Pattern formation using L-systems: A case study in forming Neyname’s words
Mohammad Mahdi Dehshibi, Ali Shirmohammadi, and Andrew Adamatzky

Citation: AIP Conference Proceedings 1648, 580015 (2015); doi: 10.1063/1.4912823
View online: http://dx.doi.org/10.1063/1.4912823
View Table of Contents: http://scitation.aip.org/content/aip/proceeding/aipcp/1648?ver=pdfcov
Published by the AIP Publishing

Articles you may be interested in
Lz -transform application to reliability measures estimation of aging MSS refrigeration system: Case study

Pattern formation in spatially extended nonlinear systems: Toward a foundation for meaning in symbolic forms
AIP Conf. Proc. 437, 469 (1998); 10.1063/1.56319

System for recognizing sentence patterns and a system for recognizing sentence patterns and grammatical cases
J. Acoust. Soc. Am. 98, 3029 (1995); 10.1121/1.413849

System for recognizing words continuously spoken according to a format
J. Acoust. Soc. Am. 79, 591 (1986); 10.1121/1.393497

Language used in UK research case studies reveals how words affect assessment scores
Phys. Today
Pattern Formation using L-Systems: A Case Study in Forming Neyname’s Words

Mohammad Mahdi Dehshibi \textsuperscript{a}, Ali Shirmohammadi \textsuperscript{b} and Andrew Adamatzky \textsuperscript{c}

\textsuperscript{a} I.A.U-Science and Research Branch, Tehran, Iran, info@dehshibi.com
\textsuperscript{b} Adiban Institute of Higher Education, Garmsar, Iran, shirmohammadi19@gmail.com
\textsuperscript{c} University of the West of England, Bristol, United Kingdom, andrew.adamatzky@uwe.ac.uk

Abstract. L-System is a parallel rewriting system and a type of formal grammar, which was introduced to be used in describing the behavior of plant cells, modeling the growth processes of plant development, the morphology of organisms, and generating self-similar fractals. The mentioned applications lie in the field of pattern formation. However, to the best of our knowledge, visual forming of a language’s words using L-Systems has not been studied yet. This paper aims to fill the gap by introducing grammars and rules of a Persian poem from Rumi, so-called Neyname, in which 108 words were generated. The main reason for selecting the Persian language is its nature in terms of complexity, dealing with the baseline, and being cursive. The results of this study show the superiority of the proposed method in comparison with geometrical and fractal approaches in case of the absolute and relative complexity in word production as well as the simplicity of the extracted rules.

Keywords: L-Systems; Pattern Formation; Persian Orthography; Persian Word Production; Rumi Poem.

INTRODUCTION

The science of pattern formation deals with the visible, (statistically) orderly outcomes of self-organization and the common principles behind similar patterns in nature. In developmental biology, pattern formation refers to the generation of complex organizations of cell fates in space and time. One of the 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. To this aim, the biological, physical or chemical processes that lead to pattern formation are simulated, which the results can be displayed in a realistic way. Calculations using models like L-Systems \cite{1}, Reaction-diffusion \cite{2} or MClone \cite{3} are based on the actual mathematical equations designed by the scientists to model the studied phenomena.

An L-System was originally described in terms of linear or branching chains of finite automata, but its subsequent reformulation in terms of rewriting systems proved more elegant. The close relationship between L-systems, abstract automata, and formal languages attracted the interest of computer scientists, who vigorously developed the mathematical theory of L-systems. This progress was followed by applications of L-systems to the modeling of plants. 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.

Because a few works \cite{4}, \cite{5} have been reported in the field of script generation, we concentrate our aim on generating one of the most famous poems of Rumi, so-called Neyname, using L-Systems. It should be noted that generating words of a language using L-Systems has not been studied yet. This paper aims at achieving this end by introducing grammars and rules of a Persian poem in which 108 words were generated. The main reason for selecting the Persian language is its nature in terms of complexity, dealing with the baseline, and being cursive.

The roadmap of this paper is as follows. Section 2 describes the preliminary definitions. The proposed method is explained in Section 3. Experimental results are depicted in Section 4, and finally this paper concludes in Section 5.

PRELIMINARY DEFINITIONS

Drawing on cellular automata and Chomsky grammars, L-systems were developed as a theoretical framework for studying development in multicellular organisms and were one of the first models used to simulate growth and development in plants. Although they bear a resemblance to cellular automata, they differ in that arrays can grow and shrink. L-systems differ from grammars in that they require parallel rewriting of all symbols and do not distinguish between terminal and non-terminal symbols.
This section presents the simplest class of L-systems, those which are deterministic and context-free. Formal definitions describing L-systems and their operation are given as: Let \( V \) denote L-system an alphabet, \( V^* \) the set of all words over \( V \), and \( V^+ \) the set of all nonempty words over \( V \). A string L-system is an ordered triplet \( G = \langle V, \omega, P \rangle \) where \( V \) is the alphabet of the system, \( \omega \in V^+ \) is a nonempty word called the axiom and \( P \subset V \times V^* \) is a finite set of productions. A production \((a, \gamma) \in P\) is written as \( a \rightarrow \gamma \). The letter “\( a \)” and the word “\( \gamma \)” are called the predecessor and the successor of this production, respectively. It is assumed that for any letter \( a \in V \), there is at least one word \( \gamma \in V^* \) such that \( a \rightarrow \gamma \). If no production is explicitly specified for a given predecessor \( a \in V \), the identity production \( a \rightarrow a \) is assumed to belong to the set of productions \( P \). An L-system is deterministic if and only if for each \( a \in V \) there is exactly one \( \gamma \in V^* \) such that \( a \rightarrow \gamma \).

Let \( \mu = \alpha_1 \ldots \alpha_m \) be an arbitrary word over \( V \). The word \( \nu = \chi_1 \ldots \chi_m \in V^* \) is directly derived from \( \mu \), noted \( \mu \Rightarrow \nu \), if and only if \( \forall i \in 1,\ldots, m \) \( \alpha_i \rightarrow \chi_i \). A word \( \nu \) is generated by \( G \) in a derivation of length \( n \) if there exists a developmental sequence of words \( \mu_0, \mu_1, \ldots, \mu_n \) such that \( \mu_0 = \omega, \mu_n = \nu \) and \( \mu_i \Rightarrow \mu_{i+1} \). The basic idea of turtle interpretation is given below. A state of the turtle is defined as a triplet \((x, y, \delta)\), where the Cartesian coordinates \((x, y)\) represent the turtle’s position, and the angle \( \alpha \), called the heading, is interpreted as the direction in which the turtle is facing. Given the step size \( d \) and the angle increment \( \delta \), the turtle can respond to commands represented by the following symbols:

- **F**: Move forward a step of length \( d \). The state of the turtle changes to \((x', y', \delta)\), where \( x' = x + d \cdot \cos \alpha \) and \( y' = y + d \cdot \sin \alpha \). A line segment between points \((x, y)\) and \((x', y')\) is drawn.
- **f**: Move forward a step of length \( d \) without drawing a line.
- **+**: Turn left by angle \( \delta \). The next state of the turtle is \((x, y, \alpha + \delta)\). The positive orientation of angles is counterclockwise.
- **-**: Turn right by angle \( \delta \). The next state of the turtle is \((x, y, \alpha - \delta)\).

Given a string \( \nu \), the initial state of the turtle \((x_0, y_0, \alpha_0)\) and fixed interpretation parameters \( d \) and \( \delta \), the turtle interpretation of \( \nu \) is the figure (set of lines) drawn by the turtle in response to the string \( \nu \).

**PROPOSED METHOD**

In order to forming the poem of Neyname, first, the related rules of words should be extracted. The easiest way to achieve this aim is to draw the Persian alphabet on a grid paper; this way makes the extraction of rules easy. Following describes our proposed method to implement this beautiful and meaningful poem, as well as rules abstraction.

One of the problems that we are dealing with the production of Persian’s words, especially those which have dot, is the correct placement of dots. The 90° in drawing words causes this problem (see Figure 1).

![Image](https://via.placeholder.com/150)

**FIGURE 1.** The problem with dot’s placement in the middle of “...” character. (a) Left alignment of the dot, (b) right alignment of the dot, and (c) deforming the shape of the dot.

In order to solve this problem we consider two solutions. First, it is possible to draw the dot in a neighbor cell and change the dot’s alignment, which is the easiest solution, or to deform the shape of the dot. However, this solution leads to deformation of the Persian characters (see Figure 1c). The second and the furthermost promising solution is to change the angle of drawing in the string from 90° to 45°. This solution has two advantages which are (I) resolving the problem of drawing the diacritical mark in case of misplacement or deformation and (II) better shaping of all characters of the Persian alphabet which makes it closer to the realistic orthography. The result of using the second solution is illustrated in Figure 2a. Despite the mentioned advantages for using a 45° angle in the visual forming of the Persian orthography in an L-System, new problem would be rose which is returning to the baseline in order to continue drawing of the rest of the word, as lots of the Persian words have dissociation.
FIGURE 2. (a) Better orthography of the "ﭗﭮﭭ" word using 45° in the rule of an L-System. (b) Procedure of synchronizing top-down and bottom-up movement for the character which has tooth-like shape. (c) The problem of returning to the baseline in case of using 45° when return up straightly.

This problem arises when using the 45° results in the production of horizontal, vertical, and diagonal directions. The vertical and horizontal directions have the length of 1; however, the length of diagonal direction is approximately 0.7. To solve this problem, it is necessary to synchronize our top-down and bottom-up movement; for example see Figure 2b which shows one of the tooth-like shape of the "ﭮ" character. In some characters, in case of moving from baseline to the bottom of the character orthogonally and returning to the baseline vertically, the end of character would not be on the baseline. In this type of characters the procedure of movements is automatically synchronized; however, in a character like "ﭮ," when moves to the bottom of the baseline then the only way is to move back to the baseline orthogonally on the drawn character using "F" (see Figure 2c). Since then, other characters can be drawn based on the designed string. The differences between the methods of returning to the baseline are as follows. Although the second string is longer, it helps drawing the Persian/Arabic characters more beautiful.

<table>
<thead>
<tr>
<th>The Start string of the &quot; pornost&quot; character (Straight movement)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-F-F-F-F-F-F-FFFFF-FF--FFF++FFF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The Start string of the &quot;牰牰&quot; character (Proposed method)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-F-F-F-F-F-F-FFFFF-FF----ff+ffff+++fffffFFF</td>
</tr>
</tbody>
</table>

EXPERIMENTAL RESULTS

Previous section introduced a novel application of L-Systems in visual forming of a cursive font using an example, a famous poem which is written by Rumi. This procedure faces several problems and we propose simple solution for them. This section will show the rule of three word from the first verse of Neyname as well as a full discussion about rule abstraction and theory of L-System’s complexity.

Rules of Neyname

In this sub-section rules of each word from the first verse of Neyname are presented as follow and subsequently their illustration will be demonstrated. All grammars have the “X” variable, the set of {’F‘ ‘f‘ ‘+‘ ‘-‘} as constants, and “X → F++F++F++F” as Rule string for the word which have dot. Moreover, the angle is set to 45°.

<table>
<thead>
<tr>
<th>The Start string of the “牰牰牰” word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start: --F-F-F++f-X-f++FF,F-F+---f-F--ff-X-ff--F-F--FF++F--FF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The Start string of the “牰牰” word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start: --FFFFFF--ff--ff-X-ff--FFFF-FF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The Start string of the “牰” word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start: --F.F-F--ff-X-ff--FFFF+FFF+F+F-F-F-F-F-F-FFFF-FF</td>
</tr>
</tbody>
</table>

These rules lead to visual forming of Figure 3 illustration:
Theory of Complexity

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 the Persian spelling of words. Therefore, we have absolute and relative complexity as follows. Figure 4 shows the words’ absolute complexity of the Neyname in comparison with its relative complexity.

<table>
<thead>
<tr>
<th>The Start string of the “سپند&quot; word</th>
</tr>
</thead>
</table>
| Start: --F-F-F++f-X-f++FF-F4+f-F-F--ff-X-ff--F-F-F-F-F-
F4+f-F-3F-F-F+F-F--ff-X-ff--4F-F-F-F-F-F-F-4F-FF |

<table>
<thead>
<tr>
<th>The Start string of the “ژ&quot; word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start: --6F--ff--fFf-X-ff-4F-FF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The Start string of the “س&quot; word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start: --F-F--ff-X-ff--3F+F+F+F+F-F-F-F-5F-FF-3F</td>
</tr>
</tbody>
</table>

FIGURE 4. Absolute complexity in comparison with relative complexity.

CONCLUSION

To the best of our knowledge, graphical word’s forming of a language using L-Systems has not been studied yet. This paper aimed at achieving this end by introducing grammars of and rules of a Persian poem from Rumi, so-called Neyname, in which 108 words were generated. The main reason for selecting the Persian language is its nature in terms of complexity, dealing with the baseline, and being cursive. The results of this study showed the superiority of the proposed method in case of the absolute and relative complexity. Moreover, there are several issues which caused the length of Start string increases abnormally, which we discuss here. (i) Those characters which are written below baseline have a longer Start string, e.g., “س," “ژ." The main reason is that we must return to the baseline while finishing the visual forming of the character. (ii) As the Persian orthography is cursive, we were forced to use a longer Start string. For example, the fracture of word “سپند" is less than the fracture of word “ژس," so needs shorter Start string.
REFERENCES