NUMBER THEORY-BASED INDUCTION OF DETERMINISTIC CONTEXT-FREE L-SYSTEM GRAMMAR

Ryohei Nakano
Department of Computer Science, Chubu University, 1200 Matsumoto-cho, Kasugai 487-8501, Japan

Naoya Yamada
Department of Computer Science and Engineering, Nagoya Institute of Technology
Gokiso-cho Showa-ku, Nagoya 466-8555, Japan

Keywords: Grammatical induction, L-system, Number theory, Plant model.

Abstract: This paper addresses grammatical induction of deterministic context-free L(D0L)-system. Considering the parallel feature of L-system production and the deterministic context-free feature of D0L-system, we take a number theory-based approach. Here D0L-system grammar is limited to one or two production rules. Basic equations for the methods are derived and utilized to narrow down the parameter value ranges. Our experiments using plants models showed the proposed methods induced the original production rules very efficiently.

1 INTRODUCTION

L-systems were originally developed by Lindenmayer as a mathematical theory of plant development (Prusinkiewicz and Lindenmayer, 1990). The central concept of L-systems is rewriting. In general, rewriting is a mechanism for generating complex objects from a simple initial object using production rules.

The most extensively studied rewriting systems operate on character strings, and Chomsky's work on formal grammars is well known. Formal grammars and L-systems are both string rewriting systems, but the essential difference between them is that in formal grammars productions are applied sequentially while in L-systems productions are applied in parallel.

The reverse process of rewriting is grammatical induction, which infers a set of production rules given a set of strings. Grammatical induction of formal grammars has been studied for decades and induction of context-free grammars is still an open problem.

Induction of L-system grammars is also an open problem little explored so far. L-systems can be classified using two axes: (1) deterministic or stochastic, and (2) context-free or context-sensitive.

(McCormack, 1993) addressed computer graphics modeling through evolution of context-free L-systems. (Nevill-Manning, 1996) proposed a simple algorithm called Sequitur, which reveals structure like context-free grammars from a wide range of sequences, however, with small success for grammatical induction of deterministic context-free L-system grammar. (Schlecht, et al., 2007) proposed statistical structural inference for microscopic 3D images through learning stochastic L-system model. (Damasevicius, 2010) addressed structural analysis of DNA sequences through evolution of stochastic context-free L-system grammars.

This paper addresses grammatical induction of deterministic context-free L(D0L)-system. Considering the parallel feature of L-system production and the deterministic context-free feature of D0L-system, we take a number theory-based approach. Here D0L-system grammar is limited to one or two production rules. Our experiments using plants models showed the proposed methods induced the original production rules quite efficiently.

2 D0L-SYSTEMS

D0L-systems. The simplest class of L-systems are called D0L-system (deterministic context-free L-system). D0L-system is defined as $G = (V, C, \omega, P)$, where V and C denote sets of variables and constants,

Nakano R. and Yamada N..
NUMBER THEORY-BASED INDUCTION OF DETERMINISTIC CONTEXT-FREE L-SYSTEM GRAMMAR.
DOI: 10.5220/0003088101940199
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2010), pages 194-199
ISBN: 978-989-8425-28-7

ω is an initial string called axiom, and P is a set of production rules. A variable is a symbol that is replaced in rewriting, and a constant is a symbol that remains fixed in rewriting and is used to control turtle graphics.

Notation. Shown below is the notation employed in the following sections. Here we assume the following form of production rules.

$$\text{rule A} : A \rightarrow ????????$$
$$\text{rule B} : B \rightarrow ??????$$

Y: given string.

n: the number of rewritings.

$Z^{(n)}$: string obtained after n times rewritings.

$\alpha_A, \alpha_B, \alpha_K$: the numbers of variables A, B and constant K occurring in the right side of rule A.

$\beta_A, \beta_B, \beta_K$: the numbers of variables A, B and constant K occurring in the right side of rule B.

y_A, y_B, y_K: the numbers of variables A, B and constant K occurring in Y.

$z_A^{(n)}, z_B^{(n)}, z_K^{(n)}$: the numbers of variables A, B and constant K occurring in $Z^{(n)}$.

3 INDUCTION OF L-SYSTEM GRAMMAR HAVING ONE RULE

When D0L-system has only one production rule, the number theory-based induction is easy. The method proposed below is called LGIN1 (L-system Grammar Induction based on Number theory for 1 rule). Here the following situation is assumed.

$$n = ?, \ \text{axiom} : A$$
$$\text{rule A} : A \rightarrow ????????$$

Given string Y, we are asked to estimate the number of rewritings n and to induce the rule A. Through simple observation we have the following.

$$z_A^{(n)} = \alpha_A^n \quad (1)$$

$$z_K^{(n)} = \begin{cases} \dfrac{1-\alpha_A^n}{1-\alpha_A}\,\alpha_K & \text{if } \alpha_A \neq 1 \\ n\,\alpha_K & \text{if } \alpha_A = 1 \end{cases} \quad (2)$$

Then we obtain the following, which we call basic equations of LGIN1.

$$y_A = \alpha_A^n \quad (3)$$

$$y_K = \begin{cases} \dfrac{1-\alpha_A^n}{1-\alpha_A}\,\alpha_K & \text{if } \alpha_A \neq 1 \\ n\,\alpha_K & \text{if } \alpha_A = 1 \end{cases} \quad (4)$$

From eq.(3) we get candidate pairs $(\alpha_A, \ n)$ by factorizing y_A into prime factors. For each candidate pair we get α_K using eq.(4) for each constant K. Since given string Y includes the right side of rule A as a substring, we exhaustively extract from Y a substring having α_A A's and α_K K's to form rule A candidate. Then we rewrite the axiom n times using the rule A candidate, and check whether the obtained string is equal to Y. If the equality holds, the rule A candidate is a solution.

Example.

$$n = 3, \ \text{axiom} : A$$
$$\text{rule A} : A \rightarrow A[+]A$$

Consider string Y shown below.

$$A[+A]A[+A[+A]A]A[+A]A[+A[+A]A[+A[+A]$$
$$A]A[+A]A]A[+A]A[+A[+A]A]A[+A]A$$

By scanning Y we get the following values.

$$y_A = 27, \quad y_+ = 13, \quad y_[= 13, \quad y_] = 13 \quad (5)$$

Since $y_A = 3^3$, we have the following two sets.
(i) $n = 1, \ \alpha_A = 27, \ \alpha_+ = 13, \ \alpha_[= 13, \ \alpha_] = 13$
(ii) $n = 3, \ \alpha_A = 3, \ \alpha_+ = 1, \ \alpha_[= 1, \ \alpha_] = 1$
The case n=1 is always a trivial one; thus, discard it. By scanning Y, we have the following two substrings having 3 A's, one +, one [, and one]:

$$A \rightarrow A[+A]A$$
$$A \rightarrow A]A[+A$$

By rewriting each of them $n(= 3)$ times and checking the equality, we select the original rule A.

4 INDUCTION OF L-SYSTEM GRAMMAR HAVING TWO RULES

When D0L-system has two production rules, the induction gets immensely complicated. The method explained below is called LGIN2 (L-system Grammar Induction based on Number theory for 2 rules). In this case the following is assumed.

$$n = ?, \ \text{axiom} : A$$
$$\text{rule A} : A \rightarrow ????????$$
$$\text{rule B} : B \rightarrow ?????$$

Given string Y, we are asked to estimate the number of rewritings n and to induce the rules A and B. LGIN2 goes in the following order.

(1) Derivation of Basic Equations. Focusing on variables, we consider the growth of the numbers of occurrences of A and B.

$$(1\ \ 0)\ \mathbf{T}^n = (z_A^{(n)}\ \ z_B^{(n)}), \quad \mathbf{T} = \begin{pmatrix} \alpha_A & \alpha_B \\ \beta_A & \beta_B \end{pmatrix} \tag{6}$$

Let λ_1 and λ_2 be eigen values of matrix $\mathbf{T}$, and $\mathbf{v}_1$ and $\mathbf{v}_2$ be their eigen vectors. Then we have the following.

$$\mathbf{T}\ \mathbf{V} = \mathbf{V}\ \Lambda, \ \ \mathbf{V} = (\mathbf{v}_1\ \mathbf{v}_2), \ \ \Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \tag{7}$$

By simple calculation we get the following.

$$\mathbf{T}^n = \mathbf{V}\ \Lambda^n\ \mathbf{V}^{-1}, \qquad \Lambda^n = \begin{pmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{pmatrix} \tag{8}$$

By substituting eigen vectors into the above we have the following.

$$\mathbf{T}^n = \begin{pmatrix} D_1 & \alpha_B\, x_n \\ \beta_A\, x_n & D_2 \end{pmatrix} \tag{9}$$

$$D_1 = \alpha_A\, x_n - (\alpha_A\beta_B - \alpha_B\beta_A)x_{n-1} \tag{10}$$

$$D_2 = \beta_B\, x_n - (\alpha_A\beta_B - \alpha_B\beta_A)x_{n-1} \tag{11}$$

$$x_n = \begin{cases} \dfrac{\lambda_1^n - \lambda_2^n}{\lambda_1 - \lambda_2} & if\ \lambda_1 \neq \lambda_2 \\ n\ \alpha_A^n & if\ \lambda_1 = \lambda_2 \end{cases} \tag{12}$$

From eqs.(6) and (9) we have the following, which we call basic equations of LGIN2.

$$y_A = \alpha_A\, x_n - (\alpha_A\beta_B - \alpha_B\beta_A)\ x_{n-1} \tag{13}$$

$$y_B = \alpha_B\, x_n \tag{14}$$

(2) Narrowing down of Variable Parameters. From eq.(14) we have candidate pairs $(\alpha_B,\ x_n)$ by factorizing y_B. Equation (13) can be used to narrow down the value ranges of α_A, β_A, and β_B. For example, considering $n = 2$ we have $z_A^{(2)} = \alpha_A^2 + \alpha_B\beta_A$, and $z_B^{(2)} = \alpha_A\alpha_B + \alpha_B\beta_B$. Using these as lower bounds, we have $y_A \geq z_A^{(2)} \geq \alpha_A^2$, and $y_B \geq z_B^{(2)} = \alpha_A\alpha_B + \alpha_B\beta_B$.

(3) Estimating the Number of Rewritings. At this stage we have candidates of $(\alpha_A, \alpha_B, \beta_A, \beta_B, x_n)$. For each candidate set we estimate the number of rewritings n in the following way. As for x_n, we can easily show that x_n is an integer and strictly increasing. Moreover, by simple calculation we have the following recurrence formula.

$$x_n = x_{n-1}\ (\alpha_A + \beta_B) - x_{n-2}\ (\alpha_A\beta_B - \alpha_B\beta_A) \tag{15}$$

Starting with x_1=1 and $x_2 = \alpha_A + \beta_B$, we increase and find n whose x_n is equal to a candidate x_n. If x_n exceeds the candidate x_n, discard the candidate.

(4) Narrowing down of Constant Parameters. For each constant K we repeat the following. Using the following we can calculate $r_A^{(n)}$ and $r_B^{(n)}$, the numbers of A and B rewritings occurred until n rewritings.

$$r_A^{(n)} = 1 + z_A^{(1)} + z_A^{(2)} + \ldots + z_A^{(n-1)} \tag{16}$$

$$r_B^{(n)} = z_B^{(1)} + z_B^{(2)} + \ldots + z_B^{(n-1)} \tag{17}$$

Then we have the following equation whose coefficients and solution are integers.

$$r_A^{(n)}\ \alpha_K + r_B^{(n)}\ \beta_K = y_K \tag{18}$$

In general, this is an indeterminate equation and can be solved easily using extended Euclidean algorithm.

(5) Generate-and-test of Rule Candidates. Now we have candidates of $(\alpha_A, \alpha_B, \beta_A, \beta_B, \alpha_K, \beta_K)$. Since given string Y always includes the right sides of rules A and B as substrings, we exhaustively extract from Y the following two substrings:

(a) a substring having α_A A's, α_B B's, and α_K K's to form rule A candidate,
(b) a substring having β_A A's, β_B B's, and β_K K's to form rule B candidate.

Then for each combination of rules A and B candidates, we rewrite the axiom n times using the candidates, and check whether the obtained string is equal to Y. If the equality holds, a pair of the candidates is a solution.

5 EXPERIMENTS

We evaluate the proposed LGIN1 and LGIN2 using plants models. The experiments were performed using a PC with 3.0 GHz CPU and 2MB main memory.

5.1 Experiments using LGIN1

Plants model ex01p, ex02p and ex03p are drawn from (Prusinkiewicz and Lindenmayer, 1990), and ex01y, ex02y and ex03y are their corresponding variations with fewer n.

$$\begin{array}{ll} \text{(ex01p)} & n = 5, \ \ \text{axiom} : F \\ & \text{rule} : F \rightarrow F[+F]F[-F]F \\ \text{(ex01y)} & n = 4, \ \ \text{axiom} : F \\ & \text{rule} : F \rightarrow F[+F]F[-F]F \end{array}$$

Shown below is string Y for ex01y whose length is 1,561.

$F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]$
$F[+F]F[-F]F[+F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[$
$-F[+F]F[-F]F]F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]$
$F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F[$
$+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F]$
$F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]$
$F[+F]F[-F]F[+F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[$
$-F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F[+F[+F]F[-F]F$
$]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F]F[+F]F[-F]F[$
$+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[$
$-F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F$
$]F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[$
$-F[+F]F[-F]F]F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]$
$F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F[$
$+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F]$
$F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]$
$F[+F]F[-F]F[-F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[$
$-F[+F]F[-F]F]F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]$
$F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F[$
$+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[$
$+F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F$
$]F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[$
$-F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F[+F[+F]F[-F]F$
$]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F]F[+F]F[-F]F[$
$+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F]$
$F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]$
$F[+F]F[-F]F[+F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[$
$-F[+F]F[-F]F]F[+F]F[-F]F]F[+F]F[-F]F[+F[+F]F[-F]F]$
$F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F[$
$+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]F[+F]F[-F]F]$
$F[+F]F[-F]F[+F[+F]F[-F]F]F[+F]F[-F]F[-F[+F]F[-F]F]$
$F[+F]F[-F]F$

Figures 1 to 6 show plants graphics for these six D0L-systems.

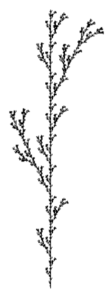

Figure 1: Model ex01p.

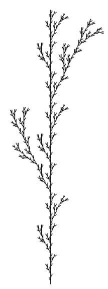

Figure 2: Model ex01y.

$$
\begin{aligned}
&\text{(ex02p)} && n=5,\ \ \text{axiom}: F\\
& && \text{rule}: F \to F[+F]F[-F][F]\\
&\text{(ex02y)} && n=4,\ \ \text{axiom}: F\\
& && \text{rule}: F \to F[+F]F[-F][F]
\end{aligned}
$$

$$
\begin{gathered}
\text{(ex03p)}\ \ n=4,\ \ \text{axiom}: F\\
\text{rule}: F \to FF-[-F+F+F]+[+F-F-F]\\
\text{(ex03y)}\ \ n=3,\ \ \text{axiom}: F\\
\text{rule}: F \to FF-[-F+F+F]+[+F-F-F]
\end{gathered}
$$

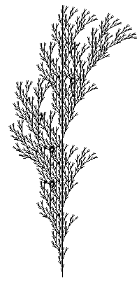

Figure 3: Model ex02p.

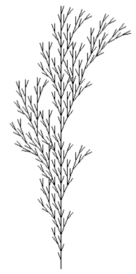

Figure 4: Model ex02y.

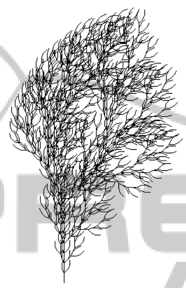

Figure 5: Model ex03p.

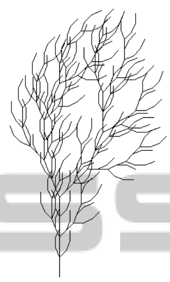

Figure 6: Model ex03y.

For these six D0L-systems LGIN1 successfully found the original grammars. When $n=4$, LGIN1 found a grammar with $n=2$ as another solution. On the other hand, when n is a prime number, LGIN1 found the original grammar as a unique solution.

Table 1: CPU time of LGIN1.

model	n	string length	CPU time (sec)
ex01p	5	7,811	0.093
ex01y	4	1,561	0.063
ex02p	5	9,373	0.082
ex02y	4	1,873	0.158
ex03p	4	11,116	0.776
ex03y	3	1,388	0.051

The CPU time required by LGIN1 is shown in Table 1. LGIN1 finished its processing within one second for each example. When we increase the number of rewritings with a production rule fixed, the string length naturally gets much larger, but the processing time does not always increase, for example, see ex02. This happened probably because $n=4$ has a factor of 2, requiring additional search, while $n=5$ has no factor other than 1.

5.2 Experiments using LGIN2

Plants model ex04p, ex05p and ex06p are drawn from (Prusinkiewicz and Lindenmayer, 1990), and ex04y, ex05y and ex06y are their corresponding variations with fewer n.

(ex04p) $n = 7$, axiom : X
rule : $X \rightarrow F[+X]F[-X]+X$
rule : $F \rightarrow FF$

(ex04y) $n = 5$, axiom : X
rule : $X \rightarrow F[+X]F[-X]+X$
rule : $F \rightarrow FF$

Shown below is string Y for ex04y whose length is 1,512.

$FFFFFFFFFFFFFFFF[+FFFFFFFF[+FFFF[+FF[+F[+X]F[-X]+X$
$]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+X]FFFF[-FF[+F[+X]F[-$
$X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+X]+FF[+F[+X]F[-X$
$]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+X]FFFFFFFF[-FFFF[$
$+FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+X]F$
$FFF[-FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]$
$+X]+FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+$
$X]+FFFF[+FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F$
$[-X]+X]FFFF[-FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[$
$+X]F[-X]+X]+FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+$
$X]F[-X]+X]FFFFFFFFFFFFFFFF[-FFFFFFFF[+FFFF[+FF[+F[$
$+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+X]FFFF[-FF$
$[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+X]+FF[$
$+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+X]FFFFF$
$FFF[-FFFF[+FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X$
$]F[-X]+X]FFFF[-FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+$
$F[+X]F[-X]+X]+FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F$
$[+X]F[-X]+X]+FFFF[+FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]$
$+X]+F[+X]F[-X]+X]FFFF[-FF[+F[+X]F[-X]+X]FF[-F[+X]F$
$[-X]+X]+F[+X]F[-X]+X]+FF[+F[+X]F[-X]+X]FF[-F[+X]F[$
$-X]+X]+F[+X]F[-X]+X]+FFFFFFFF[+FFFF[+FF[+F[+X]F[-X$
$]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+X]FFFF[-FF[+F[+X]$
$F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+X]+FF[+F[+X]F$
$[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+X]FFFFFFFF[-FF$
$FF[+FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-X]+$
$X]FFFF[-FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[$
$-X]+X]+FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+X]F[-$
$X]+X]+FFFF[+FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+F[+$
$X]F[-X]+X]FFFF[-FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]$
$+F[+X]F[-X]+X]+FF[+F[+X]F[-X]+X]FF[-F[+X]F[-X]+X]+$
$F[+X]F[-X]+X$

Figures 7 to 12 show plants graphics for these six D0L-systems.

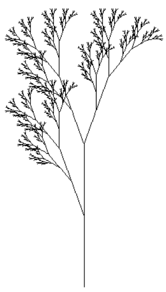

Figure 7: Model ex04p.

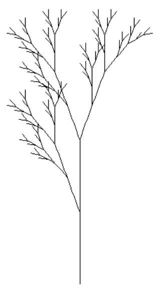

Figure 8: Model ex04y.

(ex05p) $n = 7$, axiom : X
rule : $X \rightarrow F[+X][-X]FX$
rule : $F \rightarrow FF$

(ex05y) $n = 5$, axiom : X
rule : $X \rightarrow F[+X][-X]FX$
rule : $F \rightarrow FF$

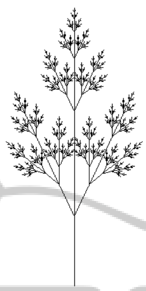

Figure 9: Model ex05p.

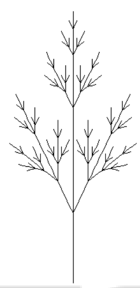

Figure 10: Model ex05y.

(ex06p) $n = 5$, axiom : X
rule : $X \rightarrow F-[[X]+X]+F[+FX]-X$
rule : $F \rightarrow FF$

(ex06y) $n = 4$, axiom : X
rule : $X \rightarrow F-[[X]+X]+F[+FX]-X$
rule : $F \rightarrow FF$

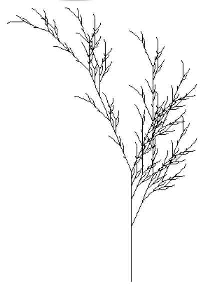

Figure 11: Model ex06p.

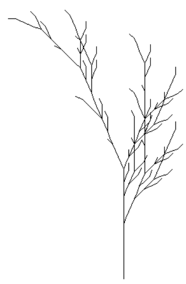

Figure 12: Model ex06y.

For these six D0L-systems having two production rules LGIN2 found exactly the same original grammars as unique solutions.

The CPU time required by LGIN2 is shown in Table 2. LGIN2 finished each task within seconds. When we increase the number of rewritings with production rules fixed, the processing time does not always increase, for example, see ex06. This happened partially because $n = 4$ has larger search space than $n = 5$; that is, $y_F = 360$ in $n = 4$ has 22 divisors (excluding 1 and 360) while $y_F = 1488$ in $n = 5$ has 18 divisors (excluding 1 and 1488).

Table 2: CPU time of LGIN2.

model	n	string length	CPU time (sec)
ex04p	7	13,956	0.680
ex04y	5	1,512	0.132
ex05p	7	12,863	0.667
ex05y	5	1,391	0.126
ex06p	5	6,263	1.440
ex06y	4	1,551	4.228

6 CONCLUSIONS

This paper proposed two methods for grammatical induction of D0L-systems having one or two production rules and simple axioms. Basic equations for the methods are derived and utilized to narrow down the parameter value ranges. In our experiments using plants models, the methods found the original grammars very efficiently. In the future we plan to extend our induction methods for wider class of L-systems.

ACKNOWLEDGEMENTS

This work was supported by Grants-in-Aid for Scientific Research (C) 22500212 and Chubu University Grant 22IS27A.

REFERENCES

Damasevicius, R. (2010). Structural analysis of regulatory DNA sequences using grammar inference and support vector machine. *Neurocomputing*, 73:633–638.

McCormack, J. (1993). Interactive evolution of L-system grammars for computer graphics modelling. *Complex Systems: From Biology to Computation*, ISO Press, Amsterdam, 118–130.

Nevill-Manning, C. G. (1996). Inferring sequential structure. Ph.D. thesis, Dept. of Computer Science, Univ. of Waikato, New Zealang.

Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer-Verlag, New York.

Schlecht, J., Barnard, K., Springgs, E., and Pryor, B. (2007). Inferring grammar-based structure models from 3d microscopy data. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*. 1–8.