Available online at www.sciencedirect.com

SCIENCE DIRECT®

Physica D 191 (2004) 178–192

PHYSICA D

www.elsevier.com/locate/physd

Self-replicating loop with universal construction

Daniel Mange, André Stauffer[*], Enrico Petraglio, Gianluca Tempesti

Swiss Federal Institute of Technology, Logic Systems Laboratory, CH-1015 Lausanne, Switzerland

Received 1 May 2003; received in revised form 18 November 2003; accepted 18 November 2003
Communicated by C.K.R.T. Jones

Abstract

After a survey of the theory and some realizations of self-replicating machines, this paper presents a novel self-replicating loop endowed with universal construction properties. Based on the hardware implementation of the so-called Tom Thumb algorithm, the design of this loop leads to a new kind of cellular automaton made of a processing and a control units. The self-replication of the "LSL" acronym serves as an artificial cell division example of the loop and results in a new and straightforward methodology for the self-replication of computing machines of any dimensions.
© 2003 Elsevier B.V. All rights reserved.

PACS: 84.30.Sk

Keywords: Self-replication; Universal construction; Universal computation; Cellular automaton; Artificial cell division

1. Introduction and survey

1.1. Self-replicating loops

The main goal of this paper is to present a new self-replicating loop endowed with universal construction properties. While the early history of the theory of self-replicating machines is basically the history of von Neumann's thinking on the matter [18], a practical implementation requires a sharply different approach. It was finally Langton, in 1984, who opened a second stage in this field of research. In order to construct a self-replicating automaton simpler than this of von Neumann, Langton [6] adopted more liberal criteria. He dropped the condition that the self-replicating unit must be capable of universal construction and computation.

Langton proposes a configuration in the form of a loop, endowed notably of a constructing arm and of a replication program or genome, which turns counterclockwise. After 151 time steps, the original loop (mother loop) produces a daughter loop, thus obtaining the self-replication of Langton's loop.

To avoid conflicts with biological definitions, we do not use the term "cell" to indicate the parts of a cellular automaton, opting rather for the term "molecule". In fact, in biological terms, a *cell* can be defined as the smallest part of a living being which carries the complete blueprint of the being, that is the being's *genome*.

According to the biological definitions of a cell, we end up with the following observations:

- Langton's self-replicating loop is a unicellular organism: its genome requires 28 molecules and is

[*] Corresponding author. Tel.: +41-21-693-26-52;
fax: +41-21-693-37-05.
E-mail addresses: daniel.mange@epfl.ch (D. Mange), andre.stauffer@epfl.ch (A. Stauffer).

0167-2789/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.physd.2003.11.003

a subset of the complete loop which requires 94 molecules.

- The size of Langton's loop is perfectly reasonable, since it requires 94 molecules, thus allowing complete simulation.
- There is no universal construction nor calculation: the loop does nothing but replicate itself. Langton's self-replicating loop represents therefore a special case of von Neumann's self-replication of a universal constructor. The loop is a non-universal constructor, 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 uninterpreted (*transcription*). In his loop, translation is accomplished when the instruction signals are executed as they reach the end of the construction arm, and upon collision of signals with other signals. Transcription is accomplished by duplication of signals at the arm junctions.

More recently, Byl [1] proposed a simplified version of Langton's automaton. Last but not least Reggia et al. [11] 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 which also have simpler transitions functions. Moreover, they found that relaxing the strong symmetry requirement consistently led to transition functions that required fewer rules than the corresponding strong symmetry version.

1.2. Self-replicating loops with computing capabilities

All the previous loops lack any computing and constructing capabilities, their sole functionality being that of self-replication. Lately, new attempts have been made to redesign Langton's loop in order to embed such calculation possibilities. Tempesti's loop [16] is thus a self-replicating automaton, with an attached executable program that is duplicated and executed in each of the copies. This was demonstrated for 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 [10] shows some kind of universal computational capabilities. The system consists of three parts, loop, program, and data, all of which are replicated, followed by the program's execution on the given data.

So far, all self-replicating loops are lacking universal construction, i.e. the capability of constructing a computing machine of any dimensions, even if this goal is of highest interest for developing new cellular automata, for example three-dimensional reversible cellular automata designed by Imai et al. [5] for the emerging field of nanotechnologies.

1.3. Self-replicating loop with universal construction

Our main goal is to show that a new algorithm, the *Tom Thumb algorithm*, will make it possible to design a self-replicating loop with universal construction easily implemented into silicon.

A second goal is to generalize the notion of the classical "cellular automaton" by introducing the *data and signals of cellular automaton* which perfectly suits the specifications of our basic molecule. Moreover, such an automaton will allow a straightforward and systematic methodology for synthesizing cellular automata, a methodology which completely lacks at the moment.

In Section 2, our new algorithm will be described by means of a minimal mother cell composed of four molecules which will grow and then divide for triggering the growth of two daughter cells. This example is sufficient for deriving the detailed architecture of the basic molecule. Section 3 deals with the generalization of the methodology previously described and its application to a real example, the self-replication of the "LSL" acronym. Universal construction and computation are briefly demonstrated. Section 4 will conclude by opening new avenues based on the self-replicating loop with universal construction.

2. The Tom Thumb algorithm for cell division

2.1. Cell division in living organisms

Before describing our new algorithm for the division of an artificial cell, let us remember the two key roles that cellular division plays in the existence of living organisms:

- The construction of two daughter cells in order to grow a new organism or to repair an existing one (genome *translation*).
- The distribution of two identical sets of chromosomes in order to create two copies of the genome from the mother cell aimed at programming the daughter cells (genome *transcription*).

Starting with a minimal cell made up of four artificial molecules, we will propose a new algorithm, the *Tom Thumb algorithm*, aimed at constructing both the daughter cells and the associated genomes. This algorithm will finally allow us to derive the detailed architecture of our final molecule. A tissue of such molecules will in the end be endowed of both universal construction and computation properties.

2.2. Initial conditions

The minimal cell compatible with our algorithm is made up of four molecules, organized as a square of 2 rows $\times$ 2 columns (Fig. 1). Each molecule is able to store in its four memory positions four hexadecimal characters of our artificial genome, and the whole cell thus embeds 16 such characters.

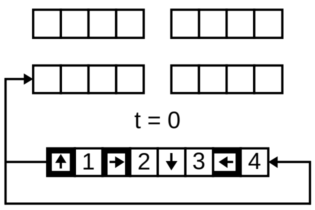
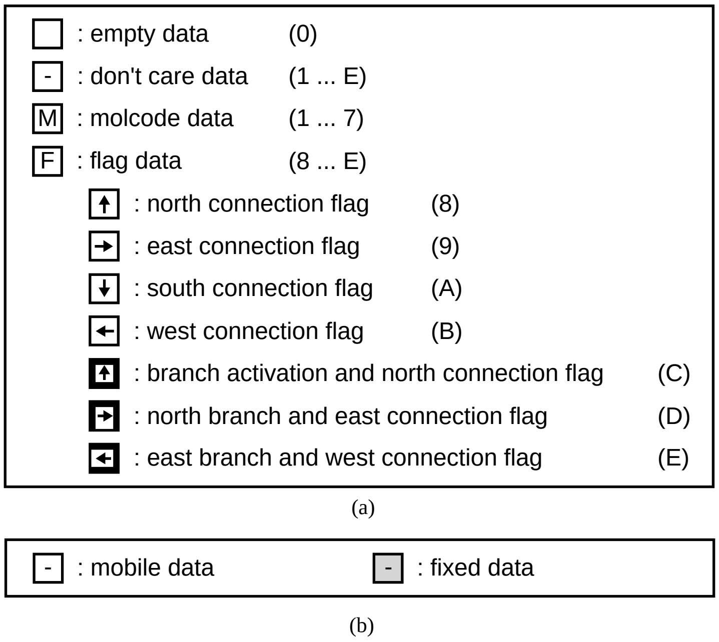


Fig. 1. The minimal cell (2×2 molecules) with its genome at the start ($t = 0$).

The original genome for the minimal cell is organized as a string of eight hexadecimal characters, i.e. half the number of characters in the cell, moving counterclockwise by one character at each time step ($t = 0, 1, 2, \ldots$).

The 15 used hexadecimal characters composing the alphabet of our artificial genome are detailed in Fig. 2. They are either *empty data* (0), *molcode data* (for molecule code data, from 1 to 7) or *flag data*


: empty data (0)
: don't care data (1 ... E)
: molcode data (1 ... 7)
: flag data (8 ... E)
: north connection flag (8)
: east connection flag (9)
: south connection flag (A)
: west connection flag (B)
: branch activation and north connection flag (C)
: north branch and east connection flag (D)
: east branch and west connection flag (E)
(a)
: mobile data
: fixed data
(b)
Fig. 2. The 15 characters forming the alphabet of an artificial genome: (a) graphical and hexadecimal representations of the 15 characters; (b) graphical representation of the status of each character.

(from 8 to E). Molcode data will be used for configuring our final artificial organism, while flag data are indispensable for constructing the skeleton of the cell. Furthermore, each character is given a status and will eventually be *mobile data*, indefinitely moving around the cell, or *fixed data*, definitely trapped in a memory position of a molecule.

2.3. Constructing the cell

At each time step, a character of the original genome is shifted from right to left and simultaneously stored in the lower leftmost molecule (Figs. 1 and 3). Note that, due to our algorithm, the first, third, etc., character of the genome (i.e. each odd character) is always a flag F, while the second, fourth, etc., character (i.e. each even character) is always a molcode M. The construction of the cell, i.e. storing the fixed data and defining the paths for mobile data, depends on two major patterns (Fig. 4).

- If the two, three or four rightmost memory positions of a molecule are empty (blank squares), the characters are shifted by one position to the right (shift data).
- If the rightmost memory position is empty, the characters are shifted by one position to the right (load data). In this situation, the rightmost F' and M' characters are trapped in the molecule (fixed data), and a new connection is established from the second leftmost position toward the northern, eastern, southern or western molecule, depending on the fixed flag information ($F' = 8$ or C, 9 or D, A, B or E).

At time $t = 16$, 16 characters, i.e. twice the contents of the original genome, have been stored in the 16 memory positions of the cell (Fig. 3). Eight characters are fixed data, forming the phenotype of the final cell, and the eight remaining ones are mobile data, composing a copy of the original genome, i.e. the genotype. Both *translation* (i.e. construction of the cell) and *transcription* (i.e. copy of the genetic information) have been therefore achieved.

The fixed data trapped in the rightmost memory position of each molecule remind us of the pebbles left by Tom Thumb for memorizing his way.

2.4. Dividing the mother cell into two daughter cells

In order to grow an artificial organism in both horizontal and vertical directions, the mother cell should be able to trigger the construction of two daughter cells, nothward and eastward.

At time $t = 11$ (Fig. 3), we observe a pattern of characters which is able to start the construction of the northward daughter cell; the upper leftmost molecule is characterized by two specific flags, i.e. a fixed flag indicating a north branch ($F = D$) and the branch activation flag ($F = C$). This pattern is also visible in Fig. 5 (northward signal, third row). The new path to the northward daughter cell will start from the second leftmost memory position.

At time $t = 23$, another particular pattern of characters will start the construction of the eastward daughter cell; the lower rightmost molecule is characterized by two specific flags, i.e. a fixed flag indicating an east branch ($F = E$), and the branch activation flag ($F = C$). This pattern appears also in Fig. 5 (eastward signal, third row). The new path to the eastward daughter cell will start from the second leftmost memory position.

The other patterns in Fig. 5 are needed for constructing the inner paths of the minimal cell (Fig. 3) as well as a cell more complex than the minimal cell, for example that of Fig. 11b.

2.5. Growing a multicellular organism

In order to analyze the growth of a multicellular artificial organism, we are led to carefully observe the interactions of the different paths created inside and outside each individual cell.

As for the initial conditions, we suppose that at time $t = -1$ a first path is constructed from the shift register storing the original genome (Fig. 1) to the lower leftmost molecule. After each period of eight time steps (i.e. $t = 7, 15, 23, \dots$), the same order will trigger again the construction of this path (Fig. 6a).

The construction of the cell is characterized by the successive launch of four inner paths northward ($t = 3$), eastward ($t = 7$), southward ($t = 11$), and westward ($t = 15$) (Fig. 6a). Due to our algorithm, this

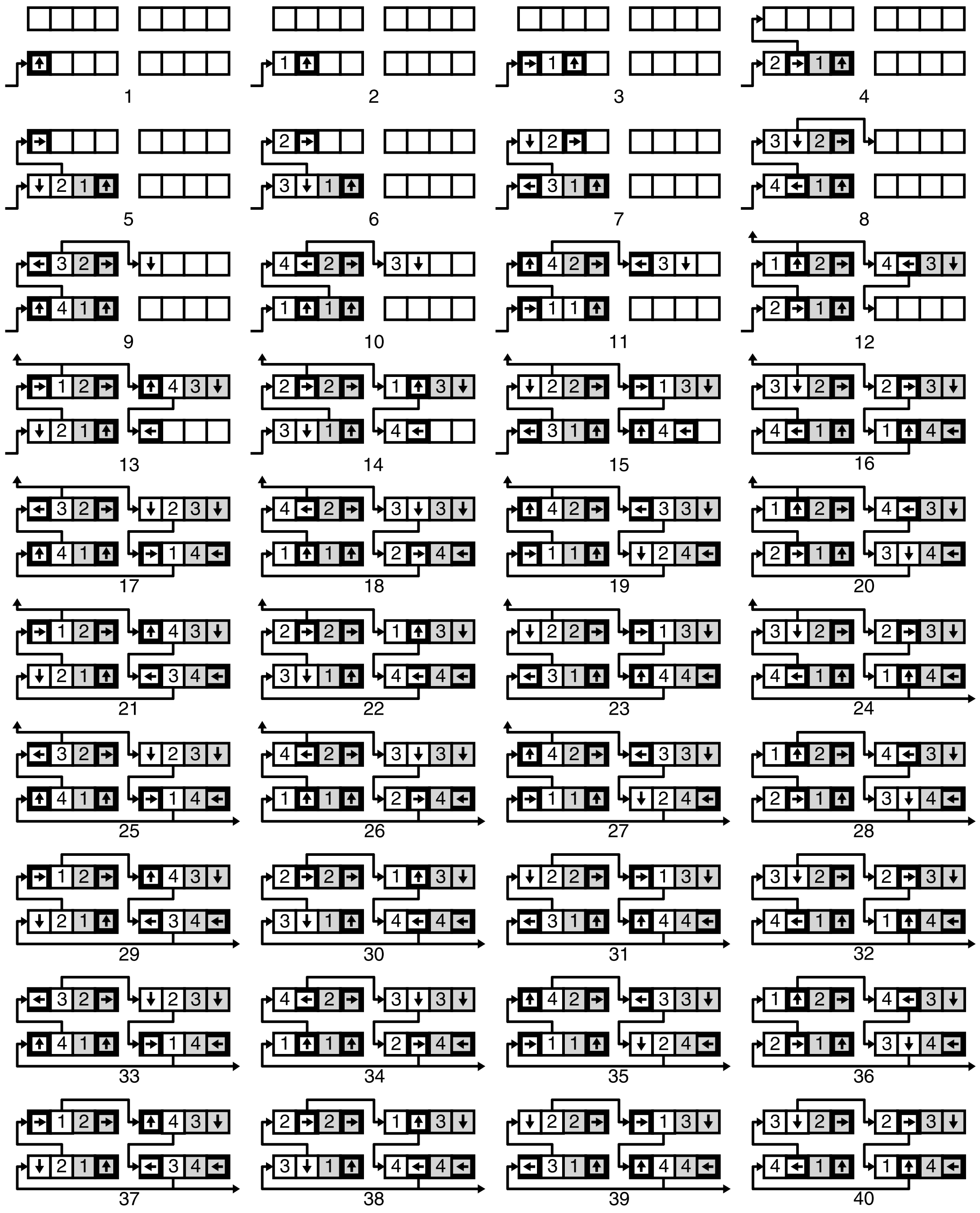


Fig. 3. Constructing the minimal cell ($t = 4$: north path, $t = 8$: east path, $t = 12$: south path and north branch, $t = 16$: west path and loop completion, $t = 24$: east branch, $t = 28$: north branch cancellation, $t = 40$: east branch cancellation).

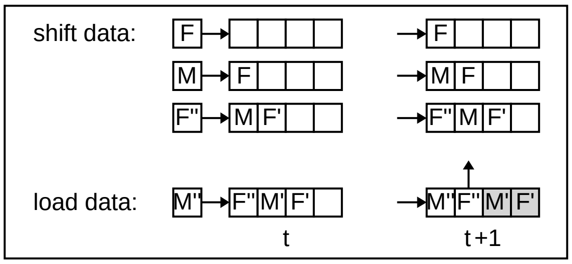


Fig. 4. The two memory patterns for constructing a cell.

construction is carried out only once, and these paths are never reactivated. Just notice the collision between the two signals at time $t = 15$ where priority is given to the westward inner path.

Finally, the division of the mother cell into two daughter cells will trigger a northward outer path at time $t = 11$. Due to our algorithm, this path is reactivated periodically every eight time steps, i.e. $t = 19, 27, 35, \ldots$. For the same reason, the cell division will trigger another eastward outer path at time $t = 23$; this path is also reactivated periodically every eight time steps, i.e. $t = 31, 39, \ldots$.

A macroscopic representation of the mother cell is given in Fig. 6b where the different activation times of the initial path ($t_i = -1, 7, 15, 23, 31, 39, \ldots$), the northward outer path ($t_n = 11, 19, 27, 35, \ldots$), the eastward outer path ($t_e = 23, 31, 39, \ldots$), and the inner path closing the loop ($t_c = 15$) are summarized. It is then possible to derive the number of time steps Δt_n between the occurrence of the first initial path ($t_i = -1$) and the first northward outer path ($t_n = 11$):

$$\Delta t_n = t_n - t_i = 12. \qquad (1)$$

The number of time steps Δt_e, until the first eastward outer path, becomes

$$\Delta t_e = t_e - t_i = 24, \qquad (2)$$

while Δt_c, the one until the inner path closing the loop, corresponds to

$$\Delta t_c = t_c - t_i = 16. \qquad (3)$$

Fig. 7a shows the macroscopic representation of a multicellular organism made up of $2 \times 2 = 4$ cells where each path activation (northward, eastward, and closing the loop) is given its precise timing according to the temporal characteristics of the minimal cell (Fig. 7b). In this cell t_n, t_e and t_c are defined as follows:

$$t_n = t_i + \Delta t_n + K.8 = t_i + 12 + K.8, \qquad (4)$$

$$t_e = t_i + \Delta t_e + K.8 = t_i + 24 + K.8, \qquad (5)$$

$$t_c = t_i + \Delta t_c = t_i + 16, \qquad (6)$$

where K is an integer $(0, 1, 2, 3, \ldots)$.

2.6. Defining the priorities between cells

When two or more paths are simultaneously activated, a clear priority should be established. We have therefore chosen three growth patterns (Fig. 7a):

- For cells in the lower row (1.1 and 2.1) a collision occurs at time $t_c = t_i + \Delta t_c = t_i + 16$ between the closing loop and the path entering the lower leftmost molecule. As already mentioned, the inner loop, i.e. the westward path, will have the priority over the eastward path.
- At the exception of the mother cell 1.1, for cells in the leftmost column (1.2), the inner loop, i.e.

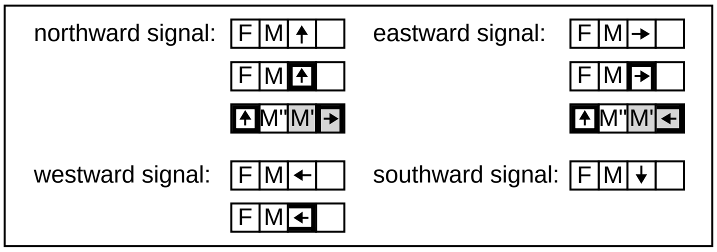
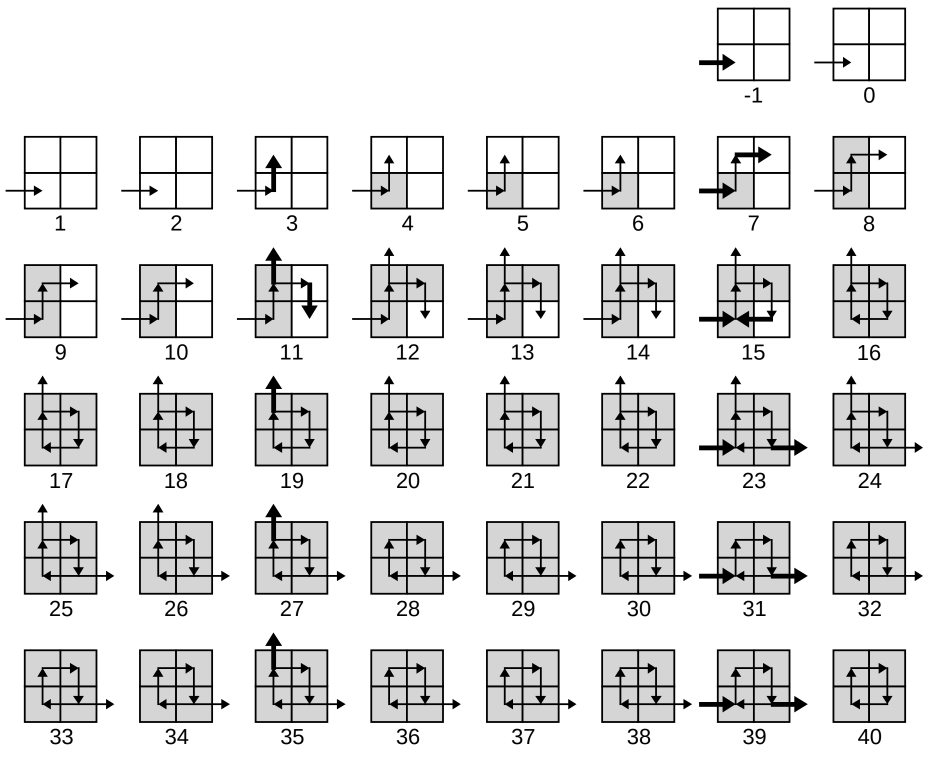
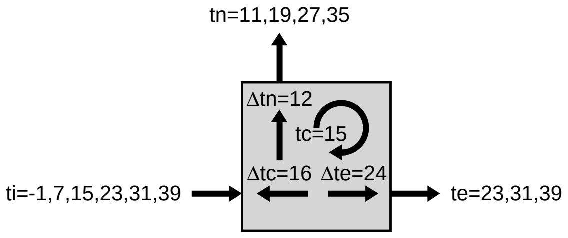


Fig. 5. Patterns of characters triggering the paths to the north, east, south and west molecules.



(a)



(b)

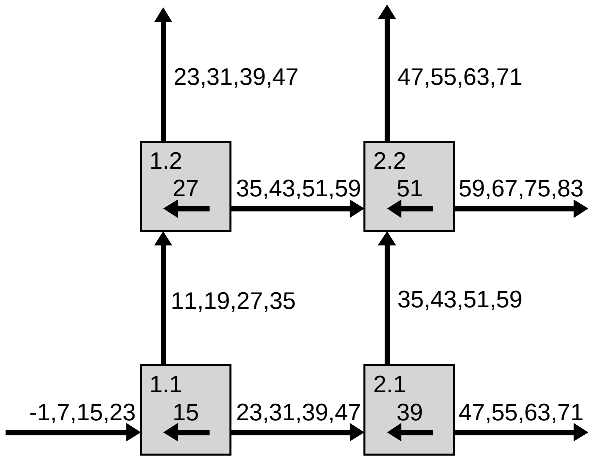
Fig. 6. Macroscopic representations of the mother cell: (a) activated path from $t = -1$ to $t = 40$; (b) number of time steps Δt_n, Δt_e and Δt_c.

the westward path, will take the priority over the northward path.

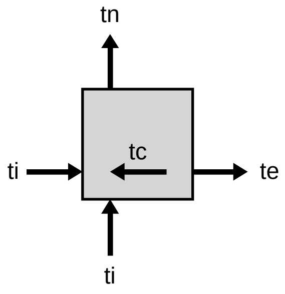
- For all other cells (2.2), two types of collisions may occur, between the northward and eastward paths (2-signal collision) or between these two paths and a third one, the closing loop at time t_c (3-signal collision). In this case, the northward path will have priority over the eastward path (2-signal collision), and the westward path will have priority over the two other ones (3-signal collision).

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

It is now possible to come back to the detailed representation of a multicellular organism made up of 2 × 2 minimal cells (Fig. 8) and exhibit the



(a)



(b)

Fig. 7. Macroscopic representations of a multicellular organism: (a) the 2×2 organism; (b) temporal characteristics of the minimal cell with the different activation times of the initial path (t_i), the northward outer path (t_n), the eastward outer path (t_e), and the inner path closing the loop (t_c).

latter at different time steps in accordance with the above-mentioned priorities.

2.7. Toward a hardware implementation: the data and signals cellular automaton (DSCA)

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

- An input unit, the multiplexer DIMUX, selecting one out of the four input data (NDI3:0, EDI3:0, SDI3:0 or WDI3:0), plus the empty data 0000; this selection is operated by a 3-bit control signal I2:0.
- A 4-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) according to the definitions of Fig. 4. The two phenotypic registers are idle (i.e. performing the HOLD operation) only when the rightmost memory position of the molecule 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 *control unit* is itself decomposed into two units:

- An input encoder ENC, a finite state machine calculating the 3-bit 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 (Figs. 6a and 7a), is described by the state graph of Fig. 9d. The five internal states QZ, QN, QE, QS, and QW will control the multiplexer DIMUX for choosing the input value 0000 or the input data NDI3:0, EDI3:0, SDI3:0 or WDI3:0, respectively.
- An output generator GEN, which is a combinational system producing the northward, eastward, southward, and westward signals (NSO, ESO, SSO, and WSO) according to the patterns described in Fig. 5.

The processing unit (PU) and control unit (CU) as well as the final molecule are represented at macroscopic levels in Fig. 9b and c; these figures define a new kind of generalized cellular automaton, the *data and signals cellular automaton* (DSCA) [15].

2.8. What's new with the data and signals cellular automaton (DSCA)?

A look at Fig. 9a allows the calculation of the number of the state variables involved in the molecule (each sequential register being represented by a small triangle), i.e. 16 for the stack (GA3:0, GB3:0, PA3:0, PB3:0) and three for the control signals of the

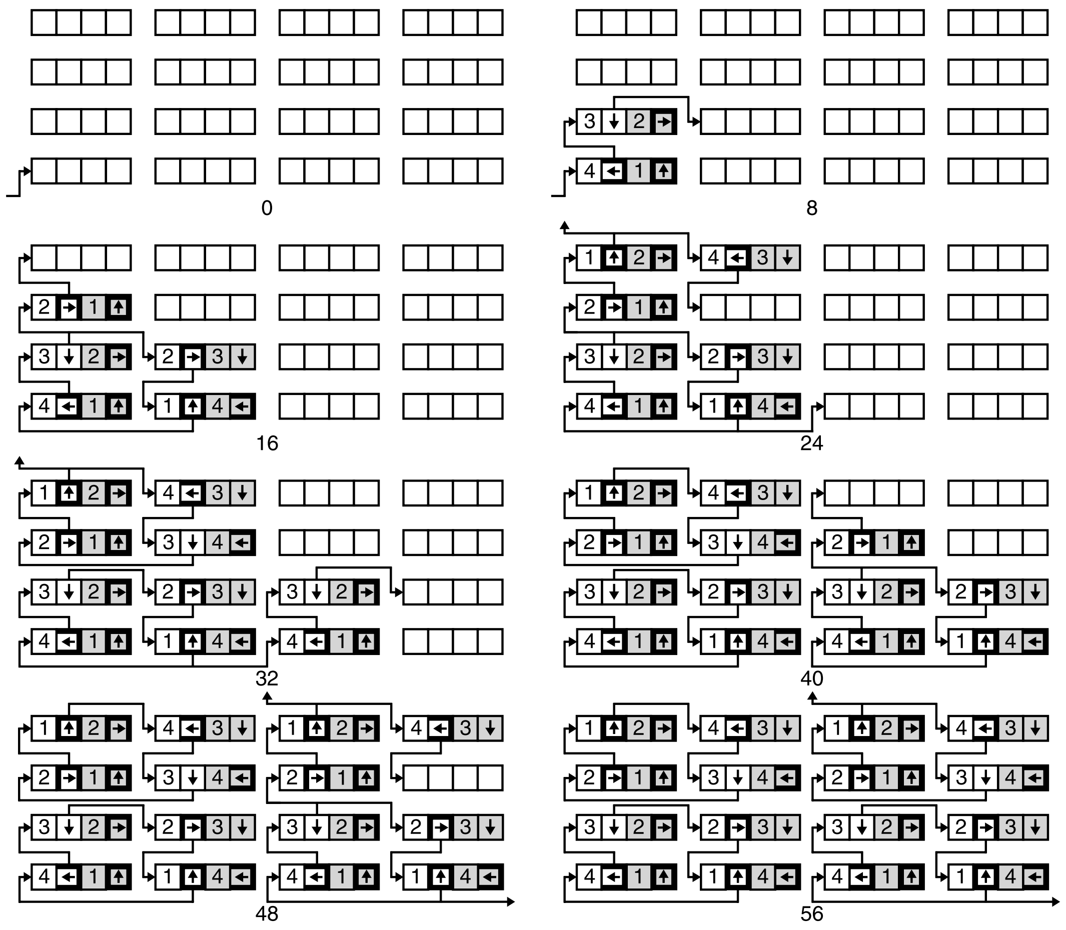


Fig. 8. Analyzing a multicellular organism made up of 2 × 2 minimal cells ($t = 32$: cell 1.2 closed on itself and independent of its mother cell 1.1, $t = 40$: cell 2.1 closed on itself and independent of its mother cell 1.1, $t = 56$: cell 2.2 closed on itself and independent of its mother cell 2.1).

input multiplexer (I2:0), which amounts to a total of 19. Therefore, the number of possible states is 2^{19}. Thanks to our methodology, i.e. decomposing the molecule into a processing unit and a control unit, we do not need to carry out the whole state table with 2^{19} rules.

- The 16 variables (GA3:0, GB3:0, PA3:0, PB3:0) are data variables, required for transferring or storing the flags and molcodes of the original genome.
- Three variables (I2:0) are control variables, required for coding the five states of the graph in Fig. 9d and for controlling the different priorities.

The information containing all the characteristics of the self-replicating loop (height, width, changes of direction, useful information) is entirely included in the genome (flags and molcodes), which is easily programmable by the user (see the next section): it constitutes the data part of the 19 variables.

The only information needed for controlling our DSCA is used for priorities calculation. Any change of specifications will then necessitate a transformation of the graph of Fig. 9d. As an example, if we wish to build our multicellular organism row by row (and not column by column as in Fig. 8), we would have to start with the modified graph of Fig. 9e.

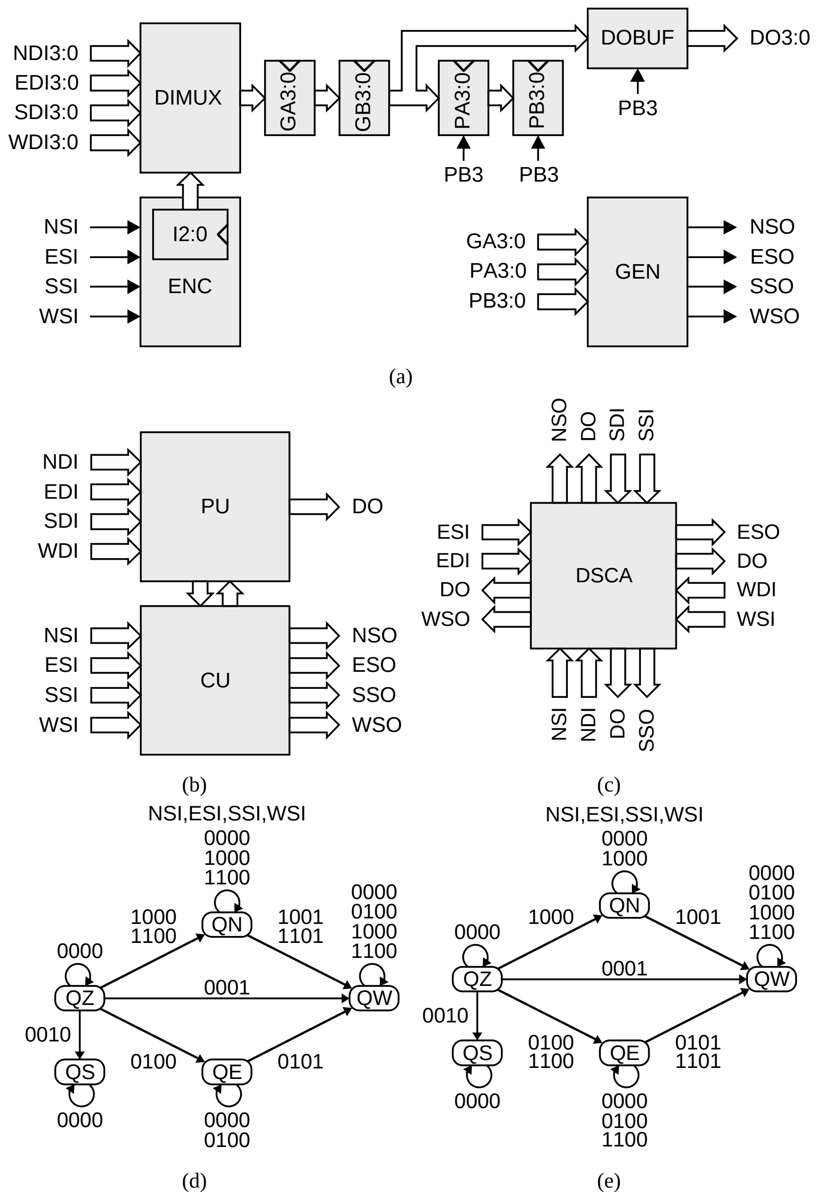


Fig. 9. A possible implementation of the basic molecule as a novel data and signals cellular automaton (DSCA): (a) detailed architecture; (b) macroscopic representation made up of a processing unit (PU) and a control unit (CU); (c) macroscopic representation of the DSCA; (d) state graph of the finite state machine ENC; (e) modified graph of the finite state machine ENC.

3. Generalization and design methodology

3.1. Non-minimal loops

The self-replicating loops in Fig. 10 are the two examples of non-minimal loops. Note that the molcode data can be directly used to display some useful information, such as in the example of Section 3.2, or can be indirectly used as a configuration string able to control a programmable device such as a field-programmable gate array (FPGA) (for such an application, see [14]).

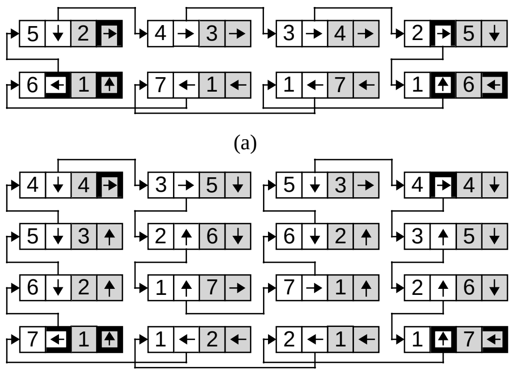


Fig. 10. Two examples of non-minimal self-replicating loops: (a) a $4 \times 2 = 8$ molecules loop ($\Delta t_n = 20$, $\Delta t_e = 28$, $\Delta t_c = 32$); (b) a $4 \times 4 = 16$ molecules loop ($\Delta t_n = 40$, $\Delta t_e = 60$, $\Delta t_c = 64$).

If C is the number of columns of the cell and R its number of rows, it is easy to derive the following relations:

- The total number of the molecules M in a cell is

$$M = CR. \tag{7}$$

- The total number T of hexadecimal characters in a cell is therefore

$$T = 4CR, \tag{8}$$

while the length L of the artificial genome is half the value of T, i.e.

$$L = 2CR. \tag{9}$$

The period of a cell, i.e. the time needed for a complete circulation of the genome, is equal to L time steps. A careful examination of the new self-replicating loop allows to derive the following relations defining the different numbers of time steps Δt_n (first northward outer path), Δt_e (first eastward outer path), and Δt_c (inner path closing the loop):

$$\Delta t_n = L + 2R = 2(C + 1)R, \tag{10}$$

$$\Delta t_e = 3L = 12R \quad \text{if} \quad C = 2, \tag{11}$$

$$\Delta t_e = 2L - 2(C - 2) = 4CR - 2(C - 2) \quad \text{if} \quad C > 2, \tag{12}$$

$$\Delta t_c = T = 2L = 4CR. \tag{13}$$

3.2. The LSL acronym design example

In [16], Tempesti has already shown how to embed the acronym "LSL" (for Logic Systems Laboratory) 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.

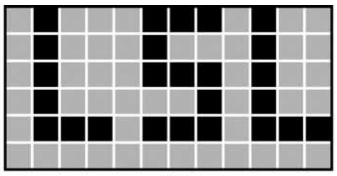
Unlike heuristic Tempesti's method, we will show that the same example can be designed in a straightforward and systematic way, thanks to the use of our new data and signals cellular automaton (DSCA) associated to the Tom Thumb algorithm.

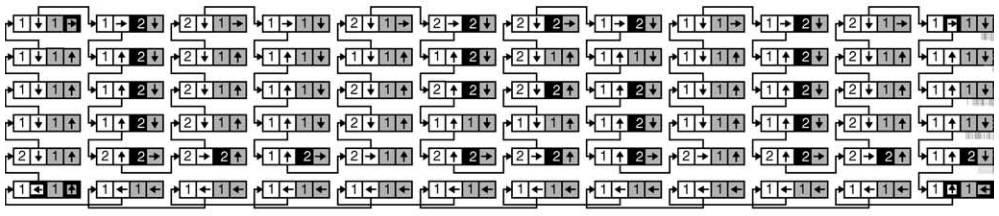
The "LSL" acronym is first represented in a rectangular array of 12 columns × 6 rows (Fig. 11a). While the number of rows is indifferent, the number of columns should be even in order to properly close the loop (Fig. 11b). The cell is therefore made up of 12 × 6 = 72 molecules connected according to the pattern in Fig. 11b: 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 molecule (gray characters in Fig. 11b) without forgetting the branch activation and north connection flag in the lower molecule of the first column, the north branch and east connection flag in the upper molecule of the first column, and the east branch and west connection flag in the lower molecule of the last column.

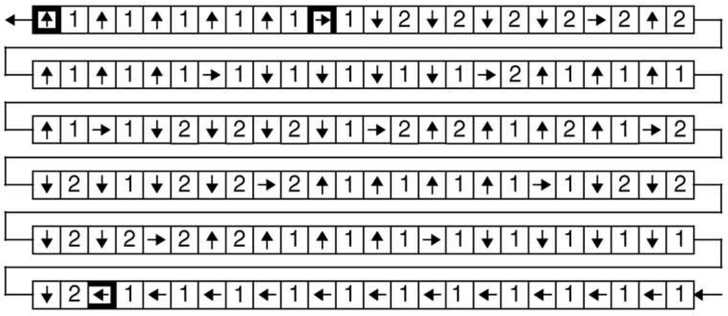
Among the 72 molecules, 25 are used to display the three letters "L", "S" and "L", and are given the character "2" as molcode (black data in Fig. 11a and b), while 47 are black (molcode "1").

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

Last, it was possible to embed the basic molecule of Fig. 9a in each of the 2000 field-programmable gate arrays of the BioWall [17] and to show the rather spectacular self-replication of our original cell (equivalent to a unicellular artificial organism), the "LSL"


(a)


(b)


(c)

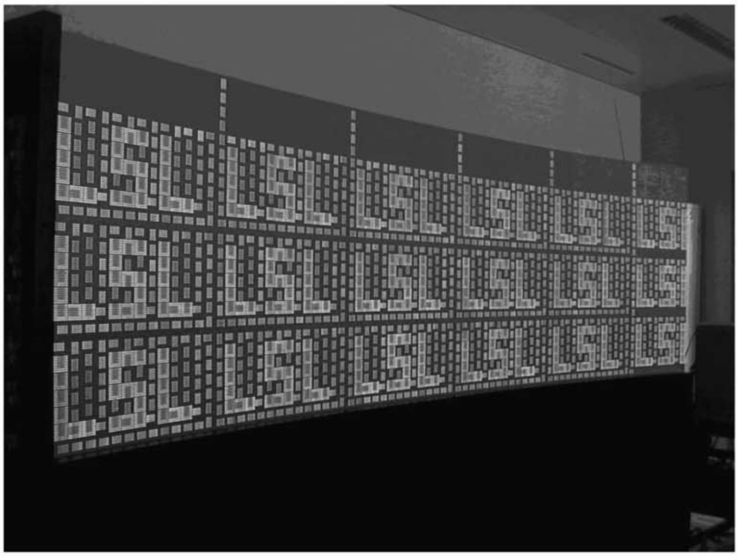

(d)

Fig. 11. Self-replication of the “LSL” acronym: (a) original specifications (LSL: Logic Systems Laboratory); (b) the $12 \times 6 = 72$ molecules of the basic cell; (c) genome; (d) BioWall implementation displaying both the genotypic path and the phenotypic shape (photograph by Petraglio).

acronym, towards both vertical and horizontal directions (Fig. 11d).

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 $\times$ 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 molecule 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 molecule of the first column, the north branch and east connection flag (D) in the rightmost memory position of the upper molecule of the first column, and the east branch and west connection flag (E) in the rightmost memory position of the lower molecule of the last column, in order to trigger the two daughter loops northwards and eastwards, respectively.
4. According to the original specifications, complete all the molcode data in the second rightmost memory position of each molecule. These molcode data constitute the phenotypic information of the artificial cell.
5. The detailed information of the final genome, i.e. the genotypic information of the artificial cell, is derived by reading clockwise along the original path of the fixed characters of the whole loop, i.e. the two rightmost characters of each molecule, starting with the lower molecule of the first column. The genotypic information, or artificial genome, is used as the configuration string of the artificial cell and will eventually take place in the two leftmost memory positions of each molecule.

3.3. Classical cellular automaton versus data and signals cellular automaton

Coming back to Tempesti's self-replicating loop [16], we can now point out the major differences between his method and our new approach.

Tempesti used a classical cellular automaton (CA). With its self-replicating mechanism, the "LSL" acronym is entirely wired inside the CA, by means of more than thousand rules, written thanks to a heuristic "cut-and-try" methodology. Even a slight modification of the original specifications could be very painful. It is impossible to demonstrate the property of universal construction (see Section 3.4).

With the Tom Thumb algorithm and its implementation as a data and signals cellular automaton (DSCA) [15], all the description of the "LSL" acronym is part of an external program, the artificial genome, simply flowing through the processing units of the DSCA. The design is straightforward, and the modifications are immediate. Changes inside the DSCA are only necessary for modifying the priorities which regulate the growth of the successive self-replicating loops.

3.4. Universal construction

In his original contribution [18], von Neumann defined construction (or constructibility) as the capability of constructing, i.e. assembling and building from appropriately defined "raw materials" an automaton using another automaton, the constructor. More precisely, the constructor, a two-dimensional automaton, is able to build in the two-dimensional array defined by von Neumann a specimen of another automaton described by a one-dimensional string of characters (the artificial genome) stored into the tape of the constructor.

According to von Neumann [18], a constructor is endowed with universal construction if it is able to construct every other automaton, i.e. an automaton of any dimensions. This concept is pointed out by Freitas and Merkle [4], where construction universality implies the ability to manufacture any of the finitely sized machines which can be formed from specific kinds of parts, given a finite number of different kinds of parts but an indefinitely large supply of parts of each kind.

If we assume (1) the existence of an array, as large as desired, of molecules such as that described in Fig. 9a and (2) the existence of a string of characters, as large as desired, the artificial genome, we claim that we are able to construct a computing machine of any dimensions into the array. Remember that the molcode data M, limited to the range 1–7, may be directly used, as in the previous example, for displaying the given specifications or may configure any kind of field-programmable gate array aimed at defining a more complex digital architecture. There are only two restrictions involved by our actual implementation:

- The number of rows and/or columns should be even, in order to properly close the loop.
- For any artificial organism characterized by a molcode alphabet greater than 1–7, we would be led to slightly modify the architecture of the actual molecule (Fig. 9a) and either use a deeper stack (with an even number of registers: 4, 6, 8, ...) or use larger registers (with more than 4-bits). For a flag alphabet greater than 8, ..., F (particularly for addressing the three-dimensional case), larger registers would also be required.

If the two conditions are met, we can embed onto an array of molecules any array of Boolean (octal, hexadecimal) values and observe the self-replication of the original pattern.

On the other hand, we have already shown that a universal Turing machine may be embedded in a regular array of identical cells [12], themselves decomposed and implemented onto a regular array of molecules. Our new loop with universal construction can therefore verify *universal computation*, thus meeting the two basic properties of the historical self-replicating cellular automaton designed by von Neumann [18], i.e. *universal construction and computation*.

4. Conclusion

Several years before the publication of the historical paper by Watson and Crick [19] revealing the existence and the detailed architecture of the DNA double helix, von Neumann was already able to point out that a self-replicating machine necessitated the existence of a one-dimensional 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. Self-replication will allow not only to divide a mother cell (artificial or living) into two daughter cells, but also to grow and repair a complete organism. Self-replication is now considered as a central mechanism indispensable for those circuits which will be implemented through the nascent field of nanotechnologies [3,13].

A first field of application of our new self-replicating loops with universal construction is quite naturally the classical self-replicating automata, such as three-dimensional reversible automata [5] or asynchronous cellular automata [9].

A second, and possibly more important field of application is Embryonics, where artificial multicellular organisms are based on the growth of a cluster of cells, themselves produced by cellular division [7,8].

A major by-product of this research is the introduction of a new kind of cellular automaton, the data and signals cellular automaton (DSCA) [15], decomposed in a processing and a control units, which allows for a systematic and straightforward design methodology which is lacking at the moment.

Other possible open avenues are about the evolution of such loops and/or their capability of carrying out massive parallel computation [2].

Acknowledgements

This work was supported in part by the Swiss National Science Foundation under grant No. 20-100049.1, by the Leenaards Foundation, Lausanne, Switzerland, and by the Villa Reuge, Ste-Croix, Switzerland. We would like to thank reviewers 1 and 2 for their invaluable contribution.

References

[1] J. Byl, Self-reproduction in small cellular automata, Physica D 34 (1989) 295–299.
[2] H.-H. Chou, J.A. Reggia, Problem solving during artificial selection of self-replicating loops, Physica D 115 (3–4) (1998) 293–312.
[3] K.E. Drexler, Nanosystems: Molecular Machinery, Manufacturing, and Computation, Wiley, New York, 1992.

[4] R.A. Freitas Jr., R.C. Merkle, Kinematic Self-replicating Machines, Landes Bioscience, Georgetown, TX, in press.

[5] K. Imai, T. Hori, K. Morita, Self-reproduction in three-dimensional reversible cellular space, Artif. Life 8 (2) (2002) 155–174.

[6] C.G. Langton, Self-reproduction in cellular automata, Physica D 10 (1984) 135–144.

[7] N.J. Macias, L.J.K. Durbeck, Self-assembling circuits with autonomous fault handling, in: A. Stoica, J. Lohn, R. Katz, D. Keymeulen, R.S. Zebulum (Eds.), Proceedings of the 2002 NASA/DOD Conference on Evolvable Hardware, Alexandria, VA, IEEE Computer Society, Los Alamitos, CA, 2002, pp. 46–55.

[8] D. Mange, M. Sipper, A. Stauffer, G. Tempesti, Toward robust integrated circuits: the Embryonics approach, Proc. IEEE 88 (4) (2000) 516–541.

[9] C.L. Nehaniv, Self-reproduction in asynchronous cellular automaton, in: A. Stoica, J. Lohn, R. Katz, D. Keymeulen, R.S. Zebulum (Eds.), Proceedings of the 2002 NASA/DOD Conference on Evolvable Hardware, Alexandria, VA, IEEE Computer Society, Los Alamitos, CA, 2002, pp. 201–209.

[10] J.-Y. Perrier, M. Sipper, J. Zahnd, Toward a viable, self-reproducing universal computer, Physica D 97 (1996) 335–352.

[11] J.A. Reggia, S.L. Armentrout, H.-H. Chou, Y. Peng, Simple systems that exhibit self-directed replication, Science 259 (1993) 1282–1287.

[12] H.F. Restrepo, D. Mange, An Embryonics implementation of a self-replicating universal Turing machine, in: Y. Liu, K. Tanaka, M. Iwata, T. Higuchi, M. Yasunaga (Eds.), Evolvable Systems: From Biology to Hardware (ICES 2001), Lecture Notes in Computer Science, vol. 2210, Springer-Verlag, Berlin, 2001, pp. 74–87.

[13] M.C. Roco, W.S. Bainbridge (Eds.), Converging Technologies for Improving Human Performance. Nanotechnology, Biotechnology, Information Technology and Cognitive Science, NSF/DOC Sponsored Report, Arlington, VA, 2002.

[14] A. Stauffer, D. Mange, G. Tempesti, C. Teuscher, Biowatch: a giant electronic bio-inspired watch, in: D. Keymeulen, A. Stoica, J. Lohn, R.S. Zebulum (Eds.), Proceedings of the Third NASA/DOD Conference on Evolvable Hardware, Long Beach, IEEE Computer Society, Los Alamitos, CA, 2001, pp. 185–192.

[15] A. Stauffer, M. Sipper, Data and signals: a new kind of cellular automaton for growing systems, in: J. Lohn, R. Zebulum, J. Steincamp, D. Keymeulen, A. Stoica, M.I. Ferguson (Eds.), Proceedings of the 2003 NASA/DOD Conference on Evolvable Hardware, Chicago, IL, IEEE Computer Society, Los Alamitos, CA, 2003, pp. 235–241.

[16] G. Tempesti, A new self-reproducing cellular automaton capable of construction and computation, in: F. Morán, A. Moreno, J.J. Merelo, P. Chacón (Eds.), Proceedings of the Third European Conference on Artificial Life (ECAL'95), Lecture Notes in Computer Science, vol. 929, Springer-Verlag, Heidelberg, 1995, pp. 555–563.

[17] G. Tempesti, D. Mange, A. Stauffer, C. Teuscher, The BioWall: an electronic tissue for prototyping bio-inspired systems, in: A. Stoica, J. Lohn, R. Katz, D. Keymeulen, R.S. Zebulum (Eds.), Proceedings of the 2002 NASA/DOD Conference on Evolvable Hardware, Alexandria, VA, IEEE Computer Society, Los Alamitos, CA, 2002, pp. 221–230.

[18] J. von Neumann, in: A.W. Burks (Ed.), Theory of Self-reproducing Automata, University of Illinois Press, Illinois, 1966.

[19] J.D. Watson, F.H.C. Crick, A structure for deoxyribose nucleic acid, Nature 171 (1953) 737–738.