Cellular Automata equivalent to D0L Systems

Abdel Latif Abu Dalhoum, Alfonso Ortega, Manuel Alfonseca
Ingeniería Informática
Universidad Autónoma de Madrid
Campus de Cantoblanco, 28049 Madrid
Spain
{Abdel.Latif;Alfonso.Ortega; Manuel.Alfonseca }@ii.uam.es

Abstract: - This paper describes an algorithm for the construction of a certain one-dimensional cellular automaton equivalent to a given D0L system. A cellular automaton is considered equivalent to an L-system if both generates the same words in the same order. Our cellular automata produce the same words and in the same order as the given D0L system for a finite number of derivations. There is no constraint to the D0L system considered, so the method is a general algorithm and can be used as a proof for an equivalence theorem.

Key-Words: - Cellular automata, Lindenmayer systems, parallel derivation grammar, theoretical computer science, formal languages

1 Introduction

A D0L system [1] is a three-fold with an alphabet (a finite non-empty set of symbols); a set of production rules that determines the only way each symbol of the alphabet can be changed by a word; and a starting word or axiom. A derivation of a word in a D0L system is the new word obtained when each symbol in the first word is replaced by applying the allowed transformations. The set of words derived from the axiom is named the language of a D0L system.

Formally a D0L system is $S=(\Sigma,P,\omega)$

where

- $\Sigma\neq\varnothing$ is a non-empty set of symbols, the alphabet.
- $P\subseteq\Sigma\times\Sigma^*\mid\forall a\in\Sigma\exists!\alpha\in\Sigma^*,,a\in P$, is the set of production rules.
- $\omega=\omega_0\cdots\omega_{|\omega|-1}\in\Sigma^*$, is the axiom and $|\omega|$ is the length of the word ω (its number of symbols).

The expression $x \Rightarrow_S y$ or $x\Rightarrow y$ when it is clear which system is used, means that the word y is derived from the word x by means of the production rules of the system S.

Given a D0L system S its associated homomorphism will be named h. $h^i(\omega)$ is the word obtained from the axiom after i derivations.

1.1 L Systems equivalent to cellular automata

Different aspects of the relationship between L Systems and Cellular Automata have been explored: in reference [2] the structural similarities between both formalisms have been underlined by applying genetic programming to them. Stauffer and Sipper [3-5] built cellular automata equivalent to the turtle graphic interpretation of some self-replicating L systems.

The authors have previously studied the possibility of generating L Systems equivalent to given cellular automata [6-7]. this paper studies the opposite direction. An algorithm is provided to build cellular automata equivalent to D0L systems.

Signals are one of the tools used to design cellular automata for a given task, which have been widely studied in one-dimensional cellular automata [8,9], whose behavior can be represented in a bidimensional cartesian diagram: the individual automata are placed on the horizontal axis, the vertical axis is time. At time t_j, the automaton at position x_i in the grid takes the state shown at (x_i, t_j) in the diagram.

Signals are identified on this diagram as digitized continuous curves, because only integer positions are meaningful in the diagram. The simplest signals are digitized lines.

It is possible to design linear cellular automata by drawing sets of lines on R^2. But it must be possible to locate all the interesting points (origins and ends of signals, for example) over integer positions on the diagram. A more detailed description of this topic can be found in [8].

Other authors have previously tackled the design of cellular automata equivalent to D0L systems. In [8] it is shown that there exist interactive linear cellular automata able to generate signals with frequencies equal to the growth functions of D0L systems. These cellular automata are designed by means of sets of signals able to solve the problem.

In a similar way, we design linear cellular automata equivalent to given D0L systems, but our approach

differs from [8] in several main features: first, our cellular automata are not interactive, no input is needed apart from the initial configuration; second, we are not interested in the growth functions but in the generation of the same languages, that is, our cellular automata generate the same words and in the same order as the D0L systems; finally, although a description of the behavior of our cellular automata in terms of signals is also possible, we are mostly interested in the explicit definition of the whole cellular automata.

2 Formal definition of one-dimensional cellular automata

Given a set *E*, a one-dimensional grid on *E* is a function $G:Z \rightarrow E$ where $G[i]$ or G_i is the element of *E* at the i^{th} position in the grid.

A one-dimensional neighborhood *V* is a pair (k,N) where

- $k \in N$ is the number of neighbors of every automaton in the grid.
- $N \in Z^k$ is a vector of *k* integer offsets. Given the index of a position in the grid, each offset point to a different neighbor of the automaton in this position.

The predecessor / successor neighborhood is formalized as $V_{p/s}=(3,(-1,0,1))$.

A one-dimensional deterministic cellular automaton is the six-fold (G,G_0,V,Q,f,T) where

- *G* is a one-dimensional grid of automata.
- *Q* is the finite and non-empty set of possible states of the automaton in the grid.
- G_0 is the initial configuration.
- $V=(k,N)$ is a one-dimensional neighborhood.
- $f:Q \times Q^k \rightarrow Q$ is the transition function that assigns the next state to each automaton in the grid, depending on its actual state and the states of its *k* neighbors.

3 One-dimensional cellular automata n-equivalent to D0L systems

3.1 Informal description

A cellular automaton is said to be *n*-equivalent to an L system if it is possible to find the first *n* words in the language generated by the L system in a finite number of successive configurations of the cellular automaton. That is, there must be a way to link the initial configuration of the cellular automaton and the axiom of the L system and each subsequent word in the language of the L system and a configuration of the cellular automaton, provided that the order of the derivation is preserved. That is, the configuration corresponding to the first derived word must be obtained before the configuration corresponding to the second derived word, and so forth.

The way in which the cellular automaton simulates the D0L system is explained by means of the example $S=\{\Sigma=\{A,B\},P=\{A::=BC, B::=AC, C::=\lambda\}, ACCCB\}$

In order to build the equivalent one-dimensional cellular automaton, the following decisions have been taken:

- There will be cells in the linear grid of the cellular automaton to contain the symbols of the words derived by the L system (one per cell).
- The states of the automaton must provide some mechanism for the following situations: to insert new symbols in a given position and hence to move the old ones (to the right), because the words can increase their length as they are derived by the L system; to delete symbols that are derived to λ (the empty word) and therefore to move the old ones (to the left), because the words can decrease their length if the number of symbols deleted is greater than the symbols generated by symbols not derived to λ. We will use different signals to supply these mechanisms.
- As a consequence of the previous points, the beginning and the end of the word must be marked. The symbols > and < are, respectively, the left and the right marker.
- If a generation only contains a set of contiguous cells embraced by the left and the right markers and every cell that has not a marker has the symbol ◊, then the generation contains a word derived by the L system.
- Provided that the length of the derived strings can decrease, we must consider that the grid has no right end, so there are always as many right markers as needed.

Figure 1 shows the initial generation of the cellular automaton. The symbol contained in each cell appears over the displacing sub-string. This configuration represents the axiom (*ACCCB*). In every cell, the upper symbol represents the state of the cell, while the lower symbols show the string to be displaced at a given point and the signals used in the process.

The behavior of the cellular automaton is summarized in figure 2, which shows twenty-one steps in the evolution of the automaton that simulate the first derivation of the D0L system. Each row in the figure

corresponds to a different step for the cellular automaton. Time increases downwards. The first row shows the initial generation of the cellular automaton, which represents the axiom (*ACCCB*) of the D0L system.

This behavior can be summarized as follows:

- The right marker is transmitted to the left until it reaches the left marker.
- The production rules of the D0L system are applied in the opposite direction. At the same time, the extra symbols and the sub-string not yet processed are displaced until they find their final position.
- As it was previously mentioned, the displacing sub-string may become empty at a given cell. In this case, a signal is sent to the right and every symbol traversed is displaced one position to the left. When the signal reaches the right marker, it rebounds until arriving at the original cell and then derivations continue.
- When a word has been derived (rows 1 and 21 in figure 2) the automaton is ready for a new derivation.

Notice the following situations:

- The right marker propagates to the left when its left neighbor contains a symbol distinct from the markers (>, <) and an empty displacing sub-string.
- After the application of a production rule, the new symbols must be moved to their final position. This is indicated by the fact that the displacing sub-string begins with the left marker (>).
- The production rules are applied in cells that contain a non-marker symbol and a displacing sub-string beginning with the left marker (>). (a) and (b) label the cells in which production rules *A::=BC* and *B::=AC* are applied.
- When a sub-string is displaced, the first symbol after the left marker becomes the state of the cell containing the sub-string, because it has found its final position. The left marker and the remaining symbols are propagated.
- If the rule applied has the empty word (λ) as its right hand side, it is possible that the sub-string that has not been yet processed (the cells between the current position and the right marker) must be displaced to the left. In this case the signal is transmitted to the right until it rebounds against the right marker. The cell sends the signal when the lower component of the state of its left neighbor belongs to the set $\{>s,,s\in\Sigma\}$.
- Cells that have just transmitted the signal ← are marked with the displacing sub-string $<_{\leftarrow}$. They copy their next symbols from their right neighbor. A discontinuous arrow remarks this circumstance.
- When the signal ← reaches the right marker ((<,<)) it moves to the left until it finds a cell with an empty displacing sub-string (◊). Meanwhile, the symbol $<_{\rightarrow}$ is used to unmark the cells that were waiting for the return of the signal.
- An empty displacing sub-string (◊) indicates a cell that contains a symbol that is already in its final position.

When symbols are displaced, they propagate the left marker (>). In a displacing sub-string, the first symbol after the left marker will be finally placed in the cell containing the sub-string. So the next state of this cell represents a symbol that has found its final position. The next state of each cell depends only on its left and right neighbors.

This method is generalized and formalized in the following sections.

3.2 Formal description

3.2.1 Theorem

Given A D0l system $S=(\Sigma,P,\omega)$ where all the components have been previously defined, and being $L(S,n)$ the set of the first n words generated by S, starting at and including ω, there exists a one dimensional cellular automata whose states contain $L(S,n)$ in the same order. This automaton is n-equivalent to S.

3.2.2 Proof

Our proof is constructive. In the following paragraphs, a one-dimensional cellular automaton (A) that generates the same language that a given D0L system (S) is built.

To make the proof clearer, the following notation will be used:

- $\Sigma_{m_l}=\{>\}$ is the alphabet of the left marker.
- $\Sigma_{m_r}=\{<\}$ is the alphabet of the right marker.
- $\Sigma_m=\{<,>\}$ is the alphabet of both markers.
- $\Sigma_{m_s}=\{<_{\leftarrow}, <_{\rightarrow}, \leftarrow\}$ are the signal symbols used for shortening the string due to the λ rules.

- $\Sigma_{em} = \Sigma \cup \Sigma_m \cup \Sigma_{m_s}$ is the alphabet Σ extended with the markers and the signals.
- $\Sigma_e = \Sigma \cup \Sigma_m \cup \Sigma_{m_s} \cup \{\Diamond\}$
- $\Sigma_{e,k} = \bigtimes_{i=1}^{k} \Sigma_e^i$
- $\Sigma_k = \bigtimes_{i=1}^{k} \Sigma^i$

The cellular automaton we want to build is one-dimensional. As indicated in the informal description, its grid (*G*) is, at least by the right, an infinite vector of finite and deterministic automata.
The predecessor / successor neighborhood ($V_{p/s}=(3,(-1,0,1))$) is used.
As suggested by the informal description, each automaton must be able to contain the following information:

- A symbol from the alphabet of the D0L system plus the end markers.
- A displacing string of the same kind of symbols plus the signals needed to adjust the length of the next derived word.
- Thus, each possible state of the cellular automaton will be a pair formally defined as follows

$$Q \subseteq \Sigma_{e_m} \times \left(\Sigma_{m_l} \cdot \left(\Sigma_k \cup \{\lambda\}\right)\right) \cup \Sigma_{m_s} \cup \Sigma_m \cup \{\Diamond\}$$

where

- $k \leq max_{A \in \Sigma}\{|\rho(A)|\} \times |h^n(\omega)|$, where *max* represents the maximum.
- (>,>) and (<,<) are used respectively as left and right markers that fill the portion of the grid that remains unused.
- $(A,<),,A \in \Sigma$ are used to propagate the right marker.
- $(A, \Diamond),,A \in \Sigma$ are used when the symbol A reaches its final position.
- $(B,>\alpha),,B \in \Sigma \cup \{<\} \wedge \alpha \in \Sigma_k$ are used to move the displacing sub-strings.
- $(C,D),,C \in \Sigma \cup \{>\} \wedge \alpha \in \Sigma_k \wedge D \in \Sigma_s$ are used to shorten the derived string.

The cellular automaton shows initially the axiom of the D0L system. 0 will be considered always the index value corresponding to the first symbol of the words. So G_0 is defined as follows:

$$G_0(i) = \begin{cases} (\omega_j,\lambda) & if \quad i=j \\ (>,>) & if \quad i<0 \\ (<,<) & if \quad i \geq |\omega| \end{cases}$$

The set of final states is empty: $T=\varnothing$
The transition is formally defined by table 1.
This cellular automaton is designed to be *n*-equivalent to the given D0L system.

3.3 Examples

The example shown in section 3.1 can be formalized thus:
Let us take the following D0L example
S={Σ={A,B},P={A::=BC, B::=AC, C::=λ}, ACCCB}
The cellular automaton(C) equivalent to D0L system S is defined as follows

$$C=\{G_c, G_{0c}, V_{p/s}, Q_c, f_c, T_c=\varnothing\}$$

where

- The initial configuration is

$$G_0(i) = \begin{cases} (>,>) & if \quad i < 0 \\ (A,\Diamond),(C,\Diamond),(C,\Diamond),(C,\Diamond),(B,\Diamond) & if \quad 0 \leq i \leq |\omega| \\ (<,<) & if \quad i > |\omega| \end{cases}$$

- The set of states is Q_c=

$$\left\{\begin{array}{c}(>,>),(<,<),(A,\Diamond),(C,\Diamond),(B,\Diamond),(A,<),(B,<),(C,<), \\ (A,>\alpha),(B,>\alpha),(C,>\alpha)(C,\leftarrow),(C,<_{\leftarrow}),(B,<_{\leftarrow}), \\ (A,<_{\leftarrow}),(<,<_{\rightarrow}),(C,<_{\rightarrow}),(B,<_{\rightarrow}),(A,<_{\rightarrow})\end{array}\right\}$$

- The transition function is defined as shown in table 1 and the first derivation in the *S* system corresponds to the cellular automaton evolution as shown in figure 2.

Cases in table 1 not included in figure 2 can be used if the reader traces the algorithm for the following D0L systems:

- {Σ={*A,B,C*},*P*={*A::=λ,B::=λC::=λ*},*ACCC CCCB*} for cases numbers 38 and 46
- {Σ={*A,B,C*}, *P*={*A::=BCA,B::=B,C::=B*},*A*} Cases numbers 36 and 39 are applied in the first derivation.
- {Σ={*C*}, *P*={*C::=*λ}, *C*} for other interesting cases.

4 Conclusions

The relationship between L systems and cellular automata is one of the main topics of interest for the authors. In this paper, the definition of a cellular automaton equivalent to a given D0L system is formally tackled for a finite number of derivations. The authors plan to extend this result to an infinite number of derivations and more complex types of L systems in order to get a more general result.

References:

[1] A. Lindenmayer, Mathematical Models for Cellular Interactions in Development (two parts), *J. Theor. Biol.*, 18, 280-315 (1968).

[2] J.R. Koza: Discovery of Rewrite Rules in Lindenmayer Systems and State Transition Rules

in Cellular Automata via Genetic Programming, *Symposium on Pattern Formation (SPF-93)*, 1993.
[3] M.Sipper, D.Mange, A.Stauffer: Ontogenetic hardware, *BioSystems*, 44, p.193-207, 1997.
[4] A.Stauffer, M.Sipper: On the relationship between cellular automata and L-systems: The self-replication case, *Physica D*, 116, p.71-80, 1998.
[5] A.Stauffer, M.Sipper: L-hardware: Modeling and implementing cellular development using L-systems. In D. Mange and M. Tomassini, editors, *Bio-inspired Computing Machines: Toward Novel Computational Architectures*, Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, p. 269-287, 1998.
[6] Alfonseca, M.; Ortega, A.; Representation of some cellular automata by means of equivalent L Systems. *Complexity International*, Volume 7 ISSN 1320-0682
[7] Alfonseca, M.; Ortega, A.; Suárez, A.: Cellular automata and probabilistic L systems: An example in Ecology, in *Grammars and Automata for String Processing: from Mathematics and Computer Science to Biology, and Back*, ed. C. Martin-Vide & V. Mitrana, Taylor and Francis Publishers. Marzo 2003. ISBN: 0415298857.
[8] Terrier, V., Temps réel sur automates cellulaires, Ph. D. Thesis, *LIP ENS* Lyon, 1991
[9] Mazoyer, J., Terrier, V., Signals in one-dimensional cellular automata, *Theoretical Computer Science*, 217 (1999) 53-80.

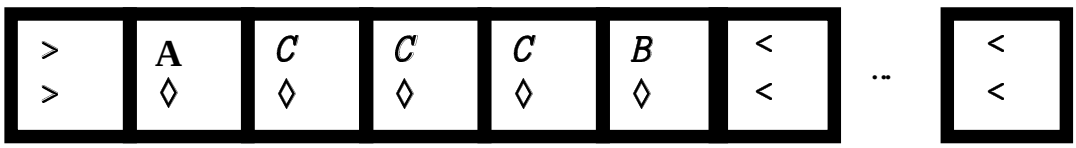
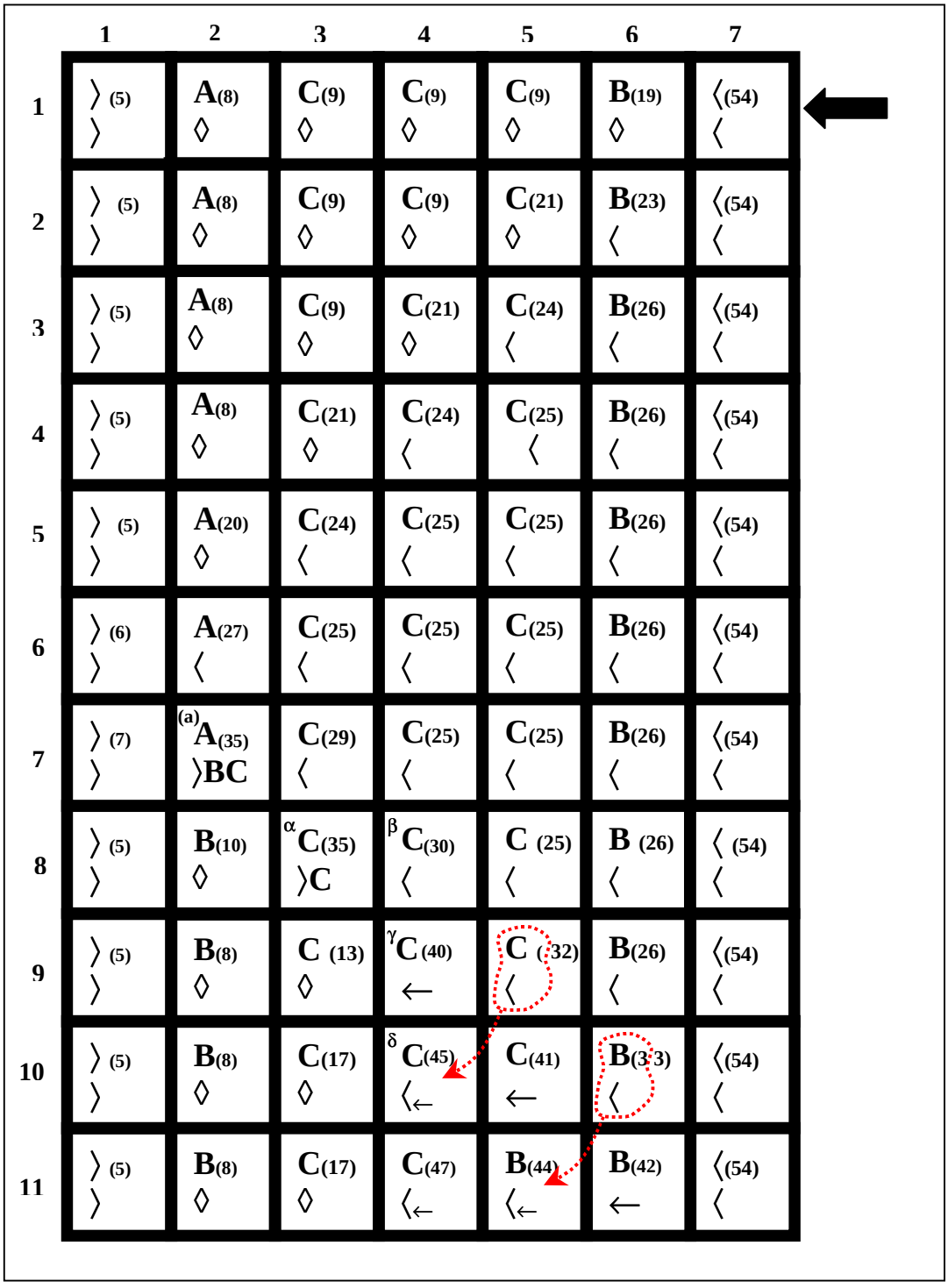


Figure 1: Initial generation of the cellular automaton.


1 2 3 4 5 6 7
1 〉(5) 〉 | A(8) ◊ | C(9) ◊ | C(9) ◊ | C(9) ◊ | B(19) ◊ | 〈(54) 〈
2 〉(5) 〉 | A(8) ◊ | C(9) ◊ | C(9) ◊ | C(21) ◊ | B(23) 〈 | 〈(54) 〈
3 〉(5) 〉 | A(8) ◊ | C(9) ◊ | C(21) ◊ | C(24) 〈 | B(26) 〈 | 〈(54) 〈
4 〉(5) 〉 | A(8) ◊ | C(21) ◊ | C(24) 〈 | C(25) 〈 | B(26) 〈 | 〈(54) 〈
5 〉(5) 〉 | A(20) ◊ | C(24) 〈 | C(25) 〈 | C(25) 〈 | B(26) 〈 | 〈(54) 〈
6 〉(6) 〉 | A(27) 〈 | C(25) 〈 | C(25) 〈 | C(25) 〈 | B(26) 〈 | 〈(54) 〈
7 〉(7) 〉 | (a) A(35) 〉BC | C(29) 〈 | C(25) 〈 | C(25) 〈 | B(26) 〈 | 〈(54) 〈
8 〉(5) 〉 | B(10) ◊ | α C(35) 〉C | β C(30) 〈 | C (25) 〈 | B (26) 〈 | 〈 (54) 〈
9 〉(5) 〉 | B(8) ◊ | C (13) ◊ | γ C(40) ← | C (32) 〈 | B(26) 〈 | 〈(54) 〈
10 〉(5) 〉 | B(8) ◊ | C(17) ◊ | δ C(45) 〈← | C(41) ← | B(33) 〈 | 〈(54) 〈
11 〉(5) 〉 | B(8) ◊ | C(17) ◊ | C(47) 〈← | B(44) 〈← | B(42) ← | 〈(54) 〈

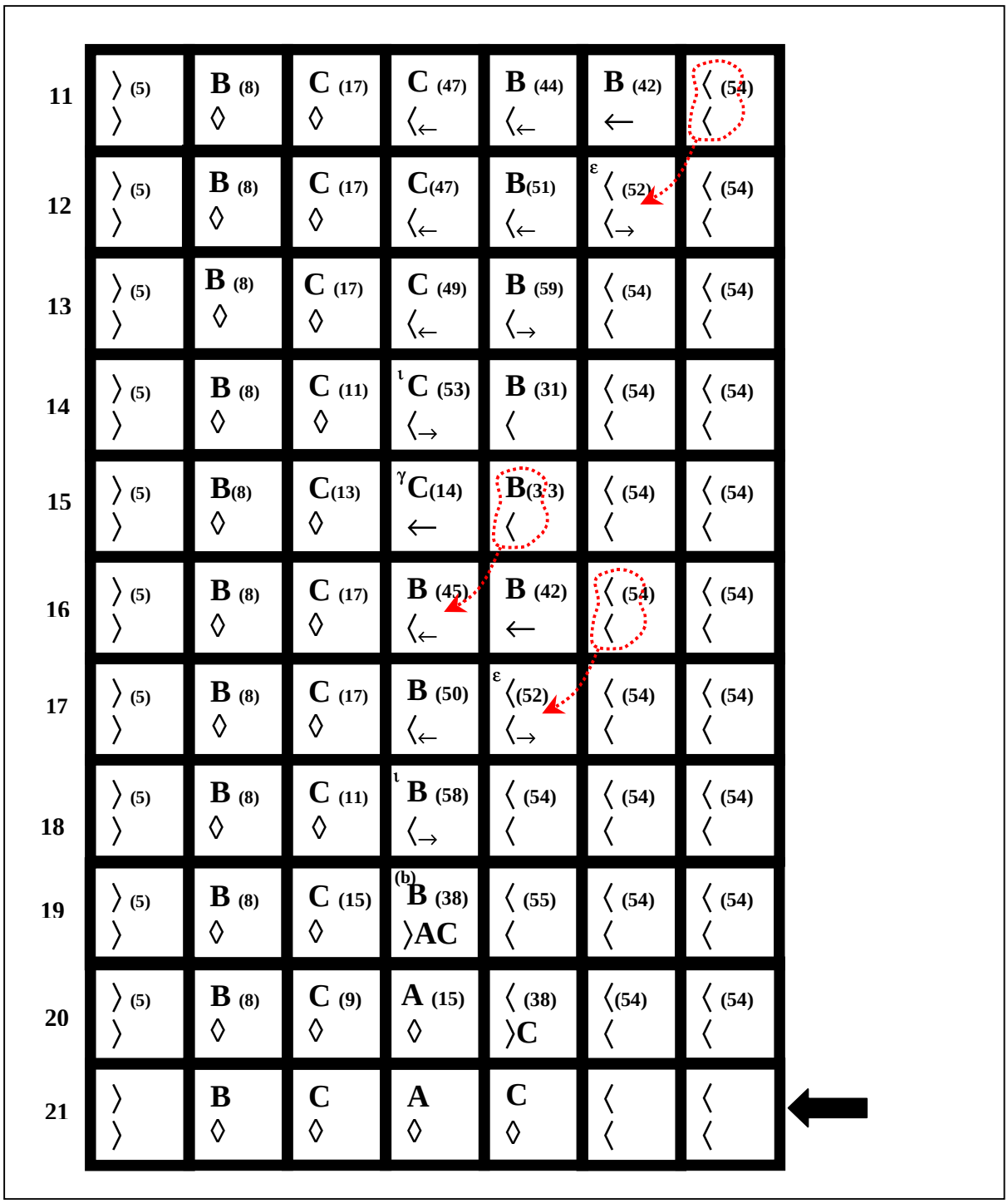


Figure 2: Cellular Automaton – D0L system comparison. (α) An erasing rule is applied. The sub-string not yet processed is not displaced to the left, because the sub-string being displaced to the right compensates that displacement. (β) Symbol C must disappear, because the sub-string being displaced to the right is empty and the applied rule (C::=λ) does not append new symbols. (γ) At this moment, a signal (←) is sent until it reaches the right markers. When the signal goes across a cell, its symbol is displaced to the left. The displacement to the left of the whole sub-string ends when the signal comes back after rebounding against the right end of the word. (δ) Cells that wait for the return of signal ← are marked with the symbol ⟨←. (ε) The signal ← rebounds against the right end of the word, the cells that receive the returning signal are marked with the symbol ⟨→. (ι) When the returning signal encounters a cell with an empty sub-string, the displacement to the left finishes and a new rule can be applied.

	$c[i-1,0]$	$c[i,0]$	$c[i+1,0]$	$f_s(C[i,0],C[i-1,0],C[i+1,0],C[i,1,0],C[i,2,0])$
1	(>,>)	(<, <)	(<, <)	(<, <)
2	(>,>)	(>,>)	(<, <)	(>,>)
3	(>,>)	(>,>)	$(<,<_{\leftarrow})$	(>,>)
4	(>,>)	(>,>)	$(s,\leftarrow)$	(>,>)$\forall s\in\Sigma$
5	(>,>)	(>,>)	$(s,\Diamond)$	(>,>)$\forall s\in\Sigma$
6	(>,>)	(>,>)	$(s,<)$	(>,>)$\forall s\in\Sigma$
7	(>,>)	(>,>)	$(s,>\alpha)$	(>,>)$\forall s\in\Sigma,\forall\alpha\in\Sigma^*\|\|\alpha\|\leq k$
8	(>,>)	$(s,\Diamond)$	$(s',\Diamond)$	$(s,\Diamond)_{\forall s,s'\in\Sigma}$
9	$(s',\Diamond)$	$(s,\Diamond)$	$(s'',\Diamond)$	$(s,\Diamond)_{\forall s,s',s''\in\Sigma}$
10	(>,>)	$(s,\Diamond)$	$(s',>\alpha)$	$(s,\Diamond)_{\forall s,s'\in\Sigma,\forall\alpha\in\Sigma+\|\|\alpha\|\leq k}$
11	$(s',\Diamond)$	$(s,\Diamond)$	$(s'',<_{\rightarrow})$	$(s,\Diamond)_{\forall s,s',s''\in\Sigma}$
12	(>,>)	$(s,\Diamond)$	$(s',<_{\rightarrow})$	$(s,\Diamond)_{\forall s,s'\in\Sigma}$
13	$(s',\Diamond)$	$(s,\Diamond)$	$(s'',\leftarrow)$	$(s,\Diamond)_{\forall s,s',s''\in\Sigma}$
14	(>,>)	$(s,\Diamond)$	$(s',\leftarrow)$	$(s,\Diamond)_{\forall s,s'\in\Sigma}$
15	$(s',\Diamond)$	$(s,\Diamond)$	$(<,>\alpha)$	$(s,\Diamond)_{\forall s,s'\in\Sigma,\forall\alpha\in\Sigma+\|\|\alpha\|\leq k}$
16	(>,>)	$(s,\Diamond)$	$(<,>\alpha)$	$(s,\Diamond)_{\forall s\in\Sigma,\forall\alpha\in\Sigma+\|\|\alpha\|\leq k}$
17	$(s',\Diamond)$	$(s,\Diamond)$	$(s'',<_{\leftarrow})$	$(s,\Diamond)_{\forall s,s'\in\Sigma}$
18	(>,>)	$(s,\Diamond)$	$(s',<_{\leftarrow})$	$(s,\Diamond)_{\forall s,s'\in\Sigma}$
19	$(s',\Diamond)$	$(s,\Diamond)$	(<, <)	$(s,<)_{\forall s,s'\in\Sigma}$
20	(>,>)	$(s,\Diamond)$	$(s',<)$	$(s,<)_{\forall s,s'\in\Sigma}$
21	$(\bar{s},\Diamond)$	$(s,\Diamond)$	$(s'',<)$	$(s,<)_{\forall s,s'\in\Sigma}$
22	(>,>)	$(s,\Diamond)$	(<, <)	$(s,<)_{\forall s\in\Sigma}$
23	$(s',\Diamond)$	$(s,<)$	(<, <)	$(s,<)_{\forall s,s'\in\Sigma}$
24	$(s',\Diamond)$	$(s,<)$	$(s'',<)$	$(s,<)_{\forall s,s',s''\in\Sigma}$
25	$(s',<)$	$(s,<)$	$(s'',<)$	$(s,<)_{\forall s,s',s''\in\Sigma}$
26	$(s',<)$	$(s,<)$	(<, <)	$(s,<)_{\forall s,s'\in\Sigma}$
27	(>,>)	$(s,<)$	$(s',<)$	$((s,>\rho(s))_{\forall s\in\Sigma\|\rho(s)\neq\lambda,\ \forall s'\in\Sigma}$
28	(>,>)	$(s,<)$	(<, <)	$(s,>p(s))_{\forall s\in\Sigma\|\rho(s)\neq\lambda}$
29	$(s',>s''\alpha)$	$(s,<)$	$(s''',<)$	$(s,>\alpha\rho(s))_{\forall s\in\Sigma\|\rho(s)\neq\lambda,}$ $\forall s',s'',s'''\in\Sigma\|,\forall\alpha\in\Sigma+\|\|\alpha\|\leq k$
30	$(s',>s'')$	$(s,<)$	$(s''',<)$	$(s,\leftarrow)_{\forall s\in\Sigma\|\rho(s)=\lambda,\ \forall s',s'',s'''\in\Sigma}$
31	$(s',<_{\rightarrow})$	$(s,<)$	(<, <)	$(s,<)_{\forall s,\ s'\in\Sigma}$
32	$(s',\leftarrow)$	$(s,<)$	$(s'',<)$	$(s,\leftarrow)_{\forall s,\ s',\ s''\in\Sigma}$
33	$(s',\leftarrow)$	$(s,<)$	(<, <)	$(s,\leftarrow)_{\forall s,\ s'\in\Sigma}$
34	$(s',<)$	$(s,<)$	(<, <)	$(s,<)_{\forall s,\ s'\in\Sigma}$

35	$(>,>)$	$(s,> s'\alpha)$	$(s'',<)$	$(s',\Diamond)_{\forall s, s', s'' \in \Sigma, \forall \alpha \in \Sigma^* \| \alpha \| \leq k}$
36	$(>,>)$	$(s,> s'\alpha)$	$(<,<)$	$(s',\Diamond)_{\forall s, s' \in \Sigma, \forall \alpha \in \Sigma^* \| \alpha \| \leq k}$
37	$(s''',\Diamond)$	$(s,> s'\alpha)$	$(s'',<)$	$(s',\Diamond)_{\forall s, s', s'', s''' \in \Sigma, \forall \alpha \in \Sigma^* \| \alpha \| \leq k}$
38	$(s'',\Diamond)$	$(s,> s'\alpha)$	$(<,<)$	$(s',\Diamond)_{\forall s, s', s'' \in \Sigma, \forall \alpha \in \Sigma^* \| \alpha \| \leq k}$
39	$(s,\Diamond)$	$(<,> s'\alpha)$	$(<,<)$	$(s',\Diamond)_{\forall s, s' \in \Sigma, \forall \alpha \in \Sigma^* \| \alpha \| \leq k}$
40	$(s',\Diamond)$	$(s,\leftarrow)$	$(s'',<)$	$(s'',<_{\leftarrow})_{\forall s, s', s'' \in \Sigma}$
41	$(s',<_{\leftarrow})$	$(s,\leftarrow)$	$(s'',<)$	$(s'',<_{\leftarrow})_{\forall s, s', s'' \in \Sigma}$
42	$(s',<_{\leftarrow})$	$(s,\leftarrow)$	$(<,<)$	$(<,<_{\rightarrow})_{\forall s, s' \in \Sigma}$
43	$(>,>)$	$(s,\leftarrow)$	$(<,<)$	$(<,<_{\rightarrow})_{\forall s \in \Sigma}$
44	$(s',<_{\leftarrow})$	$(s,<_{\leftarrow})$	$(s',\Diamond)$	$(s,<_{\leftarrow})_{\forall s, s', s'' \in \Sigma}$
45	$(s',\Diamond)$	$(s,<_{\leftarrow})$	$(s'',\leftarrow)$	$(s,<_{\leftarrow})_{\forall s, s', s'' \in \Sigma}$
46	$(s',<_{\leftarrow})$	$(s,<_{\leftarrow})$	$(s'',<_{\rightarrow})$	$(s,<_{\rightarrow})_{\forall s, s', s'' \in \Sigma}$
47	$(s',\Diamond)$	$(s,<_{\leftarrow})$	$(s'',<_{\leftarrow})$	$(s,<_{\leftarrow})_{\forall s, s', s'' \in \Sigma}$
48	$(s',<_{\leftarrow})$	$(s,<_{\leftarrow})$	$(s'',<_{\leftarrow})$	$(s,<_{\leftarrow})_{\forall s, s', s'' \in \Sigma}$
49	$(s',\Diamond)$	$(s,<_{\leftarrow})$	$(s'',<_{\rightarrow})$	$(s,<_{\rightarrow})_{\forall s, s', s'' \in \Sigma}$
50	$(s',\Diamond)$	$(s,<_{\leftarrow})$	$(<,<_{\rightarrow})$	$(s,<_{\rightarrow})_{\forall s, s' \in \Sigma}$
51	$(s',<_{\leftarrow})$	$(s,<_{\leftarrow})$	$(<,<_{\rightarrow})$	$(s,<_{\rightarrow})_{\forall s, s' \in \Sigma}$
52	$(s,<_{\leftarrow})$	$(<,<_{\rightarrow})$	$(<,<)$	$(<,<) \forall s \in \Sigma$
53	$(s',\Diamond)$	$(s,<_{\rightarrow})$	$(s'',<)$	$(s,\leftarrow)_{\forall s \in \Sigma \mid \rho(s)=\lambda, \forall s', s'' \in \Sigma}$
54	(s',x)	$(<,<)$	$(<,<)$	$(<,<) \forall s \in \Sigma, \forall x \neq \rangle \alpha ,, \alpha \in \Sigma+$
55	$(s,> s'\alpha)$	$(<,<)$	$(<,<)$	$(<,>\alpha)_{\forall s, s' \in \Sigma, \forall \alpha \in \Sigma+}$
56	$(>,>)$	$(s,<)$	$(<,<)$	$(s,\leftarrow)_{\forall s \in \Sigma \mid \rho(s)=\lambda}$
57	$(>,>)$	$(<,<_{\rightarrow})$	$(<,<)$	$(<,<)$
58	$(s',\Diamond)$	$(s,<_{\rightarrow})$	$(<,<)$	$(s,>\rho(s))_{\forall s \mid \rho(s)\neq\lambda, \forall s' \in \Sigma}$
59	$(s',<_{\leftarrow})$	$(s,<_{\rightarrow})$	$(<,<)$	$(s,<)_{\forall s, s' \in \Sigma}$
60	$(<,> s'\alpha)$	$(<,<)$	$(<,<)$	$(<,>\alpha)_{\forall s, s' \in \Sigma, \forall \alpha \in \Sigma+}$

Table 1: Cellular Automaton's transition function.