One important factor in the modelling of a $(\mathrm{DS})^2$ is the treatment of time. The model of time favoured in RS is clearly an event-driven, atomic and discrete model: time passes when an evolution occurs somewhere in the system, the application of a rule corresponds to an event and specifies an atomic and instant change in the state of the system. The concept of duration is not taken into account (although it could be, within this formalism, by considering the first and last events). The choice of a strategy of application provides a certain degree of control over the model of time: for example, a maximal parallel application of rules to change from one global state to another corresponds to synchronous dynamics, while the application of one sole rule corresponds to asynchronous dynamics.

### 2.2.2 Dealing with space

A rule of the form $c \oplus i \to c'$ presupposes that a signal $i$ produced by a certain cell will reach its target $c$ somewhere else in the system. The operation $\oplus$ used to amalgamate the states of the subsystems and the messages of interaction into the state of a complete system must therefore express the spatial dependencies and functional organisation of the system studied.

The concept of rewriting has mainly been developed and studied for the rewriting of terms. These represent a severe restriction on RS, because their use requires the encoding of the highly organised structure of $(\mathrm{DS})^2$ in tree form. The possibility of defining rules of evolution depends on this encoding. This work demands a great deal of creativity and intuition. It is difficult to represent in a satisfactory manner the organisation of a biological system into molecules, compartments, cells, tissues, organs and individuals, and this has motivated an extension of the concept of rewriting to structures more sophisticated than terms (for example, we can define a concept of rewriting on a graph, see also [21, 22]).

Nevertheless, even when they are limited to trees, RS offer remarkable examples of modelling of $(\mathrm{DS})^2$, particularly in the biological domain. By playing on the properties of the operators, it is possible to model several types of organisation. In the following sections, we shall give examples where:

- the operation $\oplus$ is associative and commutative, which allows to model a “chemical soup”;

- several operations can be considered simultaneously, as a means to introduce the idea of compartmentalisation;
- the operation $\oplus$ is simply associative, which allows to represent sequences and tree structures.

## 3 Multiset rewriting and chemical modelling

The state of a chemical solution can be represented by a *multiset*: a set in which one element can appear several times, as in a chemical solution where several molecules of the same species are present at the same time. A multiset can be formalised by a formal sum in which the operator $\oplus$ is *associative* and *commutative*. For example:

$$(a \oplus b) \oplus (c \oplus b)$$

represents a multiset (e.g. a chemical solution) containing the elements (e.g. the molecules) $a$, $b$ and $c$, where two copies of $b$ appear. Since the operation $\oplus$ is associative, we can discard the brackets, and the property of commutativity allows us to reorganise the elements in this sum as we like:

$$(a \oplus b) \oplus (c \oplus b) = a \oplus b \oplus c \oplus b = a \oplus b \oplus b \oplus c = c \oplus b \oplus a \oplus b = \ldots$$

A multiset therefore corresponds to a tree in which associativity allows us to "flatten" the branches, and commutativity allows us to permute the leaves.

In a chemical solution, Brownian motion agitates the molecules, and after a sufficiently long time each molecule will have had the opportunity to meet and interact with any other molecule in the solution. Once we have represented the state of a chemical solution as a multiset, it is therefore easy to formulate the chemical reactions as rewriting rules on multisets. The associativity and commutativity of the operator $\oplus$ play the role of Brownian motion and allow to "group together" arbitrarily the elements of the multiset corresponding to a left side of the rule before that rule is applied. For example, the three rules:

$$r_1 : a \oplus a \to a \oplus a \oplus b \qquad r_2 : a \oplus b \to a \oplus b \oplus b \qquad r_3 : b \oplus b \to b \oplus b \oplus a$$

represent second-order catalytic reactions between type $a$ and type $b$ molecules (a collision between two molecules catalyses the formation of a third molecule, without consuming the first two). Thus, if a reaction $r_1$ occurs in state $a \oplus c \oplus a \oplus b$, the result will be the state $a \oplus c \oplus a \oplus b \oplus b$ where an extra $b$ has been produced. Note that it is not necessary for the two $a$ molecules to be side by side, because we can always rearrange the term to make it so.

Several chemical reactions can happen at the same time, *in parallel*. This corresponds to the simultaneous application of several rules to different molecules. The strategy of applying as many rules as possible at a given time step is called a *maximal parallel* application. Such a strategy is non-deterministic: on the multiset

$a \oplus a \oplus b$ we can apply $r_1$ or $r_2$, but not both at the same time, due to a lack of resources. In this case, one of the rules is chosen at random. A reduction step is then repeated to simulate the evolution of the state of the chemical solution. Several approaches are possible, in terms of adjusting the strategy of rule application, to take into account the kinetics of chemical reactions [5, 24].

Note that in this approach, each molecule is explicitly represented and each interaction is explicitly treated: this is known as *agent-based simulation*. This approach can be compared to more classic approaches which represent the concentration of each chemical species rather than each molecule. Obviously, in this particular case, the agent-based approach is more costly in computing time and memory, but it allows to simulate finely the complex phenomena, such as fluctuations and correlations, that are beyond the reach of global approaches.

This abstract formalisation of chemical reactions constitutes a domain of research called *artificial chemistry* [11, 12], tackling problems ranging from the automatic generation of combustion reactions [6] to the study of mechanisms of self-organisation in the evolution of self-catalytic networks [18].

## *3.1 Some examples of application*

A simple example of population growth

To illustrate multiset rewriting and its application to modelling, we shall look at an example of a biological nature: the multiplication of a unicellular organism in a test-tube. We assume that a cell exists in two forms, $A$ and $b$: $A$ represents a mature cell ready to divide and $b$ a young cell that will evolve to form $A$. Each cell division of $A$ produces one cell of type $A$ and one cell of type $b$. These evolutions can be formalised by the two rules:

$$\begin{aligned} r_1 : &\quad A \longrightarrow A \oplus b \\ r_2 : &\quad b \longrightarrow A \end{aligned}$$

If the initial state of our test-tube is represented by $m_0 = A \oplus b \oplus b$, the first three evolutions give us:

$$m_0 \to A \oplus b \oplus A \oplus A \to A \oplus b \oplus A \oplus b \oplus A \oplus b \oplus A \to$$
$$\to A \oplus b \oplus A \oplus b \oplus A \oplus b \oplus A \oplus b \oplus A \oplus A \oplus A \to \ldots$$

Simulation of this process can be used to determine, for example, the ratio of forms $A$ to forms $b$ in the population after a given time. Moreover, as we mentioned earlier, we can test properties verified by all the processes satisfying these rules of evolution. For example, Fibonacci[5] proved that the ratio $\#A/\#b$ of the number of $A$ to the

[5] In 1602, Fibonacci studied the question of how fast a population of rabbits would grow under ideal conditions. Imagine that a pair of rabbits, one male and one female, are put in a field. These

number of $b$ converges asymptotically towards the golden number, whatever the initial state.

### Applications to the modelling of networks of biological interactions

The modelling of networks of biological interactions (genetic control networks, signalling networks, metabolic cascades, etc.) is a relatively new domain of application of these techniques. In [17], Fisher and his co-authors proposed using the concept of multiset to represent proteins involved in a cascade of interactions in a signalling pathway. This avenue has been widely developed, in particular to take into account the different complexes that proteins can form [14, 15].

If multiset rewriting has been used to model signalling networks and metabolic pathways, we should not deduce that the cell can be compared to a test-tube containing a chemical soup. On the contrary, the cell is a spatially highly organised medium, with compartments, vesicles, cargos, membranes, etc., which allow to localise the different chemical species involved (for example the receptors are localised on the cell membrane, while the genes are located in the nucleus; other proteins are anchored and diffuse in membranes like the endoplasmic reticulum). Among other things, this localisation helps to make certain reactions much more efficient. Other phenomena, such as the extreme density of proteins in the intracellular medium, render the simple model of chemical soup simply inadequate. Taking into account this spatial organisation is one of the main challenges currently faced in the modelling of cellular processes [31, 42].

### Heat diffusion in a bar

Above, we stated that the properties of associativity and commutativity allow us to deconstruct a term so that each element can interact with any other element, in the manner of molecules in a well-mixed chemical soup. But with the appropriate encoding, multiset rewriting can be "diverted", so as to take into account geometric information.

The process we want to model is the diffusion of a set of particles along a line. This problem also corresponds to the diffusion of heat in a thin bar, with each particle representing a quantum of heat. The line is discretized into a sequence of small intervals indexed by consecutive integers. Each interval contains a number of particles (possibly zero). At each time step, a particle can stay in the same interval or diffuse into the neighbouring interval (see Fig. 5). We can represent a state of the

---

rabbits are capable of reproducing after one month, so that at the end of the second month, the female has given birth to another pair of rabbits. To simplify, we assume that the rabbits never die and that each female gives birth to a new pair, composed of a male and a female, every month starting from the second month. If we represent a newly-born pair by $b$ and a mature pair by $A$, then the rule $r_1$ corresponds to the breeding of a new pair and the rule $r_2$ to the maturing of a young pair.

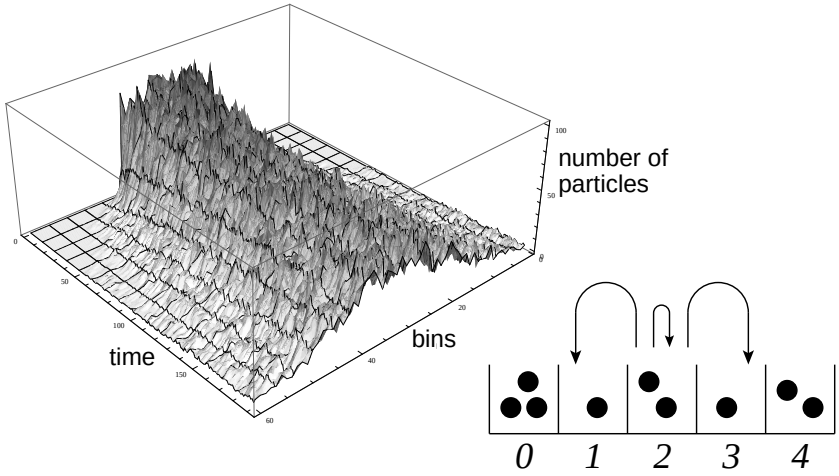


**Fig. 5** Diffusion of particles in a bar. The left-hand figure shows the evolution over 160 time steps of 1500 particles initially distributed over 20 intervals in the middle of a bar discretized into 60 intervals. The diagram on the right specifies the behaviour of a particle (see text). The number of particles in an interval corresponds to the concentration of a chemical product or a quantity of heat. The multiset representing the state of the line (which is discretized into 5 intervals numbered from *0* to *4*) is given by: $0 \oplus 0 \oplus 0 \oplus 1 \oplus 2 \oplus 2 \oplus 3 \oplus 4 \oplus 4$

line by means of a multiset in which each number $n$ represents a particle present in the interval numbered $n$. The evolution of the system is then specified by thel following three rules:

$$\begin{aligned} r_1 : &\quad n \longrightarrow n \\ r_2 : &\quad n \longrightarrow n-1 \\ r_3 : &\quad n \longrightarrow n+1 \end{aligned}$$

where $n$ is an integer and the operations $+$ and $-$ which appear on the right side are the usual arithmetic operations. The rule $r_2$ (respectively $r_3$) specifies the behaviour of a particle that diffuses into the interval on its left (respectively right) and the rule $r_1$ specifies a particle that remains in the same interval.

## *3.2 Păun systems and compartmentalisation*

The above encoding allows us to deal with linear geometry. Other variations have been proposed to facilitate the representation of more complex biological structures, such as the nesting of membranes and compartments in a cell: the elements of a multiset can be molecules or other multisets, which can in turn contain molecules or other multisets.

This nesting is studied using the formalism of *Păun systems* (P systems) [36], in which the classical rewriting of multisets is extended by the concept of *membrane*. A membrane is a nesting of compartments represented, for example, by a Venn diagram[6] without intersections and with one sole superset: the *skin* of the system (see Fig. 6).

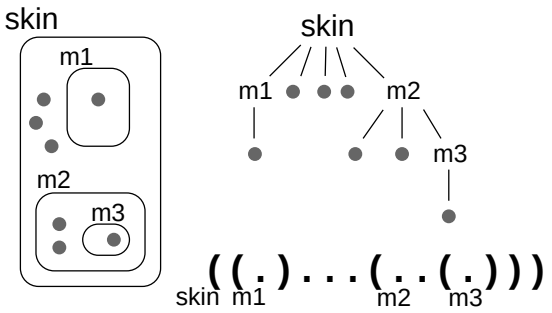


**Fig. 6** Păun systems. Membrane nesting can be represented by a Venn diagram without intersections and with one sole superset, by a nesting tree or by a well-bracketed word

Objects are placed in each region delimited by a membrane, and they then evolve according to diverse mechanisms: an object (or a multiset of objects) can change into other objects, but it can also cross a membrane or provoke the dissolution or creation of a membrane. Fig. 7 shows some examples of rules of evolution in P systems. Formally, such a system can be specified by using several operations $\oplus, \oplus', \oplus'', \ldots$, each corresponding to a certain membrane. These operations are associative and commutative, but they are not associative with each other (in order to keep the membranes separate).

## *3.3 In parenthesis: the application to parallel programming*

The dialogue between computing and the other scientific disciplines is not all one-way. Here is an example. Inspired by the chemical metaphor, computer scientists have used multiset rewriting not only to simulate chemical reactions or biological processes, but also as a parallel programming language. The idea was first developed in the language GAMMA [3]. Here is a particularly elegant example of a parallel programme:

$$x \oplus y \,/\, (x \bmod y = 0) \longrightarrow y \,.$$

This rule specifies that the pair of numbers $x, y$ must be replaced by $y$ when the condition "$y$ divides $x$" is satisfied (the condition is written after the $/$ symbol). If we

[6] Venn diagrams, invented by the English logician of the same name, are a means of visualising set operations by representing the sets as surfaces delimited by closed curves.

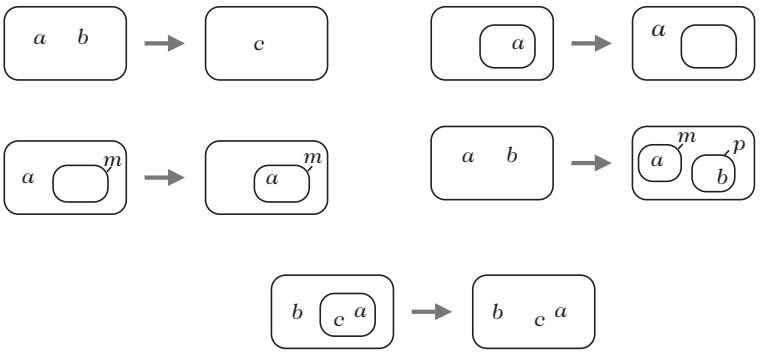


**Fig. 7** Example of a rule of evolution in a Păun system. The symbol $\delta$ on the right side of a rule entails the dissolution of the enclosing membrane. The destination membranes are indicated by the suffix (the membranes are named). The enclosing membrane can always be referred to by the name "out"

apply this rule as far as possible to the multiset composed of all the integers between 2 and *n*, we obtain a multiset in which the rule cannot be applied (because the condition is no longer satisfied) and which therefore contains all the prime numbers up to *n*.

In the above programme, there is no trace of artificial sequencing in the calculations: the rule can be applied in any order whatsoever. Note that the parallelism comes from the simultaneous application of rules and that the "unfolding" of the programme consists simply in repeating the application of the rules until a normal form (a fixed point) is obtained. These programmes are non-deterministic, unless the rewriting rules are confluent: in that case, when the programme ends, we do obtain a perfectly determined result, although the intermediate values calculated during the execution of the programme can differ (we speak of deterministic results despite a non-deterministic execution environment).

## 4 Lindenmayer systems and the growth of linear structures

In the previous section, we considered a process of rewriting on associative and commutative terms, allowing us to model a "chemical soup". In this section, we shall explore associative terms: these terms then correspond to sequences and we speak of *rewriting strings* (of symbols). Chomsky's work on formal grammars [7] marked the beginning of a long series of works on string rewriting, and these works have been at the origin of developments concerning syntax, semantics and formal languages in computing. Grammars are generative formalisms. In other words, they allow us to construct families of objects, by generating sets of *phrases*: a phrase is

a sequence of symbols generated by successive rewritings. The set of phrases that can be generated is a *language*.

In 1968, the biologist Aristid Lindenmayer (1925–1989) introduced a new type of string rewriting to serve as the foundation for a formal theory of developmental biology [32]: Lindenmayer systems, more often abbreviated to L-systems. The main difference from Chomsky grammars lies in the strategy of rule application. In Chomsky grammars, only one rewriting is applied at a time, whereas in L-systems the rewritings take place in parallel, replacing all the symbols in a phrase at each step. Lindenmayer justified this strategy by analogy to cell development: all the cells in an organism divide independently and in parallel.

The objective of L-systems is to construct a complex object (like a plant) by successively replacing the different parts of aa simpler object, by means of rewriting rules. Symbols are interpreted as components of a living organism, such as cells or organs, rather than words. L-systems have found numerous applications not only in the modelling of plant growth, but also in computer graphics, with the generation of fractal curves or virtual plants.

## *4.1 Growth of a filamentous structure*

A simple example of L-system is the one that describes the growth of cyanobacteria *Anabaena Catenula*. These blue-green algae form filaments composed, for our example, of four types of cells, `G`, `g`, `D` and `d`, which can be interpreted as follows: `G` and `D` are mature cells capable of dividing; `g` and `d` are quiescent cells. In addition, the cells are polarised: `D` and `d` are polarised towards the right in the filament; `G` and `g` are polarised towards the left. When we examine a filament, we can see that the cells do not succeed each other in any old order. And the rewriting system presented below generates sequences very similar to those observed in nature.


The derivations on the right show that at each step all the symbols are rewritten in parallel according to the rules on the left. Numerous variations can be developed from this basic mechanism, with the aim of extending the expressivity of the formalism. One of the most important extensions involves attaching attributes to the symbols, for example a number representing a size, or the concentration of a chemical product. Fig. 8 illustrates the use of one such extension in a more realistic model of *Anabaena* growth. Instead of simply considering mature cells and quiescent cells, each cell possesses a size that grows over time. Furthermore, in a nitrogen-free medium, some cells become specialised: the heterocysts. Wilcox *et*

*al*. [44] proposed that a cell differentiates into a heterocyst under the action of two chemical substances, an activator and an inhibitor, which diffuse in the filament and react with each other. This reaction-diffusion model allows to explain the appearance of an isolated heterocyst cell every $n$ vegetative cells, as observed in nature, with a relatively constant $n$. Prusinkiewicz and Hammel [25] used an L-system to specify and simulate this system, thereby achieving the simulation of a reaction-diffusion in a growing medium (the filament), an important example of $(\mathrm{DS})^2$.

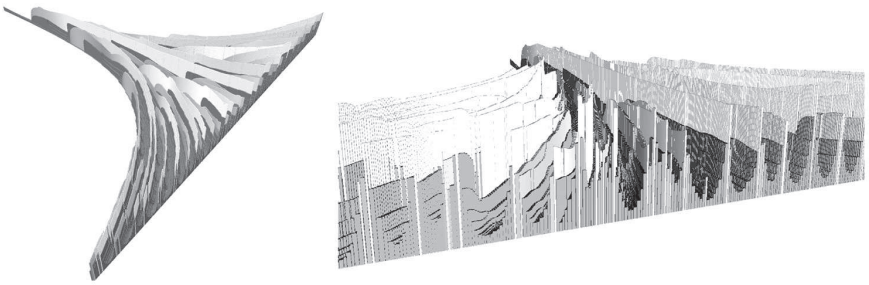

**Fig. 8** Differentiation of heterocysts in an *Anabaena* filament. The left-hand figure represents the same diagram as the right-hand one, but seen from another angle. This graphical representation, called an extrusion in space-time, was introduced in [25]. In this diagram, times moves from the top-left corner to the bottom-right corner. Each "slice" represents the cells of a filament at a given moment in time. The height of each cell represents the concentration of activator, as does the shading of the cell (from black to white). The black cells are vegetative. Differentiation occurs when the concentration of activator rises above a threshold level

## *4.2 Development of a branching structure*

It is easy to represent a branching structure by a string, by introducing two symbols that serve as "brackets". There are subtle differences between the rewriting of such strings and the direct rewriting of terms. String rewriting is the method used in L-systems to represent the branching structure of a plant and its rules of development. The example below is caricatural, and does not correspond to the growth of any real plant. But it does allow us to illustrate the power of this approach.

Let us assume that a plant is made up of two types of "branch": simple branches `b` and budded branches `B`. From one year to the next, budded branches lose their buds and become simple branches. We therefore have the rule $\mathtt{B} \to \mathtt{b}$. A simple branch grows and produces a section of plant comprising an axis made up of three simple branches with two budded branches branching off it, $1/3$ and $2/3$ of the way up. This specification is expressed by the rule:

$$\mathtt{b} \longrightarrow \mathtt{b}\langle q\mathtt{B}\rangle\mathtt{b}\langle p\mathtt{B}\rangle\mathtt{b}\,.$$

In this rule, we use the brackets $\langle$ and $\rangle$ the represent the development of the lateral axes. We use the additional symbols $p$ and $q$ to indicate development on the left or the right of these axes.

At the beginning of the 1980s, P. Prusinkiewicz introduced a graphical interpretation of words produced by an L-system [40]. This interpretation is based on the concept of graphical turtle, as used in the LOGO programming language, for example. This allows to directly visualise the structure of objects described by a word generated by the L-system. Thus, we use the successive derivations of an L-system to represent the successive states of a developing plant, or to draw the successive curves that tend towards a fractal curve.

A state of the graphical turtle is the triplet $(x,y,\theta)$, where $(x,y)$ represents the current position of the turtle in Cartesian coordinates and $\theta$ represents the orientation of the turtle. This orientation is interpreted as the angle between the body of the turtle and the horizontal axis. The turtle moves following commands represented by the symbols of a word.

- In our example, the symbol `b` corresponds to the command "move forward one length $\Delta$". So if the current state was $(x,y,\theta)$, then after reading the symbol `b` it becomes $(x+\Delta\cos\theta, y+\Delta sin\theta, \theta)$.
- The symbol `B` is interpreted in the same way, except that after drawing the corresponding segment, we also draw a circle centred on the current position.
- The symbols $\langle$ and $\rangle$ save and restore the current position respectively. The position is saved in a stack. When the turtle meets the symbol $\rangle$, it "jumps" to the position corresponding to the open bracket.
- Finally, the symbol $p$ (resp. $q$) increments (resp. decrements) the current angle $\theta$ by a predefined angle.

Using this graphical interpretation, the first three derivations of a simple branch are illustrated in Fig. 9.

## 5 Beyond linear structures: calculating a form in order to understand it

L-systems have proved to be perfectly suited to the modelling of plant growth [40]: they allow to define in a particularly compact and synthetic way the creation of the complex form of a plant and above all, in their recent extensions, to couple the process of form creation with the physical-chemical processes that take place within that form.

However, although L-systems are suitable for the representation of linear forms (filaments or trees), their use for the construction of more complex shapes (ordinary graphs, surfaces or volumes) depends on arbitrary encoding that rapidly becomes inextricably complex. Researchers are therefore trying to design more suitable formalisms. The import of this search for formalisms to specify the processes of development reaches far beyond the question of simulation, for two reasons: these

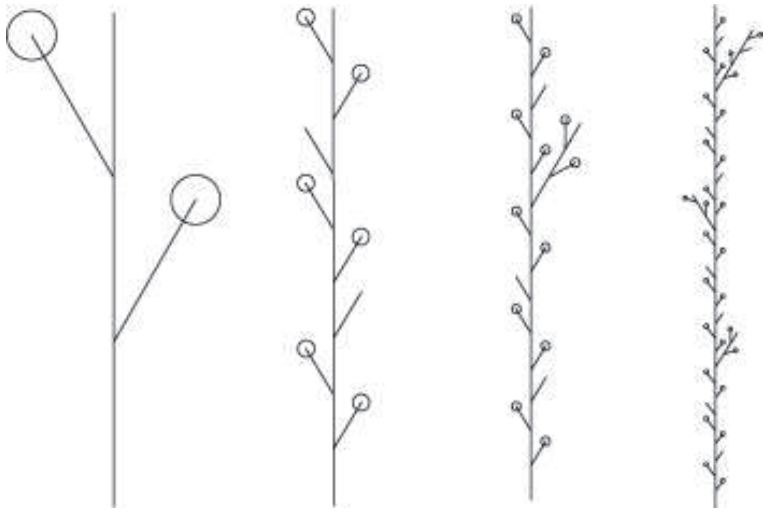

**Fig. 9** Graphical representation of the first three derivations in a Lindenmayer system. The initial state is given by the word $\mathtt{b}\langle q\mathtt{B}\rangle\mathtt{b}\langle p\mathtt{B}\rangle\mathtt{b}$ and the rules of derivation are the two rules defined in the text. The scale of the representation of each "plant" is different

formalisms could fill a conceptual vacuum in biology, and they could potentially have an enormous epistemological impact. What is more, their application could extend far beyond the domain of biology.

## Simulation and explanation

Drawing firstly on purely physical models (osmotic growth with Leduc, optimal forms with D'Arcy Thompson, reaction-diffusion processes with Turing, etc.), then purely genetic models (with concepts such as gene action or the genetic programme), the different formalisms proposed over the course of the last century to specify the processes of development have filled a conceptual vacuum and modified the perception of what has explanatory value for biologists [20].

As an example, advances in computing and the data produced in biology allow the simulation of certain processes of development with predictions that can then be empirically validated [8]. Very recently, for instance, several cell-level models [4, 41] of meristem development (the meristem being the growing tissue of the plant) have succeeded in reproducing characteristic phyllotactic patterns observed in nature and in linking them to the circulation of auxin (a plant hormone) in this tissue. The accumulation of auxin triggers the development of new organs, which modify the form of the meristem and consequently the flow of auxin: a marvellous example of $(\mathrm{DS})^2$.

However, no matter how predictive these simulations are, they can only have an explanatory value if they allow us to express the processes of development in a form

that is intelligible to the human mind, so that we can analyse them and reason about them [29]. After all, what kind of understanding can we hope to derive from the simple observation of a succession of complex calculations? We might just as well observe these processes in nature, instead of reproducing them on a computer.

Computer morphogenesis allows us to define a formal framework, in which we can speak rigorously of genetic programme, memory, information, signal, interaction, environment, etc. and to relate these concepts to a completely mechanistic view of development processes. It introduces the concept of *computation* as an explanatory scheme in the modelling of development. But if the embryo can be deduced by computation from a description of the egg and its interactions with the environment, the embryo must be considered both as the result of a computation and as part of the computer that produces this result. This problem is studied in computer science (reflexive interpreters, meta-circular evaluators). The future will tell whether these concepts will enable us to grasp that most specific aspect of living beings: their development.

## Giving form to a population of autonomous agents

The modelling of development processes is important for biologists, but it is also important for computer scientists, who are always looking for new computational models and for whom biology is clearly a great source of inspiration.

Computational models are constrained by the particularities of a material model or inspired by a metaphor of what a computation should be. Today, new material supports for computation are being studied. One celebrated example is the experiment that Adleman performed in 1994 [1], proving that a combinatorial problem[7] can be solved using DNA molecules in a test-tube. But other possibilities are currently the subject of very active research, including using the growth of colonies of bacteria, the diffusion of chemical reagents or the self-assembly of biomolecules . . . to compute. The programming of these new computational supports certainly raises some substantial problems, and is driving the development of new languages and algorithms to allow us to use an immense population of autonomous entities (biomolecules, viruses or cells) that interact locally and irregularly, to construct and develop a reliable computation (a form).

But the mechanisms offered by a programming language, or by new algorithms, can also be directly inspired by a biological metaphor without resorting to biological machines built using biotechnologies. For example, evolutionary algorithms are inspired by the mechanisms studied in evolutionary theory, even though they are executed on electronic machines like present-day computers.

In the same order of idea, formalisms providing a conceptual grasp of the mechanisms of development could well revitalise the concept of “programme”, by suggesting new approaches in the development of very big software, notably in the

[7] The problem he chose was the Hamiltonian path problem, consisting in determining whether a given graph contains a path that starts at the first vertex, ends at the last vertex, and passes exactly once through each remaining vertex.

specification of their architecture and the interconnection of their different parts, or by offering new mechanisms for hiding useless information, abstracting details or capitalising and reusing code. Computer scientists are actively seeking, for their software, properties usually attributed to living matter: autonomy, adaptability, self-repair, robustness, self-organisation. Clearly, the dialogue between computing and biology [28, 35], so ambiguous and so fertile, is not about to end.

## References


1. Adleman, L. (1994) Molecular computation of solutions to combinatorial problems, *Science* **266**, 1021–1024.
2. Atlan H. (1999) *La Fin du " tout-génétique" ? Vers de nouveaux paradigmes en biologie*, INRA Éditions (Paris).
3. Banatre J.-P., Coutant A., and Metayer D.L. (1988) A parallel machine for multiset transformation and its programming style, *Future Generation Computer Systems* **4**, 133–144.
4. Barbier de Reuille P., Bohn-Courseau I., Ljung K., Morin H., Carraro J., Godin C., and Traas J. (2006) Computer simulations reveal properties of the cell-cell signaling network at the shoot apex in Arabidopsis. *PNAS* **103**, 1627–1632.
5. Bournez O. and Hoyrup M. (2003) Rewriting logic and probabilities, in *14th Int. Conf. on Rewriting Techniques and Applications (RTA'03)*, Valencia, June 2003, Lecture Notes in Computer Science, vol. 2706, edited by R. Nieuwenhuis, Springer (Berlin), pp. 61–75.
6. Bournez O., Côme G.-M., Conraud V., Kirschner H., and Ibanescu L. (2003) A rule-based approach for automated generation of kinetic chemical mechanisms, in *14th Int. Conf. on Rewriting Techniques and Applications (RTA'03)*, Valencia, June 2003, Lecture Notes in Computer Science, vol. 2706, edited by R. Nieuwenhuis, Springer (Berlin), pp. 30–45.
7. Chomsky N. (ed.) (1957) *Syntactic structures*, Mouton & Co. (The Hague).
8. Coen E., Rolland-Lagan A.-G., Matthews M., Bangham J.A., and Prusinkiewicz P. (2004) The genetics of geometry, *PNAS* **101**, 4728–4735.
9. Dershowitz N. (1993) A Taste of Rewrite Systems, Lecture Notes in Computer Science, vol. 693, Springer Verlag (Berlin), pp. 199–228.
10. Dershowitz N. and Jouannaud J.-P. (1990) Rewrite systems, in *Handbook of Theoretical Computer Science*, vol. B, Elsevier (Amsterdam), pp. 244–320.
11. Dittrich P. (2001) Artificial chemistry webpage, `ls11-www.cs.uni-dortmund.de/achem`
12. Dittrich P., Ziegler J., and Banzhaf W. (2001) Artificial chemistries — a review, *Artificial Life* **7**, 225–275.
13. Eigen M. and Schuster P. (1979) *The Hypercycle: A Principle of Natural Self-Organization*, Springer (Berlin).
14. Eker S., Knapp M., Laderoute K., Lincoln P., Meseguer J., and Sonmez J. (2002) Pathway logic: Symbolic analysis of biological signaling, in *Proceedings of the Pacific Symposium on Biocomputing*, January 2002, pp. 400–412.
15. Eker S., Knapp M., Laderoute K., Lincoln P., and Talcott C. (2002) Pathway logic: Executable models of biological networks, in *Fourth International Workshop on Rewriting Logic and Its Applications (WRLA'2002)*, Electronic Notes in Theoretical Computer Science, vol. 71, Elsevier (Amsterdam).
16. The Elan project. Elan home page, 2002. `http://www.loria.fr/equipes/protheo/SOFTWARES/ELAN/`
17. Fisher M., Malcolm G., and Paton R. (2000) Spatio-logical processes in intracellular signalling, *BioSystems* **55**, 83–92.
18. Fontana W. and Buss L. (1994) "The arrival of the fittest": Toward a theory of biological organization, *Bulletin of Mathematical Biology* **56**, 1–64.
19. Fox Keller E. (1995) *Refiguring Life: Metaphors of Twentieth-century Biology*, Columbia University Press (New York).

20. Fox Keller E. (2002) *Making Sense of Life: Explaining Biological Development with Models, Metaphors, and Machines*, Harvard University Press (Cambridge MA).
21. Giavitto J.-L. (2003) Invited talk: Topological collections, transformations and their application to the modeling and the simulation of dynamical systems, in *14th Int. Conf. on Rewriting Techniques and Applications (RTA'03)*, Valencia, June 2003, Lecture Notes in Computer Science, vol. 2706, edited by R. Nieuwenhuis, Springer (Berlin), pp. 208–233.
22. Giavitto J.-L. and Michel O. (2002) The topological structures of membrane computing, *Fundamenta Informaticae* **49**, 107–129.
23. Giavitto J.-L. and Michel O. (2003) Modeling the topological organization of cellular processes, *BioSystems* **70**, 149–163.
24. Gillespie D.T. (1977) Exact stochastic simulation of coupled chemical reactions, *The Journal of Physical Chemistry* **81**, 2340–2361.
25. Hammel M. and Prusinkiewicz P. (1996) Visualization of developmental processes by extrusion in space-time, in *Proceedings of Graphics Interface '96*, pp. 246–258.
26. Head T. (1987) Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors, *Bull. Math. Biology* **49**, 737–759.
27. Head T. (1992) Splicing schemes and DNA, in *Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology*, Springer (Berlin), pp. 371–383. Reprinted in *Nanobiology* **1**, 335–342 (1992).
28. INTERSTICE (web site presenting research activity in the domain of the science and techniques of information and communication), *Dossier sur la bio-informatique*, in French. `http://interstices.info/display.jsp?id=c_6474`
29. Israel G. (1996) *La mathématisation du réel*, Seuil (Paris), in French.
30. Itkis Y. (1976) *Control Systems of Variable Structure*, Wiley (New York).
31. Lemerle C., Di Ventura B., and Serrano L. (2005) Space as the final frontier in stochastic simulations of biological systems, *Minireview. FEBS Letters* **579**, 1789–1794.
32. Lindenmayer A. (1968) Mathematical models for cellular interaction in development, Parts I and II, *Journal of Theoretical Biology* **18**, 280–315.
33. The Maude project, Maude home page, 2002. `http://maude.csl.sri.com/`
34. von Neumann, J. (1966) *Theory of Self-Reproducing Automata*, Univ. of Illinois Press (Urbana-Champaign).
35. Paton R. (ed.) (1994) *Computing With Biological Metaphors*, Chapman & Hall (London).
36. Păun G. (2002) *Membrane Computing. An Introduction*, Springer-Verlag, 2002.
37. Prusinkiewicz P. (1998) Modeling of spatial structure and development of plants: a review, *Scientia Horticulturae* **74**, 113–149.
38. Prusinkiewicz P. (1999) A look at the visual modeling of plants using L-systems, *Agronomie* **19**, 211–224.
39. Prusinkiewicz P. and Hanan J. (1990) Visualization of botanical structures and processes using parametric L-systems, in *Scientific visualization and graphics simulation*, edited by D. Thalmann, J. Wiley & Sons (Chichester), pp. 183–201.
40. Prusinkiewicz P., Lindenmayer A., Hanan J., *et al.* (1990) *The Algorithmic Beauty of Plants*, Springer-Verlag (Berlin).
41. Smith R.S., Guyomarc'h S., Mandel T., Reinhardt D., Kuhlemeier C., and Prusinkiewicz P. (2006) A plausible model of phyllotaxis, *PNAS* **103**, 1301–1306.
42. Takahashi K., Vel Arjunan S.N. and Tomita M. (2005) Space in systems biology of signaliting pathways — towards intracellular molecular crowding in silico, *Minireview. FEBS Letters* **579**, 1783–1788.
43. Varela F.J. (1979) *Principle of Biological Autonomy*, McGraw-Hill/Appleton & Lange (New York).
44. Wilcox M., Mitchison G.J., and Smith R.J. (1973) Pattern formation in the blue-green alga, *Anabaena*. I. Basic mechanisms, *Journal of Cell Science* **12**, 707–723.