Chapter 10

L-hardware: Modeling and Implementing Cellular Development using L-systems

André Stauffer and Moshe Sipper

10.1 Introduction

In this chapter, we study cellular development using the L-system model. Introduced almost three decades ago as a mathematical theory of plant development, L-systems capture the essence of growth processes [141]. Observing that cellular development can be considered a special case of growth [241] motivated us to carry out the investigation described in this chapter. 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 one-, two-, or three-dimensional images. The basic idea elaborated herein can be stated as follows: we employ an L-system developmental model to design a growing structure, with the graphical interpretation being that of a cellular automaton (CA).

CAs are dynamical systems in which space and time are discrete. A cellular automaton 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 n-dimensional, where $n = 1, 2, 3$ is used in practice [233, 264] (see Chapter 3 for a detailed account of CAs).

Our goals herein are: (1) to show how L-systems can be used to specify growing structures, and (2) to explore the relationship between L-systems and CAs.

We begin in Section 10.2 with an introduction to L-systems. Section 10.3 demonstrates how a number of elemental development mechanisms used in CAs can be described by L-system rewriting rules. Section 10.4 delineates the modeling of a square-root growth function using an L-system, followed by its implementation as a one-dimensional CA. Our chapter ends in Section 10.5 with the modeling of a space divider and its implementation as a two-dimensional CA.

10.2 L-systems

Lindenmayer systems – or L-systems for short – were originally conceived as a mathematical theory of plant development [141, 203]. 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 [203], 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 [39]. L-systems, introduced by Lindenmayer [141], 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 [203]. 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 Figure 10.1 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, where the

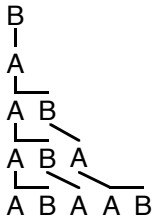


Figure 10.1 Example of a derivation in a context-free L-system. The set of productions, or rewriting rules is: $\{A \rightarrow AB, B \rightarrow A\}$. The process is shown for four derivation steps.

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). Figure 10.2 demonstrates a context-sensitive L-system. We note 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 [203].

```
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
```

(a)

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

(b)

Figure 10.2 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.

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

10.3 Cellular development

In this section we demonstrate how a number of basic components, or operations, related to cellular development can be modeled by L-systems. This developmental model involves four main processes or mechanisms: (1) simple growth, (2) branching growth, (3) signal propagation, and (4) signal divergence.

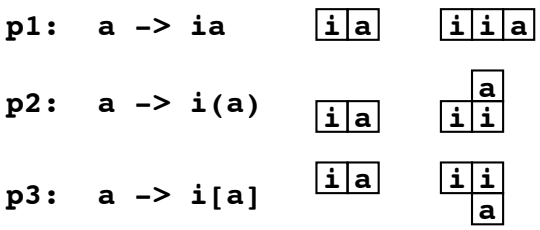


Figure 10.3 Some simple growing structures along with their CA interpretation. The ( ) and [ ] symbol pairs represent a left and right branch, respectively. As the cellular space considered is a two-dimensional grid, the branching angle is 90°.

Simple growth arises from the application of productions p1 to p3 (Figure 10.3). In these productions the symbol a represents the apex and i the internode. The terminology introduced here is borrowed from the description of tree-like shapes [203]. A tree has edges that are labeled and directed. In the biological context, these edges are referred as branch segments. A segment followed by at least one more segment is called an *internode*. A terminal segment (with no succeeding edges) is called an *apex*. In the productions, 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 [203]: 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 multi-dimensional growing structures. The corresponding CA interpretation of a single-step derivation of the simple growth productions are also given in Figure 10.3.

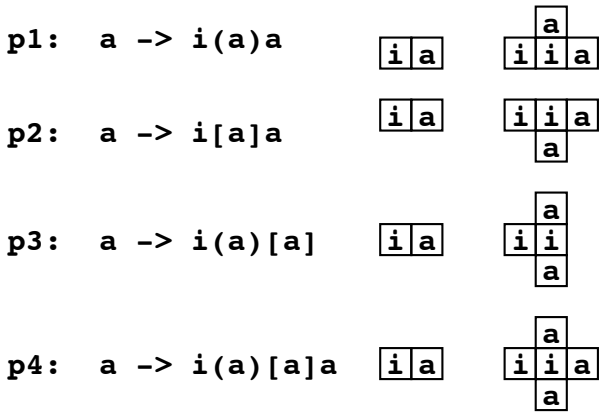


Figure 10.4 Some branching structures along with their CA interpretation.

Figure 10.5 Productions used to obtain signal propagation, along with their CA interpretation. The CA state s is propagated to the right.

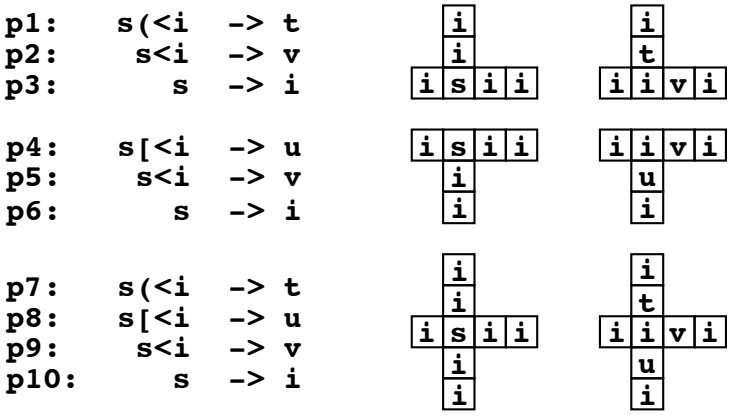


Figure 10.6 Productions used to obtain signal divergence, along with their CA interpretation. These implement the cases of a given signal, i.e., CA state (denoted s) that breaks up to yield two or three new states (denoted t, u, and v).

Productions p1 to p4 of Figure 10.4 give rise to *branching growth*. In these productions, the bracketed notations ( ) and [ ] express again left and right branches, respectively. Figure 10.4 shows also their CA interpretation for a single step derivation. Note that both in Figure 10.3 and Figure 10.4, all growth productions are context-free.

Signal propagation of a given signal s is modeled by productions p1 and p2 of Figure 10.5. The context-sensitive production p1 means that internode i with a signal s to its left becomes an s while, in conformity with the context-free production p2, the signal s itself becomes an internode i. Consequently, the CA interpretation of a single step derivation of these productions causes signal s to move one cell to the right (Figure 10.5).

Productions p1 to p10 of Figure 10.6 model *signal divergence* introduced by the branching structures. A given signal s can thus become up to three individual ones t, u, and v. Some of these productions are context-sensitive. Their corresponding CA interpretation for a single step derivation appears also in Figure 10.6.

10.4 Square-root growth function

As a first example of L-hardware, we shall implement in a CA the productions of the square-root growth function described above (Figure 10.7a). Our intended CA is one-dimensional with a connectivity radius $r = 1$. In such a cellular space, a given cell C can only access the states of its immediate left (L) and right (R) neighboring cells (Figure 10.8).

```
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
```

(a)

```
p2':  U<A>X -> D
p2": UA<X   -> AX
```

(b)

Figure 10.7 Square-root L-system. (a) The original production set. (b) The decomposition of production p2.

In order to be implemented, the productions of the L-system must therefore have at most a one-letter left context and a one-letter right context. This is not the case for production p2 since the X letter must "see" two left context letters in order to be transformed into the letter A. This is better seen by decomposing p2 into two productions (Figure 10.7b). In this decomposition, production p2″ has a two-letter left context that cannot be implemented in our CA.

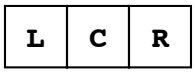


Figure 10.8 One-dimensional, $r = 1$ CA neighborhood.

```
U: propagate up (right) signal
D: propagate down (left) signal
A: grown symbol
X: string end symbol
```

(a)

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

(b)

```
0: XUX      5: XADAX
1: XADX     6: XUAAX
2: XUAX     7: XAUAX
3: XAUX     8: XAAUX
4: XAADX    9: XAAADX
```

(c)

Figure 10.9 Square-root L-system. (a) Symbol representation. (b) The adapted production set. (c) A sample derivation.

Consequently, the original L-system has to be adapted and its new productions become those of Figure 10.9b with their symbol representation given in Figure 10.9a. These new productions fall into four categories: (1) the signal U propagation productions p1 and p5, (2) the signal D propagation productions p3 and p6, (3) the simple growth (with signal change) production p2, and (4) the signal change production p4.

Using the new productions and starting this time from the axiom XUX, the first nine derivation steps lead us to the developmental process shown in Figure 10.9c. With the growth function defined once again as the number of

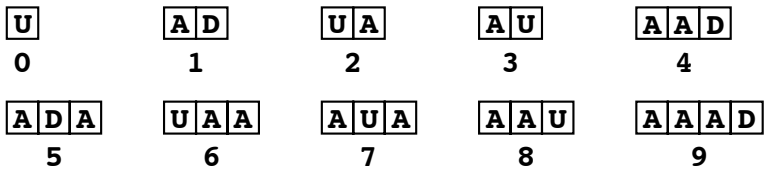
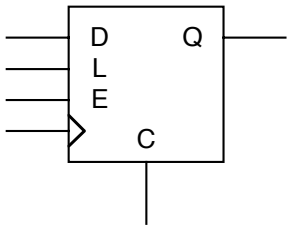


Figure 10.10 CA interpretation of the square-root L-system.

letters in the word in terms of its derivation length, it is still square-root growth but after n derivation steps the length of the string (X symbols excluded) is now $\lfloor\sqrt{n}\rfloor + 1$. The CA interpretation of this development process appears in Figure 10.10.

L	C	R	C+
U	X	-	D
U	A	-	U
A	A	D	D
X	A	D	U
-	U	-	A
-	D	-	A

Figure 10.11 Transition rules of the square-root CA cell.



(a)

operation	description	C	L	E
CLEAR	Q := 0	1	-	-
LOAD	Q <- D	0	1	-
HOLD	Q <- Q	0	0	0
TOGGLE	Q <- $\overline{Q}$	0	0	1

(b)

Figure 10.12 T-type flip-flop. (a) Logic diagram. (b) Operation table.

The one-dimensional CA implementation of the square-root growth function requires the expression of six productions as CA transition rules. These transition rules define the future state $C+$ of the basic cell as a function of its present state C and of the present states L and R of its left and right neighbors, respectively. The translation of the productions (Figure 10.9) into their corresponding transition rules is straightforward (Figure 10.11). This is done by directly transforming each production into a CA rule.

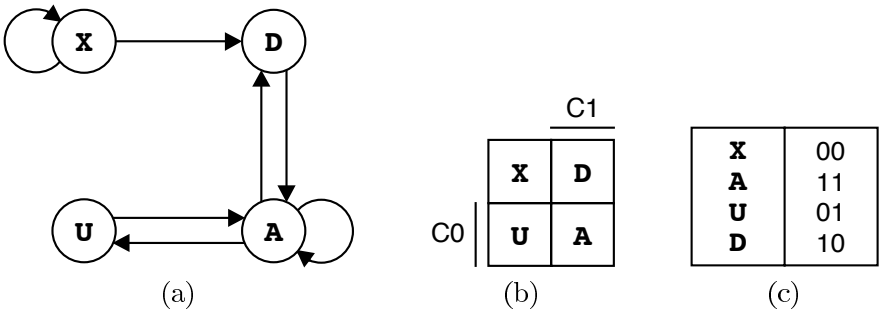


Figure 10.13 Square-root CA cell. (a) State graph. (b) Karnaugh map. (c) State codes.

L	C	R	C+	E
01	00	-	10	10
01	11	-	01	10
11	11	10	10	01
00	11	10	01	10
-	01	-	11	10
-	10	-	11	01

(a)

$$E1 = \overline{L1}\ L0 \cdot \overline{C1}\ \overline{C0} + \overline{L1}\ \overline{L0} \cdot C1\ C0 + \overline{L1}\ L0 \cdot C1\ C0 \cdot R1\ \overline{R0} + \overline{C1}\ C0$$

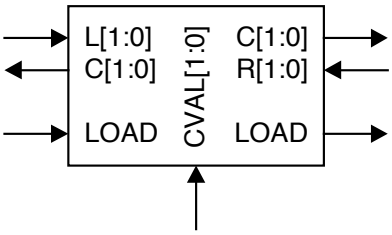
$$E0 = L1\ \overline{L0} \cdot C1\ C0 \cdot R1\ \overline{R0} + C1\ \overline{C0}$$

(b)

Figure 10.14 Square-root CA cell. (a) Truth table. (b) Equations.

The hardware implementation of the square-root growth function involves a minimal number of two state bits per cell. These two bits can be memorized in T-type flip-flops (Figure 10.12a). Depending on the values of their load L and enable E variables, these devices will allow the loading of the axiom and the replacement of the letters in the developmental process (Figure 10.12b). The minimization of the combinational circuit generating these control variables depends on the state graph representation of the transition rules (Figure 10.13a). This graph helps to find a two-bit minimal code that limits the number of state variable commutations for every transition. It leads to a Karnaugh map where each transition happens between two adjacent cells (Figure 10.13b). The resulting codes are given in Figure 10.13c.

When applied to the transition rules of Figure 10.11, the two-bit minimal codes of Figure 10.13 define the truth table of the square-root basic cell (Figure 10.14a). Given the states of the cell C and of its immediate left (L) and right (R) neighbors at derivation step n, this table specifies the cell's state $C+$ at step $n+1$. The comparison of C and $C+$ allows us to find out the commutations of



(a)

operation	description	LOAD
axiom loading	C <- CVAL	1
derivation performing	C <- C+	0

(b)

Figure 10.15 Square-root CA cell. (a) Block diagram. (b) Operation table.

the state variables. As shown in the table, for each of these commutations the corresponding control variable E takes the logic value 1 in order to involve the toggle operation of the flip-flop. A direct translation of the truth table produces the equations of the control variables E of the two flip-flops (Figure 10.14b).

The block diagram of the square-root basic cell appears in Figure 10.15a. Besides the two-bit states of the cell $C[1:0]$, of its left neighbor $L[1:0]$ and of its right neighbor $R[1:0]$, we have the two-bit value of the axiom $CVAL[1:0]$ and the control variable $LOAD$. According to its operation table (Figure 10.15b), at each clock pulse, the cell loads the axiom when $LOAD = 1$ and performs a derivation step when $LOAD = 0$.

In order to define a description able to be synthesized in any given technology, we create a register transfer level VHDL file of the square-root basic cell (Figure 10.16). The entity design unit of this file translates the block diagram of the cell (Figure 10.15a). The two processes of the architecture design unit reflect respectively the synchronous sequential register of the cell and its combinational next-state computation. This computation is based on the equations of Figure 10.14b.

Using the synthesis tool from the VIEWlogic system along with the xc4000 family from XILINX as the implementation technology, we end up with a logic diagram comprising 25 cells (Figure 10.17). After placement and routing, this design occupies 5 configurable logic blocks (CLBs) of a XILINX 4010 chip.

10.5 Space divider

The space divider is a two-dimensional CA whose purpose is to structure a "sea" of functional cells. The structure of this cellular sea ultimately consists of pro-

```
entity srcell is
  port (clk, rst :in vlbit;
        load     :in vlbit;
        cval     :in vlbit_1d(1 downto 0);
        l, r     :in vlbit_1d(1 downto 0);
        c        :inout vlbit_1d(1 downto 0));
end srcell;

architecture rtl of srcell is
  signal cd : vlbit_1d(1 downto 0);
begin
  process
  begin
    wait until rst='1' or prising(clk);
    if    rst='1' then
      c <= "00";
    else
      c <= cd;
    end if;
  end process;
  process (load, cval, l, c, r)
  begin
    if load='1' then
      cd <= cval;
    else
      if   (c(1)='0' and c(0)='0'
        and l(1)='0' and l(0)='1')
      or   (c(1)='1' and c(0)='1'
        and l(1)='0' and l(0)='1')
      or   (c(1)='1' and c(0)='1'
        and l(1)='0' and l(0)='0'
        and r(1)='1' and r(0)='0')
      or   (c(1)='0' and c(0)='1')
      then
        cd(1) <= not c(1);
      else
        cd(1) <= c(1);
      end if;
      if   (c(1)='1' and c(0)='1'
        and l(1)='1' and l(0)='1'
        and r(1)='1' and r(0)='0')
      or   (c(1)='1' and c(0)='0')
      then
        cd(0) <= not c(0);
      else
        cd(0) <= c(0);
      end if;
    end if;
  end process;
end rtl;
```

Figure 10.16 VHDL synthesizable file of the square-root cell.

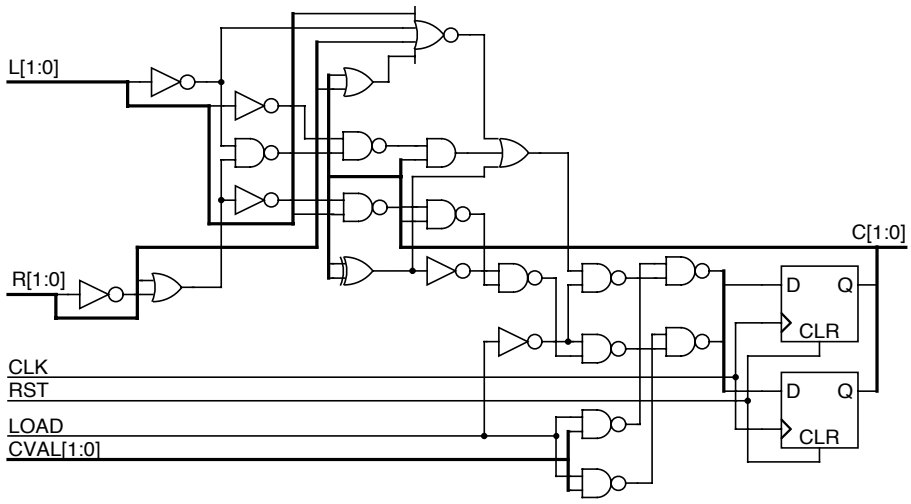


Figure 10.17 Logic diagram of the square-root cell.

grammable squares of functional cells. The structuring process is essentially a growth process, starting from the lower left-hand side automaton cell, where the programming data are continuously fed.

```
h: horizontal apex
v: vertical apex
e: east growing apex
n: north growing apex
l: left branching apex
r: right branching apex
i: internode
b: branching signal
```

Figure 10.18 Symbol representation of the space divider L-system model.

As noted, L-systems are naturally suited for modeling growth processes. Therefore, we first describe the space divider as an L-system developmental model. We transform it then to a physical realization using the corresponding CA spatial model. The representation of the symbols used in the developmental model of the space divider is given in Figure 10.18.

The L-system description of the space divider starts with a single-letter axiom l which corresponds to a left branching apex. The productions applied to the

```
p1:   b<i   -> b      p10:  i<v   -> n
p2:  b(<i   -> b      p11: i(<v   -> n
p3:  b[<i   -> b      p12:  i<n   -> iv
p4:    b    -> i      p13: i(<n   -> iv
p5:   i<h   -> e      p14:  b<n   -> bv
p6:  i[<h   -> e      p15:  b<h   -> l
p7:   i<e   -> ih     p16:    l   -> i(v)h
p8:  i[<e   -> ih     p17:  b<v   -> r
p9:   b<e   -> bh     p18:    r   -> i[h]v
```

(a)

```
0:  i|l
1:  i|i(v)h
2:  b|i(n)e
3:  i|b(iv)ih
4:  i|i(bn)be
5:  b|i(ibv)ibh
6:  i|b(iir)iil
7:  i|i(bii[h]v)bii(v)h
8:  b|i(ibi[e]n)ibi(n)e
9:  i|b(iib[ih]iv)iib(iv)ih
10: i|i(bii[be]bn)bii(bn)be
11: b|i(ibi[ibh]ibv)ibi(ibv)ibh
12: i|b(iib[iil]iir)iib(iir)iil
13: i|i(bii[bii(v)h]bii[h]v)bii(bii[h]v)bii(v)h
```

(b)

Figure 10.19 Space divider L-system model. (a) The production set. (b) A sample derivation.

axiom in order to obtain the square structure are listed in Figure 10.19a. They fall into three categories: (1) the branching signal propagation productions p1 to p4, (2) the simple growth productions p5 to p14, and (3) the branching growth productions p15 to p18.

Their application in order to derive a string like $a|b(cd[e]f)g$ is context-sensitive and this context depends upon the position of the letter that has to be replaced. We have thus $a<b$, $b(<c$, $c<d$, $d[<e$, $d<f$, and $b<g$ (thus, e.g., the context of f is not the e to its immediate left but rather d). The first character of the string to the left of the vertical separator is part of the program that is fed to the space divider. It is applied from the outside and corresponds to the left context of the first character after the vertical separator. Thirteen derivation steps of the developmental process are shown in Figure 10.19b. The CA interpretation

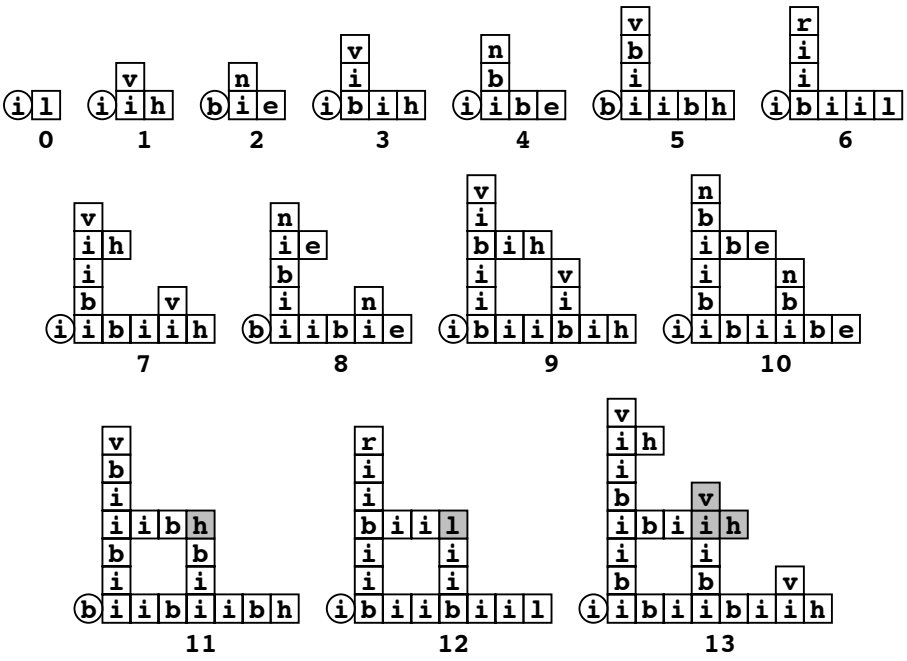


Figure 10.20 CA interpretation of the space divider L-system model.

of this process defines a division of the cellular space into squares of size 2×2 cells (Figure 10.20).

In the interpretation of Figure 10.20, the successive characters of the programming data appear in the left circle of the first cell. The number of cells on each side of the squares is thus programmable and always equal to the number of internodes i between two successive branching signals b. Starting from derivation step 11, the shaded cells in the CA interpretation (Figure 10.20) denote developments that are duplicated in Figure 10.19b and represented as underlined characters or strings.

The L-system description of the space divider is partly *relative* in terms of orientation: it only deals with left and right branches. The CA implementation of the divider is *absolute* in terms of orientation: in a von Neumann neighborhood (Figure 10.21), the future state $C+$ of a cell depends on the present state C of the cell itself and on those of its east, north, west, and south neighbors, respectively (E, N, W, and S). In order to adapt our L-system description to such a neighborhood, we can replace the former horizontal and vertical apices by a single one. The left and right branching apices have also to be merged. The new L-system symbols of Figure 10.22a become thus the states of the cell.

The 18 productions of the space divider can now be expressed eastward and

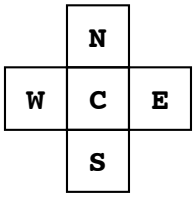


Figure 10.21 Von Neumann CA neighborhood.

```
a: apex (horizontal and vertical)
e: east growing apex
n: north growing apex
c: branching apex (east and north)
i: internode
b: branching signal
```

(a)

```
p1:   b<i   -> b      b<i   -> b
p2:                  b(<i   -> b
p3:  b[<i   -> b
p4:    b    -> i        b   -> i
p5:   i<a   -> e
p6:  i[<a   -> e
p7:   i<e   -> ia
p8:  i[<e   -> ia
p9:   b<e   -> ba
p10:                  i<a   -> n
p11:                 i(<a   -> n
p12:                  i<n   -> ia
p13:                 i(<n   -> ia
p14:                  b<n   -> ba
p15:  b<a   -> c
p16:    c   -> i(a)a
p17:                  b<a   -> c
p18:                    c   -> i[a]a
```

(b)

Figure 10.22 Space divider CA implementation. (a) The cell states. (b) The eastward and northward oriented production set.

northward. Figure 10.22b shows the 20 resulting productions in two columns containing respectively the expressions for the east and north directions. The

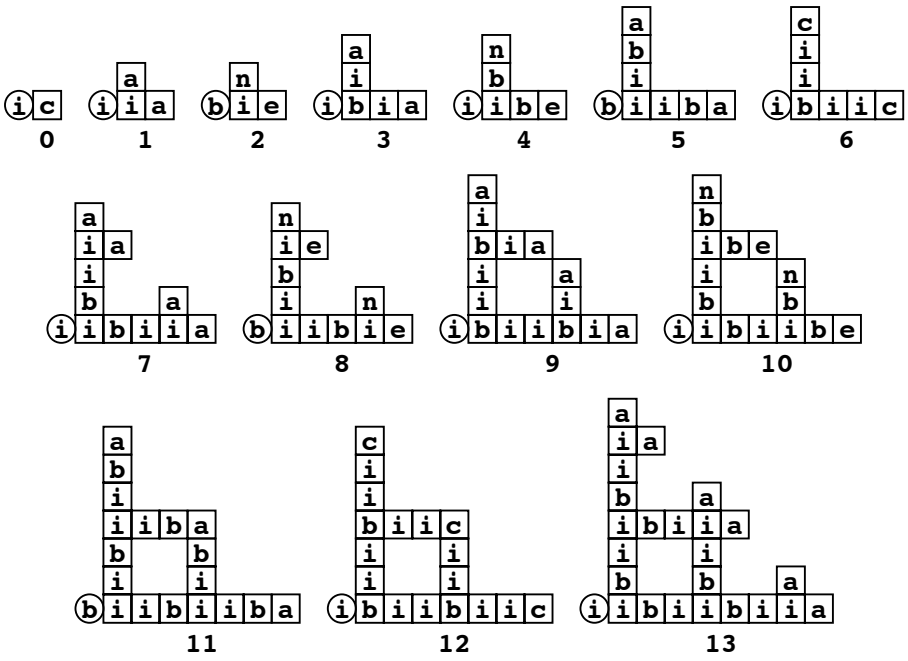


Figure 10.23 Evolution of the space divider CA implementation.

evolution of the CA implementing the 20 productions appears in Figure 10.23.

We can translate the 20 productions of Figure 10.22b in order to find the transition rules of our CA. These transition rules define the future state $C+$ of the basic cell as a function of its present state C and of the present states W and S of its neighboring cells to the west and the south only. Figure 10.24 lists the 16 transition rules that came out of the former 20 productions. In these rules the symbol o stands for the empty state.

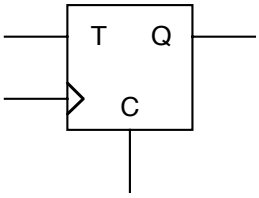
The hardware implementation of the space divider automaton requires a minimal number of three state bits per cell. The idea is to store these three bits in T-type flip-flops. According to the operation table of this device (Figure 10.25b), the control variable $T = 1$ will allow the replacement of the letters in the developmental process. In order to minimize the combinational circuit generating the T variables of the three flip-flops, we can first represent the transition rules as a state graph (Figure 10.26). We have then to find a three-bit minimal code corresponding to the variation of a limited number of state variables for every transition in the graph. This is done in the Karnaugh map of Figure 10.27a.

Based on the position of the states in the table and on the blocks that can be drawn around them, we derive respectively the full three-bit minimal codes and the corresponding existing simplified ones (Figure 10.27b). Applying these codes

C	W	S	C+
o	e	-	a
o	-	n	a
o	c	-	a
o	-	c	a
a	i	-	e
a	-	i	n
a	b	-	c
a	-	b	c

C	W	S	C+
e	i	-	i
e	b	-	b
n	-	i	i
n	-	b	b
c	-	-	i
i	b	-	b
i	-	b	b
b	-	-	i

Figure 10.24 Transition rules of the space divider CA cell.



(a)

operation	description	C	T
CLEAR	Q := 0	1	-
HOLD	Q <- Q	0	0
TOGGLE	Q <- $\overline{Q}$	0	1

(b)

Figure 10.25 T-type flip-flop. (a) Logic diagram. (b) Operation table.

to the transition rules of Figure 10.24, we find out the truth table of Figure 10.28a. Given the state of the cell C and of its immediate west (W) and south (S) neighbors at derivation step n, this table defines the cell's state $C+$ at step $n+1$. Comparing these two C and $C+$ states for each rule, we also assign the logic value 1 to the controlvariable T corresponding to the changing state variables. The straightforward translation of the truth table leads us to the equations of the control variable T of the three flip-flops (Figure 10.28b).

Figure 10.29 shows the block diagram of the CA basic cell. The minimal binary coding of this cell requires the three state bits $C[2:0]$. In order to define a description able to be synthesized in any given technology, we create a register transfer level VHDL file of the CA basic cell. Using the synthesis tool from the VIEWlogic system along with the XC4000 family from XILINX as the

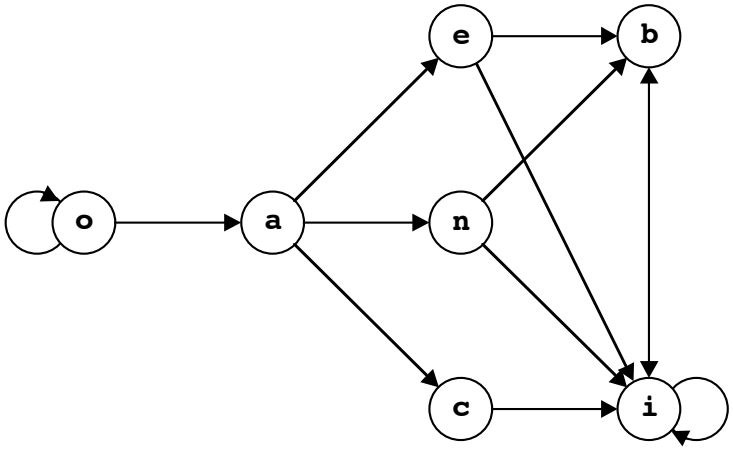


Figure 10.26 State graph of the space divider CA cell.

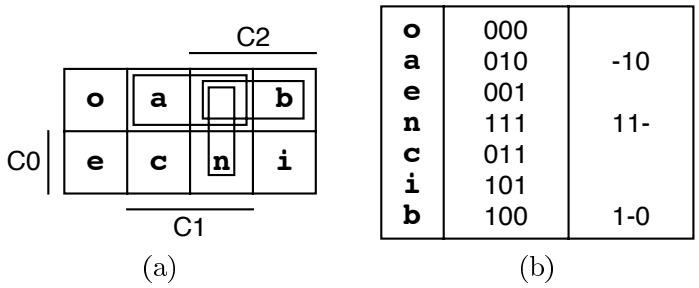


Figure 10.27 Space divider CA cell. (a) Karnaugh map. (b) State codes.

implementation technology, we end up with a logic diagram comprising 30 cells. After placement and routing, this design occupies 6 configurable logic blocks (CLBs) of a XILINX 4010 chip.

10.6 Concluding remarks

In this chapter we have shown how L-systems can be used to specify growing structures. These are then transformed into CAs, ultimately resulting in a hardware implementation. Two examples were given: a one-dimensional square-root growth function, and a two-dimensional space divider.

The study of systems that exhibit growth is interesting both from a theoretical standpoint as well as from a practical one. This chapter has shed light on the

C	W	S	C+	T
000	001	-	010	010
000	-	11-	010	010
000	011	-	010	010
000	-	011	010	010
010	101	-	001	011
010	-	101	111	101
010	1-0	-	011	001
010	-	1-0	011	001
001	101	-	101	100
001	1-0	-	100	101
111	-	101	101	010
111	-	1-0	100	011
011	-	-	101	110
101	1-0	-	100	001
101	-	1-0	100	001
100	-	-	101	001

(a)

$$T2 = \overline{C1}\,\overline{C0} \cdot S2\,\overline{S1}\,S0 + C2\,C0$$

$$T1 = \overline{C2}\,\overline{C1}\,\overline{C0} \cdot \overline{W2}\,W0 + C2\,\overline{C1}\,C0 \cdot \overline{S1}\,S0 + C1\,\overline{C0} \cdot W2\,W1\,W0 + C1\,C0$$

$$T0 = \overline{C1}\,\overline{C0} + C1\,C0 \cdot W2\,\overline{W0} + C2\,\overline{C0} \cdot S2\,\overline{S0} + C2\,\overline{C0}$$

(b)

Figure 10.28 Space divider CA cell. (a) Truth table. (b) Equations.

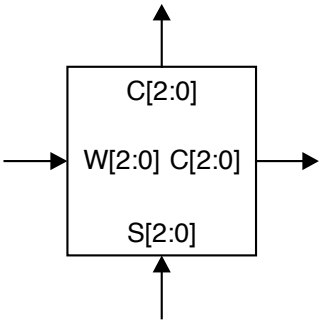


Figure 10.29 Space divider cell block diagram.

possible use of L-systems as an exploratory tool within the realm of growth and development.