A Macroscopic View of Self-Replication

DANIEL MANGE, MEMBER, IEEE, ANDRÉ STAUFFER, MEMBER, IEEE, LEONARDO PEPAROLO, AND GIANLUCA TEMPESTI, MEMBER, IEEE

Contributed Paper

In 1953, Crick and Watson published their landmark paper revealing the detailed structure of the DNA double helix. Several years earlier, von Neumann embedded a very complex configuration, a universal interpreter–copier, into a cellular array. Astoundingly, the structure of this configuration, able to realize the self-replication of any computing machine, including a universal Turing machine, shares several common traits with the structure of living cells as defined by Crick and Watson's discovery.

To commemorate the 100th anniversary of von Neumann's birth, this paper presents a macroscopic analysis of self-replication in computing machines using three examples. After describing self-replication in von Neumann's universal interpreter–copier, we will revisit the famous self-replicating loop designed by Langton in 1984. In order to overcome some of the major drawbacks of Langton's loop, namely, its lack of functionality and the fact that it is ill-adapted for a realization in electronic circuits, we present a novel self-replicating loop, the Tom Thumb loop. Endowed with the same capabilities as von Neumann's interpreter–copier, i.e., the possibility of replicating computing machines of any complexity, our loop is moreover specifically designed for the implementation of self-replicating structures in programmable digital logic.

Keywords*—Cellular automata, emergence, John von Neumann, Lindenmayer system (L-system), self-replicating loop, self-replication, Tom Thumb algorithm.*

I. INTRODUCTION

Several years before the publication of the historical paper by Crick and Watson [1] revealing the detailed structure of the DNA double helix, von Neumann was already able to point out that a self-replicating machine required the existence of a one-dimensional (1-D) description, the genome, and a universal constructor able to both interpret (translation process) and copy (transcription process) the genome in order to produce a valid daughter organism [2]. Self-replication allows not only the division of a mother cell (artificial or living) into two daughter cells, but also the growth and repair of a complete organism. Self-replication is now considered as a central mechanism indispensable for those circuits that will be implemented through the nascent field of nanotechnologies [3]–[5].

In our laboratory, we have traditionally been interested in the self-replication of digital circuits. The main goal of this paper is to present a macroscopic analysis of three major families of self-replicating machines: the historical self-replicating cellular automaton of von Neumann, the self-replicating loop due to Langton, and the more recent self-replicating loop implemented by the so-called Tom Thumb algorithm. Thanks to a new two-dimensional (2-D) Lindenmayer system (L-system) (see Appendix), we will compare the basic mechanisms of the three systems under consideration, pointing out the emergence of high-level behaviors produced by the interaction of a myriad of microscopic components, the basic cells of the cellular automaton.

The machines described in this paper are all based on the general hypotheses laid down for von Neumann's cellular automaton.

1) The automaton deals exclusively with the flow of information; the physical material (in our case, a silicon substrate) and the energy (power supply) are given *a priori*.
2) The physical space is 2-D and as large as desired (it is theoretically infinite for von Neumann's and Langton's automata, but the Tom Thumb loop is able to grow in a finite surface).
3) The physical space is *homogeneous*, that is, composed of identical cells, all of which have the same internal architecture and the same connections with their neighbors; only the *state* of a cell (the combination of the values in its memories) can distinguish it from its neighbors.
4) Replication is considered as a special case of growth: this process involves the creation of an identical organism by duplicating the genetic material of a mother entity onto a daughter one, thereby creating an exact clone.

Manuscript received May 13, 2004; revised September 1, 2004. This work was supported in part by the Swiss National Science Foundation under Grant 20-100049.1 and by the Leenaards Foundation, Lausanne, Switzerland. The work of G. Tempesti is supported by a grant from the government of Switzerland.

The authors are with the Logic Systems Laboratory, Swiss Federal Institute of Technology, Lausanne CH-1015, Switzerland (e-mail: daniel.mange@epfl.ch; andre.stauffer@epfl.ch; pandora2378@hotmail.com; gianluca.tempesti@epfl.ch).

Digital Object Identifier 10.1109/JPROC.2004.837631

In Section II, a very simple 2-D L-system [6], [7] is introduced to describe the double mechanism of interpreting and copying the description of any computing machine; this mechanism will allow us to characterize self-replication according to von Neumann's theory. Section III recalls the behavior of Langton's self-replicating loop, which will be described by another 2-D L-system. Section IV is devoted to the new Tom Thumb universal self-replicating loop, described by both a 1-D and a 2-D L-system. Section V presents a general comparison of critical features of the three models under consideration, notably including a calculation of the number of daughter organisms produced at each time step of the self-replication process (thus providing an estimate of the performance of the three systems in terms of speed). The microscopic and macroscopic models of each self-replicating system are then considered from the point of view of emergent behavior. Finally, Section VI concludes the paper with a summary of the most relevant features of the Tom Thumb loop for the implementation of self-replication in electronic devices.

II. Von Neumann's Self-Replicating Automaton

A. Overview

Existing approaches to the self-replication of computing systems are essentially derived from the work of von Neumann [2], who pioneered this field of research. Unfortunately, the state of the art in the 1950s restricted von Neumann's investigations to a purely theoretical level, and the work of his successors mirrored this constraint. In this section, we will analyze von Neumann's research on the subject of self-replicating computing machines and, in particular, his *universal constructor*, a self-replicating cellular automaton.

The computers von Neumann was familiar with were based on vacuum-tube technology, much more prone to failure than modern transistors. Confronted with this lack of reliability, he turned to nature to find inspiration in the design of fault-tolerant computing machines. Natural systems are among the most reliable complex systems known to man, and their reliability is a consequence not of any particular robustness of the individual cells (or organisms), but rather of their extreme redundancy. The basic natural mechanism which provides such reliability is self-replication, both at the cellular level (where the survival of a single organism is concerned) and at the organism level (where the survival of the species is concerned).

Thus, von Neumann, drawing inspiration from natural systems, attempted to develop an approach to the realization of self-replicating computing machines (which he called *artificial automata*, as opposed to *natural automata*, that is, biological organisms). In order to achieve his goal, he imagined a series of five distinct models for self-reproduction: the kinematic model, the cellular model, the excitation–threshold–fatigue model, the continuous model, and the probabilistic model. However, of all these models, the only one von Neumann developed in some detail was the cellular model, which became the basis for the work of his successors and is the subject of this section.

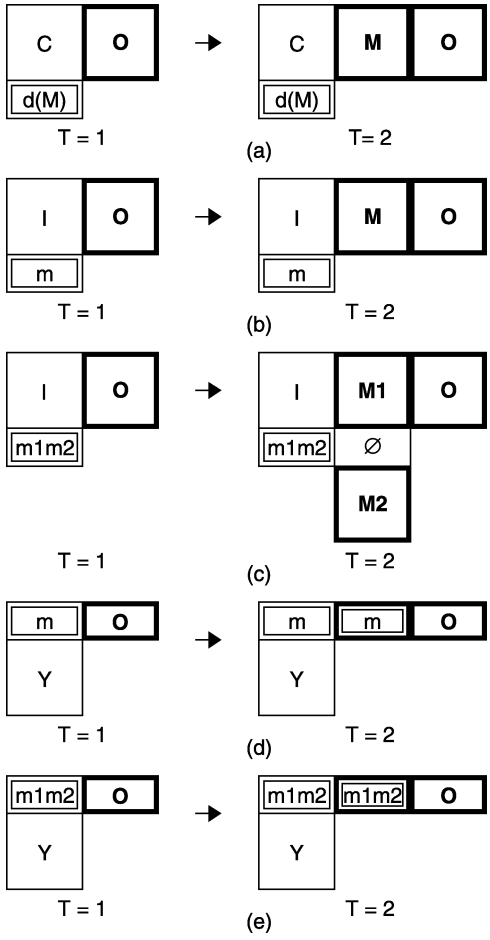


Fig. 1. Von Neumann self-replicating automaton. (a) Constructing a computing machine `M` given by its description `d(M)` (`C`: constructor). (b) Interpreting a description `m` in order to build a computing machine `M` (`I`: interpreter). (c) Interpreting a concatenated description `m1m2` in order to build two computing machines `M1` and `M2` (`I`: interpreter). (d) Copying the description `m` of a computing machine `M` (`Y`: copier). (e) Copying the concatenated description `m1m2` of two computing machines `M1` and `M2` (`Y`: copier).

B. Construction

Starting with a 29-state cell and a five-cell neighborhood, von Neumann was able to show that a specific configuration (a set of cells in a given state) could implement a *constructor* `C` able to transform the 1-D description `d(M)` of any computing machine into the actual 2-D machine `M`. According to Fig. 1(a), this transformation can be described by a very simple 2-D L-system defined by an axiom and a set of rewriting rules or productions.

In this trivial example, there exists in fact one single rule (rule #1) described by the following 2-D expression:

$$\begin{array}{c} \mathrm{C} \\ \mathrm{d(M)} \end{array} \quad < \quad \mathrm{O} \quad \rightarrow \quad \mathrm{M} \quad \mathrm{O} \tag{1}$$

which indicates that an empty character `O`, situated at the right side of the constructor `C`, itself placed at the north side of a tape containing the description `d(M)`, will produce at the next time step a specimen of the machine `M` [Fig. 1(a)].

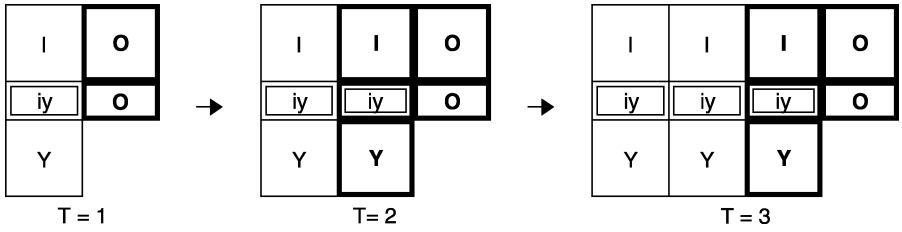


Fig. 2. Von Neumann self-replicating automaton. Self-replication of an interpreter–copier IY given by its concatenated description iy.

The axiom, at time step $T = 1$, is simply

$$\begin{matrix} \mathrm{C} & \mathrm{O} \\ \mathrm{d(M)} & \end{matrix} \tag{2}$$

while the first and unique production, at time step $T = 2$ after applying rule #1, is

$$\begin{matrix} \mathrm{C} & \mathrm{M} & \mathrm{O} \\ \mathrm{d(M)} & & \end{matrix}. \tag{3}$$

The constructor itself is not a trivial machine M, as it includes a tape d(M) describing the machine M to be built. We are, therefore, led to describe the constructor with an expression such as C[d(M)], the contents of the brackets being the description of the machine M.

The construction of the constructor, i.e., the process needed to obtain self-replication, implies the existence of a tape describing the constructor, including its own description, that is, its tape. Such a tape can be described by the following expression:

$$d(C[d(C[d\ldots \tag{4}$$

where an infinite regression may be observed.

In order to remove this infinite regression, von Neumann decided to split the construction process into two successive mechanisms:

1) an *interpretation* (or translation) mechanism carried out by an *interpreter* I able to build any computing machine M given a tape containing its description d(M) (where M can be the constructor itself, but with an empty tape);
2) a *copying* (or transcription) mechanism carried out by a *copier* Y able to realize a copy of the contents of the tape, i.e., the description d(M) of any computing machine M (including the description of the constructor itself, with its empty tape).

C. Interpretation

If we simply refer to the 1-D description of any computing machine M by m, the interpretation mechanism is described by a single rule (rule #2) exactly similar to rule #1 used above for defining the construction process [Fig. 1(b)]

$$\begin{matrix} \mathrm{I} & < & \mathrm{O} & \rightarrow & \mathrm{M} & \mathrm{O} \\ \mathrm{m} & & & & & \end{matrix}. \tag{5}$$

This rule can be modified slightly in order to transform a concatenated description m1m2 designed to build two computing machines M1 and M2 [rule #3: Fig. 1(c)]

$$\begin{matrix} \mathrm{I} & < & \mathrm{O} & \rightarrow & \mathrm{M1} & \mathrm{O} \\ \mathrm{m1m2} & & & & \varnothing & \\ & & & & \mathrm{M2} & \end{matrix}. \tag{6}$$

where $\varnothing$ designates a *don't care* condition, i.e., an empty space that can be used for writing a character produced by another rewriting rule.

D. Copying

The copying mechanism is described by a single rule [rule #4: Fig. 1(d)]

$$\begin{matrix} \mathrm{m} & < & \mathrm{O} & \rightarrow & \mathrm{m} & \mathrm{O} \\ \mathrm{Y} & & & & & \end{matrix} \tag{7}$$

which can be modified in order to transform the concatenated description m1m2 of two computing machines M1 and M2 [rule #5: Fig. 1(e)]

$$\begin{matrix} \mathrm{m1m2} & < & \mathrm{O} & \rightarrow & \mathrm{m1m2} & \mathrm{O} \\ \mathrm{Y} & & & & & \end{matrix}. \tag{8}$$

E. Self-Replication

Combining both interpretation and copying mechanisms on a concatenated description, i.e., applying both rules #3 (6) and #5 (8) to the combination of the interpreter I and the copier Y will produce the desired result, the complete self-replication of the original artificial organism, the interpreter–copier IY with its description iy (Fig. 2)

$$\begin{matrix} \mathrm{I} & \mathrm{O} \\ \mathrm{iy} & \mathrm{O} \\ \mathrm{Y} & \end{matrix} \rightarrow \begin{matrix} \mathrm{I} & \mathrm{I} & \mathrm{O} \\ \mathrm{iy} & \mathrm{iy} & \mathrm{O} \\ \mathrm{Y} & \mathrm{Y} & \end{matrix} \rightarrow \begin{matrix} \mathrm{I} & \mathrm{I} & \mathrm{I} & \mathrm{O} \\ \mathrm{iy} & \mathrm{iy} & \mathrm{iy} & \mathrm{O} \\ \mathrm{Y} & \mathrm{Y} & \mathrm{Y} & \end{matrix} \rightarrow \cdots \tag{9}$$

Starting from the axiom, each application of both rules #3 and #5 will give a new production, exhibiting at each time step T a new copy of the original interpreter–copier.

A generalization of this result is immediate: in order to perform the self-replication of any given computing machine M, it is sufficient to embed this machine into a subpart of the interpreter I and/or the copier Y, with the corresponding description of M included into a subpart of the interpreter description i and/or the copier description y. Such a system is generally referred to as a *universal interpreter–copier*.

The dimensions of von Neumann's interpreter–copier are substantial; it has, thus, never been physically implemented

and has been simulated only partially [8]. To the best of our knowledge, the only attempt to implement a complete specimen is the current work of Buckley [9], whose latest result specifies that the interpreter–copier (without its tape) is bounded by a region of $751 \times 1048 = 787\,048$ cells. If von Neumann and his successors Burks, Thatcher, Lee, Codd, Banks, Nourai, and Kashef demonstrated the theoretical possibility of realizing self-replicating automata with universal calculation, i.e., the capability of embedding a universal Turing machine [10], a practical implementation in silicon requires a sharply different approach. It was finally Langton, in 1984, who opened a second stage in this field of research.

III. Langton's Self-Replicating Loop

A. Microscopic View

In order to construct a self-replicating automaton simpler than that of von Neumann, Langton [11] adopted more liberal criteria: he dropped the condition that the self-replicating unit must be a universal interpreter–copier. Langton's mechanism is based on an extremely simple configuration in Codd's automaton [12] called the periodic emitter, itself derived from the periodic pulser organ in von Neumann's automaton [2]. Starting with an eight-state cell and a five-cell neighborhood, Langton proposed a configuration in the form of a loop, endowed notably of a constructing arm and of a replication program or genome, which turns counterclockwise within a sheath. After 151 time steps, the original loop (mother loop) produces a daughter loop, thus obtaining the self-replication of Langton's loop [Fig. 3(a)].

There is no universal interpretation–copying nor universal calculation: the loop does nothing but replicate itself. Langton's self-replicating loop represents, therefore, a special case of von Neumann's self-replication, as the loop is a specialized interpreter–copier capable of building, on the basis of its genome, a single type of machine: itself.

As did von Neumann, Langton emphasized the two different modes in which information is used: interpreted (translation) and copied (transcription). In his loop, translation is accomplished when the instruction signals are executed as they reach the end of the constructing arm and upon collision with other signals. Transcription is accomplished by the duplication of the signals at the start of the constructing arm. Unlike von Neumann's automaton, it is not possible to split Langton's loop into an interpreter, a copier, and a tape: both interpreter and copier are implemented by the loop as a whole. Such a system may be represented by the rewriting rule in Fig. 3(b), derived directly from Fig. 2 but including a monolithic interpreter–copier `IY`.

According to Fig. 3(a), after each offspring has been created, the constructing arm is moved 90° counterclockwise, and a new offspring is created in a different area of the space. When a loop encounters another loop residing in a potential offspring site, the arm is retracted and its path blocked with a sheath cell. When the looping instructions run into this sheath-cell blockade, they are simply erased one by one

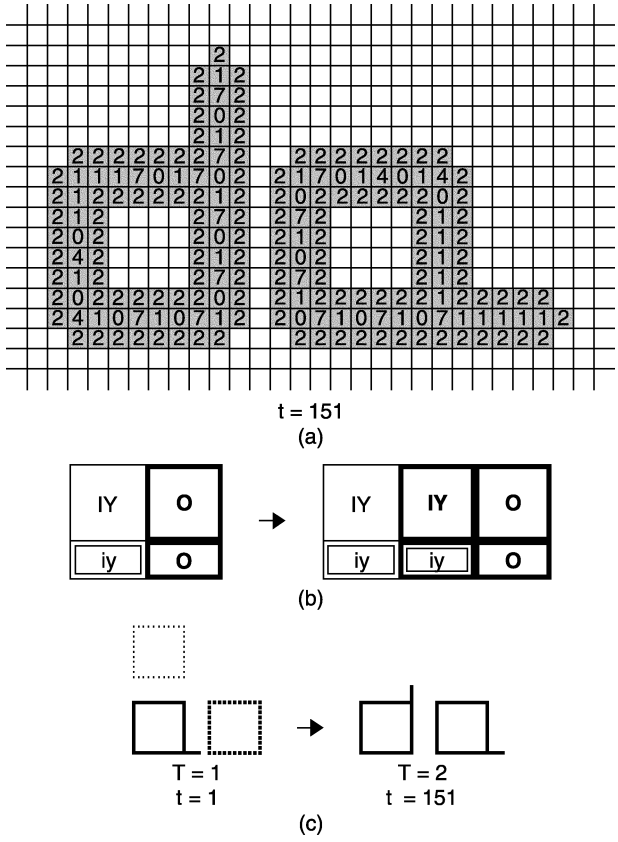


Fig. 3. Langton's self-replicating loop. (a) The genome, which turns counterclockwise, is characterized by the sequence, read clockwise: 701 701 701 701 701 701 401 401 1111; the signals 1 are ignored, the signals 70 cause the extension of the constructing arm by one molecule, while the signals 40, repeated twice, cause the arm to turn 90° counterclockwise. (b) Self-replication of a monolithic interpreter–copier `IY` and its description `iy`. (c) Macroscopic representation of the first replication of the original Langton's loop.

(dying loop), until the loop is left empty of instructions (dead loop) [11].

Thus, after a period, there will emerge an expanding *colony* of loops growing out into the array. Fig. 4(a) shows seven generations of the growth of the colony, which consists of a reproductive fringe surrounding a growing core of dying and dead loops. In an infinite cellular array, the colony would keep growing indefinitely. In a finite array, the self-replication mechanism breaks down completely upon reaching the borders.

B. Macroscopic View

As it appears from Figs. 3(c) and 4(a), the macroscopic behavior of Langton's loop can itself be represented by a new macroscopic cellular automaton based on a seven-state *macrocell* or *loop* [Fig. 4(b)]. Each of these loops may be given one out of seven states: pointing eastward (`E`), northward (`N`), westward (`W`), southward (`S`), dying (`Y`), dead, (`D`) or empty (`O`). The $\varnothing$ state (*don't care* condition) designates any one out of the six states `E`, `N`, `W`, `S`, `Y`, or `D`. The macroscopic time step T, necessary for producing a new generation of loops, is equal to 151 microscopic time steps t of the original automaton [Fig. 3(c)].

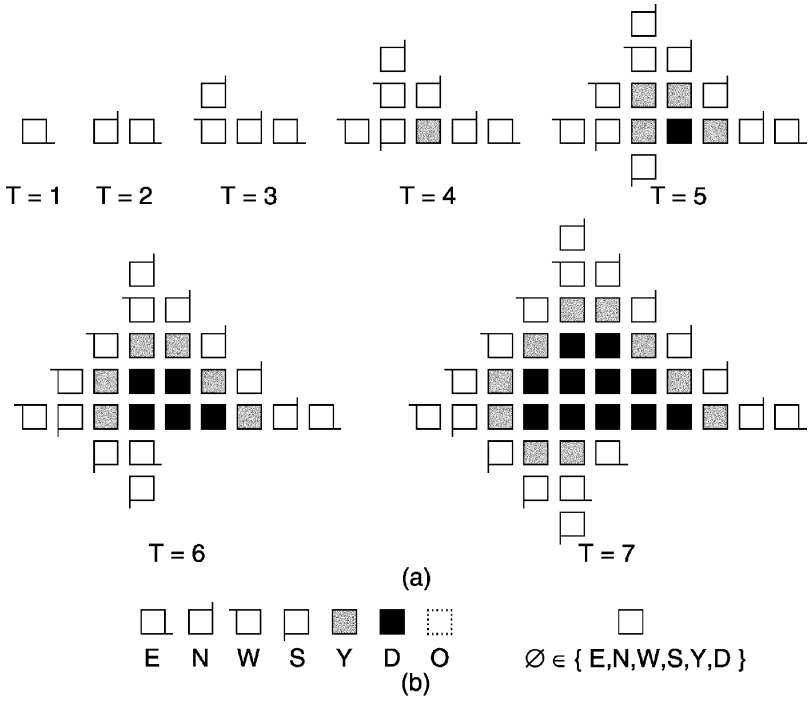


Fig. 4. Macroscopic description of Langton's loop. (a) Growth of loop colony; seven generations of growth in a colony of loops. (b) Each loop is in one out of seven states: pointing eastward (E), northward (N), westward (W), southward (S), dying (Y), dead (D), or empty (O); the ∅ state (*don't care* condition) represents any one out of the six states {E, N, W, S, Y, D}.

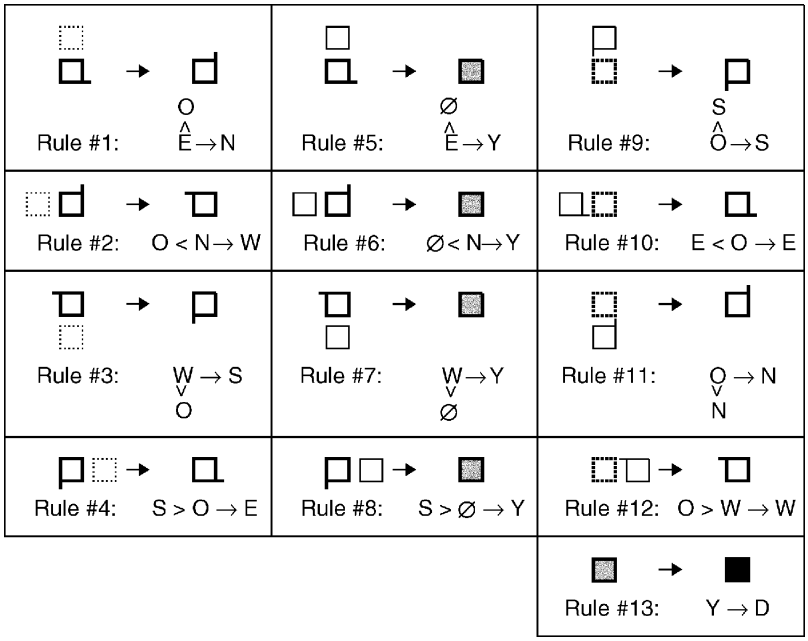


Fig. 5. Thirteen rewriting rules describing the macroscopic behavior of Langton's loop.

Each future state of such a macrocell depends on the present state of its four immediate neighbors (to the north, east, south, and west) plus its own present state, implying a transition table of $7^5 = 16\,807$ rules. While it is practically intractable to represent the entire set of transition rules by means of a state table or a state graph, a very simple 2-D L-system with only 13 rewriting rules is sufficient to completely describe the behavior of the macro automaton (Fig. 5).

Starting with the axiom describing generation 1 (Fig. 4(a), $T = 1$), it is possible to systematically derive the first four generations ($T = 2 \ldots 5$) by applying the following rewriting rules (Fig. 6).

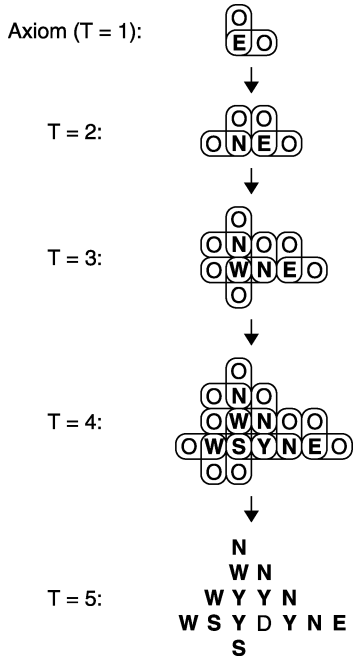


Fig. 6. Derived by means of a 2-D L-system, the first four generations of the macroscopic Langton's loop; the left part of each rewriting rule is outlined by means of an oval symbol.

- Generation 2 (time step $T = 2$) is obtained by applying rules #1 and #10 to the axiom $(T = 1)$.
- Generation 3 $(T = 3)$ is obtained by applying rules #2, #1, #11, and #10 to the derivation at time $T = 2$.
- Generation 4 $(T = 4)$ is obtained by applying rules #11, #1, #3, #2, #12, #10, and #6 to the derivation at time $T = 3$.
- Generation 5 $(T = 5)$ is obtained by applying rules #9, #8, #3, #12, #7, #2, #11, #6, #13, #1, and #10 to the derivation at time $T = 4$.

Note that for clarity, empty loops (state 0), which are missing in Fig. 4(a), have been introduced in Fig. 6 in order to facilitate the application of the rewriting rules.

In this particular 2-D L-system, 12 out of the 13 rewriting rules (Fig. 5) can be divided in two groups:

- the horizontal group (#2, #4, #6, #8, #10, and #12) in which rewriting is performed only along the horizontal axis;
- the vertical group (#1, #3, #5, #7, #9, and #11) in which rewriting is performed only along the vertical axis.

This particularity allows the horizontal and the vertical transformations to be computed separately, thus facilitating a possible mechanization of the rewriting process.

C. Toward New Self-Replicating Loops

The size of Langton's loop is perfectly reasonable, since it requires 94 cells, allowing its complete simulation. More recently, Byl [13] proposed a simplified version of Langton's automaton, while Reggia *et al.* [14] discovered that having a sheath surrounding the data paths of the genome was not essential and that its removal led to smaller self-replicating structures that also have simpler transitions functions.

All the previous loops lack any functional capabilities, their sole goal being that of self-replication. Lately, new attempts have been made to redesign Langton's loop in order to embed some calculation capabilities. Tempesti's loop [15] is a self-replicating automaton with an attached executable program that is duplicated and executed in each of the copies. Its capabilities were demonstrated using a simple program that writes out (after the loop's replication) "LSL," acronym of the Logic Systems Laboratory. Finally, Perrier *et al.*'s self-replicating loop [16] shows universal computational capabilities. The system consists of three parts: loop, program, and data, all of which undergo self-replication, followed by the program's execution on the data provided.

All these self-replicating loops lack universal interpretation–copying, i.e., the capability of constructing a 2-D computing machine of any dimensions. This is unfortunate, since this feature is of the highest interest for the development of automata in the framework of next-generation technologies, such as, for example, the three-dimensional reversible cellular automaton designed by Imai *et al.* [17] for nanoelectronic devices.

Our final goal is to show that a new algorithm, the Tom Thumb algorithm, allows the design of a self-replicating loop with universal interpretation–copying that can be implemented easily and systematically into a cellular array and is moreover compatible with the requirements of digital electronics.

IV. Tom Thumb Self-Replicating Loop

A. Microscopic View

In the past years, we have devoted considerable effort to research on self-replication, studying this process from the point of view of the design of high-complexity multiprocessor systems (with particular attention to next-generation technologies such as nanoelectronics). When considering self-replication in this context, Langton's loop and its successors share several weaknesses. Notably, besides the lack of functionality of Langton's loop (remedied only partially by its successors), which severely limits its usefulness for circuit design, each of these automata is characterized by a very loose utilization of the resources at its disposal: the majority of the elements in the cellular array remain in the quiescent state throughout the entire colonization process.

In designing a new loop, we specifically addressed these very practical issues. In fact, the system is targeted to the implementation of self-replication within the context of digital circuits realized with programmable logic devices (the states of the cellular automaton can then be seen as the configuration bits of the elements of the device). To face the drawbacks of the above-mentioned systems in this context, we have designed and verified experimentally a new loop, based on an original algorithm, the so-called *Tom Thumb algorithm* [18], [19].

The minimal loop compatible with this algorithm is made up of four cells, organized as a square of two rows by two columns [Fig. 7(c)]. Each cell is able to store in its four

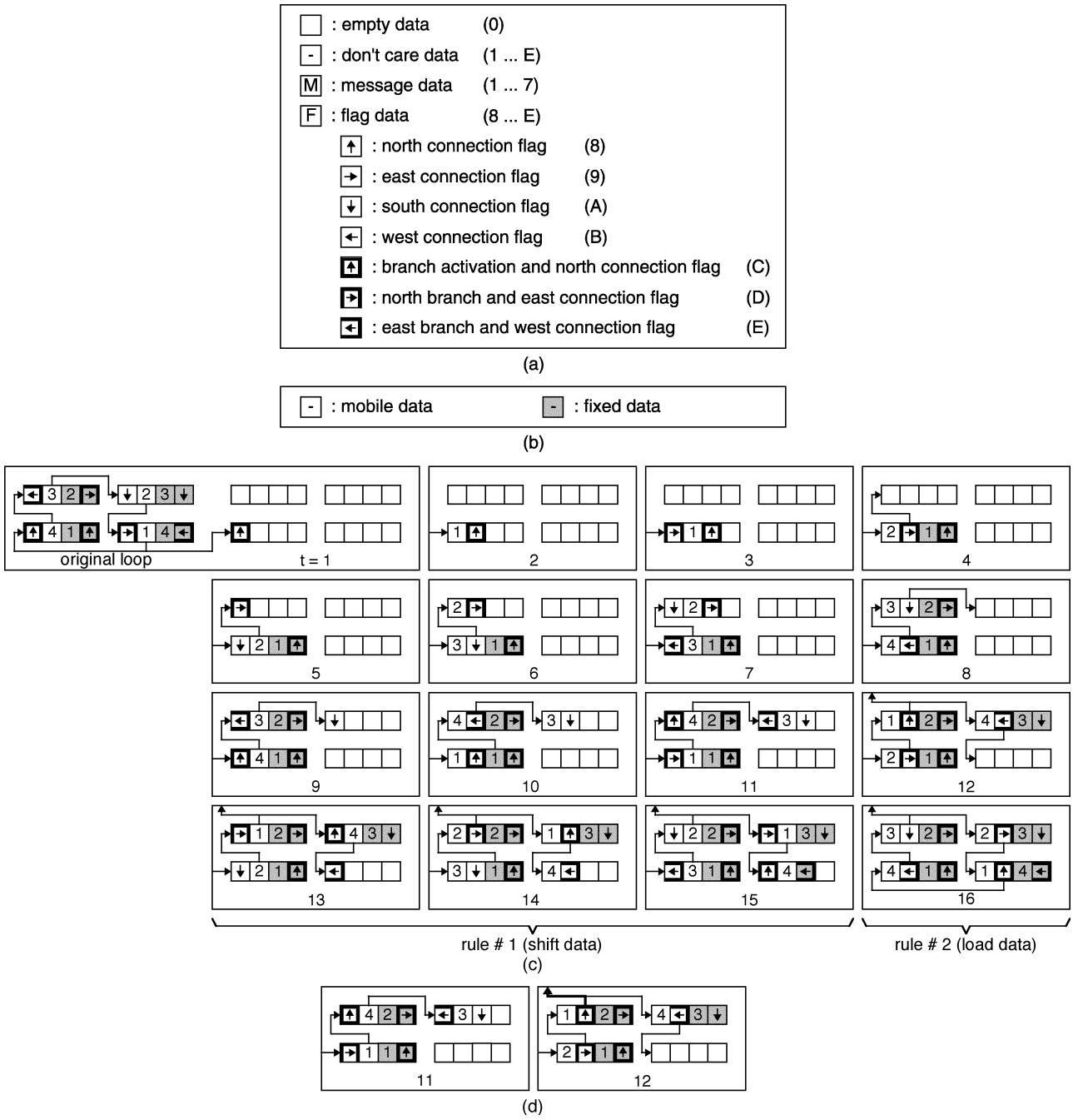


Fig. 7. Microscopic behavior of the Tom Thumb self-replicating loop. (a) Graphical and hexadecimal representations of the 15 characters forming the alphabet of the artificial genome. (b) Graphical representation of the status of each character. (c) Construction of a first specimen of the loop. (d) Creation of a new daughter loop to the north (rule #3).

memory positions four hexadecimal characters of an artificial genome. The whole loop, thus, embeds 16 such characters.

The original genome for the minimal loop is organized as another loop, the *original loop*, of eight hexadecimal characters, i.e., half the number of characters in the minimal loop, moving clockwise one character at each time step $(t = 1, 2, \ldots)$ (Fig. 7(c), $t = 1$).

The 15 hexadecimal characters that compose the alphabet of the artificial genome are detailed in Fig. 7(a). They are either *empty data* (0), *message data* $(M = 1 \ldots 7)$, or *flag data* $(F = 8 \ldots \mathrm{E})$. Message data will be used to configure our final artificial organism, while flag data are indispensable for constructing the skeleton of the loop. Furthermore, each character is given a status and will eventually be *mobile data*, moving indefinitely around the loop, or *fixed data*, definitely trapped in a memory position of a cell [Fig. 7(b)]. It is important to note that while in this simple example the message data can take a value from one to seven, the Tom Thumb algorithm is perfectly scalable in this respect; that is, the size of the message data can be increased at will, while the flag data remain constant. This is a crucial observation in view of the exploitation of this algorithm in a programmable logic

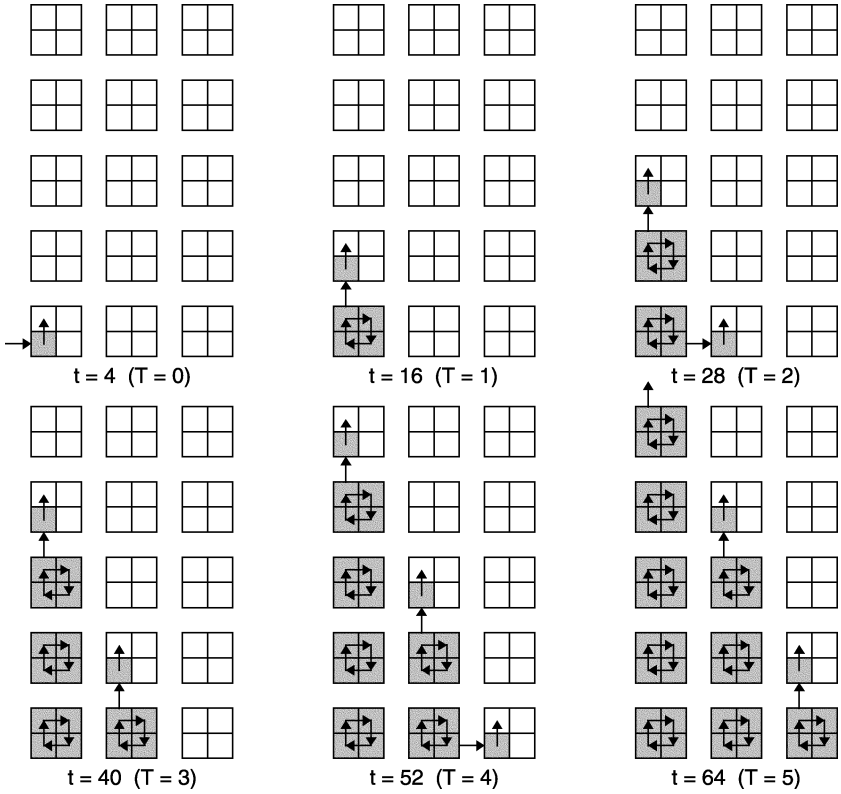


Fig. 8. Growth of a colony of minimal loops represented at different time steps (t: microscopic time step, T: macroscopic time step).

device, where the message data (the configuration data for the programmable elements of the circuit) are usually much more complex.

At each time step, a character of the original loop is sent to the lower leftmost cell [Fig. 7(c)]. The construction of the loop, i.e., storing the fixed data and defining the paths for mobile data, depends on two rules [Fig. 7(c)].

- If the four, three, or two rightmost memory positions of the cell are empty (blank squares), the characters are shifted by one position to the right (rule #1: shift data).
- If the rightmost memory position is empty, the characters are shifted by one position to the right (rule #2: load data). In this situation, the two rightmost characters are trapped in the cell (fixed data), and a new connection is established from the second leftmost position toward the northern, eastern, southern, or western cell, depending on the fixed flag information (in Fig. 7(c), at time $t = 4$, the fixed flag $F = \mathrm{C}$ determines a northern connection).

At time $t = 16$, 16 characters, i.e., twice the contents of the original loop, have been stored in the 16 memory positions of the loop. Eight characters are fixed data, forming the phenotype of the final loop, and the eight remaining ones are mobile data, composing a copy of the original genome, i.e., the genotype. Both interpretation (the construction of the loop) and copying (the duplication of the genetic information) have been, therefore, achieved.

The fixed data trapped in the rightmost memory positions of each cell remind us of the pebbles left by Tom Thumb to remember his way in the famous children's story, an analogy that gives our algorithm its name.

B. Toward a Colony of Loops

In order to grow a colony of loops in both horizontal and vertical directions, the mother loop should be able to trigger the construction of two daughter loops, northward and eastward. Two new rules are then necessary.

- At time $t = 11$ [Fig. 7(d)], we observe a pattern of characters that is able to start the construction of the northward daughter loop; the upper leftmost cell is characterized by two specific flags, i.e., a fixed flag, in the rightmost position, indicating a north branch $(F = \mathrm{D})$ and the branch activation flag $(F = \mathrm{C})$, in the leftmost position. The new path to the northward daughter loop will start from the second leftmost position $(t = 12)$ (rule #3: daughter loop to the north).
- At time $t = 23$, another particular pattern of characters starts the construction of the eastward daughter loop; the lower rightmost cell is characterized by two specific flags, i.e., a fixed flag indicating an east branch $(F = \mathrm{E})$, in the rightmost position, and the branch activation flag $(F = \mathrm{C})$, in the leftmost position (rule #4: daughter loop to the east).

When two or more loops are activated simultaneously, a clear priority should be established between the different paths. We have, therefore, chosen three growth patterns (Fig. 8) leading to four more rules.

- For loops in the lower row a collision could occur between the closing path, inside the loop, and the path

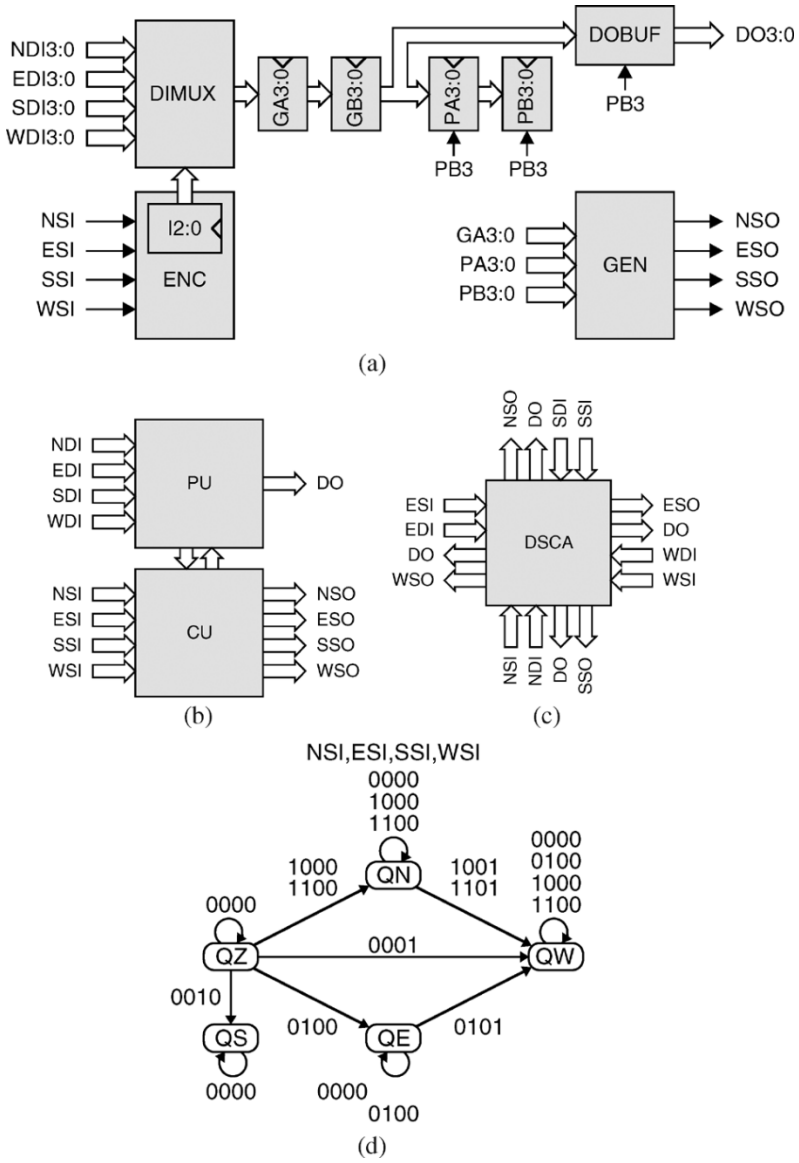


Fig. 9. Possible implementation of the basic cell as a novel DSCA. (a) Detailed architecture. (b) Macroscopic representation made up of a PU and a CU. (c) Macroscopic representation of the DSCA. (d) State graph of the finite-state machine ENC.

entering the lower leftmost cell. The westward path has priority over the eastward path (rule #5).

- With the exception of the bottom loop, the inner path (i.e., the westward path) has priority over the northward path (rule #6) for the loops in the leftmost column.
- For all other loops, two types of collisions may occur: between the northward and eastward paths (two-signal collision) or between these two paths and a third one, the closing path (three-signal collision). In this case, the northward path has priority over the eastward path (two-signal collision) and the westward path has priority over the two other ones (three-signal collision) (rules #7 and #8).

We finally opted the following hierarchy: an east-to-west path has priority over a south-to-north path, which has priority over a west-to-east path.

The results of such a choice are as follows (Fig. 8): a closing loop has priority over all other outer paths, which makes the completed loop entirely independent of its neighbors, and the colony will grow by developing its individuals bottom-up vertically. This choice is quite arbitrary and may be changed according to other specifications.

It is now possible to come back to a simplified representation (i.e., without the hexadecimal characters) of a colony of minimal loops and exhibit the latter at different time steps ($t = 4,\ 16,\ 28,\ 40,\ 52,\ 64$) in accordance with the above-mentioned priorities (Fig. 8).

C. Toward a Hardware Implementation: The Data and Signals Cellular Automaton (DSCA)

We are now able to describe the detailed architecture of our actual cell [Fig. 9(a)], which is made up of two main parts: an upper part or *processing unit* (PU) and a lower part or *control unit* (CU) [Fig. 9(b)]. The PU itself consists of three units.

- An input unit, the multiplexer DIMUX, which selects one out of the four input data (NDI3: 0, EDI3: 0,

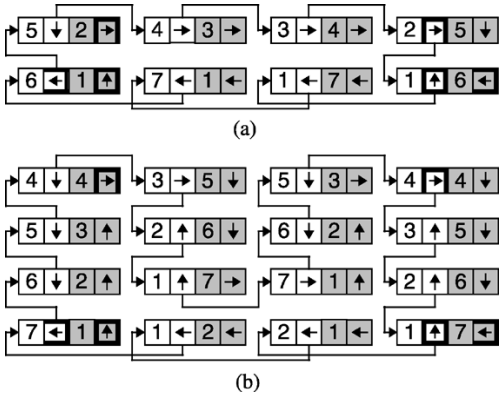


Fig. 10. Two examples of nonminimal self-replicating loops. (a) $4 \times 2 = 8$ cells loop. (b) $4 \times 4 = 16$ cells loop.

SDI3: 0, or WDI3: 0), or the empty data 0000; this selection is operated by a 3-b control signal $I2{:}\,0$.

- A four-level stack organized as two *genotypic registers* GA3 : 0 and GB3 : 0 (for mobile data), and two *phenotypic registers* PA3 : 0 and PB3 : 0 (for fixed data). The two phenotypic registers are idle (i.e., execute the HOLD operation) only when the rightmost memory position of the cell is a flag (i.e., HOLD $= PB3 = 1$).
- An output unit, the buffer DOBUF, which is either active (PB3 $= 1$, flag in the rightmost memory position) or inhibited.

The CU is itself composed of two units.

- An input encoder ENC, a finite-state machine calculating the 3-b control signal $I2{:}\,0$ from the four input signals NSI, ESI, SSI, and WSI. The specification of this machine, which depends on the priorities between cells as mentioned above, is described by the state graph of Fig. 9(d). The five internal states QZ, QN, QE, QS, and QW control the multiplexer DIMUX and select the input value 0000 or the input data NDI3: 0, EDI3: 0, SDI3: 0, or WDI3: 0, respectively.
- An output generator GEN, a combinational system producing the northward, eastward, southward, and westward signals (NSO, ESO, SSO, and WSO) according to our set of rules.

The PU and the CU as well as the final cell are represented at macroscopic levels in Fig. 9(b) and (c); these graphics define a new kind of generalized cellular automaton, the DSCA [20].

D. Generalization and Design Methodology

The self-replicating loops in Fig. 10 are two examples of nonminimal loops. Note that the message data can be used directly to display useful information, as in the "LSL" acronym example below, or can be indirectly used as a configuration string able to control a programmable device such as a field-programmable gate array (FPGA).

In [15], Tempesti has already shown how to embed the acronym "LSL" into a self-replicating loop implemented on a classical cellular automaton. Thanks to a "cut-and-try" methodology and a powerful simulator, he was able to carry out the painful derivation of over 10 000 rules for the basic cell.

Unlike in Tempesti's heuristic method, we will show that the same example can be designed in a straightforward and systematic way thanks to the use of our new DSCA associated with the Tom Thumb algorithm.

The "LSL" acronym is first represented in a rectangular array of 12 columns by six rows [Fig. 11(a)]. While the number of rows is indifferent, the number of columns should be even in order to properly close the loop [Fig. 11(b)]. The loop is, therefore, made up of $12 \times 6 = 72$ cells connected in the pattern shown in Fig. 11(b): bottom-up in the odd columns, top-down in the even columns, with the lower row reserved for closing the loop. It is then possible to define all the flags in the rightmost memory position of each cell [gray characters in Fig. 11(b)] without forgetting the branch activation and north connection flag in the lower cell of the first column, the north branch and east connection flag in the upper cell of the first column, and the east branch and west connection flag in the lower cell of the last column.

Among the 72 cells, 25 are used to display the three letters "L," "S," and "L" and are given the character "2" as message data [black data in Fig. 11(a) and (b)], while 47 are blank (message data "1").

The detailed information of the final genome, i.e., $72 \times 2 = 144$ hexadecimal characters [Fig. 11(c)], is derived by reading clockwise the fixed characters [black and gray characters in Fig. 11(b)] of the whole loop, starting with the lower cell of the first column.

Lastly, it was possible to embed the basic cell of Fig. 9(a) in each of the 2000 FPGAs of the BioWall [21] and to show the rather spectacular self-replication of our original loop (equivalent to a unicellular artificial organism), the "LSL" acronym, in both the vertical and horizontal directions [Fig. 11(d)].

The LSL acronym design example can be easily generalized to produce the following algorithm

1) Divide the given problem in a rectangular array of C columns by R rows. While the number of rows R is indifferent, the number of columns C should be even in order to properly close the loop.
2) Define all the flags in the rightmost memory position of each cell according to the following patterns: bottom-up in the odd columns and top-down in the even columns, with the lower row reserved for closing the loop.
3) Complete the path by adding the branch activation and north connection flag (C) in the rightmost memory position of the lower cell of the first column, the north branch and east connection flag (D) in the rightmost memory position of the upper cell of the first column, and the east branch and west connection flag (E) in the rightmost memory position of the lower cell of the last column, in order to trigger the two daughter loops to the north and east, respectively.
4) According to the original specifications, complete all the message data in the second rightmost memory position of each cell. These message data constitute the

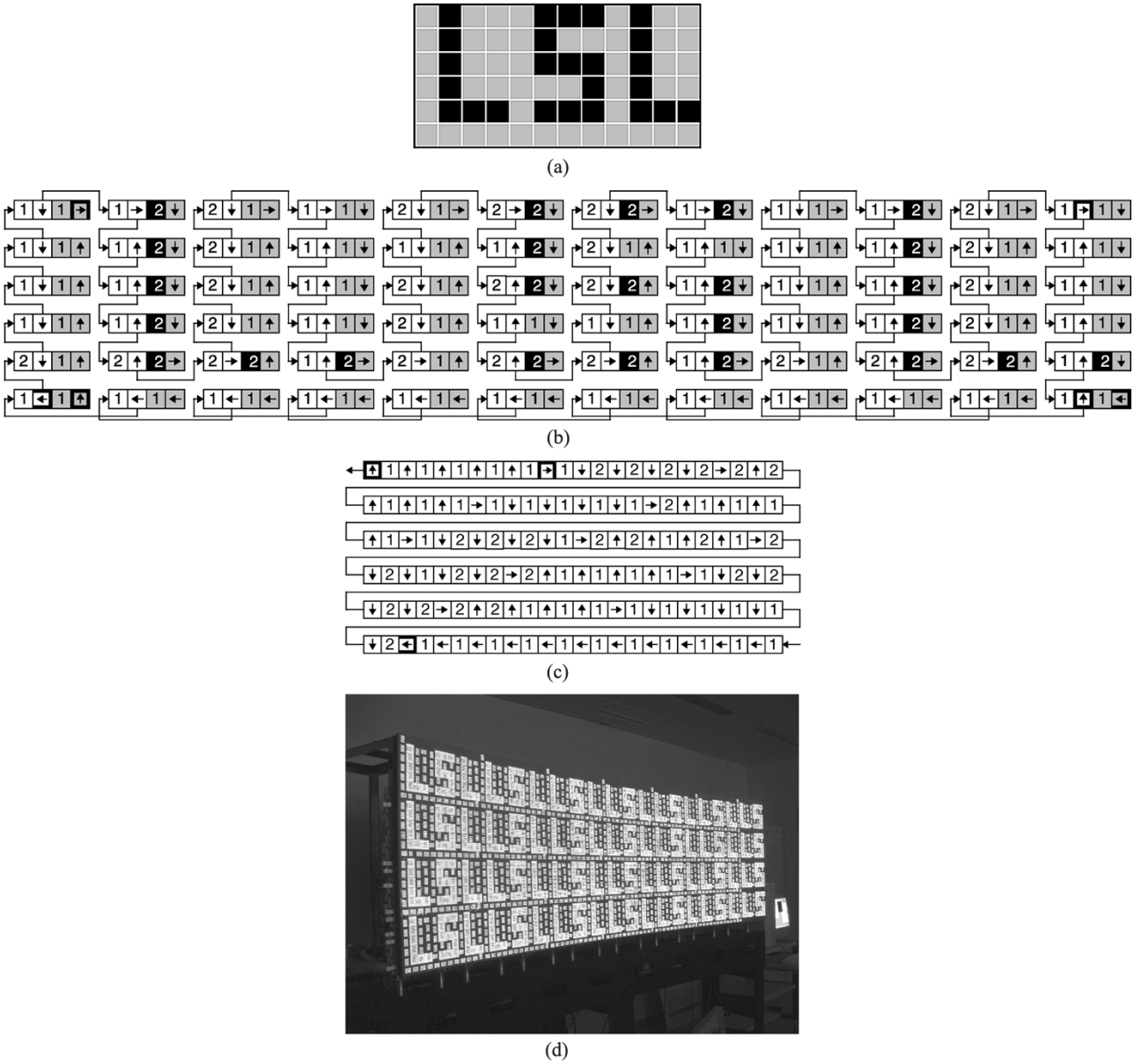


Fig. 11. Self-replication of the "LSL" acronym. (a) Original specifications. (b) The $12 \times 6 = 72$ cells of the basic loop. (c) Genome. (d) BioWall implementation displaying both the genotypic path and the phenotypic shape (photograph by E. Petraglio).

phenotypic information of the loop, equivalent to an artificial organism.

5) The detailed information of the final genome, i.e., the genotypic information of the loop, is derived by reading clockwise along the original path the fixed characters of the whole loop, i.e., the two rightmost characters of each cell, starting with the lower cell of the first column. The genotypic information, or artificial genome, is used as the configuration string of the loop and will eventually take place in the two leftmost memory positions of each cell.

E. Macroscopic View

As it appears from Fig. 8, the macroscopic behavior of the Tom Thumb loop can be itself represented by a new macroscopic cellular automaton based on a four-state *macrocell* or *loop* [Fig. 12(a)]. Each of these loops may be given one out of four states: full loop (L), pointing northward (N), pointing eastward (E), or empty (O). The macroscopic time step T, necessary for producing one or several daughter loops, is equal to 12 microscopic time steps of the original automaton (Fig. 8).

Two possible L-systems may completely describe the behavior of the macro automaton. The first is a classical 1-D L-system, the six rewriting rules of which are described in Fig. 12(b). The description of a 2-D cellular macro automaton is made possible by introducing the literals n (go to the next northern loop) and s (go to the next southern loop). Starting with the configuration at time $t = 16$ or $T = 1$ (Fig. 8), the axiom can be described by the following expression:

$$\mathrm{OnNnOssO} \tag{10}$$

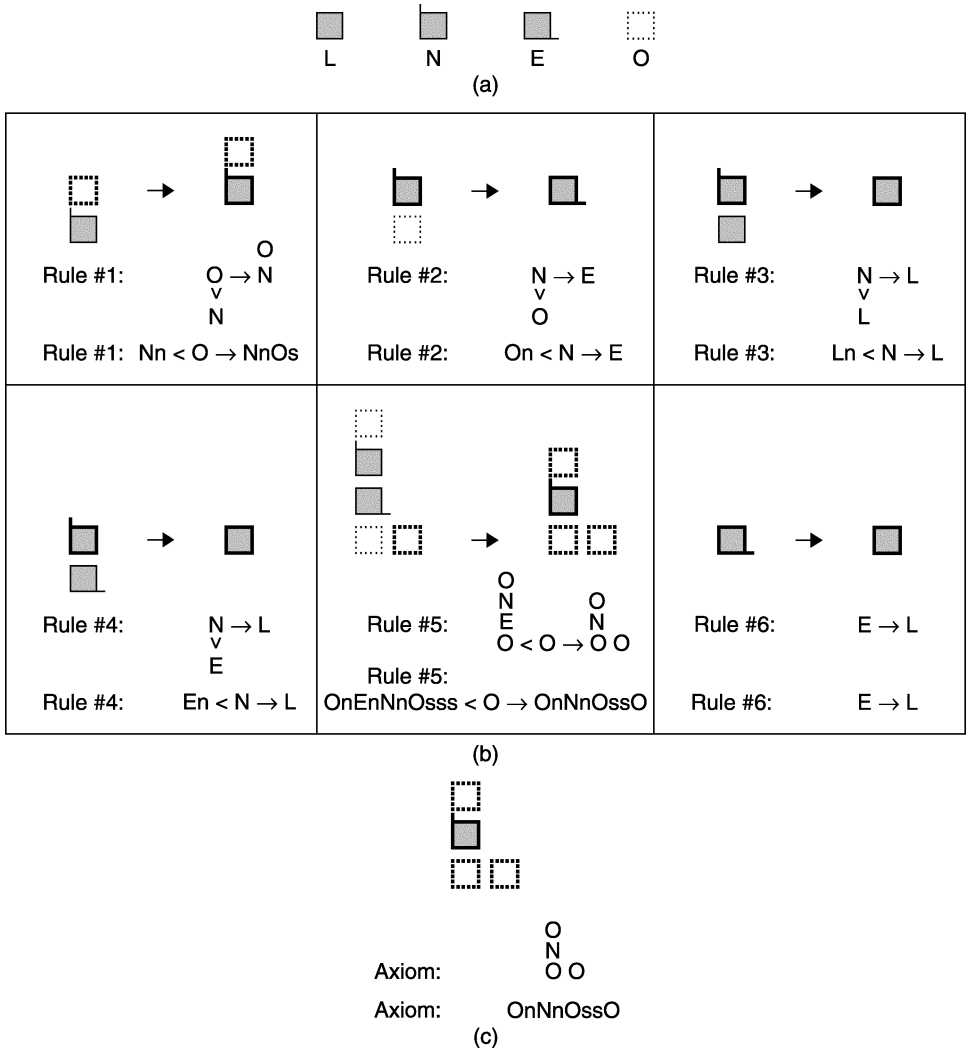


Fig. 12. Macroscopic description of Tom Thumb loop. (a) Each loop is in one out of four states: full loop (L), pointing northward (N), pointing eastward (E), or empty (0). (b) 1-D and 2-D L-systems: the six rewriting rules. (c) Axiom at $T = 1$.

which takes in account several empty loops (0) at its periphery (see also Fig. 13 at time $T = 1$).

Applying the six different rewriting rules to (10) will produce the first four derivations (Fig. 13)

$$t = 28 \ (T = 2): \quad \text{OnEnNnOsssO} \tag{11}$$

$$t = 40 \ (T = 3): \quad \text{OnLnLnNnOsssOnNnOssO} \tag{12}$$

$$t = 52 \ (T = 4): \quad \text{OnLnLnLnNnOssssOnEnNnOsssO} \tag{13}$$

$$t = 64 \ (T = 5): \quad \text{OnLnLnLnLnNnOssssssOnLnLnNnOsssOnNnOssO}. \tag{14}$$

Another 2-D L-system can also describe the axiom [Fig. 12(c)] and the six rewriting rules [Fig. 12(b)]. Applying the rules to the original axiom will produce the first four derivations of Fig. 13, where the right part of each rewriting rule is outlined by an oval symbol.

V. Discussion

A. Comparison of Growth Characteristics

Fig. 14 defines seven major parameters that can be used to compare the three self-replicating systems described in this paper: von Neumann's interpreter–copier, Langton's loop, and the Tom Thumb loop.

As for von Neumann, the precise number of cells of the mother organism, i.e., the universal interpreter–copier (without its tape) has been recently estimated [9]. The number of microscopic time steps necessary for one generation is not yet known, but the work by Buckley will certainly provide the missing figures in the near future. At each macroscopic time step T, the mother organism will produce one daughter organism, and the growth is definitely linear.

Unlike von Neumann's interpreter–copier, the growth of a colony of Langton's loops is polynomial. The number of organisms (i.e., loops) of a generation at macroscopic time T is given by the following expression:

$$N(T) = 2T^2 - 12T + 25 \quad (\text{for } T \geq 5) \tag{15}$$

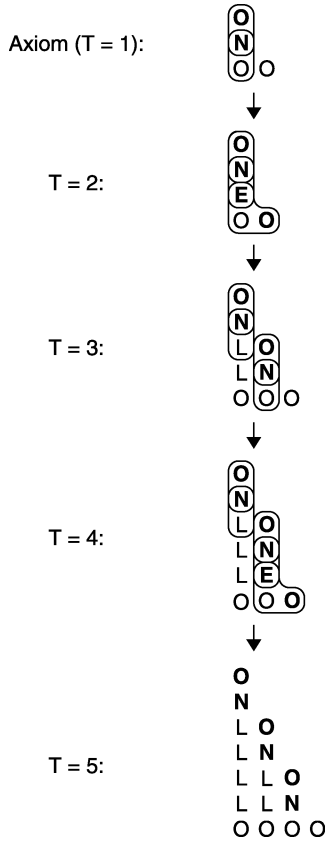


Fig. 13. 2-D L-system: axiom ($T = 1$) and four derivations ($T = 2 \ldots 5$) describing the growth of a colony of loops.

whose detailed calculation is beyond the scope of this paper.

The number of new organisms (loops) created at macroscopic time step T is given by the following expression:

$$\Delta N(T) = N(T) - N(T-1) = 4T - 14 \quad (\text{for } T \geq 6). \tag{16}$$

Due to its specific architecture (a shift register with 16 flip-flops associated to a three-variable finite-state machine), the number of states of a single Tom Thumb cell is enormous and accounts for 2^{19} states [18]. As in Langton's loop, the growth of a colony of loops is polynomial and may be described by the following four expressions, the calculation of which is again beyond the scope of this paper:

$$N(T) = \frac{(T^2 + 2T + 1)}{4}, \qquad \text{if } T \text{ odd} \tag{17}$$

$$N(T) = \frac{\left(\frac{T^2}{2} + T\right)}{2}, \qquad \text{if } T \text{ even} \tag{18}$$

$$\begin{aligned} \Delta N(T) &= N(T) - N(T-1) \\ &= \frac{(T-1)}{2} + 1, \qquad \text{if } T \text{ odd} \end{aligned} \tag{19}$$

$$\begin{aligned} \Delta N(T) &= N(T) - N(T-1) \\ &= \frac{T}{2}, \qquad \text{if } T \text{ even.} \end{aligned} \tag{20}$$

The polynomial growth of both Langton's loop and Tom Thumb loop is of major interest to achieve a rapid colonization of large cellular arrays, such as those that could potentially be provided by the introduction of nanoelectronic devices.

B. Comparison of Genome Size

Comparing our current artificial systems with those found in nature would be an extremely complex task, requiring the development of a sophisticated scheme, involving a set of comparison criteria and measures. While we shall not attempt such a comparison in this paper, it is nonetheless interesting to consider a single comparison, that of genome size. Fig. 15 presents the genome sizes of some natural and artificial organisms, clearly demonstrating our present, comparatively primitive, state. Even the most complex of the artificial systems described in this paper (in term of genome size)—namely, von Neumann's interpreter–copier—is smaller than a simple bacterium (*Escherichia coli*) [22].

What is the smallest molecular population that is able to constitute a self-replicating system? Two famous experiments, by Spiegelman and Eigen [23], finally produced artificial viruses able to self-replicate. The so-called *Spiegelman monster* and *Eigen replicator* have genome sizes of, respectively, 440 and 240 b, a figure not so far from that of Langton's loop. The experiments of Spiegelman and Eigen together give a clear answer to the question: a single RNA molecule with 100 or 200 nucleotides, i.e., with 200–400 b. It may be conjectured that this estimation is of a purely logical nature, and can also be applied to artificial organisms such the Langton's and Tom Thumb self-replicating loops.

C. Microscopic and Macroscopic Models Viewed as an Emergent Behavior

A behavior or structure is usually considered as emergent when the whole is more that the sum of its parts. More precisely, a property that applies at a given level is emergent if it does not apply at any lower level. Although there is nothing "magic" behind an emergent behavior, one might consider it a form of creativity, since new states in the systems' state-space are reached. One of the more illustrative examples is the *Game of Life*, where one can observe the "emergence" of gliders and other patterns from a random initial configuration [24].

Several attempts to formalize emergence were recently made. Ronald *et al.* [25], [26] proposed an emergence test. This test centers on an observer's avowed incapacity (amazement) to reconcile his perception of an experiment in terms of a global world view with awareness of the atomic nature of the elementary interactions.

The test basically consists of three steps.

- *Design*. The system is described by a designer by means of local interactions between the components in a language $L1$.
- *Observation*. The global behavior of the system (as desired by the designer) is observed and described using a language $L2$.
- *Surprise*. The observer experiences surprise when the language of design $L1$ and the language of observation

	von Neumann	Langton	Tom Thumb (minimal loop)
Number of states per cell	29	8	2^{19}
Number of cells of the neighborhood	5	5	5
Theoretical number of rules	29^5 = 20'511'149	8^5 = 262'144	2^{95}
Number of cells of the mother organism	787'048	94	4
Number of microscopic time steps per generation	?	151	16
N(T): number of organisms of a generation at macroscopic time T	T	$2T^2$ - 12T + 25 for T ≥ 5	$(T^2 + 2T + 1)/4$ if T odd $(T^2/2 + T)/2$ if T even
ΔN(T) = N(T) - N(T-1): number of new organisms per generation at macroscopic time T	1	4T - 14 for T ≥ 6	(T - 1)/2 + 1 if T odd T/2 if T even

Fig. 14. Comparison of the major parameters for von Neumann interpreter–copier, Langton's loop, and Tom Thumb loop.

Organism	Genome size [bits]
Bacterium (*Escherichia coli*)	8.10^6
Von Neumann interpreter-copier	4.10^6
Spiegelman monster	440
Eigen replicator	240
Langton's loop	84
Tom Thumb minimal loop	32

Fig. 15. Comparison of the genome size for various natural and artificial self-replicating organisms.

$L2$ are distinct, and the causal link between the elementary interactions programmed in $L1$ and the behaviors observed in $L2$ is nonobvious.

The test naturally depends on how big the surprise effect is, i.e., how easy it is for the observer to bridge the $L1 - L2$ gap.

Both the Langton's and Tom Thumb loops are good candidates for this test, as they are both described in the microscopic language $L1$ (the rules of the cell) and in the macroscopic language $L2$ (the rules of the macrocell, i.e., the loop). Figs. 4(a) and 8 are the key representations for launching a bridge between the microscopic $(L1)$ and macroscopic $(L2)$ levels. Depending on the reaction of the reader (surprise?), we could consider to have clearly recognized an emergent behavior.

It should be emphasized that this test of emergence is rather subjective and speculative. For Kubik [24], the category of surprise obscures emergent phenomena and a more profound treatment of emergence is based on the four following points:

- a multiagent system approach, the system being decomposable into parts with autonomous behavior (the agents);
- the use of the reduction principle, which states that the macro behavior of an observed system is reducible to the interactions of its components;
- heavy reliance on the fact that the whole system generates richer behavior than the sum of the behaviors of its components;
- the nonessentiality of the moment of surprise in revealing emergent behavior.

A crucial question is still open: is it possible to automatize recognition of this emergent behavior, i.e., to systematically derive the language $L2$ (the L-system) from the language $L1$ (the transition rules)? According to Kubik [24], there is some hope to use in the future "grammar systems in the theory of hierarchies of complex systems where the generated languages on one level can become parts of the alphabets for the systems on the higher levels of description. But this is now a very hard problem."

VI. Conclusion

Unlike its predecessors, the Tom Thumb loop has been developed with a specific purpose beyond the theoretical study of self-replication. We believe that in the not-so-distant future, circuits will reach densities such that conventional design techniques will become unwieldy. Should such a hypothesis be confirmed, self-replication could become an invaluable tool, allowing the engineer to design a single processing element, part of an extremely large array that would build itself through the duplication of the original element.

Current technology does not, of course, provide a level of complexity that would render this kind of process necessary. However, programmable logic devices (already among the most dense circuits on the market) can be used as a first approximation of the kind of circuits that will become available

in the future. Our loop is then targeted to the implementation of self-replication on this kind of device.

To this end, our loop introduces a number of features that are not present in any of the historical self-replicating loops we presented. Most notably, the structure of the loop (that is, the path used by the configuration data) is determined by the sequence of flags in the genome, implying that structures of almost any shape and size can be constructed and replicated using this algorithm, as long as the loop can be closed and that there is space for the daughter organisms. In practice, this implies that if the Tom Thumb algorithm is used for the configuration logic of a programmable device, any of its configurations and, hence, any digital circuit can be made capable of self-replication.

In addition, particular care was given to develop a self-replication algorithm that is *efficient* (in the sense that it fully exploits the underlying medium, rather than leaving the vast majority of elements inert, as past algorithms did), *scalable* (all the interactions between the elements are purely local, implying that organisms of any size can be implemented), and amenable to a *systematic design process*. These features are important requirements for the design of highly complex systems based on either silicon or molecular-scale components [28].

Always in this context, we are currently working on introducing some degree of fault tolerance in the algorithm: the loop will then be able to develop in a faulty substrate (a key feature from the point of view of nanoelectronics and of increasing importance even for silicon), using a set of spare cells to avoid the faulty areas of the circuit [27].

Of course, the overhead implied by the algorithm is significant, probably limiting its commercial usefulness for current programmable devices. However, if we take into account the ever-increasing densities achieved in integrated devices and the probable development of molecular-scale electronics, the negative impact of this overhead is likely to be reduced and compensated by the features made possible by the algorithm.

Appendix
Lindenmayer Systems

Lindenmayer systems (L-systems) were originally conceived as a mathematical theory of plant development [6], [7]. The central concept of L-systems is that of rewriting, which is essentially a technique for defining complex objects by successively replacing parts of a simple initial object using a set of *rewriting rules* or *productions*. Generally, L-systems operate on character strings and productions are applied in parallel to all letters in a given string. This behavior reflects the biological inspiration of L-systems: productions are intended to mimic cell divisions in multicellular organisms, where many divisions may occur at the same time.

As a simple example of an L-system, consider strings (words) built of two letters, A and B. Each letter is associated with a rewriting rule. The rule $A \rightarrow AB$ means that the letter A is to be replaced by the string AB, and the rule $B \rightarrow A$ means that the letter B is to be replaced by the letter A [7]. The rewriting process starts from a distinguished

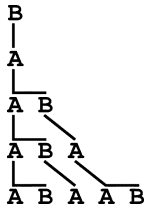


Fig. 16. Example of a derivation in a context-free L-system. The set of productions (or rewriting rules) is $\{A \rightarrow AB, B \rightarrow A\}$. The process is shown for four derivation steps.

```
p1: U<A>A -> U
p2: U<A>X -> DA
p3: A<A>D -> D
p4: X<A>D -> U
p5:   U   -> A
p6:   D   -> A

 0: XUAX      5: XADAAX
 1: XADAX     6: XUAAAX
 2: XUAAX     7: XAUAAX
 3: XAUAX     8: XAAUAX
 4: XAADAX    9: XAAADAX
```

Fig. 17. Context-sensitive L-system. (a) Production set. (b) Sample derivation. Note that if no rule applies to a given letter, then the letter remains unchanged.

string called the *axiom*. For example, let the axiom be the single letter B. In the first derivation step (the first step of rewriting), axiom B is replaced by A using production $B \rightarrow A$. In the second step, production $A \rightarrow AB$ is applied to replace A with AB. In the next derivation step, both letters of the word AB are replaced *simultaneously*: A is replaced by AB and B is replaced by A. The behavior of the system is shown in Fig. 16 for four derivation steps.

In the above example, the productions are *context free*, i.e., applicable regardless of the context in which the predecessor appears. However, production application may also depend on the predecessor's context, in which case the system is referred to as *context sensitive*. This allows for interactions between different parts of the growing string (modeling, for example, interactions between parts of a plant). Several types of context-sensitive L-systems exist, the following being just one example.

Context-sensitive productions include, as mentioned, the context of the predecessor. For example, the production $U < A > X \rightarrow DA$ replaces the letter A (called the *strict predecessor*) with the string DA if and only if A is preceded by the letter U and followed by the letter X. Thus, letters U and X form the context of A in this production. When the strict predecessor has a one-sided context, to the left or to the right, only the < or the > symbol, respectively, is used (e.g., $U < A \rightarrow DA$ is a left-context rule and $A > X \rightarrow DA$ is a right-context one). Fig. 17 shows a context-sensitive L-system. If the *growth* of the system is defined as a function of the number of symbols in a word in terms of the number of derivation steps, then this L-system exhibits square-root growth: after n derivation steps, the length of the string (not counting the X symbols, which represent the beginning and end of the string) is $\lfloor \sqrt{n} \rfloor + 2$. Other growth functions can also be obtained, including polynomial, sigmoidal, and exponential [7].

A direct extension of context-sensitive L-systems is possible in the framework of a two-dimensional space. In this case, the context must be extended: in addition to the left and

right (west and east, to use a terminology more appropriate for two-dimensional spaces) neighbors, the context can include the letters above and below (north and south neighbors) the strict predecessor. The notation is modified accordingly

$$U < \begin{matrix} R \\ \wedge \\ A \\ \vee \\ T \end{matrix} > S \quad \rightarrow BC \tag{21}$$

denotes a production where the strict predecessor A is replaced by the string BC if and only if its north, east, south, and west neighbors (i.e., its context) are the letters R, S, T, and U, respectively.

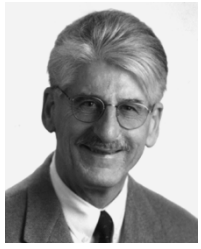
L-systems are a powerful tool for modeling and describing growth processes. In this paper, they are used to describe the growth of colonies of self-replicating loops, a process that is quite complex to describe with some of the other common formalisms such as state tables, state graphs, or transition rules. The use of L-systems, on the contrary, allows this kind of growth to be described very simply by hiding the complexity of the underlying microscopic processes and concentrating on the macroscopic behavior of the system.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their invaluable contribution. The authors would also like to thank B. Buckley for providing the dimensions of the von Neumann interpreter–copier.

REFERENCES

[1] J. D. Watson and F. H. C. Crick, "Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid," *Nature*, vol. 171, pp. 737–738, 1953.
[2] J. von Neumann, *Theory of Self-Reproducing Automata*, A. W. Burks, Ed. Urbana, IL: Univ. of Illinois Press, 1966.
[3] K. E. Drexler, *Nanosystems: Molecular Machinery, Manufacturing, and Computation*. New York: Wiley, 1992.
[4] M. C. Roco and W. S. Bainbridge, Eds., "Converging technologies for improving human performance. Nanotechnology, biotechnology, information technology and cognitive science," Nat. Sci. Found./Dept. Commerce, Arlington, VA, 2002.
[5] R. A. Freitas Jr. and R. C. Merkle, *Kinematic Self-Replicating Machines*. Georgetown, TX: Landes Biosci., to be published.
[6] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. New York: Springer-Verlag, 1990.
[7] P. Prusinkiewicz and J. Hanan, *Lindenmayer Systems, Fractals, and Plants*. Berlin, Germany: Springer-Verlag, 1989.
[8] U. Pesavento, "An implementation of von Neumann's self-reproducing machine," *Artif. Life*, vol. 2, no. 4, pp. 337–354, 1995.
[9] W. R. Buckley, "On the complete specification of a von Neumann 29-state self-replicating cellular automaton," unpublished, 2004.
[10] D. Mange and M. Tomassini, Eds., *Bio-Inspired Computing Machines*. Lausanne, Switzerland: Presses Polytechniques et Universitaires Romandes, 1998.
[11] C. Langton, "Self-reproduction in cellular automata," *Physica D*, vol. 10, pp. 135–144, 1984.
[12] E. F. Codd, *Cellular Automata*. New York: Academic, 1968.
[13] J. Byl, "Self-reproduction in small cellular automata," *Physica D*, vol. 34, pp. 295–299, 1989.
[14] J. A. Reggia, S. L. Armentrout, H.-H. Chou, and Y. Peng, "Simple systems that exhibit self-directed replication," *Science*, vol. 259, pp. 1282–1287, Feb. 1993.
[15] G. Tempesti, "A new self-reproducing cellular automaton capable of construction and computation," in *Lecture Notes in Computer Science, ECAL'95: 3rd European Conference on Artificial Life*, F. Moran, A. Moreno, J. J. Merelo, and P. Chacon, Eds. Heidelberg, Germany: Springer-Verlag, 1995, vol. 929, pp. 555–563.
[16] J.-Y. Perrier, M. Sipper, and J. Zahnd, "Toward a viable, self-reproducing universal computer," *Physica D*, vol. 97, pp. 335–352, 1996.
[17] K. Imai, T. Hori, and K. Morita, "Self-reproduction in three-dimensional reversible cellular space," *Artif. Life*, vol. 8, no. 2, pp. 155–174, 2002.
[18] D. Mange, A. Stauffer, E. Petraglio, and G. Tempesti, "Self-replicating loop with universal construction," *Physica D*, vol. 191, pp. 178–192, 2004.
[19] ——, "Embryonic machines that divide and differentiate," in *Lecture Notes in Computer Science, Biologically Inspired Approaches to Advanced Information Technology*. Heidelberg, Germany: Springer-Verlag, 2004, vol. 3141, pp. 201–216.
[20] A. Stauffer and M. Sipper, "The data-and-signals cellular automaton and its application to growing structures," *Artif. Life*, vol. 10, no. 4, pp. 463–477, 2004.
[21] G. Tempesti and C. Teuscher, "Biology goes digital," *Xcell J.*, vol. 47, pp. 40–45, Fall 2003.
[22] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Perez-Uribe, and A. Stauffer, "A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 83–97, Apr. 1997.
[23] F. Dyson, *Origins of Life*. Cambridge, U.K.: Cambridge Univ. Press, 1985.
[24] A. Kubik, "Toward a formalization of emergence," *Artif. Life*, vol. 9, no. 1, pp. 41–65, 2003.
[25] E. M. A. Ronald, M. Sipper, and M. S. Capcarrère, "Design, observation, surprise! A test of emergence," *Artif. Life*, vol. 5, no. 3, pp. 225–239, 1999.
[26] ——, "Testing for emergence in artificial life," in *Lecture Notes in Computer Science, Advances in Artificial Life*, D. Floreano, J.-D. Nicoud, and F. Mondada, Eds. Berlin, Germany: Springer-Verlag, 1999, vol. 1674, pp. 13–20.
[27] E. Petraglio, "Fault Tolerant Self-Replicating Systems," Ph.D. dissertation no. 2973, Swiss Fed. Inst. Technol., Lausanne, Switzerland, 2004.
[28] L. J. K. Durbeck and N. J. Macias, "The cell matrix: An architecture for Nanocomputing," *Nanotechnology*, vol. 12, pp. 217–230, 2001.



Daniel Mange (Member, IEEE) received the M.S. and Ph.D. degrees from the Swiss Federal Institute of Technology in Lausanne, Switzerland, in 1964 and 1968, respectively.

Since 1969, he has been a Professor at the Swiss Federal Institute of Technology. He was also a Visiting Professor at the Center for Reliable Computing, Stanford University, Stanford, CA, in 1987. He is Director of the Logic Systems Laboratory. He has authored and coauthored several scientific papers in this areas, as well as the books *Microprogrammed Systems: An Introduction to Firmware Theory* (London, U.K.: Chapman & Hall, 1992) and *Bio-Inspired Computing Machines* (Lausanne, Switzerland: Presses polytechniques et universitaires romandes, 1998). His chief research interests include firmware theory (equivalence and transformation between hardwired systems and programs), cellular automata, artificial life, and embryonics (embryonic electronics).

Dr. Mange was Program Cochairman of the First International Conference on Evolvable Systems: From Biology to Hardware (ICES96), Tsukuba, Japan; General Chairman of the Second International Conference on Evolvable Systems: From Biology to Hardware (ICES98), Lausanne, Switzerland, in September 1998; General Chairman of the Fifth International Workshop on Information Processing in Cells and Tissues (IPCAT 2003), Lausanne, Switzerland, in September 2003; and General Cochairman of the First International Workshop on Biologically Inspired Approaches to Advanced Information Technology (Bio-ADIT 2004), Lausanne, Switzerland, in January 2004.

André Stauffer (Member, IEEE) received the M.S. and Ph.D. degrees from the Swiss Federal Institute of Technology in Lausanne, Switzerland, in 1969 and 1980, respectively.

He was a Visiting Scientist at the IBM T. J. Watson Research Center, Yorktown Heights, NY, in 1986. He is currently Senior Lecturer in the School of Computer and Communication Sciences at the Swiss Federal Institute of Technology. He is also a Professor at the HES-SO University of Applied Sciences, Yverdon, Switzerland. In addition to digital design, his research interests include cellular automata, circuit reconfiguration, and bioinspired systems.

Leonardo Peparolo received the M.S. degree from the Swiss Federal Institute of Technology, Lausanne, Switzerland, in 2004.

He entered the Computer Science Department, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 1998. While at EPFL, he worked with Dynamic Parallel Schedules (DPS), a high-level framework for developing parallel applications on distributed memory computers, in the Peripheral System Laboratory (EPFL-IC-LSP), supervised by Prof. R. D. Hersch and Dr. S. Gerlach. In this context, he implemented a Java version of this library, named JDPS, using the DPS kernel. He collaborated on the L-system project in the Logic System Laboratory (EPFL-IC-LSL) with the supervision of Prof. D. Mange. He is currently an Informatics Engineer at Coris, a consulting company in Geneva, Switzerland.

Gianluca Tempesti (Member, IEEE) received the B.S.E. degree in computer science from Princeton University, Princeton, NJ, in 1991, the M.S.E. degree from the University of Michigan, Ann Arbor, in 1993, and the Ph.D. degree from the Swiss Federal Institute of Technology, Lausanne, Switzerland, in 1998, with a thesis based on the development of a self-repairing bioinspired field-programmable gate array.

After pursuing his research as a Postdoctoral Fellow at the Swiss Federal Institute of Technology, he became Assistant Professor in the same institution in 2003. His research activity centers on bioinspired system architectures for high-complexity digital and molecular-scale devices.