A survey on L-system inference

FARAH BEN-NAOUM

EEDIS, UDL, Computer Science Department
Sidi Bel Abbes
P.O. 89, 22000 Algeria
ben_naoumfr@yahoo.fr

Abstract. Many algorithms of grammatical inference were developed for several types of grammars. The grammatical inference problem consists of finding, from a set of strings, a grammar that produces all the strings of this set [3]. We are interested here by the inference of particular grammars, noted L-systems, which are parallel rewriting systems most famously used to model the growth processes of plant development. We present a survey on methods of L-system inference proposed since the creation of this rewriting system by Aristid Lindenmayer in 1968 [15]. The grammatical inference of L-systems has been studied over the past 30 years, and that in relation to several areas of application of produced L-systems. We are interested in looking at this problem from the point of view of a possible use of these methods for an application in biological modeling, particularly in the modeling of plants. In order to provide a better understanding of the research challenges of L-system inference, this article presents a detailed investigation of current state-of-the-art algorithms in L-system inference, with an analysis highlighting their positive and negative points. Open research issues are also discussed, with an objective to spark new research interests in this field.

Keywords: L-system, grammatical inference, tree structure, positive sample.

(Received December 19, 2008 / Accepted May 14, 2009)

1 Introduction

1.1 Grammatical inference

1.1.1 Definition. The problem of inducing, learning or inferring grammars has been studied for some time, as a practical problem of attempting to represent some knowledge about strings or trees through a typical representation of sets of trees and strings, such as an automaton or a grammar [3].

The *grammatical inference* is a specific instance of inductive learning which may be formulated as the discovery of structures from examples that are supposed to have been generated by the same process. In this particular case, all examples known as *positive sample* usually consists of a set of defined strings on a specific alphabet. A negative sample, ie a set of counter-examples of the target language, can also help the process of induction.

In a general way, grammatical inference consists of finding the grammar or automaton for a language of which we are given an indirect presentation through strings, sequences and words, trees, terms and structures, or graphs.

1.1.2 Parameters of learning process. de la Higuera proposes in [10] arguments to measure the quality of the inferred grammar, which should also somehow give a measure of the quality of the learning process:

- Definition of the target[1]: the hardness of the learning task can depend on the complexity of the target, and questions linked with teaching (is this particular target teachable?).

- To choose identification or approximation as convergence criteria: Now we have a target, and we are given some data to work from. So the question is: How

[1] The *target* of an inference process correspond to the type or the class of the sought grammar.

are we going to measure success? A first school of thought defines success only when the actual target is found. This leads to the elegant concept of *identification in the limit* (named Gold identification [8]). A second school of thought consists of saying that we could accept to be a little wrong. Again an elegant setting, called *Probably Approximately Correct learning* has been introduced by Valiant [22] to deal with this case.

- To choose a particular presentation[2] of a language with grammar (or an equivalent way): As languages can be infinite, there is no way we can manipulate languages directly: We can only use the grammars.

1.1.3 Difficulties in the learning grammars process. One thing is to build algorithms, another is to be able to state that these work. For this fact, difficulties in learning grammars process are invoked in [10] as questions about validation and comparison of learning applications. The main questions are: (1) Does this algorithm work? (2) Do we have enough learning data? Can we put a higher bound on the number of data needed to have some guarantee? Or some lower bound on the quantity of data without which learning is impossible? (3) Do we need some extra bias? If what we are looking for is just going to be impossible to find, is there someway we can artificially (or not) reduce the search space in order to perhaps have more hopes to find something? (4) Is this algorithm better than the other? and finally (5) Is this problem easier than the other?

de la Higuera proposes also alternatives as to how to answer these questions among them: (1) Use well admitted benchmarks: Algorithms could compare one against the other and hopefully allow engineers to choose the better algorithm for a given task, (2) Build your own benchmarks, (3) Solve a real problem: In all cases, solving a new unsolved problem on true data is meaningful, (4) Mathematical proofs of convergence.

1.2 Grammatical inference of L-systems

1.2.1 L-systems. L-systems or *Lindenmayer systems* were introduced and developed in 1968 by the biologist and botanist Aristid Lindenmayer [15]. An L-system is a parallel rewriting system most famously used to model the growth processes of plant development, but also able to model the morphology of a variety of non-filamentous organisms. They can also be used to generate self-similar fractals. Originally L-systems were devised to provide a formal description of the development of simple filamentous multicellular organisms, and to illustrate the neighborhood relationships between plant cells. Later on, they were extended to describe higher plants and complex branching structures.

[2] A way to formally define a language.

1.2.2 Basic definitions. We shall present the notion of a L-system such as outlined by Lindenmayer in [15]. We denote an *L-system* G by the triple (Σ, P, w_0), where Σ is the alphabet, $P = \{A \rightarrow w / \ A \in \Sigma,\ w \in \Sigma^*\}$ the set of production rules , and $w_0 \in \Sigma^+$ the axiom. The major difference of L-systems with Chomsky's grammars is the parallel derivation of all symbols of a string at the same step, this corresponds to the development of biological organisms, also all symbols can produce. In a *bracketed L-system* the two symbols '[' and ']' are considered in Σ, they are used to signal the beginning and the end of a sub-branching structure in the turtle geometry [1]. An L-system is called *deterministic* if and only if there is at most one production in P for each symbol of Σ, and is called *propagating* if and only if no rule of P does contain a right part equal to the empty string, so in a propagating L-system cells cannot simply disappear. The abbreviation DPxL-system stands for *deterministic propagating* x-sided L-system, with x$\in \{0, 1, 2\}$. x indicates the type of interaction between neighboring subunits. If $x = 2$ stand for two-sided interactions where each rule of P is in the form $a < A > b \rightarrow w$ with a and b are respectively the left and right context of the symbol A. $x = 1$ stands for one-sided rules of the form $A > b \rightarrow w$ or $a < A \rightarrow w$, and $x = 0$ for no interaction *(Information-less* L-system with rules of the form $A \rightarrow w$). From an L-system $G = (\Sigma, w, P)$, an *L-scheme* can be defined as $S = (\Sigma, P)$ [20].

1.2.3 L-system inference. There are many open problems involving studies of L-systems. Among them, the inference problem formulated as: Given a structure, find an L-system that can produce that structure. The grammatical inference of L-systems has been studied over the past 30 years, and that in relation to several areas of application. We are interested in looking at this problem from the point of view of a possible use of these methods for an application in biological modeling, particularly branching structures. For this fact we only invoke the inference from only positive sample, because we can not talk about negative sample for biological structures (many justifications are given in [6]).

Several studies have been conducted in this axis, which come as response to some problems arising in learning [10]. Some of them give an answer on the learnability of several targets, and this from different inputs. Others offer solutions of L-system inference process, on the basis of certain choices, namely:

- limit the target to a special class or to a sub-class of

  L-systems,

- limit the application area of the method,

- using particular presentation of the language,
- take special assumptions on the initial data,
- choose particular convergence criteria,
- translate the problem in another search space,
- use a completely heuristic approach,
- define the inference as a resolution of a special problem in an other field, its properties help to guide the process of inference.

We will review in this article different approaches for solving the problem of L-systems inference. Each of them is based on one or more choices from those cited above. We will also give a number of observations and conclusions on all of them by highlighting their positive and negative points. Finally, we will propose solutions to improve capacities of L-system inference process.

2 Study of decidability in L-system inference problem

A number of works have been reported by Herman and Walker [9] and by Feliciangeli and Herman [6], investigating the syntactic inference problem for variety of L-language families. They give an answer on the learnability of several families, and this from different inputs. For this, they deal with the following problem.

ProblemXx: *Give a procedure which, for any finite set φ of sequences of strings, decides whether or not there exists a deterministic or non-deterministic and propagating or non-propagating xL-system $\underline{L}$ such that φ has property X with respect to $\underline{L}$ ($X \in \{A, B, C\}$[3]), and produces such an $\underline{L}$, if there is one.*

This problem means that the hypothesis space itself is dependent on the input data. We will use this notation throughout this article. An answer to 30 from 36 instances of the $problemXx$ is provided in [6], the other six are still open, Table1 summarizes this results. In each of the 36 instances, Feliciangeli and Herman assume that the input is a *finite set of finite sequences of strings* of the form $\varphi = \{\langle p_{1,1}, p_{1,2}, ..., p_{1,n_1}\rangle$, $\langle p_{2,1}, p_{2,2}, ..., p_{2,n_2}\rangle$,..., $\langle p_{m,1}, p_{m,2}, ..., p_{m,n_m}\rangle\}$, where $p_{i,j}$ denote the j^{th} string in the i^{th} sequence of φ, and $a_{i,j,k}$ denote the k^{th} symbol in the string $p_{i,j}$. The inference procedures given in [6] are guided by the proofs (which are constructive) of the decidability of existence of an inference algorithm for each type of sequences. To indicate the type of methods used, they shall give a solution to the DPB2 problem whose main idea is:

[3] *X represent information about the time intervals between strings of a sequence φ as follows, 'A': all intervals equal to 1, 'B': all intervals equal but of unknown length, 'C': intervals of arbitrary lengths.*

Type of L-system		Deterministic		Nondeterministic	
		Propagating	Nonpropagating	Propagating	Nonpropagating
(x=0)	A	Dec	Dec	Dec	Dec
	B	Dec	Dec	Dec	Dec
	C	Open	Open	Dec	Dec
(x=1)	A	Dec	Dec	Dec	Dec
	B	Open	Open	Dec	Dec
	C	Open	Open	Dec	Dec
(x=2)	A	Dec	Dec	Dec	Dec
	B	Dec	Dec	Dec	Dec
	C	Dec	Dec	Dec	Dec

Dec: decidable case, Open: open problem

Table 1. Summary of results of Feliciangeli and Herman on L-system inference algorithms decidability

1. If the input sequence φ verify the properties **(a)** and **(b)**, then step(2) constructs a correspondent DP2L-system $L = \langle \Sigma, \delta, g\rangle$, else there is no DP2L-system of the required kind: **(a)** the size of all strings of φ must increase from left to right (L is propagating), **(b)** there is no string existent in two different sequences in φ and having different successors in each of them (L is deterministic).

2. Let l be the length of the longest string in φ. Then φ is an l-regular subprocess associated with L (intervals equal to l). Let Σ_0 consist of all symbols in φ and an additional symbol g. The alphabet Σ of L consists of all symbols in Σ_0, and symbols representing all strings of symbols from Σ_0 of odd length between 3 and $2l$-1. The symbol which represents the string p will be denoted by $[p]$. δ is defined in such a way that, starting from any $p_{i,j}$, after $l-1$ steps each cell will be in a state which indicates to that cell what the whole of $p_{i,j}$ is, and how many cells from the end that particular cell is. Then each cell can change into the state of the cell in the corresponding position in $p_{i,j+1}$, with the last cell dividing into many cells if $|p_{i,j+1}| > |p_{i,j}|$. This is done in the way described in Algorithm 1.

Analysis. Herman and Walker next give an evaluation of the resulting solutions. They highlight the very fact that these algorithms produced an L-system which does the job, but which is more complicated than another model which is also appropriated. They also introduce some concepts that help to choose the best solution from those given by Feliciangeli and Herman's algorithms. One of these concepts consists on identification in the limit or to complexity measures to select the 'best' system consistent with the finite data.

From our point of view, in addition to the fact that they produce a no minimized grammar[9], the work of Feliciangeli and Herman focused on "the syntactic Inference problem applied to biological systems", these biological systems concern only filamentous structures

Algorithm 1 A solution of the DPB2 problem as proposed by Feliciangeli and Herman.

Input: - A sequence $\varphi = \{\langle p_{1,1}, p_{1,2}, \ldots, p_{1,n_1}\rangle, \langle p_{2,1}, p_{2,2}, \ldots, p_{2,n_2}\rangle, \ldots, \langle p_{m,1}, p_{m,2}, \ldots, p_{m,n_m}\rangle\}$.

Output: - Is φ a PD2L-sequence or no? If yes give a corresponding PD2L-system $\underline{L} = \langle \Sigma, g, \delta \rangle$.

Method:

If the size of all strings of φ increase from left to right and there is no string existent in two different sequences in φ having different successors in each of them, **then**

-**For all** symbols a, b, c and d in Σ_0, **for all** strings p of symbols from Σ_0 of odd length between 1 and $2l-5$, **for** $1 \le i \le m, 0 \le j < n_i, 1 \le k \le |p_{i,j}|$, and **for all** $x, z \in \Sigma$ **do**:

$\delta(a,b,c) = \{[abc]\}$,

$\delta([abp],[bpc],[pcd]) = \{[abpcd]\}$,

$\delta(g,[bpc],[pcd]) = \{[gbpcd]\}$,

$\delta([abp],[bpc],g) = \{[abpcg]\}$,

$\delta(g,[bpc],g) = \{[gbpcg]\}$,

$\delta(x, \left[g^{l-k} p_{i,j} g^{l+k-1-|p_{i,j}|}\right], z) =$

$$\begin{cases} \{a_{i,j+1,k}\}, & if\ k < |p_{i,j}|\ and\ k \le |p_{i,j+1}|, \\ \left\{a_{i,j+1,|p_{i,j}|} a_{i,j+1,|p_{i,j}|+1} \ldots a_{i,j+1,|p_{i,j+1}|}\right\}, & if\ k = |p_{i,j}|\ and\ k \le |p_{i,j+1}|, \\ \{\varepsilon\}, & if\ k > |p_{i,j+1}|. \end{cases}$$

- **For all** other symbols x, y and z in Σ: $\delta(x,y,z) = \{y\}$.

- Return L.

Else Return $\emptyset$.

that can be described by deterministic or non-deterministic and propagating or non-propagating xL-systems with $x \in \{0,1,2\}$, the positive sample set does not include more complex biological systems like tree structures that can be, at least, described by bracketed L-systems.

3 L-system inference based on changing search space

3.1 Algebraic resolution of D0L-system inference

A particular method of the syntactic inference problem of D0L-sequences is proposed, by Doucet in [5], by presenting algorithms based on an algebraic approach rather a combinatorial[4] one. The initial information may have various special forms: the words are given as a sequence which may be either consecutive (illustrated in the algorithm1) or scattered (described by algorithm2), and the rank order numbers of each string (its order in the language) are given as well as the alphabet. Essentially, the goal of this method is to find the development within strings of a language L from those existent in the sequence s, the problem is then translated in the *algebraic search space*. For this, each string is replaced by its *Parikh vector*[5]. The search of the linear dependence within Parikh vectors means that we look for a growth function for each symbol of the given alphabet, in a global way it's represented by the growth matrix (noted $A = [A_{i,j}]_{i,j \in \{1,..,k\}}$ in algorithm2). Each $A_{i,j}$ correspond to the number of occurrences of the symbol $a_i \in \Sigma$ in the right part α of the rule corresponding to the character a_j (i.e. in the rule $a_j \to \alpha$). Starting from w_0, all production rules will be then deduced from this matrix from the fact that we know time intervals separating all strings of s, it's sufficient to make the correspondence, from left to right, between symbols of each string and substrings of its successor in s.

To conclude, these algorithms make use of the number of letters present in each given word, and are able to discard the vast majority of combinations at an early stage.

Algorithm 2 Algebraic algorithm of D0L-system inference from a consecutive sequence

Input: - an alphabet $\Sigma = \{a_1, \ldots, a_k\}$ of size k,
- a sequence of $k+1$ words $s = w_0, \ldots, w_k$ (s is called initial consecutive subsequence).

Output: all D0L-systems G for which s is the initial subsequence of $\mathcal{L}(G)$.

Method:

- Form the parikh-images of s: $\bar{s} = \bar{w}_0, \ldots, \bar{w}_k$;

- Let $A \in \mathbb{N}^{k,k}$ be the growth matrix of G, with

$$A \begin{pmatrix} \bar{w}_0 & \cdots & \bar{w}_{k-1} \\ \vdots & & \vdots \end{pmatrix} = \begin{pmatrix} \bar{w}_1 & \cdots & \bar{w}_k \\ \vdots & & \vdots \end{pmatrix} \text{ noted } AS = T;$$

- **For all** matrixes A obtained by solving the equation $A = TS^{-1}$:
  - Determine the useful production rules P uniquely from A and s, and finally return P.

In algorithms 2 and 3 there is multiple solutions for A, $\chi(x)$, and E_0, thus we must compute the solution of P for all the cases, which leads to find all D0L-systems associated to the initial sequence.

Analysis. Our study of this case of D0L-systems inference has identified several remarks and conclusions. First we say that their essential strong points can be summarized as follow: Doucet treats the cases of inferring D0L-systems from A and C L-sequences, which can be referred as problems DA0 and DC0. The problem DC0 is one of the 6 such problems for L-systems posed in [6] which are still open. To solve it, Doucet

[4] A combinatorial method seeks for a solution by testing all possible combinations on input data in the aim to discover the needed result.

[5] For an alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$, the *parikh-vector* $\bar{w}$ assigned to a word w is defined as a vector in $\mathbb{N}^k$ with its i^{th} coordinate equal to the number of occurrences of σ_i in w.

gives additional information consistent on the exact rank of each string in the sequence, as well as the alphabet. The inference problem is known to be decidable as soon as the alphabet is given. Also the matrix representation of the problem leads to a simple and effective algebraic resolution, which allows saying whether a given sequence corresponds to a D0L-sequence or not. If so, it finds all possible D0L-systems associated with him.

Algorithm 3 Algebraic algorithm of D0L-system inference from a scattered sequence

Input:
- an alphabet Σ of size k,
- a sequence of words $s=w_{i_1},\ldots,w_{i_p}$, with the ranges $i_0<i_1<\cdots<i_p$ and p arbitrary (s is a scattered subsequence),

Output: - Is s part of a D0L-sequence or no? If yes give all possible G's from s.

Method:
- Form the parikh-images of s: $\bar{s}=\bar{w}_{i_1},\ldots,\bar{w}_{i_p}$;
- Find a linear dependence relation (recurrence relations) within $\bar{s}$, and find its associated polynomial $\psi(x)$;

If the linear dependence relation with integer parameters exists
  then continue /*we have a D0L-sequence*/
  else break; /*it is not a D0L-sequence and in this case there is no solution given by this method*/

For all monic divisors $\chi(x)$ of $\psi(x)$ (with integer coefficients and degree $m \leq k$) **do**
  - Compute the coefficient matrix C from $\chi(x)$ and the index set of s;
  For all $E_0 \in \mathbb{N}^{k,m}$ satisfying the matrix equation $S = E_0 C$ **do**
    - Determine $\bar{w}_m$ from $\bar{w}_0,\bar{w}_1\ldots,\bar{w}_{m-1}$ as found in E_0 and from $\chi(x)$'s associated recurrence relation;
    For all growth matrixes $A \in \mathbb{N}^{k,k}$ satisfying the matrix equation $E_1 = AE_0$ (E_1 is composed from E_0 and $\bar{w}_m$) **do**
      - Find the set of production rules P from A and the words of s, and return P.

In addition, this method presents some limitations, namely: (1) Many restrictions on the initial data: The alphabet and enough words must be given, as well the words rank order numbers. In addition, the number of words depends on the size of the alphabet. (2) Also this method works only if sufficient words are given to establish a linear dependence relation between their Parikh-vectors, even then, a certain amount of trial-and-error work is necessary. (3) In absence of a recurrence relation within the given sequence the method simply does not work. In addition, the recurrence relation is found intuitively. The most important questions are: Is there any algorithm for the detection of this recurrent relation? And is there any algorithm which confirms the inexistence of any recurrence relation within the sequence? (4) Complexity of computations. Indeed, the more the size of the alphabet or the size of the initial sequence is large, the more complex calculation becomes: handle large matrices and find recurrent relations becomes difficult.

From the biological point of view: L-systems are currently fully dedicated to modeling of plants by introduction of the graphical interpretation of their symbols (the turtle graphical defined in [1]). The simplest form that can be modeled by D0L-systems corresponds to tree structures easily represented by the brackets [and], in which a branch is modeled by a well formed string, and every production rule associate a symbol to a well formed string. This method of D0L-system inference does not take into account this aspect during the decomposition of each word of the sequence, and then the production rules may contain right parts corresponding to no well formed strings. In addition, this inference case does not meet the biological motivations of L-systems inference. Indeed, it is difficult to provide, from observations of the biologist; rank associated with each stage of development of a tree in addition to his precise description.

3.2 PD0L-Inference from developmental sequences (tree structures)

A rigorous algorithmic procedure for finding deterministic interaction-less models from developmental sequences (tree structures) are formulated by Jurgensen and Lindenmayer in [12], they infer bracketed PD0L-systems from BL-sequences, which correspond to the PDB0 problem. These models represent control mechanisms based on cell lineages[6]. They try to model the practical inference procedure as used by biologist. The basic biological assumption made is that: no interactions take place in the course of development, the biological process underlying the observations is deterministic, and that sufficiently frequent observations are available at equal time intervals. They were not primarily concerned with the start configurations but rather with potential derivation, for this reason they mainly work with 0L-schemes.

This method tries to find derivation rules from the behavior of each apex in time, which correspond to the *descendant relations* between each apex and its descendant part in the next tree of the sequence. For this it tries to associate each apex of a tree to its descendent part in the next tree of the sequence, this association, named descendant relation, will then create a production rule, of course in this case it resonates directly on a codified format of all strings representing the trees of the sequence (by assigning a unique code to each apex). This method regroups all descendent relations

[6] Differential gene expression resulting from equal or unequal cell division processes in development.

in graphs called *descendant trees*. The construction of each graph begins from the code of each apex in w_1 (the first string of the sequence $\{w_1, w_2, ...\}$), which will be related to the codes of its descendant part in the second string w_2, the same principle can be applied to apexes of w_2 with parts of w_3, and so on. Then, there will be as many graphs as there are apexes in w_1. To reduce the number of the obtained rules, the method performs the unification of the symbols associated with *isomorphic sub-graphs* [7], which correspond to identical sub-trees developmental model.

The main idea of this algorithm can be formally summarized as follows:

Algorithm 4 PD0L-system inference from a développemental sequence

Input: a P0L *based observation* Ω which is the pair of: a P0L-scheme G with special requirements on the basis of biological background. And a pair of finite sequences $W = \left((w_i)_{i=1,...,n}, (t_i)_{i=1,...,n-1}\right)$.

Output: PD0L-scheme representing W.

Method:

- Take the set of derivations D_Ω constructed from G and W;
- Define a PD0L *descendant system* $\mathfrak{D}_\circ = (\Sigma, \Theta, \Phi, \mathfrak{F}, \theta)$ consistent with $d \in D_\Omega$ by renaming the apexes in each string x_j of d with indexed symbols a_j^i $(1 \leq i \leq len(x_j))$ (the definition of the initial coding $\Phi : \Theta \to \Sigma$); and for each apex a_j^i we determine the tree T_j^i of its descendants (the set $\mathfrak{F}$), T_j^i will exhibit the lineages with a_j^i as the origin;

Repeat

- Find the isomorphic subtrees over all T_j^i (over $\mathfrak{F}$);
- Encode the isomorphic subtrees with the same symbol (modify the PD0L descendant system according to this new isomorphism found);

until stabilization of the set of descendant trees (no other isomorphism can be found from the actual PD0L descendant system);

- From the obtained trees of descendants we read off the resulting CPD0L-scheme;
- Ignore the coding in the aim to find the underlying PD0L-scheme, and return the resulting one.

1. Take a P0L *based observation* Ω which is a pair of:

   (a) a P0L-scheme G with special requirements on the basis of biological background, these requirements limits the kind of the rules permitted, for example the choice of sub-apical branching structures. G serves as the *developmental pattern* for this inference procedure.

   (b) a pair of finite sequences

   $$W = \left((w_i)_{i=1,...,n}, (t_i)_{i=1,...,n-1}\right)$$

   consists of the observed structures w_i and the time intervals t_i; one assume that w_1 has been observed at time 1, w_2 at time $1 + t_1$, w_3 at time $1 + t_1 + t_2$, etc.

2. Construct the set D_Ω of all derivations with respect to G which start with w_1 derive w_2, w_3,... with the number of steps t_1, t_2,..., than this consist on finding all possible intermediary strings between two w_i and w_{i+1} if $t_i > 1$ consistent with G.

3. For a P0L-based observation Ω and $d \in D_\Omega$, let $\mathcal{D}(d)$ be the class of all PD0L *descendant systems*[8] consistent[9] with d. A PD0L *descendant system* consists on the codification of all apexes of initial trees, as well as the set of all descendant trees constructed from d. It is the goal of this method (as described in Algorithm 4) to determine a PD0L descendant system within $\mathcal{D}(d)$ which is particularly well structured and small. These can be performed recursively by unification of codes associated to *isomorphic* descendant sub-trees. Then the inference problem is solved here in the *PD0L descendant systems search space*, from which we extract the *CP0L-scheme*[10] and then the final PD0L-scheme.

Analysis. There have been several remarks made in [12] on the performances of this algorithm: Whereas an inference procedure for such systems can always be carried out, not all solutions found are biologically acceptable. A mathematically precise and algorithmically useful description of such a procedure needed to be formulated. In addition, this algorithms used 0L-forms to describe the search space and thus to guide the algorithm. In this way they have obtained a satisfactory solution for D0L-inference problem with observations corresponding to consecutive derivation steps. The case of non-consecutive observations needs further clarification as to which criteria should govern the selection of

[7] *Isomorphic graphs* are identical except for the labels of their nodes.

[8] a P0L *descendant system* is a 5-uple $\mathfrak{D} = (\Sigma, \Theta, \Phi, \mathfrak{F}, \theta)$, where: Σ, Θ are respectively the alphabet and the code alphabet; $\Phi : \Theta \to \Sigma$ is the coding; $\mathfrak{F}$ the trees of descendants (trees with labels in Θ); $\theta : \Theta \to 2^{\mathfrak{F}}$ with $\theta(x)$ is the set of all the trees of $\mathfrak{F}$ having x as their root label.

[9] The *consistence* expressed the connection between derivations and descendant systems in a precise way in which there is a special mapping between the strings of the derivation and the coding of the PD0L descendant system.

[10] A C0L-system is a quintuple $H = (\Delta, \Sigma, \mathcal{Q}, w, P)$, where Σ, w, P are parameters like for a 0L-system, and Δ is an alphabet and $\mathcal{Q}$ is a coding from Σ to Δ.

a derivation. Also, the exact computational complexity of the PD0L-inference algorithm is still open problem, as well as the existence of less costly solutions. The question also remains whether there are other, essentially different ways to implement the inference procedure. Finally, the inference algorithm presented does not necessarily provide an acceptable PD0L-scheme for a given sequence of observations, if for instance the number of cell states is exceedingly large.

And from our point of view: The input sequence must contain sufficiently strings representing informations about the growth process, with sufficiently frequent observations available at equal time intervals. But a small heights (periods of observations) are quite unrealistic as reliable basis of inferences for practical applications. And the resulting grammar is not minimized especially the size of the resulting alphabet. Also the search of the isomorphic descendant trees are a tedious task, especially when the size of the set of descendant sub-trees and the size of its elements are large.

4 Identification in the limit of PD0L-systems

The inductive inference problem, adopted by Yokomori in [23], perform the inference procedure for PD0L-systems by considering a special presentation of the language; from particular initial data (test set), with "identification in the limit [8]" as convergence criteria. For this, they first show that the family of PD0L-languages is identifiable in the limit from positive data, and they also show that this is not the case of the family of 0L-languages. Further, it is shown that each PD0L-language can be represented as a disjoint union of a finite set and several disjoint PD0L-languages ($L(G) = F_G \bigcup_{j=0}^{m_G-1} L(G_j)$).

Algorithm 5 Identifying the family of PD0L languages

Input: a positive presentation of U (U is an unknown PD0L-language over a fixed alphabet Σ)
Output: a sequence of PD0L-systems G_T
Method:
Initialize $T = \emptyset$;
Repeat (forever)
  read the next positive example w;
  let $T := T \cup \{w\}$;
  call Procedure(T;G_T);
  If Procedure(T;G_T) succeeds **then** output G_T
    else output $G_T = (\Sigma, Id, w)$; (where Id: identity over Σ)

Intuitively, since L is infinite and is ordered by the relation $\preceq$ [11] (L is propagating), then there is necessarily a sequence of m_G alphabets corresponding to different strings of L where $\forall x_i \in L(G) - F_G$, and $\forall k > 0$, $alphabet(x_i) = alphabet(x_{i+k.m_G})$. Also, their can exist in L an initial finite sequence of elements with irregular alphabets appearance, this set is obviously F_G. We say that it's the initial sequence because of the axiom necessarily exist there. We can exploit this regularity to find general rules of growth of L. Just find the homomorphism between elements of F_G, and after find the homomorphism between the first elements of the m_G languages $L(G_j)$, i.e. between elements of the set $\{first(L(G_0)), first(L(G_1)), ..., first(L(G_{m_G-1}))\}$ where $first(L(G_j))$ is the first element of $L(G_j)$ which are ordered by the relation $\preceq$. This can be clearly illustrated in Fig. 1, in which all $x_{j,i}$ in $L(G_j)$ have the same alphabet.

Using the same reasoning, Nielsen describes in [19] the way to find from a PD0L-system the corresponding

Algorithm 6 Computing a PD0L-system from a test set T.

Procedure(T;G_T):
(A) Decomposition of T in $F \cup \bigcup_{j=0}^{m_G-1} T_{\Sigma_j}$ by computing:
  - the sets Σ_j which represents redendant alphabets computed separately for each string of T ;
  - m_G the number of this alphabets;
  - F the set of strings of T for which the alpabets didn't correspond to any Σ_j, F correspond to F_G of the original system G;
  - For each alphabet Σ_j extract the subset T_{Σ_j} from T where: T_{Σ_j} are all strings of T with alphabet equal to Σ_j; and all elements are linearly ordered with respect to $\preceq$;
(B) Let T_{Σ_j} the test set of $L(G_j)$, then construct all $G_j = (\Sigma_j, h_j, w_j)$ $(0 \leq j \leq m_G - 1)$ on the basis of Nielsen principle [19]. In this stage, one may adopt the *algebraic method* for obtaining all homomorphisms h_j satisfying a given D0L sequence of strings T_{Σ_j} with alphabet Σ_j;
(C) Definition of G_T (this step are performed using combinatorial method):
  (C.1) Find h_T rules corresponding to $\cup_j L(G_j)$. For this, find the homomorphism between the first elements of all test sets T_{Σ_j}. We must take into account the fact that each h_j is equivalent to m_G applications of the homomorphism h_T;
  (C.2) Further, find h_T rules corresponding to F in which elements must classifed according to the relation $\preceq$. We must also find a rule linking the last element of F with the first one of a particular T_{Σ_j};
  (C.3) Let w_T the first element of F, be the axiom of G_T;
Return the PD0L-system $G_T = (\Sigma, h_T, w_T)$ equivalent to G;
end of Procedure(T;G);

[11] The relation $\preceq$ is defined on $\mathbb{N}^m$ representing the parikh image of strings of Σ^* where $|\Sigma| = m$. $\preceq$ is defined as follows: for $v_i = (x_{i,1}, ..., x_{i,m})$ $(i = 1, 2)$ in $\mathbb{N}^m$, $v_1 \preceq v_2$ iff $1 \leq \forall j \leq m$: $x_{1,j} \leq x_{2,j}$.

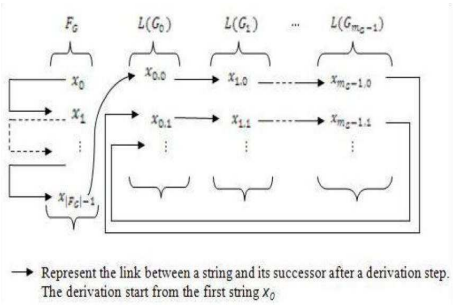

Figure 1: Illustration of strings derivations order in the sets F_G and all $L(G_j)$ $(0 \leq j \leq m_{G-1})$.

number m_G, and then explain the method of construction of each G_j. In summary, we take here the first derivations of the target language (named a test set) in the aim to infer a PD0L-system, which correspond to the problem noted PDA0, in this context the growth function of strings of L are detected from the frequency of repetitions of their alphabets, for this the test set must contain sufficiently information about the language L. It must contain the set F_G in addition to at least the sequence $\{x_{0,0}, x_{1,0}, ..., x_{m_G-1,0}, x_{0,1}, x_{1,1}, ..., x_{m_G-1,1}\}$ cited in Fig. 1, in the aim to discover the alphabet repetitions.

This principle is detailed in Algorithms 5 and 6. Algorithm 5 identify a PD0L-system $G = (\Sigma, h, w)$ from a finite set $T = F_G \cup \bigcup_{j=0}^{m_G-1} T_{G_j}$ (called *"finite telltale set"*) of $L(G)$, where each T_{G_j} is a test set of $L(G_j)$. Procedure$(T; G)$ detailed in Algorithm 6, works for all T containing a finite telltale set of $L(G)$, and obtain a PD0L-system $G_T = (\Sigma, h_T, w_T)$ equivalent to the original G for L.

Analysis. Several conclusions have been made in [23]: An open problem is whether or not the family of D0L-languages is identifiable in the limit from positive data. Also, the question remains open whether or not there exists an algorithm for identifying the family of D0L(PD0L) languages in the limit from positive data using D0L(PD0L)-systems, respectively. However, there is no work concerning the identification of 0L-languages families "in the limit" framework, and a little is known about the efficient identifiability in the limit, in particular, from positive data.

From the study of this inference method, we extract several remarks, which the most important ones are: Yokomori gives very important and useful results on the identifiability in the limit of subclasses of 0L-languages from positive data. The strength of algorithms proposed in this work is the guarantee of identifying a PD0L-system in the limit from only positive data. But we must dispose of a test set of each language generated by an unknown PD0L-system. Without this test set, the method cans not success. In the case of grammatical inference of biological systems, we are not sure of finding such a set directly from observations of a biological species. Also, the proposed identification algorithm of PD0L-systems use a special procedure which must found all homomorphisms associated to two given strings (steps (B) and (c) in algorithm5), the computation of this procedure can be so complex in the case of a sequence of too long strings. Finally, this inference method did not work on branching structures, because it aims to find correspondences between successive strings in the sequence regardless of the semantics of each symbol (its graphical interpretation [1]).

5 Heuristic inference method(GLP)

Genetic Programming (GP) has been introduced as a heuristic method to automatically develop populations of computer programs through simulated evolution [14]. Considering L-systems as rulebased development programs it is easy to define program evolution. Each program encoded as a symbolic expression has to be interpreted and is assigned a fitness value dependent on the optimization task to be solved. On the basis of these fitness the individual programs struggle for "survival of the fittest" and for the chance to become members of the next generation. In order to introduce variations into the program encoding structures genetic operators like mutation or crossover are applied. The evolution process develops new populations of programs from generation to generation, the interplay of modifying operators and selection hopefully leading to ever better programs.

L-systems have been introduced independently into the area of genetic programming (GLP) by different researchers. They have been considered as examples of generative encoding for evolutionary algorithms, we have shown some important ones related to plant-like structures represented by : bracketed D0L-systems [2], parametric L-system [4,11], D0L-systems [16], parametric D0L-systems [13], L-systems of particular tree species [21],...Than the problem is referred as DCx, because of this method is evolutionary we can't speak about their decidability point of view. In these works, several situations have been dealt. Some of them produce derivation rules from a visual result that the output L-system must produce; the calculation of fitness is then made by interpreting graphically each L-system and comparing its result to the target image. Others infer parameters from a set of already fixed rules.

Analysis. We can conclude that genetic algorithms provide a good solution for the inference of L-systems,

especially the types whose inferences is considered to be difficult, or that there is no algorithm. However, these inference methods have limitations, which favor constructive inference methods in cases where they exist, view the drawbacks of genetic algorithms [14], i.e.: (1) The computing time: unknown time of convergence, and compared with other heuristiques, they require numerous calculations, particularly at the level of the evaluation function. (2) Uncertainty on the algorithm convergence: It should also be noted the impossibility of being insured, even after a significant number of generations, that the solution is the best. We can only be sure that we approached the optimal solution without the certainty of having reached. (3) They are often difficult to implement: Parameters such as population size or the rate of mutation are sometimes difficult to determine. But success depends on the evolution and several trials are therefore needed, which further limits the effectiveness of the algorithm. In addition, choosing a good evaluation function (for the calculation of the fitness) is also critical. It must take into account the good parameters of the problem. It must therefore be carefully chosen. (4) The great spatial complexity dependent on the size of the population. (5) Another important issue is that of local optima.

6 Data compression resolution by L-system inference

Another case of grammatical inference was defined by Nevill-Manning and Witten [18] to meet the needs of data compression, they proposed an algorithm (named SEQUITUR) that infers a hierarchical structure from a string of discrete symbols by replacing repeated phrases with a D0L-system rule that generates the phrase, and continuing this process recursively. Also this algorithm works incrementally, and uses two defined properties:

p_1: no pair of adjacent symbols appears more than once in the D0L-system.

p_2: every rule is used more than once. This property ensures that each rule is useful.

These two constraints exactly characterize the grammars that SEQUITUR generates. SEQUITUR's operation consists of ensuring that both properties hold (see algorithm6). When describing the algorithm, the properties act as constraints. The algorithm operates by enforcing the constraints on a grammar: when the p_1 constraint is violated, a new rule is formed, and when the p_2 constraint is violated, the useless rule is deleted.

Analysis. In this case, a D0L-system is inferred from a sequence of size 1, and then we can say that the problem here is DA0. The great advantages of this method, if it is considered as a compression method, are its performances in data compression, in a linear time complexity. Perhaps its greatest drawback is its memory usage (to save the structure I of algorithm6), which is linear in the size of grammar. Linear memory complexity is ordinarily considered intractable, although in practice SEQUITUR works well on sequences of DNA of rather impressive size.

Algorithm 7 Summary of the SEQUITUR algorithm

Input: a string $S = s_1 s_2 ... s_n$;
Output: a D0L-systems G generating the string S;
Method:
- Initialize the set R of production rules of G by the unique rule $R_1 \rightarrow s_1 s_2 ... s_n$, with R_1 the axiom; and initialize $S' = s_1$, $i = 1$;
Repeat
  -$i = i + 1$ and $S' = S' s_i$ (add the next symbol from S);
  - Find from S' the set I of all substrings of lenght 2;
  If their is an element e of I which repeated in the right parts of rules of G (to deal with property p_1) **then**:
    - create a new rule $R_i \rightarrow e$ and replace all occurences of e in G by the symbol R_i;
    - modify the set I with the new substring obtained after the e substitution;
  If their is a symbole which not used more than once (verification of property p_2 for all productive symbols), **then**:
  - We must remove the rule and substituting its contents in place of the other non-terminal symbols. Also modify I.
until $i = n$;
Return G.

In addition to this drawback cited in [18] , we can say that in this case the grammar is inferred from a positive sample containing only one string, also this grammar are not minimized, and it can only generate this string without generalization: only non-recursive grammar are inferred by this method, whereas most useful L-systems are recursive, this process cannot represent the biological development of a plant structure in which repeated modules as well as modules generated by a regular developmental model, called self-similar modules, can be found. For this Nevill-Manning proposed in [17] an improvement of the SEQUITUR algorithm to deal with bracketed D0L-system rules format when performing the inference and this with a unification-based rule generalizer. The unification was made in conjunction with the verification that the rules right parts are well-parenthesed. Although the construction phase of grammar is built in a linear time, the process of its generalization to get a bracketed D0L-system with recursive rules remains a tedious task, when should look into isomorphic rules. This final step makes the algorithm to loose its character as a linear time.

7 Discussion

We have covered in this paper a large number of results on L-systems inference, and have made their analysis. It is a vast area in which we found that inference as presented in this work does not solve all problems for all cases. Then what has been done is that particular types of L-systems have been produced in answer to some particular situation. In addition, we must make assumptions on the initial data which can take form of a particular type of sequence, and restrictions on the hypothesis space, sometimes with a specific presentation of target L-systems.

According to [10], future research directions should focus on inference in a particular area or to solve a real problem. Each real problem is a particular instance of a more general one, which has its own characteristics, this consideration limits then the search space during the inference process. We are interested here specifically to the case of the study of tree structures. There is a good attempt of Jurgensen and Lindenmayer[12] in solving the problem in the descendent sub-trees search space, but this solution remains elusive in practice (see section 3.2). While other methods (algebraic, and PD0L-systems identification in the limit) are very effective in abstract languages, they remain inapplicable for tree structures.

Furthermore, since GLP proposition, no attempt to define a new constructive method of L-system inference has been made. GLP offered a good solution to this problem but it remains a heuristic because it does not understand the mechanism with which the plant develops: it is a black box that will give us an acceptable L-system that generates exactly an input tree without guarantees that it also generates its other stages of development. While, the goal here is not only to infer but also to analyze the growth process of a biological organism.

The ideal would be to create cooperation between several methods by dividing the problem in several levels; each of them can be solved by the method that seems to be most effective. For example, from a primary plant architecture infers rules describing the structure and the development of the basic topology, (like in [12]). And use another approach to infer rules describing the details (like GLP to describe leaves, flowers...). These methods, although different, can complement each other.

Another solution is to consider the inference not as a full-fledged problem but as a solution to another problem, such as compression of DNA biological sequence proposed by Nevill-Manning and Witten [18]. This principle allows the use of the characteristics of the solved problem to help and to guide the inference process.

Also, we can change the presentation of data on which we learn. A pretreatment will better target the result. Always wanting to infer a grammar from a sequence of strings is not the most appropriate format for all problems. For example, for trees it would be better to classify sub-trees that were generated at the same time (and therefore are identical) and use this classification to represent the whole of the tree, and this to guide its L-system inference.

8 Conclusion

This paper has considered the problem of L-system inference. All this study takes us a general conclusion, that, there is no universal method of L-system inference that meets all expectations. Hence, several problems remain open in L-system inference. Also, little was done on inference from tree structures, which are clearly related to the principle of L-systems through modeling.

We have made three propositions to improve capacities of L-system inference process: The first one was to combine several existing methods by dividing the problem into sub-problems and to solve each of them separately by the most suitable method. The second solution is to consider the inference as a solution of another problem; this principle allows the use of the characteristics of the solved problem to help and to guide the inference process. The third one is to change input data representation with one which is more adapted to the problem and that can improve its analysis.

In future, we are particularly interested in the study of tree structures. This is a largely expanding area whose needs are growing. Several studies have focused on the fact to work from their compressed format [7]. In perspective, we try to apply our second and third propositions by setting an improved tree compression method on the basis of L-systems inference, where in the inference process each tree is represented by its initial compressed format, proposed in [7], from which we search the L-system corresponding to the initial tree. This compression will complement the work of Ferraro and Godin[7] in the field of study and analysis of tree structures topology .

9 Acknowledgment

The authors thank Christophe Godin, Pascal Ferraro and Jean-Christophe Janodet for collecting the information and for insightful comments and editorial help.

10 Bibliograph References

[1] Abelson, H., DiSessa, A. *Turtle Geometry*: The Computer as a Medium for Exploring Mathematics, MIT Press, Cambridge, MA, 1981.

[2] Ashlock, D., Bryden, K.M., Gent, S.P. *Simultaneous Evolution of Bracketed L-system Rules and Interpretation*. Evolutionary Computation, CEC2006. IEEE Congress on Volume, Issue, 0-0 0 Page(s):2050-2057, 2006.

[3] Cornuéjols, A., Miclet, L. *Apprentisage artificiel : concepts et algorithmes*, Eyrolles, 2002.

[4] Curry, R., *On the Evolution of Parametric L-systems*, Technical Report, No.1999-644-07, Department of Computer Science, University of Calgary, Calgary, 1999.

[5] Doucet, P.G. *The syntactic inference problem for D0Lsequences*, lecture notes in Computer Science 15, 1974.

[6] Feliciangeli, H., Gabor, Herman, T. *Algorithms for producing grammars from sample derivations: a common problem of formal language theory and developmental biology*, journal of computer and system sciences 7, 97- 118, 1973.

[7] Godin, C., Ferraro, P. *A general method for quantifying the structural self-similarity of trees*. Technical report, INRIA, 2007.

[8] Gold, E.M. *Language Identification in the limit*, Information and control, Vol.10, No.5, pp, 447-474, 1967.

[9] Herman, G.T., Walker, A. D. *The syntactic inference problem applied to biological systems*, in "Machine Intelligence 7" (Michie, Ed.), Edinburgh University Press, Edinburgh, 1972.

[10] de la Higuera, C. *A Bibliographical Study of Grammatical Inference*, Pattern Recognition Volume 38, pp.1332- 1348, 2005.

[11] Jacob, C. *Genetic L-System Programming: Breeding and Evolving Artificial Flowers with Mathematica*, IMS 95, Proc. First International Mathematica Symposium, Southampton, Great Britain, Computational Mechanics Publications, Southampton, UK, pp. 215-222, 1995.

[12] Jürgensen, H., Lindenmayer, A. *Inference algorithms for developmental systems with cell lineages*, Bulletin of mathematical biology, volume49, number1, 93-123, Springer, 1987.

[13] Kókai, G., Tóth, Z., Ványi, R. *Evolving Artificial Trees Described by Parametric L-system*: In Proc. IEEE Canadian Conference on Electrical & Computer Engineering, Edmonton, Alberta, Canada May 9 - 12, 1722-1728, 1999.

[14] Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press, 1992.

[15] Lindenmayer, A. *Mathematical models for cellular interaction in development*, parts I and II. J Theor Biol, 18:280-315, 1968.

[16] Mock, K.J. Wildwood: *The evolution of Lsystem plants for virtual environments*. In Proceedings ICEC 98, pages 476-480. IEEE-Press, 1998.

[17] Nevill-Manning, C.G. *Inferring sequential structure*, Ph.D. thesis, Department of Computer Science, University of Waikato, New Zealand, 1996.

[18] Nevill-Manning, C.G., Witten, Ian H. *Identifying Hierarchical Structure in Sequences: A linear-time algorithm*. Journal of Artificial Intelligence Research 7, 67–82, 1997.

[19] Nielsen, M. *On the decidability of some equivalence problems for D0L-systems*. Information and control, 25:166-193, 1974.

[20] Prusinkiewicz, P. , Lindenmayer, A. *The Algorithmic Beauty Of Plants*. (review) Springer-Verlag, 1990.

[21] Runqiang, B., Chen, P., Burrage, K., Hanan, J., Room, P., Belward, J. *Derivation of L-system Models from Measurements of Biological Branching Structures Using Genetic Algorithms*: Developments in Applied Artificial Intelligence, volume 2358 of Lecture Notes in Computer Science, Springer, 2002.

[22] Valliant, L. *A theory of the learnable*. Communication of the ACM, vol.27, n11, pp.1134-1142, 1984.

[23] Yokomori, T. *Inductive inference of 0L languages*, in: G. Rozenberg and A. Salomaa (eds.), Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology, Springer, 115– 132, 1992.