ELSEVIER

Physica D 116 (1998) 71–80

PHYSICA D

On the relationship between cellular automata and L-systems: The self-replication case ★

André Stauffer, Moshe Sipper *

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

Received 21 July 1997; received in revised form 29 August 1997; accepted 23 October 1997
Communicated by A.M. Albano

Abstract

Cellular automata (CAs) have been ubiquitously used over the years to study the issue of self-replication. The L-systems model, on the other hand, is naturally suited for modeling growth processes, of which replication is a special case. The goals of this paper are: (1) to show how L-systems can be used to specify self-replicating structures, and (2) to explore the relationship between L-systems and CAs. We conclude that the bridge between CAs and L-systems seems to offer a promising approach in the study of self-replication, and, more generally, of growth processes in CAs. Copyright © 1998 Elsevier Science B.V.

PACS:
Keywords: Self-replication; Cellular automata; L-systems

1. Introduction

The study of artificial self-replicating structures or "machines" has been taking place for almost half a century. Much of this work is motivated by the desire to understand the fundamental information-processing principles and algorithms involved in self-replication, independent of their physical realization [12,21]. An understanding of these principles could prove useful in a number of ways. It may advance our knowledge of biological mechanisms of replication by clarifying the conditions that any self-replicating system must satisfy and by providing alternative explanations for empirically observed phenomena. The fabrication of artificial self-replicating machines can also have diverse applications, ranging from nanotechnology [4] to space exploration [5].

One of the central models used to study self-replication is that of cellular automata (CAs). CAs are dynamical systems in which space and time are discrete. A cellular automaton (CA) consists of an array of cells, each of which can be in one of a finite number of possible states, updated synchronously in discrete time steps, according to a local, identical interaction rule. The state of a cell at the next time step is determined by the current states of a surrounding neighborhood of cells. This transition is usually specified in the form of a rule table, delineating the cell's next state for each possible neighborhood configuration. The cellular array (grid) is

★ Supported in part by grants 20-42270.94 and 21-45630.95 from the Swiss National Science Foundation.

* Corresponding author. Tel.: +41-21-693-2658; fax: +41-21-693-3705; e-mail: moshe.Sipper@di.epfl.ch.

PII S0167-2789(97)00255-8

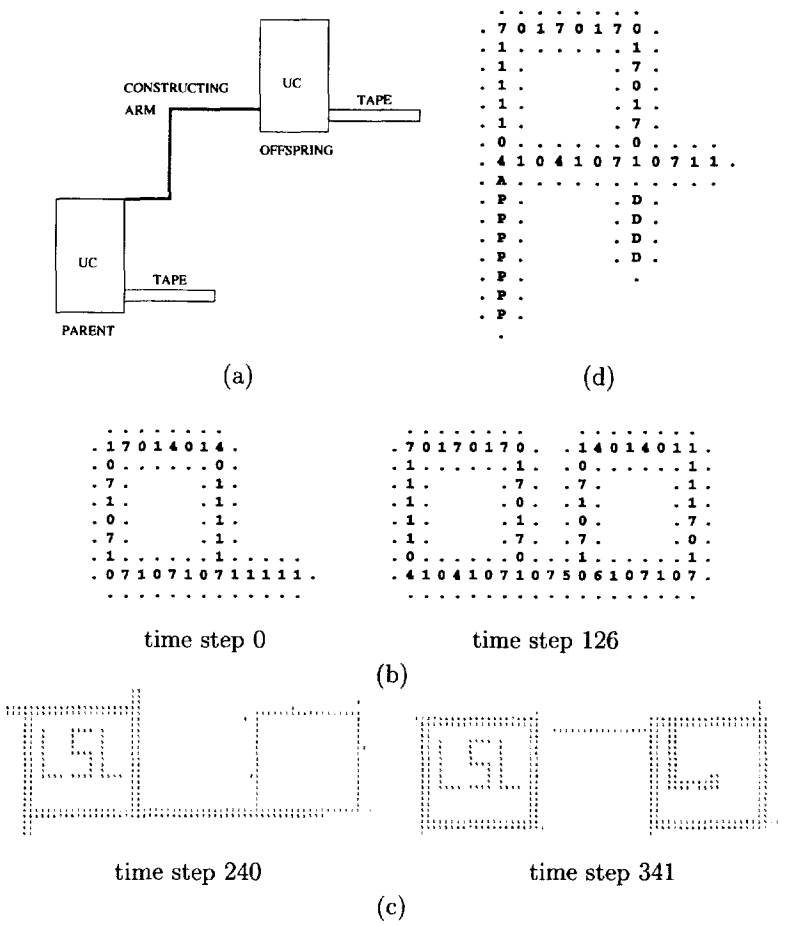

Fig. 1. Schematic illustration of some self-replicating systems, embedded in two-dimensional cellular spaces. (a) von Neumann's universal constructor (UC) is a machine (i.e., CA-embedded structure) capable of constructing, through the use of a "constructing arm", any configuration whose description can be stored on its input tape. This universal constructor is therefore capable, given its own description, of constructing a copy of itself, i.e., of self-replicating. (The machine is not drawn to scale.) (b) Langton's self-replicating loop, embedded in an 8-state cellular space (states are denoted by decimal values). The loop lacks any computing and constructing capabilities, its sole functionality being that of self-replication. (c) Tempesti's loop is 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. (d) Self-replicating loop with universal computational capabilities of Perrier et al. 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. P denotes a state belonging to the set of program states, and D denotes a state belonging to the set of data states.

n-dimensional, where $n = 1, 2, 3$ is used in practice [15,20].

Research into self-replication (within a formal framework) began in the late 1940s with von Neumann's seminal work [21]. He showed that a universal constructor can be embedded within a two-dimensional, 5-neighbor, 29-state cellular space; such a "machine" is capable of constructing any CA state configuration whose description can be stored on its input tape. The universal constructor is therefore capable, given its own description, of constructing a copy of itself, i.e., of self-replicating (Fig. 1(a)). Note that terms such as "machine" and "tape" refer to configurations of CA states – indeed the ability to formally describe such structures served as a major motivation for von Neumann's choice of the CA

model. Over the years several researchers pursued this line of research, some of the more notable milestones being (in chronological order): (1) Codd [3], who designed a universal constructor in an 8-state cellular space, simplifying von Neumann's original design; (2) Langton [6], who observed that while universal construction is a sufficient condition for self-replication it is not a necessary one, and went on to design a much simpler system that exhibits solely self-replication (Fig. 1(b)); (3) Byl [1] and Reggia et al. [12] who simplified Langton's loop; (4) Tempesti [19], who developed a self-replicating CA, similar to that of Langton's, yet with an attached executable program that is duplicated and executed in each of the copies (Fig. 1(c)); (5) Perrier et al. [10], who went beyond Tempesti's demonstration of finite computation, constructing a self-replicating loop that is capable of implementing any program, written in a simple yet universal programming language (Fig. 1(d)). For a recent review of self-replication in CAs the reader is referred to Reggia et al. [13] (a short survey is provided in [10]; see also the online self-replication page at http://lslwww.epfl.ch/~moshes/selfrep/).

A major problem with such highly local systems as CAs is the difficulty in designing them to exhibit a specific behavior or solve a particular problem. This results from the local dynamics of the system, which renders the design of local interaction rules to perform global computational tasks extremely arduous [15]. One recent approach to CA design involves the application of artificial evolution techniques. The works of Mitchell et al. [9] and Sipper [14–17] have shown that CAs can be successfully evolved to solve a number of hard problems. Lohn and Reggia [8] applied the evolutionary approach to the self-replication problem, demonstrating that a genetic algorithm can be used to discover self-replicating structures in CAs.

The advantage of evolutionary techniques lies with the fact that the (CA) designer no longer has to specify the precise solution, i.e., the CA rule table and state configuration. Rather, he or she needs to provide the evolutionary algorithm with a means of assessing the quality (or fitness) of a given solution. This facilitates the design task since in general it is easier to assess a (given) solution than to construct one. If the evolutionary setup has been well conceived, evolution may then (automatically) generate a good solution, in our case, a self-replicating CA (or, more generally, a CA that exhibits a prespecified behavior). Evolutionary techniques can be, however, a double-edged sword, the downside being the non-trivial nature of constructing the evolutionary scenario, and the non-optimized self-replication solutions thus obtained.

In this paper we explore a different avenue, studying the self-replication issue using the L-systems model. Introduced almost three decades ago as a mathematical theory of plant development, L-systems capture the essence of growth processes [7]. Observing that replication can be considered a special case of growth [18] motivated us to carry out the investigation described in this paper. Basically, an L-system is a string-rewriting grammar that is coupled with a graphical interpretation – the system can be used to churn out a plethora of finite strings that give rise (through the graphical interpretation) to two- or three-dimensional images. The basic idea elaborated herein can be stated as follows: we employ an L-system to design a self-replicating structure, with the graphical interpretation being that of a CA.

Our goals herein are: (1) to show how L-systems can be used to specify self-replicating structures, and (2) to explore the relationship between L-systems and CAs. We begin in Section 2 with an introduction of L-systems. Section 3 demonstrates how a number of elemental components used in self-replicating CAs can be described by L-system rewriting rules. Section 4 delineates the design of a self-replicating loop using an L-system, followed by its implementation as a CA in Section 5. Our paper ends in Section 6 with conclusions and directions for future research.

2. L-systems

Lindenmayer systems – or L-systems for short – were originally conceived as a mathematical theory of plant development [7,11]. 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*. The most ubiquitous rewriting systems operate on character strings. Though such systems first appeared at the beginning of this century [11], they have been attracting wide interest as of the 1950s with Chomsky's work on formal grammars, who applied the concept of rewriting to describe the syntactic features of natural languages [2]. L-systems, introduced by Lindenmayer [7], are string-rewriting systems, whose essential difference from Chomsky grammars lies in the method of applying productions. In Chomsky grammars productions are applied sequentially, whereas in L-systems they are applied in parallel and simultaneously replace all letters in a given word. This difference reflects the biological motivation of L-systems, with productions intended to capture cell divisions in multicellular organisms, where many divisions may occur at the same time.

As a simple example, 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 A [11]. The rewriting process starts from a distinguished 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. This process is shown in Fig. 2 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, e.g., interactions between plant parts). Several types of context-sensitive L-systems exist, one of which we shall concentrate on herein. In addition to context-free productions (e.g., $A \rightarrow AB$), context-sensitive ones of the form $U<A>X \rightarrow DA$ are introduced,

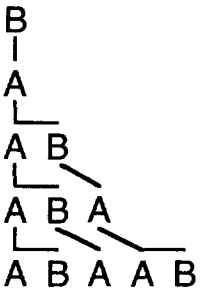

Fig. 2. Example of a derivation in a context-free L-system. The set of productions, or rewriting rules is: $\{A \rightarrow AB, B \rightarrow A\}$. The process is shown for four derivation steps.

where the letter A (called the strict predecessor) can produce word DA if and only if A is preceded by letter U and followed by 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, then only the $<$ or $>$ symbol is used, respectively (e.g., $U<A \rightarrow DA$ is a left-context rule and $A>X \rightarrow DA$ is a right-context one). Fig. 3 demonstrates a context-sensitive L-system. We note in passing that, defining a growth function as one describing the number of symbols in a word in terms of its derivation length, then this L-system exhibits square-root growth: after n derivation steps the length of the string (X symbols excluded) is $\lfloor\sqrt{n}\rfloor + 2$. Other growth functions can also be attained, including polynomial, sigmoidal, and exponential [11].

As noted above, L-systems were originally designed to model plant development. Thus, in addition to a grammar that produces finite strings over a given alphabet (as defined above), such a system is usually coupled with a graphical interpretation. Several such interpretations exist, one example of which is the so-called turtle interpretation, based on a LOGO-style turtle [11]. Here, the string produced by the L-system is considered to be a sequence of commands to a cursor (or "turtle") moving within a two- or three-dimensional space. Each symbol represents a simple command (e.g., `move forward`, `turn left`, `turn right`) such that interpretation of the string gives rise to an image.

In summary, there are two important aspects concerning L-systems, which shall serve us herein: (1)

```
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
```

(a) (b)

Fig. 3. A context-sensitive L-system. (a) The production set. (b) A sample derivation. Note that if no rule applies to a given letter then that letter remains unchanged.

such a system gives rise to a growing, one-dimensional string of characters, and (2) which can then be interpreted as a two- or three-dimensional image.

3. Using L-systems to describe components in cellular space

The basic idea elaborated herein can now be stated as follows:

(i) A self-replicating structure is designed in an L-system. Specifically, the axiom shall be replicated after a certain number of derivations.

(ii) The coupled graphical interpretation is that of operations in a cellular space, thus transposing the self-replicating structure onto a CA.

As noted, L-systems are naturally suited to model growth processes, and in particular replication.

In this section we demonstrate how a number of basic components, or operations, related to self-replication can be modeled by L-systems. Fig. 4 shows some simple growing structures along with their CA interpretations. These can implement, e.g., the extension of a constructing arm, as in the loops of Fig. 1. The ( ) and [ ] symbol pairs represent a left and right branch, respectively. These are used in so-called bracketed L-systems with the parentheses being a form of recursive application [11]: a string is interpreted from left to right to form the corresponding image. When a left bracket is encountered then the current position within the image is pushed onto a pushdown stack, with a right bracket signifying that a position is to be popped from the stack. Thus, one can model plants with branches, sub-branches, etc., or, in our case, create such constructs as extending arms and propagating signals. Fig. 5 shows a number of more complex branching structures along with their CA interpretations.

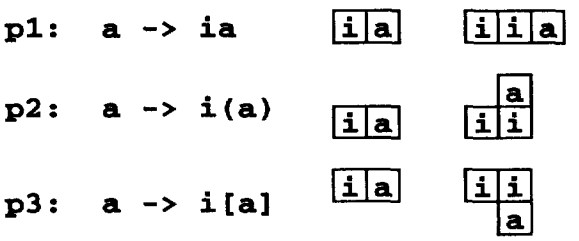

Fig. 4. Some simple growing structures along with their CA interpretations. The ( ) and [ ] symbol pairs represent a left and right branch, respectively. As the cellular space considered is a two-dimensional grid, the branching angle is 90°. The corresponding CA interpretation of a single derivation step is shown to the right. Note that the resulting operations in CA space can implement the extension of a constructing arm, similar to the loops of Fig. 1.

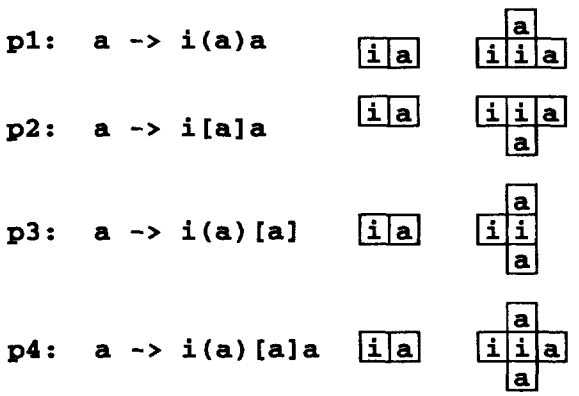

Fig. 5. Some branching structures along with their CA interpretations.

Fig. 6. Productions used to obtain signal propagation along with their CA interpretation. The CA state s is propagated to the right.

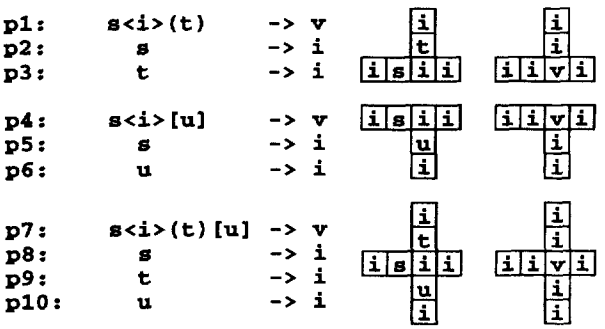

Fig. 7. Productions used to obtain signal confluence along with their CA interpretations. These implement the cases of two or three confluent signals, i.e., CA states (denoted s, t, and u) that intersect to yield a new state (denoted v).

Signal propagation can be achieved by the productions shown in Fig. 6. Note that these are context-sensitive whereas the above branching structures are context-free.

As a final example, consider signal confluence, which can be modeled by the set of productions given in Fig. 7.

4. Describing a self-replicating loop using an L-system

The L-system description of our self-replicating loop consists of a string of characters (the "genome") that is to be replicated. As emphasized by Langton [6], a distinctive property of such systems is the two different modes in which information is used: (1) as instructions that are *interpreted* to direct the development of the replica, and (2) as *uninterpreted* data that is copied onto the replica. Langton compared this with self-replication in nature, with the interpreted mode analogous to the process known as translation, and the uninterpreted mode analogous to transcription.

The self-replicating loop is a small one, inspired by Reggia et al. [12], and defined by the axiom $LGG(OO(OO(O)))OO$. It is easy to verify that the CA interpretation of this axiom corresponds to a loop (Fig. 10, derivation step 0). The system is defined in Fig. 8, and the derivation process demonstrating the axiom's replication is shown in Fig. 9. The productions of Fig. 8(b) are based on the components described in Section 3 and can be divided into five categories: (1) propagation productions (p1–p12), (2) growth productions (p13 to p16), (3) left-turn productions (p17–p19), (4) branching productions (p20–p27), and (5) loop-closing productions (p28–p31these are operative in the final stages of replication, closing the "daughter" loop and severing it from the "mother" structure). Note: when applying the above productions to derive a string $x_1x_2x_3(x_4x_5(x_6x_7(x_8)))x_9x_{10}$ then the context of a letter x_i depends upon its position. Thus, the context letters x_{i_l} and x_{i_r} of $x_{i_l}<x_i>x_{i_r}$ are defined as follows: $x_8<x_1>x_2$, $x_1<x_2>x_3$, $x_2<x_3>(x_4$, $x_3(<x_4>x_5$, $x_4<x_5>(x_6$, $x_5(<x_6>x_7$, $x_6<x_7>(x_8$, $x_7(<x_8>x_1$, $x_3<x_9>x_{10}$, and $x_9<x_{10}$. For example, when deriving the string $LGG(OO(OO(O)))OO$ (Fig. 9, step 0), production p8, $O<L>G \rightarrow O$, is applied to the left-most L, with the understanding that L's left context is the O in the innermost parentheses. This production is thus shorthand for $L>G-(--(--(O)))--\rightarrow O$, where the '–' symbol signifies "don't-care". Similarly, the above productions do not explicitly list the entire context word, with the context definition taken to be that given herein. This notation was selected for simplification purposes. Formally, L-systems where the context of a letter can be further down the string are known as IL-systems or (k, l)-systems, meaning that the left context is a word of length k and the right context is a word of length l [11]. Such an enlarged context is necessary in our case so as to obtain the loop "behavior".

5. Implementing the loop as a CA

In Section 4 we delineated the design of a self-replicating structure in an L-system. As noted, L-systems seem to offer a natural medium to study issues of growth and self-replication. In this section

```
O: building component
G: growth signal
L: left turn signal
T: turning component
B: first branch signal
C: second branch signal
e: empty component
```

(a)

```
p1:   G<O      -> G        p17:  G<L      -> T
p2:  G(<O      -> G        p18:    G>T    -> L
p3:   L<G>G    -> L        p19:    T      -> G(O)
p4:   L<G>(G   -> L        p20:  B<L      -> B
p5:   L<G>O    -> L        p21: B(<L      -> O
p6:   L<G>(O   -> L        p22:  B<e      -> O
p7:  L(<G>G    -> L        p23:  C<B      -> C
p8:   O<L>G    -> O        p24:  C<O      -> C
p9:  O(<L>G    -> O        p25:    C>B    -> O
p10:  O<L>(G   -> O        p26:    C>O    -> O
p11:  O<L>O    -> O        p27:    C>e    -> OO
p12: O(<L>O    -> O        p28:  L<G>(L)  -> B
p13:  G<G>e    -> GO       p29:    O>B    -> e
p14:  L<G>e    -> LO       p30:  B<G      -> L
p15: L(<G>e    -> LO       p31:  e<B      -> C
p16:  L<O      -> L
```

(b)

Fig. 8. An L-system implementing a self-replicating loop: (a) Symbols (letters) used; (b) Productions.

```
0:  LGG(OO(OO(O)))OO
1:  OLG(GO(OO(O)))GO          23: GGO(OO(OO(L)))OOOO(LG(O))
2:  OOL(GG(OO(O)))GG          24: LGG(OO(OO(O)))OOOO(OL(G))
3:  OOO(LG(GO(O)))LGO         25: OLG(GO(OO(O)))GOOO(OO(LO))
4:  OOO(OL(GG(O)))OLG         26: OOL(GG(OO(O)))GGOO(OO(OL))
5:  OOO(OO(LG(G)))OOLO        27: OOO(LG(GO(O)))LGGO(OO(OL))
6:  GOO(OO(OL(G)))OOOL        28: OOO(OL(GG(O)))OLGG(OO(OL))
7:  GGO(OO(OO(L)))OOOL        29: OOO(OO(LG(G)))OOLG(GO(OL))
8:  LGG(OO(OO(O)))OOOL        30: GOO(OO(OL(G)))OOOL(GG(OL))
9:  OLG(GO(OO(O)))GOOL        31: GGO(OO(OO(L)))OOOO(LG(GL))
10: OOL(GG(OO(O)))GGOL        32: LGG(OO(OO(O)))OOOO(OL(GT))
11: OOO(LG(GO(O)))LGGL        33: OLG(GO(OO(O)))GOOO(OO(LG(O)))
12: OOO(OL(GG(O)))OLGT        34: OOL(GG(OO(O)))GGOO(OO(OL(G)))
13: OOO(OO(LG(G)))OOLG(O)     35: OOO(LG(GO(O)))LGGO(OO(OO(L)))
14: GOO(OO(OL(G)))OOOL(G)     36: OOO(OL(GG(O)))OBGG(OO(OO(O)))
15: GGO(OO(OO(L)))OOOO(LO)    37: OOO(OO(LG(G)))eBLG(GO(OO(O)))
16: LGG(OO(OO(O)))OOOO(OL)    38: GOO(OO(OL(G)))eCBL(GG(OO(O)))
17: OLG(GO(OO(O)))GOOO(OL)    39: GGO(OO(OO(L)))eOCB(LG(GO(O)))
18: OOL(GG(OO(O)))GGOO(OL)    40: LGG(OO(OO(O)))eOOC(OL(GG(O)))O
19: OOO(LG(GO(O)))LGGO(OL)    41: OLG(GO(OO(O)))eOOO(OO(LG(G)))C
20: OOO(OL(GG(O)))OLGG(OL)    42: OOL(GG(OO(O)))eGOO(OO(OL(G)))OO
21: OOO(OO(LG(G)))OOLG(GL)    43: OOO(LG(GO(O)))eGGO(OO(OO(L)))OO
22: GOO(OO(OL(G)))GOOL(GT)    44: OOO(OL(GG(O)))e LGG(OO(OO(O)))OO
```

Fig. 9. Applying the productions of Fig. 8(b) to the axiom $LGG(OO(OO(O)))OO$ results in its replication after 44 derivation steps.

we implement the loop in a spatial model, namely, the CA, by transforming our structure from the L-systems language to the CA language.

Our intended cellular space is two-dimensional, 5-neighbor. We first note that the simple CA interpretation of the loop (Fig. 10) cannot in fact be implemented in such a cellular space. For example, the transition between time steps 2 and 3 cannot be carried out (since two blank cells have a state of G as their left neighbor and only one must change to state O). Thus, our graphical interpretation has so far been just that – a graphical representation of the system in question. However, it is not necessarily a viable CA, i.e., one that can be implemented as a CA rule in a two-dimensional, 5-neighbor cellular space.

In order to carry out the transformation into an actual CA one can either transform the original (non-implementable) L-system into an implementable one, or, alternatively, transform the (non-implementable) graphical interpretation into a viable CA. Herein we have opted for the former. The L-system of Fig. 8 is orientationally "neutral" in the sense that the symbols signify "turn" and "grow." The novel system, depicted in Fig. 11, contains components which allow the implementation of a CA with weak rotational symmetry [12]. Comparing the modified set of symbols

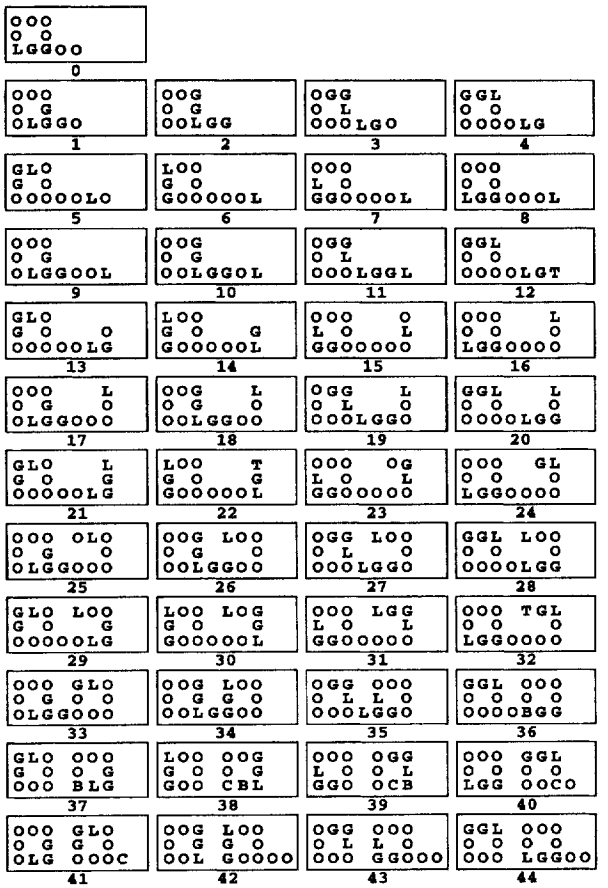

Fig. 10. The CA interpretation of the L-system of Fig. 8, with each time step corresponding to the respective derivation step of Fig. 9.

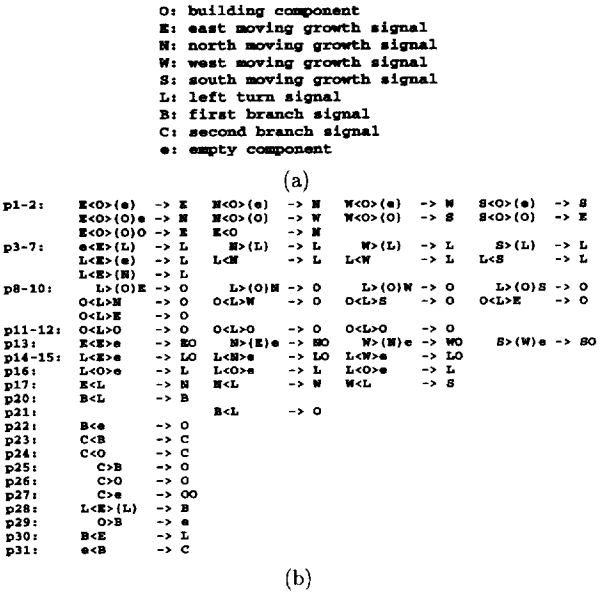

Fig. 11. Modifying the L-system of Section 4 so as to enable implementation as a viable CA: (a) Symbols (letters) used; (b) Productions. The four columns contain the expressions for the east, north, west, and south directions, respectively, thus transforming the orientationally neutral system of Fig. 8 to one which can be implemented in a CA with weak rotational symmetry. The total number of productions is 56.

(Fig. 11(a)) with the original one (Fig. 8(a)), we note that the non-oriented growth symbol G has been transformed into four oriented symbols E, N, W, and S; furthermore, the original T symbol has been eliminated. The resulting self-replicating loop is shown in Fig. 12. The production set of Fig. 11(b) can be transformed into the CA rule table depicted in Fig. 13.

6. Concluding remarks and future work

L-systems are naturally suited to model growth processes, and in particular replication, though CAs have been the model of choice for studying the latter issue. In this paper we tried to create a bridge between these two models, specifically with respect to self-replication. Basically, one defines the self-replicating structure as an L-system, with the graphical interpretation being that of a CA. This may not directly give rise to a viable CA, so that either the L-system or its graphical (CA) interpretation may have to be amended.

Apart from their natural amenability to modeling growth processes, another advantage of L-systems is their linearity, meaning that one can concentrate on the "primary structure", i.e., the one-dimensional genome, rather than on the "tertiary structure", i.e., the actual two- or three-dimensional implementation.[1] To date, this separation is not complete, as evident by our context definition that takes into account certain physical considerations (Fig. 8), thus calling for further research into the matter. As the system's physical

[1] These terms are "borrowed" from molecular biology, where the primary structure of a protein is the linear amino acid sequence, while the tertiary structure is the protein's three-dimensional organization.

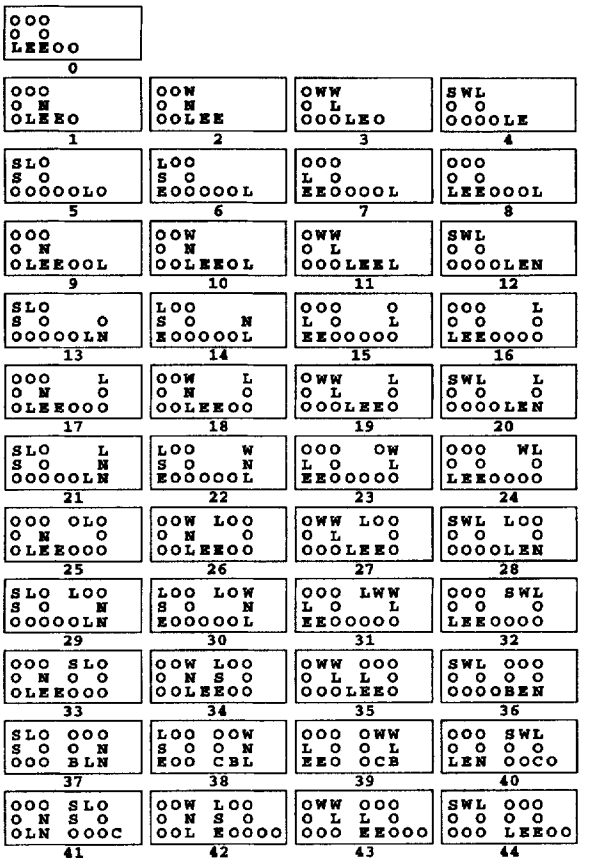

Fig. 12. The CA interpretation of the L-system of Fig. 11. This system is a viable, two-dimensional, 5-neighbor, 9-state self-replicating CA.

C	E	N	W	S	C+
e	-	-	E	-	O
e	-	-	-	N	O
e	W	-	-	-	O
e	-	S	-	-	O
e	-	-	B	-	O
e	-	-	C	-	O
O	-	e	E	-	E
O	e	O	E	-	N
O	O	O	E	-	E
O	-	-	e	N	N
O	-	-	O	N	W
O	-	-	-	E	N
O	W	-	-	e	W
O	W	-	-	O	S
O	e	S	-	-	S
O	O	S	-	-	E
O	e	-	L	e	L
O	e	e	-	L	L
O	L	e	e	-	L
O	-	-	C	-	C
O	B	-	-	-	e
E	-	L	e	-	L
E	-	e	L	-	L
E	-	N	L	-	L
E	e	-	L	-	L
E	E	L	L	-	B

C	E	N	W	S	C+
E	-	-	B	-	L
N	-	-	L	-	L
N	-	-	-	L	L
W	-	-	-	L	L
W	L	-	-	-	L
S	L	-	-	-	L
S	-	L	-	-	L
L	E	O	-	-	O
L	-	N	O	-	O
L	-	-	W	O	O
L	O	-	-	S	O
L	N	-	O	-	O
L	-	W	-	O	O
L	O	-	S	-	O
L	-	O	-	E	O
L	E	-	O	-	O
L	O	-	O	-	O
L	-	O	-	O	O
L	-	-	E	-	N
L	-	-	-	N	W
L	W	-	-	-	S
L	-	-	B	-	B
L	-	-	-	B	O
B	-	-	C	-	C
B	-	-	e	-	C
C	-	-	-	-	O

Fig. 13. Rule table of the CA of Fig. 12. The cellular space is two-dimensional, 5-neighbor, 9-state. Each entry lists the state of the cell at the next time step ($C+$) as a function of its current five neighboring states. Only non-identity transformations are shown (i.e., ones that change the central cell's state). The total number of rules is 52, with the '–' symbol signifying "don't-care".

aspect is usually of prime import, this necessitates an inherently spatial model – the CA, in our case. Thus, after designing the self-replicating L-system, it is transformed into a viable CA. These two phases were demonstrated in this paper, culminating in the self-replicating loop of Fig. 12. Some of the interesting related questions for further investigation are those concerning genotype–phenotype mappings, e.g., the stability of the phenotype (the self-replicating structure) when the genotype (L-string) is mutated, and, more generally, how do genotypic changes affect the phenotype; conversely, given a phenotype, can one derive the respective genotype that gives rise to it?

As noted, our aim has been to create a bridge between the L-systems and CA models within the framework of self-replication. Currently, this can be considered but a footbridge, with several avenues still open for future research. One major issue involves the automatization of some of the steps carried out herein. The L-system we presented was designed by hand, as were the subsequent transformations (it should be noted that virtually all self-replicating systems to date have been hand-designed, except, to some extent, that of Lohn and Reggia [8]). A step forward would involve the simplification of the design process, toward which end at least two questions need be addressed: (1) can the transformation to a viable CA, carried out in Section 5, be automated (fully or partially)? and (2) are there L-systems that lend themselves more naturally to viable, self-replicating CAs, i.e., ones that obviate the transformation step of Section 5 altogether? As for the first issue, an L-system can be trivially transformed to a CA by considering each production as an entry in the rule table. However, this usually entails a large neighborhood size and a large number of states (depending upon the L-system's context size and number of variables). The question

becomes non-trivial when one's aim is to minimize these CA parameters (neighborhood size and number of states). Taking this idea yet further one can imagine a "replication design toolkit" that would simplify the construction of such systems, automatically performing much of the underlying drudgery. Such a system could well be based on the bridge between L-systems and CAs. As noted in Section 1, replication can be considered a special case of growth. Another extension of our work, which we are currently investigating, is to examine this latter process within the framework of CAs, by adopting the L-systems point of view. Preliminary results indicate that where growth in general is concerned, the combined L-systems/CA approach is quite a promising one.

The study of artificial self-replicating systems is interesting both from a theoretical standpoint as well as from a practical one. This work has shed light on the possible use of L-systems as an exploratory tool within the realm of self-replication.

Acknowledgements

We are grateful to Jacques Zahnd, Daniel Mange, and the anonymous referees for their helpful comments.

References

[1] J. Byl, Self-reproduction in small cellular automata, Physica D 34 (1989) 295–299.
[2] N. Chomsky, Three models for the description of language, IRE Trans. Inform. Theory 2 (3) (1956) 113–124.
[3] E.F. Codd, Cellular Automata, Academic Press, New York, 1968.
[4] K.E. Drexler, Nanosystems: Molecular Machinery, Manufacturing and Computation, Wiley, New York, 1992.
[5] R.A. Freitas Jr., W.P. Gilbreath (Eds.), Adavanced automation for space missions: Proceedings of the 1980 NASA/ASEE Summer Study, Chapter 5: Replicating, Systems Concepts: Self-replicating Lunar Factory and Demonstration, NASA, Scientific and Technical Information Branch (available from US, GPO), Washington, DC, 1980.
[6] C.G. Langton, Self-reproduction in cellular automata, Physica D 10 (1984) 135–144.
[7] A. Lindenmayer, Mathematical models for cellular interaction in development, Parts I and II, J. Theoret. Biol. 18 (1968) 280–315.
[8] J.D. Lohn, J.A. Reggia, Discovery of self-replicating structures using a genetic algorithm, in: Proceedings of 1995 IEEE International Conference on Evolutionary Computation (ICEC'95), 1995, pp. 678–683.
[9] M. Mitchell, J.P. Crutchfield, P.T. Hraber, Evolving cellular automata to perform computations: Mechanisms and impediments, Physica D 75 (1994) 361–391.
[10] J.-Y. Perrier, M. Sipper, J. Zhand, Toward a viable, self-reproducing universal computer, Physica D 97 (1996) 335–352.
[11] P. Prusinkiewicz, A. Lindenmayer, The Algorithmic Beauty of Plants, Springer, New York, 1990.
[12] J.A. Reggia, S.L. Armentrout, H.-H. Chou, Y. Peng, Simple systems that exhibit self-directed replication, Science 259 (1993) 1282–1287.
[13] J.A. Reggia, H.-H. Chou, J.D. Lohn, Cellular automata models of self-replicating systems, in: Advances in Computers, Academic Press, New York, 1998, to appear.
[14] M. Sipper, Co-evolving non-uniform cellular automata to perform computations, Physica D 92 (1996) 193–208.
[15] M. Sipper, Evolution of Parallel Cellular Machines: The Cellular Programming Approach, Springer, Heidelberg, 1997.
[16] M. Sipper, The evolution of parallel cellular machines: Toward evolware, BioSystems 42 (1997) 29–43.
[17] M. Sipper, Evolving uniform and non-uniform cellular automata networks, in: D. Stauffer (Ed.), Annual Reviews of Computational Physics, vol. V, World Scientific, Singapore, 1997, pp. 243–285.
[18] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Pérez-Uribe, A. Stauffer, A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems, IEEE Trans. Evolutionary Comput. 1 (1) (1997) 83–97.
[19] G. Tempesti, A new self-reproducing cellular automaton capable of construction and computation, in: F. Morán, A. Moreno, J.J. Merelo, P. Chacón (Eds.), ECAL'95: Third European Conference on Artificial Life, Lecture Notes in Computer Science, vol. 929, Springer, Heidelberg, 1995, pp. 555–563.
[20] T. Toffoli, N. Margolous, Cellular Automata Machines, MIT Press, Cambridge, MA, 1987.
[21] J. von Neumann, Theory of Self-Reproducing Automata, University of Illinois Press, Illinois, 1966; Edited and completed by A.W. Burks.