

## Chapter 12

# Pattern Formation Using Cellular Automata and L-Systems: A Case Study in Producing Islamic Patterns

Seyyed Amir Hadi Minoofam, Mohammad Mahdi Dehshibi,  
Azam Bastanfard, and Jamshid Shanbehzadeh

**Abstract.** The science of pattern formation deals with the visible, (statistically) orderly outcomes of self-organization and the common principles behind similar patterns in nature. Cell pattern formation has an important role in both artificial and natural development. Different methods have been utilized for pattern formation such as geometrical, Cellular Automata (CAs), and L-Systems. In this chapter, we concentrate our aim on introducing the role of CAs and L-Systems in pattern formation and how to extract optimum rules in terms of numbers and functionality for this aim. Because a few works have been reported in the field of script generation, we take generating Ma'qeli script and Holy words patterns, as a case study, in hand. Results of this study show the superiority of the proposed method in comparison with geometrical and fractal approaches in case of the time complexity in word production, simplicity of extracted rules, and possible reusability of the CAs rules in generating other script patterns in other languages. Moreover, the proposed method is shape-resistance, which can be less seen in fractals and geometrical based pattern formation.

---

Seyyed Amir Hadi Minoofam  
Department of Computer Engineering, Islamic Azad University,  
Nazar Abad Center, Karaj, Iran  
e-mail: minoofam@qiau.ac.ir

Mohammad Mahdi Dehshibi  
Department of Computer Engineering,  
Islamic Azad University, Science and Research Branch, Tehran, Iran  
e-mail: mohammad.dehshibi@piaou.ac.ir

Azam Bastanfard  
Department of Computer Engineering,  
Islamic Azad University, Karaj Branch, Karaj, Iran  
e-mail: bastanfard@kiaou.ac.ir

Jamshid Shanbehzadeh  
Department of Computer Engineering, Kharazmi University, Tehran, Iran  
e-mail: jamshid@saba.tmu.ac.ir

## 12.1 Introduction

According to Wolfram [9], there are no studies that cannot be modeled by cellular automata. Nowadays, the footstep of cellular automata, either synchronous or asynchronous, can be found in different aspects of science [20] from modeling a biological system [22] to produce a virtual social network [23]. Artificial models of cellular development have been proposed over the years with the objective of understanding how complex structures and patterns can emerge from one or a small group of initial undifferentiated cells [6]. This mathematical tool has the capability of easing many proposed solutions and modeling them more intelligently.

On the other hand, an L-system (Lindenmayer system) is a parallel rewriting system and a type of formal grammar. An L-system consists of an alphabet of symbols that can be used to make strings, a collection of production rules that expand each symbol into some larger string of symbols, an initial 'axiom' string from which to begin construction, and a mechanism for translating the generated strings into geometric structures. Lindenmayer [12] used L-systems to describe the behavior of plant cells and to model the growth processes of plant development. L-systems have also been used to model the morphology of a variety of organisms [20] and can be used to generate self-similar fractals such as iterated function systems.

In [11], 'Game of Life' with complex behaviors was characterized by simple synchronous cellular automata's rules. The main contribution towards this work was to design a simple set of rules to study the macroscopic behavior of a population. The Firing Squad [10], Firing Mob [16], and Queen Bee [7] are other games in which synchronization problems are investigated adequately. Piwonska and Serebinski [19] studied the impact of utilizing genetic algorithm on extracting optimum rules for 2D cellular automata. They utilized optimum extracted rules with von Neumann neighborhood in order to reconstruct several patterns. Chavoya et al. [5] consider the problem of growing a solid French flag pattern in a 3D virtual space. They proposed an artificial development model for 3D cell pattern generation based on CAs. Cell replication is controlled by a genome consisting of an artificial regulatory network and a series of structural genes. The genome was evolved by a genetic algorithm in order to generate 3D cell patterns through the selective activation and inhibition of genes. Morphogenetic gradients were used to provide cells with positional information that constrained cellular replication in space.

Bentley et al. [4] describe a novel computer simulation that uses evolution to exhibit some of the functions of cell walls raphid pennate diatom valves. The model of valve morphogenesis used was based on theories that highlight the importance of cytoskeletal elements in valve development. An 'organic' negative imprint is grown in a grid-based system, using both local and global rules to dictate grid cell states. Silica then diffuses out into all remaining grid cells. At each stage of development the generated valves were consistent with observations on real diatom valve growth. Simulated models are extremely useful for investigating, visualizing, and developing theories of morphogenesis. It is the intention of this work to inspire further model-based experiments and to try to bridge the gap between the disparate fields of computer science and biology for the exploration of morphogenesis. This model

of diatom valve morphogenesis is interestingly similar to the negative technique used by artists in batik painting.

In [13], scholars use cellular automata for modeling shell pigmentation of molluscs. They found self-organization into stationary (Turing) structures, travelling waves, chaos, and so-called class IV behavior during their research. Class IV behavior consists of a disordered spatio-temporal distribution of periodic and chaotic patches, which differs from chaos in that it has no well-defined error propagation rate. The calculations of the modes agree well with observations in natural shells. Their results suggest evidence in nature for class IV behavior, a mode that had so far been reported only as the result of simulations. Oya et al. [18] use single-electron circuits to perform dendritic pattern formation with nature-inspired cellular automata. They propose a novel semiconductor device in which electronic-analogue dendritic trees grow on multilayer single-electron circuits. A simple cellular automaton circuit was designed for generating dendritic patterns by utilizing the physical properties of single-electron devices.

Another application of pattern formation, which is a highly demanded topic among computer graphics researchers, is producing cultural-related motifs which are still yet to mature due to their production complexities. Cellular automata show their brilliant capability in this application too; as a result, much research has been devoted to apply this mathematical tool for generating complex patterns [12]. Arata et al. [3] made an effort on applying cellular automata with Margolus neighborhood to model an interactive free-form scheme within a 3D Voxel space for designing virtual clay objects. They assumed each Voxel is allocated a finite state automaton that repeats state transitions according to the conditions of its neighbor Voxels. Typeface and language's script are other attractive patterns, which entices researchers to bring their utility and attractiveness into focus. Xu et al. used a computational approach to digital chinese calligraphy and painting [24]. Ahuja et al. [2] utilized a number of geometric shapes to tessellate the 'Ali' pattern. In [15] a prototype for performing geometrical transformations was introduced with the aim of decoration designs by the Kufic square scripts. Ma'qeli script is a sort of ancient Persian script with amazing features (refer to Fig.12.1); however, a small number of its patterns were produced by Minoofam and Bastanfard [14] utilizing synchronous cellular automata.

This chapter will review the basic foundation of CAs and L-Systems in pattern formation especially those which are related to the cultural heritage. Then, it will be concentrated on generating all forms of the Ma'qeli script with CAs and three holy Islamic words using L-Systems with two clear reasons which one of these is decidedly more glamorous than the other one. The glamorous reason deals with the distinctive features of Persian script, i.e., it is cursive, it depends on the base line, and it is the formal writing typeface of more than 150 million people. All of this has overshadowed the less glamorous side of the reason, which deals with the tessellated nature of Ma'qeli script. This reason seems to have been shown more adaption with cellular automata rather than geometric methods. The focus of this study is not only on generating Ma'qeli script pattern utilizing 2D asynchronous cellular automata and Margolus neighborhood, but also on finding optimum set of rules for this sort



**Fig. 12.1** A miniature consists of Ma'qeli pattern of “**مبارک باد**” which means ‘Congratulations’

of script which facilitates reusability possibilities in generating similar patterns. In order to show the simplicity and efficiency of the proposed method, a set of experiments is conducted in which the capability of both synchronous and asynchronous cellular automata with 1D and 2D structures and different neighborhoods are examined. It was observed in the course of experiments that the contribution towards this study has a number of advantages. First, this script, as a graphical primitive, has the capability of being utilized in graphical applications such as computer games, animations, and so forth [8]. Moreover, it will assist industrial applications to show sentences on 7-segments or dot matrix monitors. In short term too, an improvement of this research’s understanding can be used as an add-on for CAD software to make it applicable in the domain of cultural heritages, handicrafts and the calligraphies which are woven through the carpets.

The roadmap of this chapter is as follows. Section 2 describes preliminary definition and terminology. The proposed method is explained in Section 3. Experimental results are depicted in Section 4, and finally this chapter is concluded in Section 5.

**Fig. 12.2** Six ‘Ali’ written in Ma’qeli script in a coin such that three are inscribed in shape of hollows and the others are embedded between the hollows



## 12.2 Preliminary Definitions and Terminologies

The Ma’qeli script is a sort of Persian script which consists of 32 characters. However, it usually appears in 64 different letter’s shapes in four different groups of the Initial, Medial, Final, and Isolated forms. The Ma’qeli characters are constructed based on the squares which resemble the markings of a grid paper. There is no curve in this script and both horizontal and vertical lines have odd length in a grid paper. This kind of alphabet is generally utilized to tile shrines with words such as Allah and Muhammad, and has an interesting appearance. In other words, tiling has been done such that only some of these words can be seen in the first impression; however, to look carefully, more words can be recognized in that geometrical object. For instance, an observer could see only three ‘Ali’ in Fig. 12.2 at first glance, but having a closer look will result in finding six embedded ‘Ali’ in the golden coin.

This interesting feature motivated us to explore whether this script could be effectively used to generate graphical text patterns. Fig.12.3 shows all Persian alphabet letters in Ma’qeli script.

As shown in Fig. 12.3, the characters with similar shapes are grouped together and the only difference between members of these groups is in existence of additional diacritics including dots and lines, the {س، ش} and {ک، گ} groups are clarifying examples with respect to the former and the latter differences. Production of a word pattern using the Ma’qeli script can be done by the following meta-rules:

- The positions of each letter with regard to the base line.
- Attaching adequate additional diacritics to letters
- The connection or separation of each letter to/from other ones in a word, and space between each words.
- The expansion of letters and words in both vertical and horizontal directions.
- The square cells which fill a grid paper or tessellate a construction.
- Considering a white space before writing a word as the most important principle of calligraphy.

A 2D asynchronous cellular automaton consists of cells arranged in a rectangular array, in which space and time are discrete. Each cell has  $k$  finite discrete states with varying values between 0 and  $k-1$ . At each discrete time step, the states of all cells are updated asynchronously according to a local rule that depends only upon the state of the cell and its neighbors. Although there is no limitation for defining neighbors in two dimensions cellular automata, models of spatial processes usually use one of the only three regular tessellations in the plane which are squares, hexagons,

الفبا Alphabets	آخر Final	وسط Medial	اول Initial	مجزأ Isolated
ا A	ا			ا
ب ت پ ب Th T P B	ب	ب	ب	ب
ج ح چ خ Kh H Ch J	ج	ج	ج	ج
د Dh D	د			د
ز ر ژ Zh Z R	ز			ز
س Sh S	س	س	س	س
ص Dh S	ص	ص	ص	ص
ط Th T	ط	ط	ط	ط
ع غ Gh E	ع	ع	ع	ع

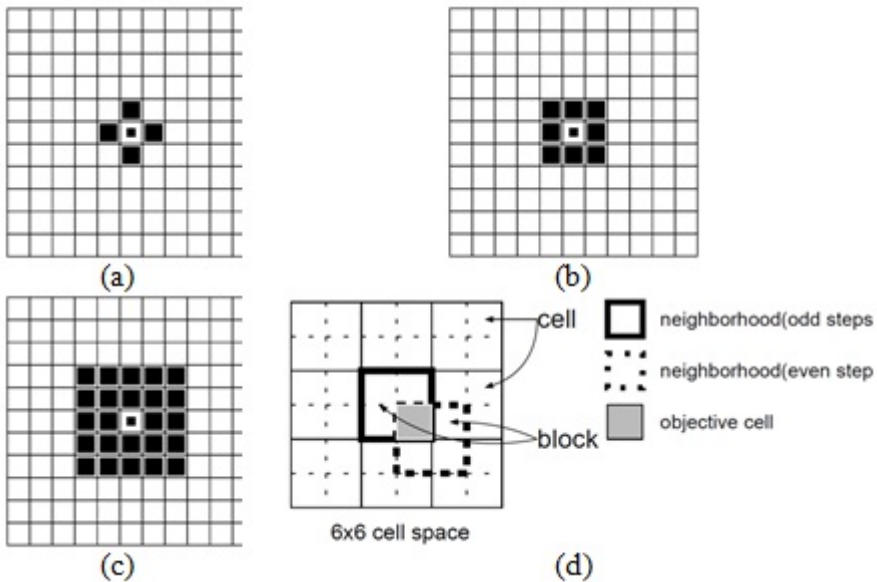
الفبا Alphabets	آخر Final	وسط Medial	اول Initial	مجزأ Isolated
ف F	ف	ف	ف	ف
ق Q	ق	ق	ق	ق
گ ک G K	گ	گ	گ	گ
ل L	ل	ل	ل	ل
م M	م	م	م	م
ن N	ن	ن	ن	ن
و W	و			و
ه H	ه	ه	ه	ه
ی Y	ی	ی	ی	ی

Fig. 12.3 Persian alphabet letters in Ma'qeli script [21]. The dashed lines show the base lines of each letter.

or triangles. The orthogonal neighbors within squares are called the von Neumann neighborhood (Fig. 12.4a), and those plus the diagonal neighbors are called the Moore neighborhood (Fig. 12.4b and Fig. 12.4c). An alternative is the Margolus neighborhood, which is used in Block Cellular Automata [17], and is implemented with two  $n \times n$ -cell blocked square tessellations. These two square tessellations are offset from each another by one row and one column. Fig. 4d visualizes a specific kind of Margolus neighborhood in which each block consists of  $2 \times 2$ -cell tessellations.

### 12.3 Proposed Method

As is previously mentioned, letters, words, and sentences expand in two-dimension and this demand inspires us to utilize 2D cellular automata. Since it is important to establish a trade-off between the form of patterns and the number of productive rules, block cellular automata are utilized by dividing the lattice of cells into

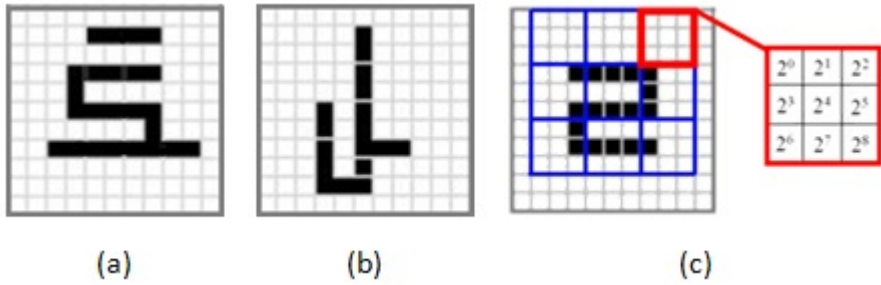


**Fig. 12.4** Topologies analyzed in the paper are (a) squares with the von Neumann neighborhood, (b) squares with the Moore neighborhood, (c) squares with the extended Moore neighborhood, and (d) the Margolus neighborhood

non-overlapping blocks. Block cellular automata are useful for generating Ma'qeli scripts, since they are straightforward to choose transition rules that obey writing laws and apply them simultaneously and synchronously to a whole block at a time rather than a single cell. In view of the fact that synchronous cellular automata simultaneously apply all transition rules to all blocks, the procedure of script generation encounters serious problems. In order to achieve a correct script generation, we have to be aware of the fact that each rule must be executed in a correct time step. Not only does this type of supervision make rule extraction process difficult, but also it is impossible in some cases. Therefore, in present study we use asynchronous cellular automata to ease both the pattern formation and the rule extraction process.

### 12.3.1 *Ma'qeli Character Generation Using Margolus Neighborhood*

The proposed block scheme consists of Margolus neighborhood itself in which the lattice is divided into  $3 \times 3$ -cell blocked square tessellations, in as much as all Ma'qeli characters can be bounded by such block (Fig. 12.5). An extension to a standard Margolus neighborhood is proposed which does not need to be shifted by one cell on alternate timestamps and can generate preferable patterns in one timestamp. As shown in Fig. 3, 64 characters are just defined without any additional



**Fig. 12.5** Three samples related to the Ma'qeli alphabets which are (a) Gaf in the form of “س” (b) Lam in the form of “ل” and (c) Ha in the form of “ه” where its partition and rule’s numbers are shown

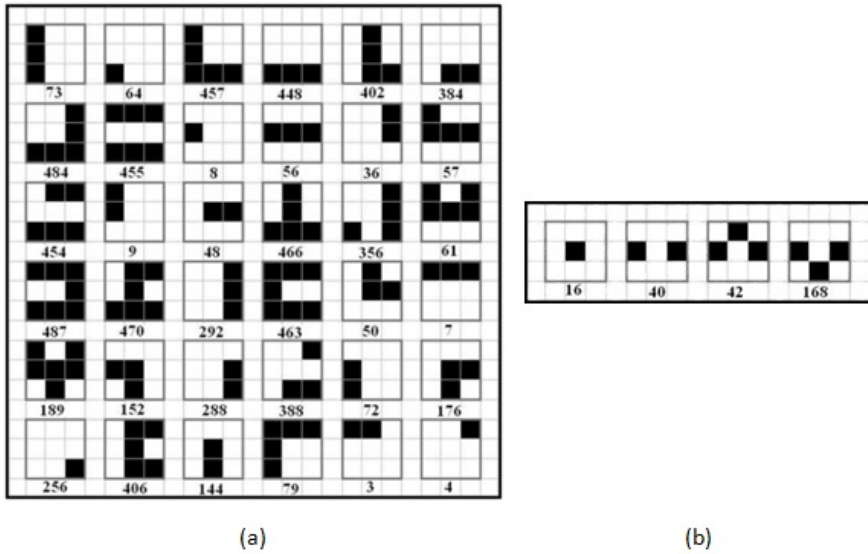
diacritics. Taking into consideration all of the additional diacritics will result in 114 distinct Persian characters in four different groups. Fig. 12.5 illustrates three out of 114 characters, which are bounded by a 3×3-block.

In order to extract rules of each 3-cells embedded in a 3 × 3-block, binary weight is utilized which is illustrated in Fig. 12.5c. In this example, the letter “ه” fills six 3×3-cells and the binary rules related to this block are 0, 0, 0, 260, 455, 74, 36, 56, and 8, respectively. As is obvious from Fig. 12.3 and Fig. 12.5, all of the 114 characters have their own rules, but extracting separated rules for them seems to be not ideal. As the second contribution towards this study, these rules were decreased down to total of 40 ones covering all 114 Ma'qeli characters. These rules are illustrated in Fig. 6 in which four rules are dedicated to put dot(s) on/under a letter.

### 12.3.2 Word and Sentence Generation Using Ma'qeli Patterns

In order to generate different words and construct sentences, additional restrictions should be observed. For the reason that Persian script is cursive in the right to left direction, connectivity rules must be obeyed in word generation, and the connection side should be specified. Four letters family {ل, ر, ج, گ} have no medial and final forms, so they cannot be connected to other letters from left, see Fig. 12.3. It may, however, be noted that other letters' family do not have this limitation and can be connected to other letters from both sides. With regard to the mentioned criteria, if the current block contains a left connected letter, the next block commences with no spaces; otherwise, the next block commences with one cell space. Considering a base line is the most important precondition for generating a well-formed sentence. To achieve this aim, three simple rules are added to each block's rules which are Rule#, Start, and End. Rule# is symbolized by a number and is composed of two parts itself in which the least significant digit shows the group of a





**Fig. 12.6** All 114 Ma'qeli script patterns are generated by means of (a) 36 body rules and (b) four dot rules. The rule's number, which is written below each 3×3-cells, presents the sum of binary value of filled cells.

letter (Initial = 1, Medial = 2, Final = 3, and Isolated = 4) and other digit(s) shows the alphabet placement of that letter.

The Start and the End rules are presented in the form of ij, where i (i = 0, 1, ..., 8) shows the ith tessellate in the letter block, and j (j = 0, 1, ..., 8) shows the jth cell in the ith tessellate. In fact, these rules regulate the initial and final cells of a block location such that each letter places on the base line. As an example, Table 1 tabulates 12 rules of a block for producing the letter "ا". The overall procedure of generating a string using Ma'qeli script is as follows below:

### 12.3.3 Holy Word Formation Using L-Systems

Word formation using L-Systems is a kind of turtle graphics in which there are three attributes including (I) a location, (II) an orientation, and (III) a pen, itself having attributes such as color, width, and up versus down. The turtle moves with commands that are relative to its position, such as "turn left 90 degrees". The pen carried by the turtle can also be controlled, by enabling it, setting its color, or setting its width. From these building blocks one can build more complex shapes like squares, triangles, circles and other composite figures. Combined with control flow, procedures, and recursion, the idea of turtle graphics is also useful in a Lindenmayer system for generating words, as we take it in hand.

L-system grammars are very similar to the semi-Thue grammar. L-systems are now commonly known as parametric L-systems, defined as a tuple  $G = (V, \omega, P)$ ,

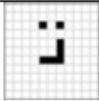
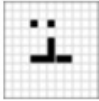
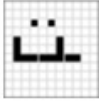
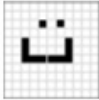
Figure	Rule#	B0	B1	B2	B3	B4	B5	B6	B7	B8	Start	End
	41	-	-	40	-	-	484	-	-	-	58	56
	42	-	40	-	-	448	457	-	-	-	58	46
	43	-	40	-	-	448	457	-	-	-	58	37
	44	-	40	-	-	448	73	-	-	-	56	37

Fig. 12.7 Different rules for producing the pattern of letter “ج” in Ma’qeli script

where V (the alphabet) is a set of symbols containing elements that can be replaced (variables),  $\omega$  (start, axiom or initiator) is a string of symbols from V defining the

---

### Generating a string using Ma’qeli script

---

**Input:** An array of string, Table of rules

**Output:** A string in the form of Ma’qeli Script

**Output** Ma’qeli\_Script (**Input**)

**Begin**

*Step1:* Get the current character and search through the alphabet file to examine whether it is Persian or not

*Step2:* If the current character does not find in the alphabet file go to **End**

*Step3:* Scan the left and right neighborhoods of character in the string to find whether it is connective or not

*Step4:* Based on output of Step 3, select the proper Rule number from rule’s table and perform **B0 - B8, Start,** and **End** rules

*Step5:* If the character belongs to the set of space, isolated, final, put a space in output by shifting current block with the amount of 1 cell

*Step6:* If there is any character go to Step 1.

**End**

---

initial state of the system, and  $P$  is a set of production rules or productions defining the way variables can be replaced with combinations of constants and other variables. A production consists of two strings, the predecessor and the successor. For any symbol 'A' in  $V$  which does not appear on the left hand side of a production in  $P$ , the identity production  $A \rightarrow A$  is assumed; these symbols are called constants or terminals. The rules of the L-system grammar are applied iteratively starting from the initial state.

The grammars for producing 'Allah', 'Muhammad', and 'Ali' are as follows:

---

### Case Allah

---

**Variables**= 'X';

**Constants** = 'F' 'f' '+' '-';

**Start** = '-FFFF-f-FFFF+f-F+f+F+F-f-F-f-ffffFF+FF-ffff-FFFF+FF+FF+FFX'

**Rules**= 'X' '-ffffFFFFFF-f-FFFF+f-F+f+F+F-f-F-f-ffffFF+FF-ffff-FFFF+FF+FF+FFX'

**Angle**= $\pi/2$ ;

---



---

### Case Muhammad

---

**Variables**= 'X';

**Constants** = 'F' 'f' '+' '-';

**Start** = '-FF+FF+FF+FFFFFF-FF+FFF+Ff+FFF-ffff-fff-f-F-F+f+F+Ff-ffff-FF-FF-FF-ffff-FF+FF+ffff+FF+FFX'

**Rules**= '-FF+FF+FF+FFFFFF-FF+FFF+Ff+FFF-ffff-fff-f-F-F+f+F+Ff-ffff-FF-FF-FF-ffff-FF+FF+ffff+FF+FFX'

**Angle**=  $\pi/2$ ;

---



---

### Case Ali

---

**Variables**= 'X';

**Constants** = 'F' 'f' '+' '-';

**Start** = '++FF+FF+FF-ffff-FFF-fff-FF+FF+FF-FF-FFFF-FFFFX'

**Rules**= 'X' '-ffffiff-FF+FF+FF-ffff-FFF-fff-FF+FF+FF-FF-FFFF-FFFFX'

**Angle**=  $\pi/2$ ;

---

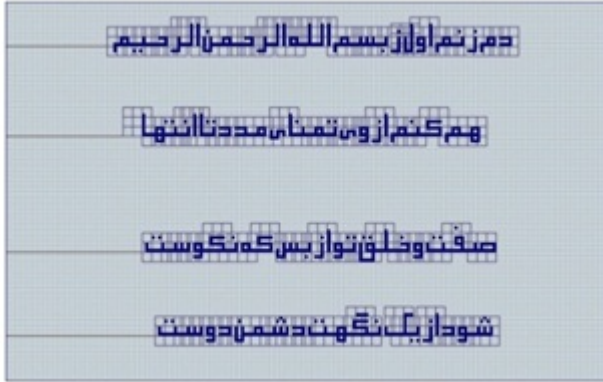


Fig. 12.8 Persian poems, which are written using Ma'qeli script

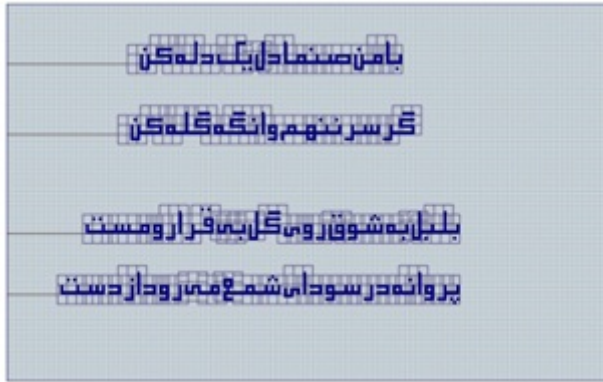
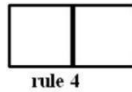


Fig. 12.9 Persian poems, which are written using Ma'qeli script. These outputs are generated in the 'Microsoft Visual Studio.Net 2008' environment.

### 12.4 Experimental Results

As is stated in the previous section, the proposed method has the capability of producing sentences without any limitation using block cellular automata and Margolus neighborhood such that the base line, as an underlying principle, is carefully observed. Figures 12.8 and 12.9 illustrate two Persian poems, which are written using Ma'qeli script. In order to demonstrate the efficiency of the proposed method, several experiments are conducted, and the capability of generating letters, words, and sentences are examined by means of 1D and 2D synchronous cellular automata with different rules and neighborhoods.



**Fig. 12.10** 1D synchronous cellular automata in which the rules number 4 is utilized to produce Aleph' environment

### 12.4.1 *Ma'qeli Script Generation Using 1D Synchronous Cellular Automata*

1D elementary cellular automata are the simplest form of automata with different applications; however, they have several problems in situation in which they are utilized in generating Ma'qeli script. The most challenging reason is that they should obey predefined rules. In this manner, Aleph, which is written in the form of "ا" is the only letter that can be produced by this type of automata in which the rule's number 4, 12, 36, 44, 68, 76, 100, 108, 132, 140, 164, 172, 196, 204, 228, or 236 needs to be utilized. Figure 12.10 illustrates one cell with rule number 4 for generation Aleph pattern.

Besides the mentioned limitation, this scheme has other shortcomings which are extracting rules for each letter is time consuming and difficult. Also even by utilizing the set of complicated rules, problems concerning the words production are considerable. These problems occur as a result of similarity between letters, and their different appearance form in a word.

### 12.4.2 *Ma'qeli Script Generation Using 2D Synchronous Cellular Automata*

The nature of letters and weakness of 1D cellular automaton directed us to examine 2D cellular automata in which three neighborhoods including von Neumann, Moore, and extended Moore are considered for generating Ma'qeli letters and words. Although these neighborhoods have the capability of using in letter and word generation, they suffer from defects, which without loss of generality we describe them by a sample.

The von Neumann neighborhood (see Fig. 12.4 a) can facilitate generating all letters through the use of several rules, although it is not free from defects. As shown in Fig. 12.11, the process of generating the pattern of letter "ا" faces a major problem regarding the production of the underneath dot, along with the high time complexity and a bad-formed illustration for this letter. This letter can be generated by the following transitional rules:

Moreover, this neighborhood cannot produce words in view of the fact that the word generation process needs to observe the location of each letter based on the

---

**Transitional rules for generation the pattern of letter <sup>46</sup> ۛ using von Neumann neighborhood**

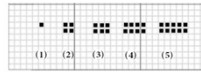
---

```

1.forn=1 to 2 do
  a.if( N= =1) then S=1 do
    // where N and S stand for North and South neigh-borhoods, respectively.
1.forn=1 to 5 do
  a.if( E= =1) then W=1 do
    //where E and W stand for East and West neigh-borhoods, respectively.
1.forn=1 to 2 do
  a.if( S= =1) then N=1 do

```

---



**Fig. 12.11** The process of generating the pattern of letter <sup>46</sup> ۛ using von Neumann neighborhoods.




**Fig. 12.12** The process of generating the pattern of letter <sup>46</sup> ۛ using Moore neighborhoods.

previous letter, which is impossible in a considerable number of cases using the structure of von Neumann neighborhood. Despite the fact that the Moore neighborhood checks more cells than von Neumann does, it is observed in the course of experiments (Fig. 12.12) that it can produce Kufic form of letter <sup>46</sup> ۛ but it still suffers from same defects and can only reduce the steps of producing letters a bit.

This letter can be produced by the following transitional rules:

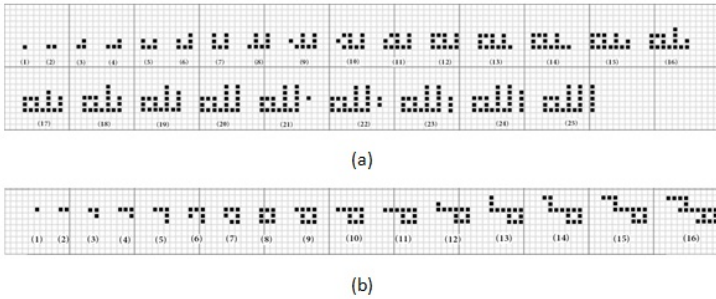
In order to resolve the mentioned weakness, the impact of using extended Moore neighborhood is examined, in which the cellular automata can supervise 25 cells simultaneously. In light of the fact that defined rules for this neighborhood have the capability of regulating more cells, it seems to have got rarely eased the process of word generation.

Minoofam and Bastanfard [14] utilized this neighborhood with the aim of simulating the holy word of ‘Muhammad’ with Ma’qeli script and this neighborhood. They reported this procedure takes 25 steps of rule execution, and the results are promising (see Fig 12.13a. In order to achieve this aim, they combined predefined rules with XOR operator ( $\otimes$ ), which are as follows below: (Consider a central cell,

**Transitional rules for generation the pattern of letter  using Moore neighborhood**



```

1.1.if( N= =1) then
{
    S=1;
    SW=1; // where SW stands for South-West neigh-borhood
}
2.for i=1 to 5 do
if(E= =1) then
{
    W=1;
    NW=1; // where NW stands for North-West neighborhood
}
if(S= =1) then N=1;
    
```



**Fig. 12.13** The procedure of generating (a) ‘Allah’ and (b) ‘Muhammad’ using Ma’qeli script and extended Moore neighborhood

Ci, and apply the following rules to: E, W, N, and S stand for East, West, North, and South neighborhood, respectively.)

Along the lines of that work, the rules for generating holy word of ‘Muhammad’ is produced by this experiment; however, it is observed that rule which is utilized for generating the letter  deforms the shape of letter  which places at the beginning of this word on account of similarity with other rules, see Fig. 12.13b. Rules for producing holy word ‘Muhammad’ with XOR operator in Ma’qeli script are as follows: (As the rule number 10 and rule number 15 follow the same structure, procedure of generating holy word ‘Muhammad’ faces problem in step 15.)

With respect to the obtained results, it is reasonable to claim that the proposed method has a considerable number of advantages over the observed competitive methods. For instance, as the proposed method works based on the separated rules, generating patterns of a word can be assigned to a parallel machine. Nevertheless, for other neighbors an adequate algorithm should be design which in the current

---



**Rules for producing holy word ‘Allah’ with XOR (⊗) operator in Ma’qeli script**

---

1.1 Ei	1.13. Wi⊗WWi⊗NWWi⊗NNWWi
1.2. Si SWi	1.14. Si SWi SWWi WWi NWWi
1.3. Ei⊗EEi⊗NEEi	1.15. Si⊗SS⊗SSWi⊗SSEEi⊗SSWi
1.4. Si⊗SEi⊗SEEi⊗EEi	1.16. Si⊗SSi⊗SSWi⊗SSWi⊗SWWi⊗WWi⊗NWWi
1.5. Si⊗SSi⊗SSWi⊗SSWi⊗SSWi	1.17. Si⊗SSi⊗SSEEi⊗SSWi
1.6. Si⊗SSi⊗SSEi⊗SSEEi⊗SEEi⊗EEi	1.18. Si⊗SSi⊗WWi⊗NWWi⊗SSWi⊗SSWi
1.7. Ei⊗EEi⊗NNEi⊗NEi	1.19. Si⊗SSi⊗SSWi⊗SSWi⊗WWi
1.8. SEi⊗SEEi⊗EEi⊗NEEi	1.20. WWi⊗NWWi⊗NNWWi⊗SWWi⊗SSWi
1.9. SWi⊗SEi⊗SSi⊗SSEi⊗SSEEi⊗EEi	1.21. Ni⊗NWWi⊗WWi⊗SSWi⊗NNWWi
1.10. Ei⊗EEi⊗Ni⊗NEEi⊗NNEi⊗NNEEi	1.22. WWi⊗NWWi⊗NNWWi⊗Ni⊗NNi
1.11. Si⊗SSi⊗SSEi⊗SSEEi⊗SEEi⊗Ei⊗EEi	1.23. Si⊗SSi⊗WWi⊗NWWi⊗SSWi⊗SSWi
1.12. Wi⊗WWi⊗NWi⊗NNWi	1.24. Si⊗SSi⊗WWi⊗SSWi⊗SSWi

---

state, it seems to be impossible. Extendibility and rule extraction are two important criteria for comparing letter generation algorithms. Extendibility means the extracted rule can be applied to produce other languages letters or pattern with a bit modification, and this issue can be seen in Margolus neighbor. As is shown in this paper, the extracted rules for Von Neumann, Moore, and extended Moore neighbors for generating letters as well as words are very complicated and in some cases are not applicable for word generation. Therefore, it is reasonable mentioning that the propose algorithm have simple rules and extendable.

Despite the fact that the obtained results are very favorable, and the proposed method tends to have the capability of being utilized in generating similar pattern, we should mention that the proposed method is not free from shortcomings. Initial and final forms of  and  cannot be produced, albeit this problem can be traced back to the predefined pattern for these letters in Ma’qeli script. If the size of letters is allowed to be modified this problem can be solved adequately; however, this alternation violates the infrastructure of this script.

---

**Rules for producing holy word ‘Muhammad’ with XOR (⊗) operator in Ma’qeli script**

---

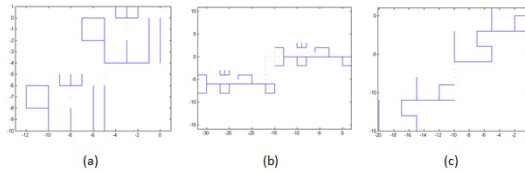
1.1 Ei	1.9. Ei⊗EEi⊗SEEi⊗SSEEi
1.2. Ni⊗NWi	1.10. Ei⊗EEi
1.3. Ei⊗EEi⊗SEEi	1.11. Si⊗SEi⊗SEEi
1.4. Ni⊗NNi⊗NNWi⊗NNWWi	1.12. Si⊗SSi⊗SSEi⊗SSEEi
1.5. Ni⊗NEi⊗NNEi⊗EEi⊗SEEi	1.13. Ei⊗SEi⊗SSEi⊗SSEEi
1.6. Ei⊗NEi⊗NNEi⊗NNi⊗NNWi	1.14. Ei⊗EEi⊗SEEi⊗SSEEi
1.7. Ni⊗NNi⊗NNEi⊗NNEEi⊗NEEi⊗EEi⊗Ei	1.15. Ei⊗EEi
1.8. Ei⊗SEi⊗SSEi⊗EEi⊗SSEEi	

---

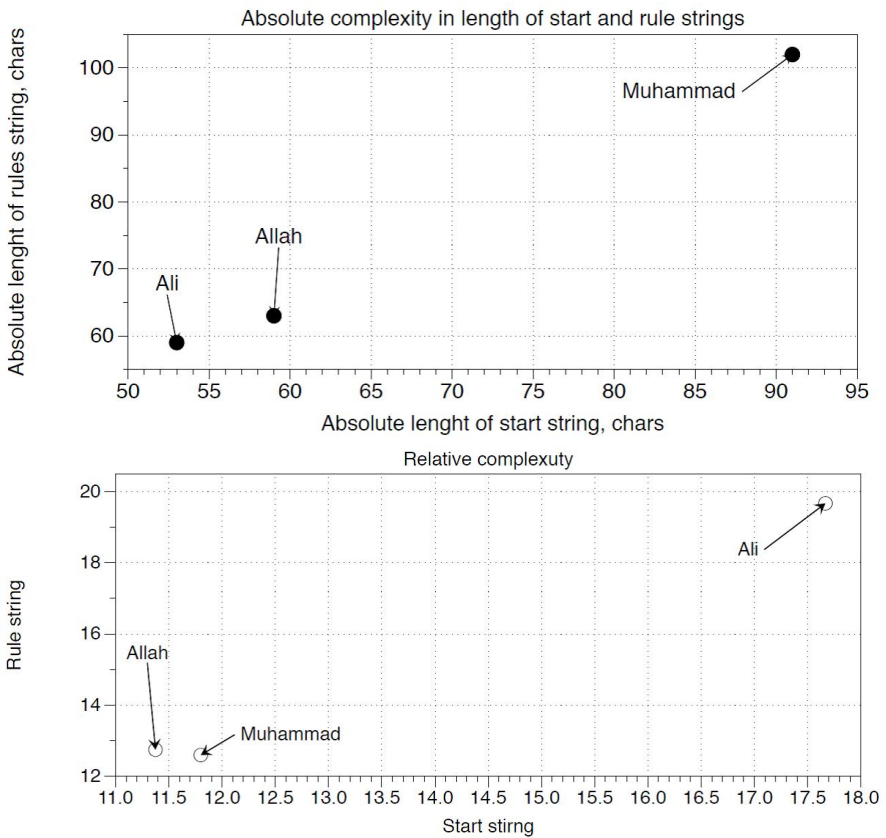


### 12.4.3 Holy Words Formation Using L-Systems

As is stated in the section 3.3, the proposed method has the capability of producing ‘Allah’, ‘Muhammad’, and ‘Ali’ using L-Systems. Figure 12.14 illustrates these words, which are written in Ma’qeli script.



**Fig. 12.14** Holy words formation using L-Systems (a) Allah, (b) Muhammad, and (c) Ali



**Fig. 12.15** (a) Absolute complexity. (b) Relative Complexity [1].

In order to demonstrate the efficiency of the proposed method, we calculate the absolute and relative complexity. Then we rewrite the rules in compressed format. For calculating absolute complexity, we count the length of the start and rule of each grammar; then, let us relative complexity which is absolute complexity divided by number of chars in Latin spelling of words. Therefore, we have absolute complexity as follows:

- Allah: Start (59), Rule (63)
- Muhammad: Start (91), Rule (102)
- Ali: Start(53), Rule (59)

And, the relative complexities are as follows:

- Allah: Start (11.8), Rule (12.6)
- Muhammad: Start (11.375), Rule (12.75)
- Ali: Start(17.6666666667), Rule (19.6666666667)

Figure 12.15 shows these two kinds of complexities.

## 12.5 Conclusion

We design cellular automaton and L-Systems based algorithms for generating the ancient Persian script Ma'qeli. We demonstrate how letters of the script can be produced using block cellular automata with Margolus neighborhood. We found a set of optimal (in terms of generation time and richness of letters/words produced) rules to generate complex Persian cursive words. The contribution towards this study has a number of advantages, and has the capability of applying to several domains. First, the script, as a graphical primitive, can be used in computer games and animations. The script can also be used in industrial applications when displaying sentences in 7-segments or dot matrix monitors. We also envisage a possible usage in CAD software in case of cultural heritages, handicrafts and the calligraphies woven through the carpets. In order to demonstrate the efficiency of the proposed method, a set of experiments is conducted on generating holy words 'Allah' and 'Muhammad' with von Neumann, Moore, and Extended Moore neighborhoods so as to compare with Margolus neighborhood. It was observed in the course of experiments that the proposed method is simple for generating words and sentences, as it can produce each letter with just one rule. In conclusion, it is worth mentioning that using L-Systems can eliminate the overhead of constructing initial blocks in the proposed method.

## References

1. Adamatzky, A., Bull, L.: Are complex systems hard to evolve? *Complexity* 14(6), 15–20 (2009)
2. Ahuja, M., Loeb, L.A.: Tessellations in Islamic calligraphy. *Leonardo* 28, 41–45 (1995)

3. Arata, H., Takai, Y., Takai, N.K., Yamamoto, T.: Free-form shape modeling by 3D cellular automata. In: International Conference on Shape Modeling and Applications, pp. 242–247. IEEE Computer Society (1999)
4. Bentley, K., Cox, E.J., Bentley, P.J.: Nature's batik: a computer evolution model of diatom valve morphogenesis. *Journal of Nanoscience and Nanotechnology* 5, 25–34 (2005)
5. Chavoya, A., Andalón-García, I.R., López-Martin, C., Meda-Campaña, M.E.: 3D cell pattern generation in artificial development. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) NICSO 2010. *SCI*, vol. 284, pp. 127–139. Springer, Heidelberg (2010)
6. Chavoya, A., Duthen, Y.: A cell pattern generation model based on an extended artificial regulatory network. *Biosystems* 94, 95–101 (2008)
7. Culik, I.K., Hurd, L., Yu, S.: Formal languages and global cellular automaton behavior. In: Howard, G. (ed.) *Cellular Automata*, pp. 396–403. MIT Press (1990)
8. Hearn, D., Baker, P.M., Carithers, W.: *Computer Graphics with Open GL*, 4th edn. Prentice Hall (2010)
9. Ilachinski, A.: *Cellular Automata: A Discrete Universe*. Emerald Group Publishing Limited (2001)
10. Kari, J.: A counter example to a conjecture concerning synchronizing words in finite automata. In: *Bulletin of the European Association for Theoretical Computer Science*, p. 146 (2001)
11. Kari, J., Moore, C.: New results on alternating and non-deterministic two-dimensional finite-state automata. In: Ferreira, A., Reichel, H. (eds.) *STACS 2001*. LNCS, vol. 2010, pp. 396–406. Springer, Heidelberg (2001)
12. Lindenmayer, A., Prusinkiewicz, P., Hanan, J.: Development models of herbaceous plants for computer imagery purposes. *SIGGRAPH Computer Graphics* 22, 141–150 (1988)
13. Markus, M., Kusch, I.: Cellular automata for modelling shell pigmentation of molluscs. *Journal of Biological Systems* 3, 999–1011 (1995)
14. Minoofam, S.A.H., et al.: Ad-hoc Ma'qeli Script Generation Using Block Cellular Automata. *Journal of Cellular Automata* 7(4) (2012)
15. Moustapha, H., Krishnamurti, R.: Arabic calligraphy a computational exploration. In: 3rd International Conference on Mathematics and Design, pp. 294–306 (2001)
16. Nasu, M.: Local maps inducing surjective global maps of one-dimensional tessellation automata. *Theory of Computing Systems* 11, 327–351 (1977)
17. Nehaniv, C.: Self-reproduction in asynchronous cellular automata. In: *Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware (EH 2002)*, p. 201 (2002)
18. Oya, T., Motoike, I.N., Asai, T.: Single-electron circuits performing dendritic pattern formation with nature-inspired cellular automata. *International Journal of Bifurcation and Chaos* 17, 3651–3655 (2007)
19. Piwonska, A., Serebinski, F.: Discovery by genetic algorithm of cellular automata rules for pattern reconstruction task. In: Bandini, S., Manzoni, S., Umeo, H., Vizzari, G. (eds.) *ACRI 2010*. LNCS, vol. 6350, pp. 198–208. Springer, Heidelberg (2010)
20. Rozenberg, G.: *The Mathematical Theory of L Systems*, vol. 90. Academic Press (1980)
21. Sakkal, M.: *Art of Arabic Calligraphy* (1993), <http://www.sakkal.com/ArtArabicCalligraphy.html>

22. Toffoli, T., Margolus, N.: Cellular Automata Machines: A New Environment for Modeling. MIT Press (1987)
23. Wolfram, S.: Cellular Automata and Complexity: Collected Papers. Westview Press (1994)
24. Xu, S., Lau, F., Pan, Y.: A computational approach to digital Chinese painting and calligraphy. Springer (2009)