5-18-1981

The biological and mathematical basis of L systems

Christine Kelly-Sacks

ROCHESTER INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

THE BIOLOGICAL AND MATHEMATICAL BASIS OF L SYSTEMS

A Thesis submitted in Partial Fulfillment of
Master of Science in Computer Science Degree Program

BY: Christine Kelly-Sacks

APPROVED BY:

Peter H. Lutz _ Peter H. Lutz, advisor

Michael J. Lutz - Michael J. Lutz

Rodger W. Baker - Rodger W. Baker

DATE: May 18, 1981

Table of Contents

OVERVIEW

The theory of Lindenmayer systems (L-systems) was developed to mathematically describe biological development models using the cell as the basic unit of development. The theories of finite state machines and automata and formal languages were incorporated into a framework which emphasized and maintained the significance of biological forces behind development.

This type of model was first proposed by Lindenmayer (1968a,b), a developmental biologist, in the form of a finite state machine, and it has evolved into a well-investigated branch of formal language theory. Other theories had been previously proposed using the same mathematical basis to simulate development, but using a different biological context. Rather than the cell, Rosen (1964) used the whole organism as a unit, while others tried using intracellular activities (Stahl and Goheen, 1963; Stahl, 1965; Chiaraviglio, 1965). For a review of alternate theories, see Apter (1966, chapter 3).

The purpose of this thesis is to describe the theory of L-systems from both a theoretical (formal language) and biological point of view. Emphasis will be placed upon the biological significance of the theoretical model.

A brief overview of formal language theory will be presented, based on the Chomsky hierarchy of languages.

Standard notation for formal languages will be taken from Hopcroft and Ullman (1969).

The notation used to describe L-systems will be defined as it is presented. Much of the terminology is drawn from formal language theory and corresponds to that described by Hopcroft and Ullman. The terminology specific to L-systems is taken primarily from Herman and Rozenberg (1975) and is summarized in Appendix A for convenience.

It is important to emphasize that the families of languages generated by classical formal language theory are not families of developmental languages. In some ways, the two systems are analogous and in other ways different or incomparable.

The Chomsky Hierarchy

The basic elements of any Chomsky grammar are the alphabet, V, which is any finite set of symbols; the set of productions, P; and a special symbol, S, which is called the start symbol. V is composed of the union of two finite sets, whose intersection is empty, called Vn (variables or non-terminals) and Vt (terminals). S must be an element of the set of non-terminals. Each member of V is exactly one symbol (that is, has a length of one). A production is an expression having the form a -> b (a derives b) where $a \subset V+$ and $b \subset V*$.

V* represents the set of any string of symbols from the alphabet, including {Λ}. V+ is V* - {Λ}. A sentence is any member of V* whose length is finite. If a ∈ V*, then |a| represents the length of the string a; that is, the total number of symbols in a. The set of productions is called P.

A grammar G is defined as a 4-tuple,

$$G = \langle Vn, Vt, P, S \rangle$$

The language generated by G is

$$L(G) = \{ w \mid w \text{ is in } Vt^* \text{ and } S \overset{*}{\underset{G}{\Rightarrow}} w \}$$

That is, w is in L(G) if it is a string comprised solely of symbols from the set of terminals and can be derived from the start symbol according to the production rules in P in one or more steps. Any language L generated by a grammar G is said to be recursively enumerable, abbreviated RE.

If G = <Vn, Vt, P, S> is a grammar, then if, for every production a -> b in P, |a| <= |b|, then G is said to be context sensitive (CS). If, for every production a -> b in P, |a| = 1 and |b| > Ø, then G is context free (CF). If every production in P has the form A -> aB or A -> a, where A and B are non-terminals and a is a terminal, then G is said to be regular (RG). It is permissible for any context sensitive, context free or regular language to contain the production S -> Λ, provided that S does not appear on the right-hand side of any production.

The notation F(RG) denotes the family of all languages which are regular languages. Note that

$$F(RG) \subsetneq F(CF) \subsetneq F(CS) \subsetneq F(RE)$$

That is, each of the families is a proper subset of the next one.

Two points regarding Chomsky languages will be emphasized here. First, only those strings which contain exclusively terminals are in the language generated by the grammar, and no symbol is both a terminal and non-terminal. Second, when applying production rules to a string over the alphabet, exactly one production is used in a single step. These two aspects of Chomsky languages are important in differentiating them from the families of developmental languages, as will be demonstrated later.

How development can be related to grammars

When studying developmental languages, it is important to understand the rationale behind the connection between formal languages and automata, and a developing organism.

Every multicellular organism develops from exactly one cell. This original cell contains all the cytological, genetic and physiological components needed to eventually yield the mature form of the organism. This cell must divide, yielding two daughter cells. Passed on to these cells is a copy of all the genetic material contained in the

original cell. This process is carried on and, for a short time,all cells that are produced are generally identical to that one original cell. That is, these cells are undifferentiated, none of them being specially equipped for any specific cellular function. However, with time, cells do differentiate, despite the fact that each one carries identical genetic material.

Exactly how the cells differentiate is not known. But how a cell behaves can be determined by its position relative to other cell types, the presence or absence of chemical cell constituents, the combinations of active or inactive genes, etc. Thus each cell is a functionally autonomous unit and the basic unit to be considered in the description of the developmental model.

In order to describe this model as a string of automata (each cell being an automaton), it is necessary to describe the states, inputs, outputs and state transition functions. The states can easily be described in terms of the cells' biochemical composition and/or what their genetic capabilities are. The inputs to the cell are whatever external stimuli it receives in a given time frame; for example, a neighbor cell produces a compound which causes a membrane excitation. Any material this cell produces for export is the output of this cell. The transition function is a description of how the genetic capabilities of a cell are altered, either by repression or derepression, and effects

of any other cytoplasmic constituents (Lindenmayer, 1974). Cell death can be represented by replacement of a symbol with the null string, cell division by replacement of one cell with two cells. One cell may be replaced with one different cell, indicating a change in the cellular capabilities or functions ascribed to that cell.

If a very simple system is being modeled, such as the growth of a filamentous organism, this concept of an organism as a string of automata can be used literally as described, with each individual automaton representing one cell. In many cases, however, it is impossible to follow the development and differentiation of an entire organism from the one-cell stage in this manner. The basic problem stems from the fact that this model is a two-dimensional one, and a growing organism is growing in three dimensions. Cellular migrations and three-dimensional cellular juxtapositions play an important role in the development of an organism, and this form of modeling cannot follow an organism from one cell to adult. If, however, the development of form or pattern is being followed with symbols representing aggregates of cells or the presence of a certain structure, or just part of the organism is being considered (a leaf instead of the entire tree), the model is much more adaptable while still maintaining the biological significance. Work is being done to use a two dimensional model to simulate three dimensional development (Rozenberg and Salomaa, 1980; Mayoh, 1974, 1976; Carlyle, Greibach and Paz, 1974;

Paz, 1976; Nagel, 1976).

In his original papers, Lindenmayer (1968a,b) modeled growth of linear arrays of cells. In one case, he described filaments in which cells could be influenced by both neighbor cells, the left neighbor cell only, or no other cells. In the first case, there are two output signals to consider; in the second, there is an output signal migrating to the right; in the third case, there is no output signal. Despite the fact the the model used was a very simple one, quite complex developmental patterns did emerge. (See Appendix B for a detailed example of each type of system). It has also been shown that Lindenmayer's systems are as powerful as Turing machines (van Dahlen, 1971).

The constructs described for L-systems are analogous to the grammars used in the Chomsky hierarchy, and can be expressed in analogous terms. The hierarchy of L-systems will in fact be defined this way and referred to henceforth in those terms. Two very important differences between classical languages and developmental languages must be kept in mind.

Developmental languages have an alphabet, but this alphabet is not divided into two disjoint sets of terminals and non-terminals. Therefore, any string which can be derived from the "start symbol(s)" (axiom) is part of the language.

Developmental languages have production rules (the next-state function), but production rules are applied for every symbol in the string simultaneously (parallel rewriting), while in a derivation in a Chomsky grammar, only one production rule is used in a single step.

Types of L-systems

As was mentioned previously, Lindenmayer took into account two general environments in which development was taking place. One was the situation in which no cell received inputs from other cells, an "interactionless" system. The other was a system in which there were cellular interactions.

If there are no cellular interactions, a mathematical construct reflecting this which includes the production rules (next-state function), the alphabet (set of state symbols) and the axiom ("start string" or starting string of states, length of at least zero, analogous to the start symbol) has been called a zero-sided, informationless or interactionless Lindenmayer system, or ØL-system. If the production rules are deterministic (each member of the alphabet has at most one production rule associated with it), then it is a DØL-system. If the next-state function is never the empty symbol (non-erasing, thus allowing for no cell death), the system is propagating, or PØL. If both propagating and deterministic, it is PDØL.

Some DØL systems exhibit the phenomenon that they produce repetitiveness of substrings. These are called "locally catenative sequences" and allow strings (other than the first ones) to be described in a formula as a concatenation of previous strings in a sequence (Herman and Rozenberg, 1975; Rozenberg and Lindenmayer, 1973). An example of just such a system will be given shortly. A similar property, the "recurrence" property, is seen in many ØL-systems, and is an extension of locally catenative sequences. A recurrence system is made up of sets of formulas which specify, by concatenation of strings, all of the strings of a given ØL-sequence. Note that a locally catenative sequence has only one formula (Herman, Lindenmayer and Rozenberg, 1974). An example of such a system is presented in Chapter 1.

If there are cellular interactions, as reflected in the production rules, the system is an interactive or IL-system. There may be any number of neighbor cells to the left and any number to the right which are influencing a given cell. If there are m left neighbors and n right ones, the system is an <m,n> system, where m and n are both non-negative integers. If both m and n are equal to zero, it is a <Ø,Ø> system, or a ØL-system. If either m or n (but not both) is zero, it is a one-sided system, sometimes called a 1L-system. Otherwise, it is a two sided or 2L-system. Again, these systems may be propagating and/or deterministic, PD<m,n>-systems, or PDIL-systems. In his original work,

Lindenmayer allowed only for m and n to be no greater than 1. This was later expanded to the definition given.

There are times when environmental conditions must be taken into account when considering the growth and development of an organism. For example, the same plant cells will act completely differently when exposed to sunlight than when in darkness. If both conditions can be described individually as ØL-systems, then it is possible to use the same axiom and alphabet and greater than one set of production rules (only one set to be used at a time) to describe this system. This is called a table ØL-system, or TØL-system. If there is only one table, then it is a ØL-system. As with ØL and IL systems, there may be PDTØL-systems as well. Table systems are particularly significant when taking changing environmental conditions into account.

A further refinement may be applied to any of the types of L-systems. The alphabet may essentially be broken down into terminals and non-terminals by designating a target alphabet, Δ, such that $\Delta \subset \Sigma$, and only those strings which consist solely of symbols from Δ are in the language. This is called an extension language, and is designated as EØL, EIL or ETØL. These systems may also be propagating and/or deterministic.

Making a symbol a "terminal" has the biological meaning that the cell represented by it has been irrevocably differentiated and can not form a different type of cell,

although possibly more of the same type of cell. An example of such a cell is the human red blood cell, which no longer has a nucleus and therefore is incapable of further differentiation or division.

That the theory behind L-systems holds true for real models has been demonstrated using the simulation language CELIA (Cellular Iterative Array Simulator). CELIA was first developed by Baker and Herman (1972). It was later improved by Herman and Liu (1973) and then again by Liu (1973).

The development of a number of organisms has been described using CELIA, for example: the distribution of pigment in sea snail shells, a function of glandular activity (Herman and Liu, 1973); development of algae and heterocyst formation in blue-green algae (Baker and Herman, 1972); the growth and flowering of Aster (Frijters, 1974); pattern formation in hydra (Herman and Schiff, 1974).

Example of ØL-systems

This example has been used by Lindenmayer (1975b; Rozenberg and Lindenmayer, 1973; Herman and Rozenberg, 1975) as a simple DØL-system which can be clearly illustrated. Many organisms contain structures which consist of a number of similar parts which repeat to form the organism. This is apparent in a compound leaf of a plant, or in flowering structures. This is what has already been described as a locally catenative sequence, and shows the biological motivation behind the theory.

If the following DØL-system is assumed:

alphabet {a,b,c,d,e,f,g,h,i,j,k}

productions

a -> bc	f -> ih
b -> kd	g -> hi
c -> ek	h -> de
d -> gb	i -> k
e -> cf	k -> k

axiom a

This system will produce the following sequences (which happen to be locally catenative).

a

bc

kdek

kgbcfk

khikdekihk

kdek kgbcfk kdek

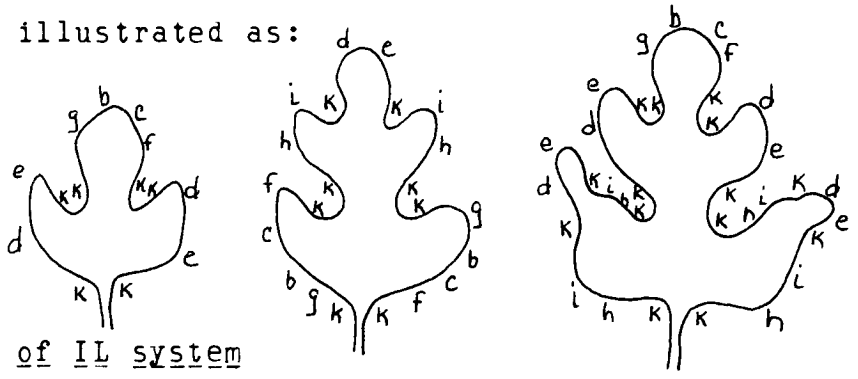
kgbcfk khikdekihk kgbcfk

kkhikdekihk kdekkgbcfkkdek khikdekihk

k represents a state in which the cells are not growing on the leaf margin (in the notches). If the leaf is at or beyond the sixth state, then the leaf has a left lobe and a right lobe which are identical to the whole leaf three stages previously, and a middle lobe which is identical to the whole leaf two stages previously. That is, for n>=6,

$$S_n = S_{n-3}\ S_{n-2}\ S_{n-3}$$

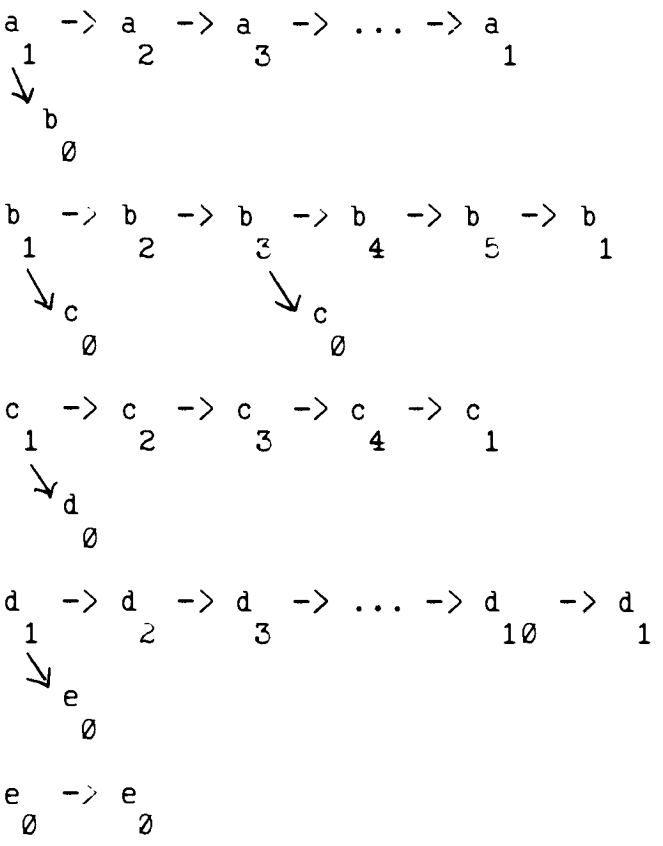
This is illustrated as:



Example of IL system

This example of an interactive system was constructed by Lindenmayer (1975b). It shows the development of the main root of maize. The root exhibits a gradient rate of growth which is a function of the distance from the tip of the root. Division is slow near the tip, increases to a maximum 4 mm. from the tip and ceases at 10 mm. from the tip. Interactions can take place in both directions (therefore this is a 2L system), time is taken in 1 hour units, and the root is divided into 2 mm. units (yielding five segments, labelled a-e).

The segments a-e have the following timing cycles: 1/6, 2/5, 1/4, 1/10, and 0/1 respectively (where x/y represents x cell divisions in y hours). This can be illustrated as:



The rules for the transition function are expressed as follows: p and q stand for any pair of different symbols from {a,b,c,d,e}, x>0, y>=0.

<p , q > -> q
y -x x+1

<q , q > -> q
-0 x x+1

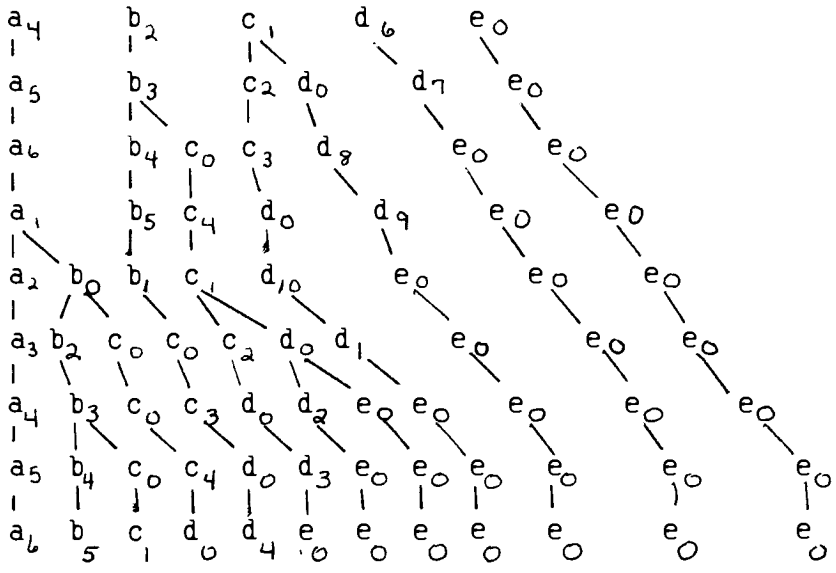
<p , p , q > -> q
0 -x y 0

<p , p > -> p
-0 0 0

The underlined term is influenced by either or both of its immediate neighbors, as indicated by the specifications.

For example, a cell in state a5 with a cell in state b5 to its immediate left will be rewritten as a6.

A developmental sequence can be illustrated as:



Example of a tabled system

This model was also presented by Lindenmayer (1975b). It is the same concept of leaf development as presented previously, but with two possible sets of production rules. This allows for terminal structures, whereas the earlier example can only expand infinitely. The alphabet and axiom are the same, but the tables of productions are as follows:

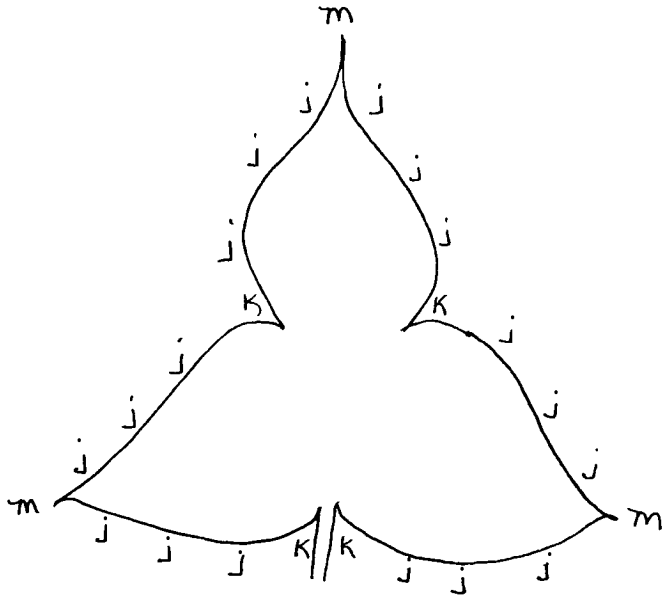
table 1: a->kbk, b->cdc, c->e, d->kek, e->jej,
j->j, k->k, m->m

table 2: a->kmk, b->e, c->j, d->kmk, e->jmj,
j->j, k->k, m->m

A possible derivation is: (which table is used is indicated)

	a
t1	kbk
t1	kcdck
t1	kekekek
t1	kjejkjejkjejk
t2	kjjejjkjjejjkjjejjk
t2	kjjjmjjjkjjjmjjjkjjjmjjjk (a terminal structure)

This structure can be illustrated by:



ØL-Systems

Definitions

ØL languages constitute the basic family of developmental languages. Each cell (symbol) is considered an autonomous pre-programmed unit, and its behavior is determined only by its cell lineage (ancestry). Since many of the biological systems which are best understood and are describable are based on knowledge of the morphogenic power of the cell lineage, it is likely that it can be described by a ØL-system. In fact, it is possible to describe a real system, which is known to depend on cellular interactions, using a ØL model (Lindenmayer, 1975). ØL-systems can be shown to make up a proper subset of context sensitive languages, and specific kinds of ØL-systems also have specific relationships to languages of the Chomsky hierarchy.

The definition of a ØL Scheme, S, is:

$$S = \langle \sum, P \rangle$$

where $\sum$, the alphabet of S, is a finite, non-empty set and P, the set of productions, is a finite, non-empty subset of $\sum \times \sum^*$ such that

$$(\forall_a)_{\sum} (\exists_A)_{\sum^*} (\langle a, A \rangle \in P)$$

which can also be expressed as

$$(\forall a \in \Sigma)\ (\exists A \in \Sigma^*)\ (\langle a, A\rangle \in P)$$

From here on, for any ØL schemes, $\langle a, A\rangle \in P$ will be written

$$a \underset{S}{\rightarrow} A$$

If the underlying scheme is understood, the letter designating it may be omitted.

A ØL system is a triple

$G = \langle \Sigma, P, w \rangle$, where

$S = \langle \Sigma, P \rangle$ is a ØL scheme and w, called the axiom of G, is a word over the alphabet. The axiom corresponds to the "start symbol" in formal language theory, but note that it may have a length greater than one, which is not true of the start symbol.

$L(G)$ is the language generated by $G = \langle \Sigma, P, w \rangle$. If G is a ØL system, $L(G)$ is defined as a ØL language such that

$$L(G) = \{ x \mid w \overset{*}{\underset{G}{\Rightarrow}} x \}$$

A language L is a ØL language if and only if $L = L(G)$ for some ØL system G.

For any ØL scheme $S = \langle \Sigma, P \rangle$, for any word x in Σ^* and any nonnegative integer n, the finite language $L_n(S,x)$ is defined by induction on n as:

$$L_{\emptyset}(S,x) = \{x\}$$

$$L_{n+1}(S,x) = \{y \mid (\exists z)\ (z \in L_n(S,x) \text{ and } z \Rightarrow_S y)\}$$

Let $S=<\sum, P>$ be a ØL scheme.

If $x = a_1 \ldots a_m$ $(m>=\emptyset)$ and $a_j \in \sum$ for $j=1,\ldots,m$

and $y \in \sum^*$
then x directly derives y in S $(x \Rightarrow_S y)$
if and only if $(\exists a_1,\ldots,a_m)_{m \sum^*}\ ((a_1 \to b_1,\ldots,a_m \to b_m)_{1\ 1\ \ m\ m}$

and $(y = b_1 \ldots b_m))$

Note that all substitutions are made simultaneously.

Basic concepts and examples

ØL languages may be finite or infinite. A deterministic ØL language (IØL) has the property that

if $x \to y$ and $x \to z$ then $y = z$
for any $x \in \sum$, $y,z \in \sum^*$
and the axiom is not the empty sentence.

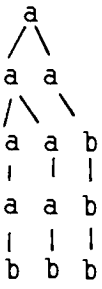
A propagating ØL system (PØL) has the property that $x \to \Lambda$ for $x \in \sum$ does not exist.
Therefore, if G is a PØL system and $x \Rightarrow y$ in G, then $|y| >= |x|$.

An example of a non-deterministic, propagating ØL system is:

$G = <\{a,b\}, \{a\to a,\ a\to b,\ a\to aa,\ b\to b\}, a>$

Note that the language generated by G is $\{a,b\}^+$.

One possible derivation tree for this system is:



An example of a non-propagating, deterministic ØL system is:

$$G = \langle \{a,b\}, \{ a \rightarrow (ab)^2, b \rightarrow \Lambda \}, ab \rangle$$

$$L(G) = \{ (ab)^{2^n} \text{ for } n \geq 0 \}$$

ab

abab

abababab

ababababababababab

.
.
.

It is possible for a DØL system which is not propagating to generate the same language as a PDØL system. For example:

$$G = \langle \{b,c,d,e\}, \{b \rightarrow cde,\ c \rightarrow b,\ d \rightarrow \Lambda,\ e \rightarrow \Lambda \},\ b \rangle$$

$$L(G) = \{b,\ cde\}$$

H = <{b,c,d,e}, {b->cde, c->c, d->d, e->e}, b>

L(H) = {b, cde}

The difference between L(G) and L(H) is the way and sequence in which the two languages were generated, and how many times each word may be generated. The two languages are, however, equivalent.

There are languages which cannot be described by a ØL system. For example, L= {a, aa} is not a ØL language. L= $\{a^n \mid n>0\}$ is not a DØL language, although it could be a non-deterministic language.

It is possible to describe a branching filamentous organism (such as an alga) using a ØL system. Parentheses are used to designate a branching from the filament. This example will be referred to again, as this PDØL-system is also an example of a recurrence system, which will be described in more detail in the next chapter.

G = <∑, P, 4>

∑ = {(,),Ø,1,2,3,4,5,6,7,8,9}

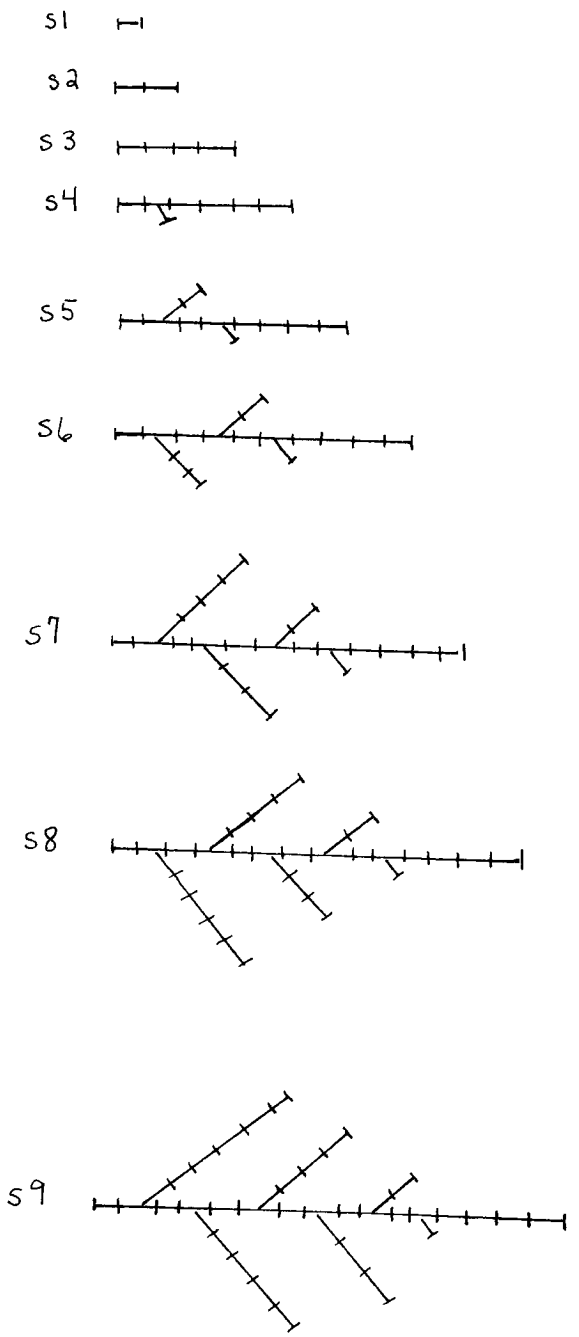
P = { Ø -> 10, 1 -> 32, 2 -> 3(4), 3 -> 3,
4 -> 56, 5 -> 37, 6 -> 58, 7 -> 3(9),
8 -> 50, 9 -> 39, (-> (,) ->) }

A sequence of productions yields

s1 4
s2 56
s3 3758
s4 33(9)3750
s5 33(39)33(9)3710
s6 33(339)33(39)33(9)3210
s7 33(3339)33(339)33(39)33(4)3210
s8 33(33339)33(3339)33(339)33(56)33(4)3210

s9 33(333339)33(33339)33(3339)33(3758)33(56)33(4)3210

This is more easily visualized as:



The growth of this organism can be verbally described. After the fifth stage (that is, for n>=6), the organism can be divided into two parts; the first six cells of the main branch being basal and the rest being apical. The odd-numbered cells of the basal part have a branch from the filament, while the even-numbered cells do not branch. These basal branches grow monotonically, without branching, with time. At stage six, the apical portion consists of six non-branching cells. After stage six, the apical part adds two new cells at its leftmost end, the second of which carries a branch which is identical to the entire organism six stages earlier. For example, the new branch at stage nine is four linear cells, which corresponds to the entire organism at stage 9-6=3.

It will be shown in the next chapter that this entire system can be described without the need for productions, but rather by a formula, and using only one symbol to represent all cells.

The length of the filament at stage n, $|W_n|$, is the number of symbols in the string, including parentheses. This can be described mathematically for n>6 as

$$|W_n| = |W_{n-1}| + |W_{n-6}| + 7$$

The added seven is for the extra three basal cells, extra two apical cells and the two parentheses of the new branch.

Comparing the production rules for this ØL system, it appears to correspond to the definition of a context-free grammar. It is due to the property of simultaneous substitution that this model corresponds in fact to a context-sensitive rather than context-free language. That this particular example is CS can be demonstrated using the "uvwxy" theorem (theorem 4.7 in Hopcroft and Ullman, 1969). This theorem states:

> Let L be any context-free language. There exist constants p and q depending only on L, such that if there is a word z in L, with $|z| > p$, then z may be written as z = uvwxy where $|vwx| <= q$ and v and x are not both Λ, such that for each integer $i>=0$.
> $uv^i wx^i y$ is in L.

In the example given, $|W_n|$ is a strictly monotonically increasing function and, for $n>6$, $|W_n| - |W_{n-1}|$ is also strictly monotonically increasing. So, for any constants p and q depending on this language, there exists a string W_n whose length is greater than p. But, for W_m where $m > n$, $|W_m| > |W_n| + q$. That is to say, subsequent strings cannot be increasing by at most a constant amount, or less than some constant amount. Since this language is infinite and its size increases with time according to a strictly increasing function, there cannot exist a constant q for which $|W_m| <= |W_n| + q$ would be true. Therefore, this language is not context free.

There exists an algorithm (Herman and Rozenberg, 1975) which allows the translation of this or any ØL grammar to a context-sensitive grammar. See appendix D for the context-sensitive grammar which produces this model.

Membership problem

Given a ØL system $S = < \sum, P, w >$, is it possible to determine if any arbitrary string x, where x is a word over $\sum$, is included in the language generated by S?

For a PØL system this is easily answered. In derivations in such a system, the length of any given string must be greater than or equal to any earlier string. It is possible to derive all strings that are less than or equal to a given length; if that length is the same as the length of this arbitrary string, and the x has not yet been derived, then x is not included in this language.

Following the productions from the axiom to some string z is called the derivation of z from w, or D(of z from w). The trace of this derivation (tr(D)) is the sequence of strings that is produced by applying the production rules. It can be proved that, for any ØL system G, every non-empty word x in L(G) can be derived in such a way that no "intermediate" word (one appearing in the trace before x) is longer than $C(|x|)$, where C is a constant dependent on the language and can be calculated. Formally (Rozenberg and Doucet, 1971):

Let $G = \langle \sum, P, \dot{w} \rangle$ be a ØL system. Then there exists a positive integer C such that, for every word x in L(G), there exists a derivation D (of x from $\dot{w}$) such that if $tr(D) = (\dot{w} = x_0, x_1, x_2, \ldots, x_f = x)$ then $|x_i| <= C(|x|+1)$, for every i in $\{0,\ldots,f\}$.
This constant C can be defined as:
max {J, K} where

$$J = \max \{ |s|, s \in L_i(G), \text{ for some } i, 0 <= i <= n \}$$

$$K = \max \{ |A|, a \rightarrow A \text{ for some } a \text{ in } \sum \}$$

Since this constant can be calculated, it is possible for the ØL system G to define, for all n, $K_n(G)$ to be the set of all elements s of L(G) which have a derivation requiring no more than n steps and whose trace does not contain any string of length greater than C(|x|). Following the same reasoning as used for PØL systems, it is possible to determine if any arbitrary string x is contained in the language generated by the ØL system G.

Equivalency Problem

Given any two arbitrary ØL systems G and H, is there an algorithm which can prove that G and H are equivalent (that is, that L(G) = L(H))?

To answer this, it is necessary to use the Post Correspondence Theorem (Hopcroft and Ullman, 1969).

If $\sum$ is an alphabet containing at least two letters, then there is no algorithm which decides for an arbitrary n>=1

and for arbitrary n-tuples $\langle a_1, a_2, \ldots, a_n \rangle$ and $\langle b_1, b_2, \ldots, b_n \rangle$ of non-empty words over $\sum$, whether or not there exists a nonempty sequence of indices i1,...,ik such that $a_{i1} \ldots a_{ik} = b_{i1} \ldots b_{ik}$

A pair of n-tuples as described is an instance of the Post Correspondence Problem over $\sum$, which has a solution if and only if the sequence of indices exist such that $a_{i1} \ldots a_{ik} = b_{i1} \ldots b_{ik}$.

It is possible to construct two arbitrary ØL systems which can be proved not to be equivalent if and only if the described instance of Post Correspondence Problem has a solution. Therefore, to decide if two arbitrary ØL systems are equivalent, there must exist an algorithm to decide if an arbitrary Post Correspondence Problem has a solution. Since this is not the case, the problem for equivalence for ØL systems is unsolvable (for proof, see Rozenberg, 1972).

Adult Languages

There exist ØL schemes which produce words over $\sum$ which derive only themselves. If this is the case, then the set of all strings in the scheme which derive only themselves constitute an adult language. Formally, given a ØL scheme $S = \langle \sum, P \rangle$ and a ØL system $G = \langle \sum, P, w \rangle$, the adult language A(S) is the set of all x in $\sum^*$ such that $x \Rightarrow x$ in S and, for any y in $\sum^*$, if $x \Rightarrow y$ in S, then $y = x$. The

adult language A(G) is the intersection of the two sets A(S) and L(G). A simple example of an adult language involves again a model for development of cells along the margin of a leaf (Lindenmayer, 1971).

Σ = {S,a,b,c,d,e,f,g,h,i,j,k,m,0,1,2}

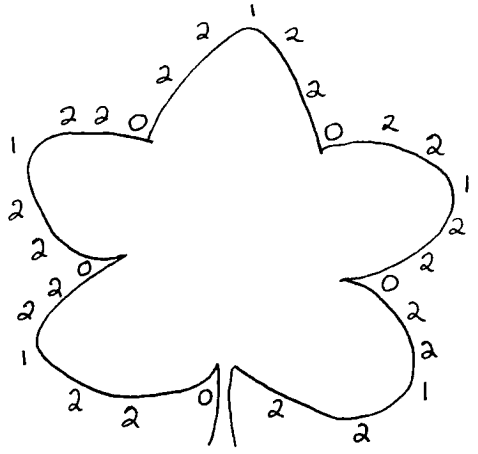
P = { S -> ab, a -> dg, b -> e0, c -> 22,
d -> 0e, e -> cf, f -> 1c, g -> hb,
h -> di, i -> jk, j -> m1, k -> c0,
m -> 0c, 0 -> 0, 1 -> 1, 2 -> 2 }

w = S

The language generated by this system consists of nine different strings:

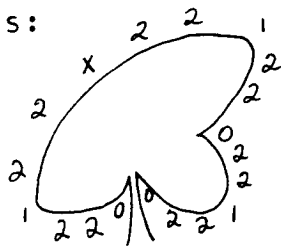
S
ab
dge0
0ehbcf0
0cfdie0221c0
0221c0ejkcf0221220
0221220cfm1c0221c0221220
0221220221c0c1220221220221220
022122022122022122022122022122022122

The column containing the letter S represents the central growing point of the leaf.
The last string derives only itself and can be illustrated as:



It is also interesting to note that if the central growing point should die at some intermediate point, the

parts which are in progress would still expand, yielding a damaged leaf, such as:



if the symbol n were replaced with x, where x -> x would be included in P, in the fourth string. It would not be at all unusual to see a damaged leaf with this morphology.

The adult languages of ØL systems (A(ØL)) are identical to the class of languages of context-free grammars (Walker, 1974; Herman and Walker, 1974). This can be demonstrated by devising a way to construct a ØL system H from any context free grammar (CFG) G such that L(G) = A(H), and how to construct G from H such that A(H) = L(G). The description for how this can be done is in appendix E, with further elaboration in the two references cited.

Extension Languages

It is possible to limit the differences between developmental languages and classical formal languages by selecting a subset of the alphabet to represent what corresponds to terminal symbols. This is referred to as the extension operation, and, for a ØL system $G = \langle \sum, P, w \rangle$, the extended ØL (EØL) system is denoted $K = \langle \sum, P, w, \Delta \rangle$ where Δ is a subset of the alphabet and the language generated by the EØL system is a subset of the language

generated by the corresponding ØL system. That is, $L(G) \cap \Delta^*$ is the language of the EOL system. If in fact $\Delta = \Sigma$, then $L(K) = L(G)$ and K is said to be full.

Any language which is finite is an EØL language. Given the finite language $L = \{a_1, a_2, \ldots, a_n\}$, $n>Ø$, and a symbol S (which is not in Σ), the EØL system $< \Sigma \cup \{S\}, P, S, \Sigma >$ with productions

$$P = \{ S \rightarrow a_i \mid 1 <= i <= n \} \cup$$

$$\{ x \rightarrow x \mid x \text{ is in } \Sigma \}$$

is an EOL system which generates the language L.

Given any EØL system $G = < \Sigma, P, w, \Delta >$, it is possible to determine if any arbitrary sentence x is in L(G). Since it is possible to determine if x is in G if G is full, it is also possible to determine if x is generated and all symbols of x are in Δ.

Locally Catenative Sequences and Recurrence Formulas

Certain ØL systems exhibit developmental patterns with periodically repeating structures. Two main types of patterns are seen: recursive sequences (left and right) and locally catenative sequences, abbreviated LCS. Since all recurrence sequences and LCS can be expressed as PDØL systems, this chapter will deal with them in this form.

Recurrence sequences have the property that the entire structure at one stage is repeated after a time at one end of a subsequent structure (at the right end for right recurrency, and the left for left recurrency). An example of a right recurrent model is the PDØL system

$G = \langle \sum, P, a \rangle$

$\sum = \{a,b,c,d,e,f,g,h,i,j,k\}$

P = { a -> bc, b -> e, c -> g, d -> h,
e -> e, f -> di, g -> da, h -> f,
i -> jk, j -> e, k -> f }

The following set of strings is obtained:

sØ	a
s1	bc
s2	eg
s3	eda
s4	ehbc
s5	efgh
s6	edieda
s7	ehjkehbc
	(etc)

The first three strings (sØ-s2) do not exhibit recurrence. Beginning with s3, however, the right-hand side of the structure is the entire structure of three stages

previous. It is always preceeded by some word over the alphabet.

LCS's consist of systems in which two or more previously-encountered strings form together a subsequent string. An example was given in chapter 1 of the morphogenesis of a compound leaf. After a certain stage, the entire structure can be given as a formula (locally catenative formula or LCF) which concatenates strings which have already been derived. This eliminates the necessity of producing each individual string by parallel rewriting after a certain stage. The first few strings must either be stated explicitly, or expressed in terms of alphabet, axiom and productions.

Locally Catenative Sequences

The most readily recognizable examples of locally catenative sequences in nature come from compound structures (such as leaves and flowering structures) and some branching structures. It has been pointed out (Rozenberg and Lindenmayer, 1971) that a cellular filament which consists of cells which are dividing at different rates whose daughters require k and m time units to divide, will grow according to the formula

$$a_n = a_{n-k}\, a_{n-m}$$

The following definitions will be needed in subsequent discussions.

Let $H = \langle \sum, P, w \rangle$ be a DØL system, and G be its underlying scheme. The sequence generated by H (denoted $E(H) = t$) is an infinite sequence of strings $a_\emptyset, a_1, a_2, \ldots$ such that $a_\emptyset$ = the axiom and for every non-negative integer i, $a_i \Rightarrow a_{i+1}$ in H. The trace of H (O(H)) is an infinite sequence of subsets of E (EØ, E1, E2, ...) such that Ei=the set of all different symbols which appear in a_i.

If $G = \langle \sum, P, w \rangle$ is a DØL system where O(G) = EØ,E1,... then G is called quasi-reduced if every letter of the alphabet is in some word derived from the axiom, and reduced if every letter is in infinitely many words. Since quasi-reduced systems contain no superfluous letters in the alphabet, all systems considered here will be quasi-reduced.

Let $\sum$ be an alphabet and $t = a_\emptyset, a_1, \ldots$ an infinite sequence of words over $\sum$. t is locally catenative if there exist integers n (n>Ø) and k (k>1), and a sequence of integers (possibly with repetitions, and not necessarily in ascending or descending order) $i_1, \ldots, i_k$ with each having a value between 1 and n inclusive, such that for every j>=n,

$$a_j = a_{j-i_1} a_{j-i_2} \ldots a_{j-i_k}$$

Then t is a $\langle i_1, \ldots, i_k \rangle$ locally catenative sequence with cut n and width k. That is, the first string to exhibit the

property of local catenation is the nth one, and it is formed by concatenation of k previous strings. $\langle i_1,\ldots,i_k \rangle$ is a locally catenative formula.

For any DØL system G where $E(G) = a_\emptyset, a_1, \ldots$, if for some integer $n>\emptyset$ there exist $k>1$ and integers (each $>\emptyset$ and $<=n$) $i1,..,ik$, such that $a_n = a_{n-i1} \ldots a_{n-ik}$, then for every $m>=n$, a_m follows the same LCF. That is, once locally catenative, always locally catenative and by the same formula.

Let $G = \langle \Sigma, P, w \rangle$ be a PDOL system with $E(G) = a_\emptyset, a_1, \ldots$ and for $j>=\emptyset$,

$$a_j = c_1^{(j)} \ldots c_{l_j}^{(j)}, \quad c_p^{(j)} \in \Sigma \quad \text{for } 1<=p<=l_j$$

We say that a_n, for some integer $n >\emptyset$, is covered by the axiom if and only if there exist $k>1$,

$$i_1,..,i_k \in \{1,2,3\ldots\},\ 1<=i_1<i_2\ldots<i_k=l_n$$

such that

$$a_n = c_1^{(n)} \ldots c_{i_1}^{(n)} c_{i_1+1}^{(n)} \ldots c_{i_2}^{(n)} \ldots c_{i_{k-1}}^{(n)} \ldots c_{i_k}^{(n)}$$

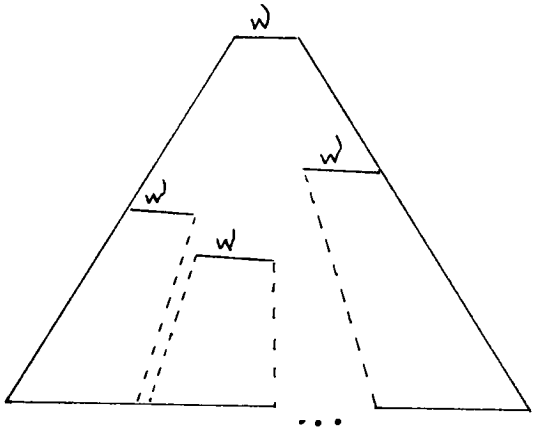
and, for each m in $\{i_1,..i_k\}$, there is an occurrence of the axiom in some a_j, $\emptyset<j<n$, such that

$$c_1^{(n)} \ldots c_{i_m}^{(n)}$$

is the whole subword of a_n derived from this occurrence of the axiom.

In other words, a_n is covered by the axiom if every occurrence of every letter in string a_n $(n>1)$ is derived from some occurrence of the axiom in some earlier string of $\{E(G) - a_\emptyset\}$, with no overlapping. Not all occurrences of the axiom must be in the same string, nor must each occurrence be in a different string.

This can be illustrated as:



Obviously, if a_n is covered, so are all subsequent strings derived from it.

If G is a PDØL system such that E(G) is covered by its axiom, then G is a locally catenative system.

Assume that $E(G) = a_\emptyset, a_1, \ldots$ is covered by the axiom. So for some $n>1$, a_n is covered by w. For $j>=\emptyset$, let $a_j = c_1 \ldots c_r$ for some $r >=1$, $c_r \in \Sigma$ for $1<= r <= r_j$. a_j can be written as a_n

$a_n = b_1 \ldots b_k$ for some $1<=k<=l$, $b_1,\ldots,b_k \in \sum^+$ where $a_n \Rightarrow b_1,\ldots,a_n \Rightarrow b_k$ for some integers

$q_1,\ldots,q_k \in \{1,2,3,\ldots,n-1\}$

Thus $b_1 = a_{q_1}$, $b_2 = a_{q_2}$, $\ldots$, $b_k = a_{q_k}$ and so

$$a_n = a_{q_1} \ldots a_{q_k} = a_{n-(n-q_1)} \ldots a_{n-(n-q_k)}$$

Hence, for every $j>=n$,

$$a_j = a_{j-(n-q_1)} \ldots a_{j-(n-q_k)}$$

Not all PDØL systems are locally catenative, but locally catenative systems form a subset of the family of PDØL systems. See Appendix C for an algorithm to generate a PDØL system given any LCS.

Recurrence Systems

In Chapter 1, a detailed example of a recurrence system was given and illustrated. This example of a branching filamentous organism will be referred to extensively in this section, but will not be described again.

Let $\sum$ be an alphabet and $t = A_\emptyset, A_1, \ldots$ an infinite sequence of words over the alphabet. t is right recurrent if and only if there exist integers n ($n>=\emptyset$) and p ($p>\emptyset$) such

that for every $j>=n$, A_j is a suffix of A_{j+p}. Any such n is called a cut of t and any such p is a period of t. That is, the first string of the sequence to demonstrate recurrence is A_n, and $A_{i+p} = BA_j$ for $B \in \sum+$. Left recurrence is similarly defined, except $A_{j+p} = A_j B$ (A_j is a prefix rather than a suffix).

It is possible to prove (Rozenberg and Lindenmayer, 1971) that if in a DØL sequence some string appears as a prefix (suffix) of a subsequent string, then this property holds for all subsequent strings. That is, similar to a LCS, once recurrent, always recurrent and by the same formula.

A formula that can be used to describe the example of recurrence given in Chapter 1 is:

$$W_n = B_{n-1}\ A_{n-1}$$

$$A_n = cc(W_{n-5})A_{n-1}$$

$$B_n = cc(F_{n-2})cc(F_{n-3})cc(F_{n-4})$$

$$F_n = F_{n-1}\ c$$

Where W refers to the whole organism, A is the apical section, B is the basal section and F is an unbranched filament.

In this formula, all cells are designated by one symbol, c. The parentheses are also constant and designate, as before, a branch.

In order to designate the "whole" organism, a number of formulas were required. In order to describe the nth string, reference had to be made to previous strings. Note that the set of formulas

$$W_n = B_n\ A_n$$

$$A_n = cc(W_{n-6})A_{n-1}$$

$$B_n = cc(c_{n-3})cc(c_{n-4})cc(c_{n-5})$$

can also describe the system, but W_n does not refer directly to earlier strings, so this does not conform to the form needed for a recurrence formula.

In order to use this formula, some of the strings must be given as axioms. The "whole organism" must be defined for stages 1 through 5 in this example in order for the formula to be solvable for later stages (due to the reference to W_{n-5}). Similarly, A and B must be defined, but only for the fifth stage as it is never necessary to refer back more than one stage for these components. F must be defined for stages 2 through 5.

The depth of a system is the maximum number of axioms that must be given for any component of the formula, in this case 5. The width is the number of component parts in the formula, in this case four (W,A,B and F). The entire system can be represented in tabular form as:

Stage	W	A	B	F
1	c			
2	cc			c^2
3	c^4			c^3
4	$cc(c)^4$			c^4
5	$cc(c^2)cc(c)c^4$	c^4	$cc(c^3)cc(c^2)cc(c)$	c^5

This represents all components that must be defined in order to utilize the formula for definition of later strings. For example, at stage 6, the formula is:

$$W_6 = B_5 A_5 = cc(c^3)cc(c^2)cc(c)c^4$$

Note that each entry at each position of the table represents an element of a set. There may be more than one element at any position; this example is limited to systems for which each set has at most one element.

Since A and B are not referred to for stages 1 through 4, and F not for stage 1, they are denoted by the empty set, {}. If they had been referred to, but would not have contributed anything to the string, they would have been denoted with {Λ} instead.

If N is the set of positive integers, for any x in N, N^x denotes the set of all integers between 1 and x, inclusive. A recurrence system can be formally defined as a 6-tuplet

$S = < \Sigma , O, d, A, F, w >$

1) Σ is the alphabet

2) O is the index set N^w where w = width of the system

3) d is a positive integer, the depth of the system

4) A is a function associating with each <x,y> in $O \times N^d$ a finite set $A_{x,y}$ of axioms, $A_{x,y} \subset \Sigma^*$

5) F is a function, associating with each x in O a non-empty, finite set F_x such that F_x is a subset of $((O \times N^x) \cup \Sigma)^*$. These are notations for the set of recurrence formulas, and may combine previous strings with members of the alphabet.

6) $w \in 0$ is the "distinguished index", indicating which of the components of the formula represents the "whole organism".

A recurrence system with width w and depth d is called a (w,d) recurrence system. The language generated by a recurrence system is a recurrence language.

The example used is a (4,5) recurrence system defined as:

$$S = \langle \{c,(,)\}, N^4,5,A,F,1 \rangle$$

$$A_{1,1} = \{c\} \qquad A_{1,2} = \{cc\} \qquad A_{1,3} = \{ccc\}$$

$$A_{1,4} = (cc(c)cccc\} \qquad A_{1,5} = \{cc(cc)cc(c)cccc\}$$

$$A_{2,1} = A_{2,2} = A_{2,3} = A_{2,4} = \emptyset$$

$$A_{2,5} = \{cccc\} \qquad A_{3,1} = A_{3,2} = A_{3,3} = A_{3,4} = \emptyset$$

$$A_{3,5} = \{cc(ccc)cc(cc)cc(c)\}$$

$$A_{4,1} = \emptyset$$

$$A_{4,2} = \{cc\} \qquad A_{4,3} = \{ccc\} \qquad A_{4,4} = \{cccc\} \qquad A_{4,5} = \{ccccc\}$$

$$F_1 = \{\langle 3,1\rangle \langle 2,1\rangle \}$$

$$F_2 = \{ cc(\langle 1,5\rangle)\langle 2,1\rangle \}$$

$$F_3 = \{ cc(\langle 4,2\rangle)cc(\langle 4,3\rangle)cc(\langle 4,4\rangle) \}$$

$$F_4 = \{ \langle 4,1\rangle c \}$$

F_1 represents the formula for the component in column one. At stage N it is the concatenation of the element in column 3 at stage (n-1), <3,1>, and the element at column 2 at stage (n-1), <2,1>. That is,

$$W_n = A_{n-1} \; B_{n-1}$$

Formula F4 is the concatenation of the fourth component at stage n-1 and the constant c. That is,

$$F_n = F_{n-1} \; c$$

In this way, a recurrence system may be represented without the necessity of specifying the production rules. Compare the difference between the 6-tuple given, and the production rules necessary to describe the same system (provided in Chapter 1).

There exists an algorithm (Herman, Lindenmayer and Rozenberg, 1971) which allows any recurrence system to be expressed as an equivalent recurrence system of depth one.

A recurrence system is deterministic if the following two conditions are met:

1) the maximum number of elements in all sets of axioms is one (there is no more than one entry in any position of the table for each axiom)

2) for each component of the formula, there is no more than one possible configuration

Therefore, if S is a deterministic recurrence system, each axiom has exactly one element, or is empty. The example

used is deterministic.

Recurrence systems are generalizations of locally catenative systems. Locally catenative systems are deterministic recurrence systems of width one, in which none of the formulas contains any elements of the alphabet. The family of locally catenative systems form a proper subset of recurrence systems. The example used demonstrates that the inclusion is proper.

An example of a system that can be expressed either way is:

S is a recurrence system such that

$S = \langle \{a,b,c,d,e,f\}, 1, 3, A, F, 1 \rangle$ where

$A_{1,1} = \{ab\}$

$A_{1,2} = \{ef\}$

$A_{1,3} = \{cd\}$

$F_1 = \{ \langle w, 2 \rangle \langle w, 3 \rangle \}$

This can be equivalently expressed as a LCS

$a_{\emptyset} = ab$

$a_1 = ef$

$a_2 = cd$

$a_n = a_{n-2} a_{n-3}$ for $n \geq 3$

TØL Systems

Definitions

Table ØL (TØL) systems were first suggested by Rozenberg (1973a) as a way of accounting for environmental changes in development.

A TØL scheme is a pair $S = \langle \sum, P \rangle$ where $\sum$ is the alphabet and P is a finite, non-empty set. Each element p of P, called a table, is a finite non-empty subset of productions in S. Obviously, a ØL scheme is a TØL scheme which has only one element in P. Let $S = \langle \sum, P \rangle$ be a TØL scheme, and $x = a_1 \ldots a_m$ $(m>=\emptyset)$ such that $a_j \in \sum$ for $j=1,\ldots,m$ and $y \in \sum^*$. Then we say that x directly derives y in S ($x \Rightarrow y$) if and only if

$$(\exists p)_P (\exists A_1,\ldots,A_m)_{\sum^*} (a_1 \xrightarrow{p} A_1, \ldots, a_m \xrightarrow{p} A_m \text{ and } y = A_1 \ldots A_m)$$

Note that all productions for one derivation step must be contained in the same table.

S is propagating if, for every p in P, there is no production of the form $a \rightarrow \Lambda$ for any a in the alphabet. S is deterministic if, for every p in P and for every a in the alphabet, there exists exactly one A in $\sum^*$ such that $a \rightarrow A$. If both of these conditions are true, it is a PDTØL scheme.

A TØL system is a triple $G = \langle \sum, P, w \rangle$ such that $S = \langle \sum, P \rangle$ is a TØL scheme and w is a word over the alphabet (the axiom of G). G is propagating (deterministic) if and

only if S is propagating (deterministic).

Examples and Concepts

An example of a DTØL system is

$< \{a,b,c,d\}, \{p1,p2\}, babab >$

$p1 = \{ a \rightarrow c^3, b \rightarrow b, c \rightarrow c, d \rightarrow d \}$

$p2 = \{ a \rightarrow d^4, b \rightarrow b, c \rightarrow c, d \rightarrow d \}$

The language generated by this system is:

$\{babab, bc^3 bc^3 b, bd^4 bd^4 b\}$.

This is a finite language which can not be generated by a ØL system. TØL systems can obviously generate more languages than ØL systems, but they still can not generate all finite languages. {a. aaa}, for example, is not a TØL language.

The TØL system $G = <\{a,b\}, \{p1,p2\}, a>$ where

$p1 = \{ a \rightarrow a, a \rightarrow a^2, b \rightarrow b\}$

$p2 = \{ a \rightarrow b, b \rightarrow b \}$

generates the infinite language

$L(G) = \{ a^n \mid n>=1 \} U \{ b^n \mid n>=1 \}$

This is an example of an infinite TØL language which can not be generated by a ØL system.

Given any TØL language L, it is possible to construct a TØL language H such that H = L U {Λ}. This is also not the case with ØL languages.

Tables are useful in imposing control on the application of productions: if two productions should not be used in the same derivation step, then they should be in separate tables.

As with ØL languages, there is no algorithm which can decide if the language generated by any arbitrary TØL language is equivalent to the language generated by any other arbitrary TØL system.

Similarly, the membership problem for TØL systems is solvable. Given a TØL system $G = \langle \sum, P, w \rangle$, it is possible to determine if any given x in $\sum^*$ is generated by L(G). If G is a PTØL system, the reasoning is analogous to the situation for a PØL system. If the TØL system is non-propagating, however, it is first necessary to generate from this system G a new system, H, such that H is propagating. There is an algorithm which essentially generates a system H such that $L(H) = L(G) - \{\Lambda\}$. Since it is possible to determine if G generates $\{\Lambda\}$, the reasoning behind the membership problem for G is the same as for any PTØL system. (See Herman and Rozenberg, 1975, for details on how to construct a system which will generate this system H).

ØL systems exhibit a so-called "linear derivation property" (for every word x in a ØL system G, it is possible to find a derivation of x in G where the length of every word in the trace of x is not greater than $C(|x|+1)$ where C is a constant dependent on G). This in not true in every TØL

system, hence the necessity for the algorithm cited to solve the membership problem (Salomaa, 1980). Despite the existence of this algorithm, erasing does increase the power of TØL systems. There are TØL languages which cannot be generated by PTØL systems.

The TØL system G = < {a,b,c}, {p1,p2,p3}, ba > where

$p1 = \{ a \rightarrow a^2, b \rightarrow b, c \rightarrow c \}$

$p2 = \{ a \rightarrow a, b \rightarrow b, c \rightarrow c^3 \}$

$p3 = \{ a \rightarrow \Lambda, b \rightarrow bc, c \rightarrow \Lambda \}$

generates the language $L = \{ba^{2^n} \mid n>=0 \} \cup \{bc^{3^n} \mid n>=0\}$

There is no PTØL system which can generate this language. Either ba or bc would have to be the axiom. If ba is the axiom, a table must exist such that ba => bc. That table must therefore contain the productions b -> b and a -> c. Since baa must also be in the language, bcc would also have to be in the language using the table required to generate bc. But bcc is not in L, so it is not possible to generate this language using a PTØL system with axiom ba. The situation is analogous if the axiom were bc. baaa would be eventually generated, and baaa is not in L.

The complexity of TØL systems is related to the number of tables that are in P. It is possible to increase the complexity of a TØL language by adding one or more tables (Rozenberg, 1973a).

Extension languages

Extended TØL (ETØL) systems are derived from TØL systems by adding the target alphabet, Δ. An ETØL system is a 4-tuple,

$G = \langle \sum, P, w, \Delta \rangle$ where $S = \langle \sum, P, w \rangle$ is a TØL system

and Δ is a subset of $\sum$.

The language generated by G is $L(G) = L(S) \cap \Delta^*$. The symbols of Δ represent the terminal symbols of the system. ETØL systems are deterministic and/or propagating if and only if the underlying TØL system is deterministic and/or propagating respectively.

Let $G = \langle \sum, P, w, \Delta \rangle$ be an EDTØL system such that

$\sum = \{S,A,B,X,a,b\}$

$\Delta = \{X,a,b\}$

$w = SSS$

p1 = { S->A, A->Sa, B->X, a->a, b->b, X->X }

p2 = { S->B, A->X, B->Sb, a->a, b->b, X->X }

L(G) is then { XyXyXy | $y \in \{a,b\}^*$ }. Note that this can not be generated by an EØL system.

An ETØL system may have finitely many set of production rules. However, for any ETØL system G, it is possible to construct an equivalent ETØL system H such that H has exactly two tables. This is in contrast to TØL systems, which can produce more complicated languages by adding one or more tables to the existing system. The algorithm to

produce H is actually quite simple.

Let G = < $\sum$, P, w, Δ > be an ETØL system with P = {p1,...,pr}.

Let H = < { $\sum$ U [a,i] | a $\in$ $\sum$ and 1 <= i <= r}, {h1,h2},w,Δ>

h1 is a finite substitution on $\sum^*$ defined as:

$[a,i] \xrightarrow[h1]{} [a,i+1]$ for a in $\sum$, 1 <= i <= r-1

$a \xrightarrow[h1]{} [a,1]$ for a in $\sum$

$[a,r] \xrightarrow[h1]{} [a,1]$ for a in $\sum$

h2 is a finite substitution on $\sum^*$ defined as:

$a \xrightarrow[h2]{} a$ for a in $\sum$

$[a,i] \xrightarrow[h2]{} x$ such that $a \xrightarrow[pi]{} x$ for a $\in$ $\sum$, x $\in$ $\sum^*$, 1 <= i <= r

Even though it is possible to express any ETØL system with two tables, it would usually defeat the purpose of having tables, as tables are used to group productions into logically related sets. This also eludes the biological significance of the model.

IL Systems (<m,n> Systems)

Definitions

IL systems (interactive L systems) allow for cellular interactions and communication from cells on both sides of a given cell. In its earliest form, Lindenmayer (1968a,b) allowed for communication from at most one cell on either side, the model being linear. This model is still linear, but allows for communication to a depth of more than one cell.

Let m and n be non-negative integers. An <m,n> system is a construct

$G = \langle \sum, P, g, w \rangle$ such that

$\sum$ is the alphabet of G

w is an element of $\sum^*$ (the axiom)

g is an element not in the alphabet (called the marker, indicating the "outside" environment)

P is the set of productions of G

$$P \subset (\sum \cup \{g\})^m \times \sum \times (\sum \cup \{g\})^n \times \sum^*$$

Any single production in P has the form $\langle X, a, Y, B \rangle$ such that if X is the left context for a and Y is the right context of a, then a is rewritten as B. The notation

$$\langle X, a, Y \rangle \rightarrow B$$

is an equivalent expression.

X and Y must follow certain restrictions. If $\langle X,a,Y \rangle \rightarrow B$ then:

a) if $X = xgy$ for some x,y in $(\sum \cup \{g\})^*$, then

x is in {g}*

b) if Y = qgz for some q,z in ($\sum$ U {g})*, then z is in {g}*

Notice that |X| = m and |Y| = n.

Any <m,n> system is an IL system. If n = m = Ø, then it is a <Ø,Ø> system, equivalent to a ØL system. If m = Ø or n = Ø (m not equal to n), then it is a one-sided IL system, sometimes called a 1L system. A <Ø,n> system is right sided, an <m,Ø> system is left sided. If n and m are both greater than Ø, it is a two-sided (2L) system.

For any non-negative integer k, if $x = a_1 a_2 \ldots a_i$ ($a \in \{ \sum U \{g\} \}$) is a word such that i >= k, then

$suf_k(x) = a_{i-k+1} a_{i-k+2} \ldots a_i$ (the last k symbols of string x).

Similarly, $pref_k(x)$ is the first k elements of string x.

If $G = \langle \sum, P, g, w \rangle$ is an <m,n> system, $x = a_1 \ldots a_k$ in $\sum^*$ and y in $\sum^*$, then x directly derives y in G (x => y) if and only if:

$$\langle g^m, a_1, pref_n(a_2 \ldots a_k g^n) \rangle \rightarrow A_1$$

$$\langle suf_m(g^m a_1), a_2, pref_n(a_3 \ldots a_k g^n) \rangle \rightarrow A_2$$

$$\vdots$$

$$\langle suf_m(g^m a_1 \ldots a_{k-1}), a_k, g^n \rangle \rightarrow A_k$$

for some $A_1, A_2, \ldots, A_k \in \sum^*$ such that $y = A_1 \ldots A_k$.

If G is an <m,n> system, L(G) is all those strings x (x in $\sum^*$) which can be derived from the axiom in zero or more steps. Note that the marker is excluded from the language, and the axiom is considered to be part of the language.

Examples and concepts

When expressing a string in an <m,n> system, it is necessary to have at least m+n+1 symbols represented. A string may be padded left and right with as many occurrences of the marker as desired.

Consider the <2,1> system G = $\langle \sum, P, g, e \rangle$ where

$\sum$ = {a,b,c,d,e}

P = { <gb, a, b> -> babcc
<gb, a, d> -> badccccc
<bb, a, b> -> bbabccc
<x, b, y> -> Λ for every $x \in (\sum \cup \{g\})^2$ and $y \in (\sum \cup \{g\})$
<x, c, y> -> c (x and y defined above)
<x, d, y> -> Λ (x and y defined above)
<gg, e, g > -> babcc
<gg,e.g> -> bbabccc
<gg, e, g> -> badccccc }

The axiom is e. Since this is a <2,1> system, the first string to be rewritten is denoted ggeg. A sample derivation is:

ggeg
ggbabccg
ggbabccccg
ggbabcccccg
ggbabccccccg
(etc.)

$L(G) = \{babc^{2n} \mid n>=1\} \cup \{bbabc^{3n} \mid n>=1\} \cup \{badc^{5n} \mid n>=1\} \cup \{e\}.$

An IL system $G = \langle \Sigma, P, g, w \rangle$ is propagating if and only if w is not Λ and for every $\langle x,a,y,A \rangle$ in P, A is not Λ. Otherwise, G is non-propagating.

For the purposes of defining determinism, consider the environmental marker to be included in the alphabet, and assume that all production rules adhere to the previously-given restrictions on the placement of the marker. G is deterministic if and only if P is a mapping of the set

$$\bigcup_{i=\emptyset}^{2} \Sigma^i \times \Sigma \text{ into } \Sigma^* \quad (\text{where } \Sigma^2 = \Sigma \times \Sigma)$$

So the set of productions is a subset of $\Sigma^{i+1} \times \Sigma^*$ such that for every u in Σ^{1+1} there exists exactly one x in Σ^* such that $\langle u,x \rangle$ is in P.

If $S = \langle \Sigma, P, g \rangle$ is a DiL scheme ($i \in \{\emptyset,1,2\}$), then the mapping of Σ^3 into Σ^* is defined as follows:
For a,b,c in Σ , X in Σ^*

$\langle b,x \rangle \in P$ if S is a DØL scheme

$\langle a,b,X) \in P$ if S is a D1L scheme

$\langle a,b,c,X \rangle \in P$ if S is a D2L scheme.

The following two notations are considered equivalent:

$\langle - , a, b \rangle \rightarrow X$

$\langle \Lambda , a, b \rangle \rightarrow X$

Redefining <m,n> systems

It is possible to describe every <m,n> system as an equivalent <1,k> system or a <k,1> system where k is a non-negative integer. When expressed in this way, the system is said to be in normal form.

For all non-negative integers m1,m2,n1,n2, if m1 <= m2 and n1 <= n2, then

$$F(\langle m1,n1\rangle) \subseteq F(\langle m2,n2\rangle)$$

In addition,

a) for every m >= 2 and n >= 0,
$F(\langle m,n\rangle) \subseteq F(\langle m-1,n+1\rangle)$

b) for every m>= 0 and n >= 2,
$F(\langle m,n\rangle) \subseteq F(\langle m+1,n-1\rangle)$

(See Herman and Rozenberg, 1975, for proof and an algorithm to allow this).

The logic behind the algorithm which generates such a transformation as in a) is as follows. If a string $a_1 \ldots a_n$ (n>0) derives a string $A_1 \ldots A_n$ (where $a_1,\ldots,a_n$ derive $A_1,\ldots,A_n$ respectively) then in the new grammar, a_1 simulates a_n in the sense that it derives A_n. a_{n-1} derives Λ. a_n simulates a_{n-1}, and so on. a_{n-2} must simulate the effect of rewriting a_{n-1} and a_1 simultaneously. There is an effective "shifting" of the productions. It is also necessary to be able to detect the leftmost and rightmost symbols a_1 and a_n.

By applying this algorithm repeatedly, it is possible to generate an equivalent normal-form IL system for any <m,n> system where m and n follow the limitations specified. This indicates that, in 2L systems, it is not the distribution of the context which is important, but rather the total context. This says that, for m>=1, n>=1,

$$F(\langle 1,m+n-1\rangle) = F(\langle m,n\rangle) \quad \text{and}$$

$F(\langle m+n-1,1\rangle) = F(\langle m,n\rangle)$

For two 2L systems, $\langle m1,n1\rangle$ and $\langle m2,n2\rangle$,

$F(\langle m1,n1\rangle) \subsetneq F(\langle m2,n2\rangle)$ iff $m1 + n1 < m2 + n2$

$F(\langle m1,n1\rangle) = F(\langle m2,n2\rangle)$ iff $m1 + n1 = m2 + n2$

It is also true that for all non-negative integers m1,n2,n1,n2, if $m1 \leq m2$ and $n1 \leq n2$ then

$F(\langle m1,n1\rangle) \subset F(\langle m2,n2\rangle)$

One-sided IL systems

Some of the results presented so far refer to two sided IL systems only. The character of any IL system can also be changed when considering one sided systems in particular.

There are some 1L languages which can be defined either as a $\langle 1,0\rangle$ or a $\langle 0,1\rangle$ system. For example, $L = \{a^n b^n \mid n>0\}$ can be either $\langle 1,0\rangle$ or $\langle 0,1\rangle$. The axiom and alphabet for either are = {a,b}, axiom = ab. For the former, the production rules would be

$\{ \langle -,b,b\rangle \rightarrow b$
$\langle -,b,g\rangle \rightarrow b$
$\langle -,a,b\rangle \rightarrow aab$
$\langle -,a,a\rangle \rightarrow a \}$

For the latter, the production rules would be

$\{ \langle a,a,-\rangle \rightarrow a$
$\langle g,a,-\rangle \rightarrow a$
$\langle b,b,-\rangle \rightarrow b$
$\langle a,b,-\rangle \rightarrow abb \}$

There are also languages which can be described by a $\langle 1,0\rangle$ system, but not a $\langle 0,n\rangle$ system for any n.

Consider the <1,Ø> system < {a,b,c},P,g,c >,

```
P = { <g,c,-> -> a,
      <g,c,-> -> baa,
      <g,a,-> -> aa,
      <b,a,-> -> a,
      <a,a,-> -> aa,
      <g,b,-> -> b       }
```

This generates the language $\{c\} \cup \{a^{2^n} \mid n>=Ø\} \cup \{ba^{2^n+1} \mid n>=Ø\}$

Explanation of why this language can not be generated by a <Ø,n> system is found in Appendix F.

From these examples it should be clear that, for every pair of positive integers m and n, F(<m,n>) and F(<Ø,n>) are incomparable but not disjoint.

Extended IL systems

An extended IL (EIL) system is a 5-tuple,

$G = < \sum, P, g, W, \triangle >$ where
$G' = < \sum, P, g, W >$ is an IL system and
$\triangle$ is a subset of $\sum$.

The language generated by G is defined as $L(G') \cap \sum^*$.

EIL systems have the interesting property that for every grammar G = <Vn, Vt, P, S>, there exists an E<1,Ø> system H such that L(G) = L(H). (The proof for this was repeated in Herman and Rozenberg, 1975, taken from Rozenberg, 1972b. It is given in Appendix G).

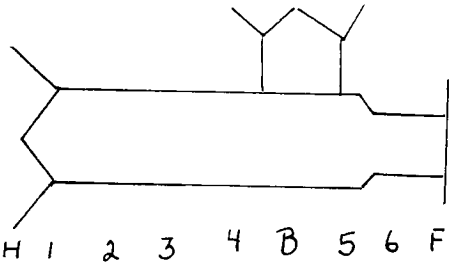
It has been shown that L systems have a sound basis in both biological and mathematical applications. The grammars described are capable of generating languages, and languages which indicate the morphology and development of living organisms. Even though the linear array model has limitations in the types of patterns that it can generate, and must generally represent a model that is essentially two dimensional, it is adequate to show many three dimensional phenomena.

To this point, the biological significance of the L system used to generate any language has been concentrated on whether the model has or lacks cellular interactions, and what the final outcome may be. The results are phenomological rather than physiological. In practice, it is often impossible to determine what molecular basis any given phenomenon may have. Quantification has always been a difficult topic of experimental biology. No theory can be proved or disproved by any given experiment when there are a variety of unknown parameters. It is possible, however, to use a theoretical model to decide if a given hypothesis (based on specific assumptions whose validity can not be verified) should be accepted or discarded. In particular, CELIA has been used for hypothesis testing in various systems. To demonstrate the use of L systems from a qualitative point of view, a specific organism, hydra, (which has been extensively investigated in biological context, and has been simulated using CELIA) has been chosen for discussion.

Information on hydra as well as experimental results referred to in grafting experiments is taken from Sacks (1978).

Hydra is a hollow tube-shaped aquatic organism, up to one half inch in length. At the distal end is the head region which consists of a hypostome (mouth-like structure) and a whorl of tentacles. The animal is closed at the proximal end, which is the peduncle. Hydra has been used as a model for growth and regeneration because of its capability to regenerate lost parts.

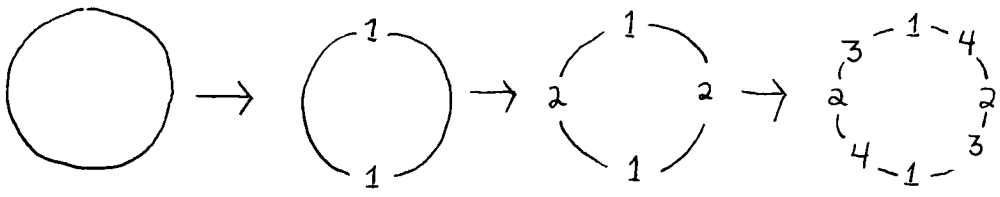
A hydra can be divided into distinct regions, as illustrated below:



H and F stand for the head and foot regions, respectively. B is the budding zone, that level on the column where buds form, budding being the most common means of reproduction in hydra. In cross section, the animal would be circular.

One of the areas that has been studied in hydra is that of tentacle regeneration. If the head region is transected, a new head will grow and produce new tentacles. In one species of hydra, the order in which the tentacles regenerate is regulated; the mechanism behind regulation is unknown.

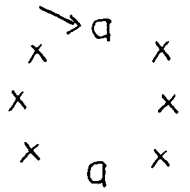
Seen in cross section, the order in which the tentacles regenerate is as follows:



It is hypothesized that there exists a dorsal-ventral axis in this species, and the first tentacle to actually appear is on the axis with the budding zone. The first set of tentacles appear at diametrically opposite points, the second set appears approximately half way between the first two tentacles. The third and fourth sets then fill in the remaining areas. The appearance of the tentacles is a function of time, the first two sets being under finer control than the remaining tentacles. Although each number in the diagram is supposed to represent one tentacle (or actually the area on the hypostome from which one tentacle arises), it is not necessarily the case that exactly eight tentacles will appear. Again, the underlying mechanism for determining tentacle number is unknown, although interactions involving the existence of both inhibitory and activating substances acting antagonistically constitute one theory.

This can be represented as an IL system, although with a slight difference in notation. The sentence will be represented in circular form, and the symbols will be scanned clockwise, rather than left to right. It is also assumed that one derivation step represents the rewriting of

only one circumference of the animal. The "beginning point" for scanning will be indicated by an arrow. The results would be the same regardless of the direction of scanning and choice of beginning point.



represents the transected hydra. The points labelled "a" represent the areas distinguished by their position on the dorsal-ventral axis. When an area begins generating a tentacle, it is rewritten as "t". With the axiom being the transected hydra, alphabet {a,x,t}, the following <2,1> system would produce the desired results.

Let A = {t,x}+ where |A| = 2

Let B = {t,x} where |B| = 1

<A,a,B> -> t
<tx,x,B> -> t
<xx,x,B> -> x
<tx,t,b> -> t
<xt,x,x> -> x
<xx,t,B> -> t

This model assumes the existence of both inhibitory and activating substances which diffuse in a lateral direction (rather than disto-proximal). The activator, according to this model, would most likely come from tentacle-producing areas, while the inhibitor could come from non-tentacle producing areas (represented by x), or is present in constant amounts and counteracted when the activator reaches a certain level. Once activation begins, it is irreversible.

This neither proves nor disproves the existence of any activator or inhibitor, not the possible source(s) of either substance. But, by making certain assumptions regarding their existence and physical properties, it would be possible to simulate their actions, compare these results with experimental data and, by chi-squared analysis, determine if this hypothesis is one which should be discarded, or further investigated.

The head region of hydra is known to be a dominant region. This means that if this region is lost, it is the first to be regenerated and once its formation is initiated, its regeneration is an autonomous process not influenced by any other region. The presence of a dominant region inhibits the formation of other such regions.

It is known that there exist head activator and head inhibitor substances, believed to be produced by nerve cells. The antagonistic effect of these substances determines the regeneration of the head region.

A way of showing the effect of the inductive ability after the loss of a head region and inhibitory effect of the head region is to graft together regions of hydra that are not normally juxtaposed. A normal hydra is denoted as H1234B56F. If an H12 piece of one hydra is grafted onto a 1234B56F (also denoted as 1-F) piece of another hydra, it is represented as H12/1-F, the slash denoting the graft border. No structures are formed at the graft border, due to the

presence of the dominant head region, which inhibits the formation of another head. If an animal were 1-F, without grafting the H12 region onto it, a head would form at the distal end. If an H123/1-F graft is made, a head will form at the graft border, indicating a gradient in concentration of head inhibitor.

Since it is theorized that the inhibitors and activators come from nervous tissue, it would be expected that removing nerve cells (possible by chemical treatment) would alter regenerative properties of hydra. If transected, a nerveless hydra demonstrates near-normal regeneration properties. There are differences if normal hydra tissue is grafted to nerveless tissue, or nerveless tissue is grafted to nerveless tissue.

Since this is a biological system and the phenomena represented are not "all or nothing", the model is a stochastic one based on experimental evidence. Percentages and p values from chi-squared analysis will be given where available.

The alphabet consists of the symbols representing a normal hydra (H, B, F, 1-5) and nerveless hydra (H′, B′, F′, 1′-5′). The production rules represent the results of grafting together nerveless and normal hydra tissue at the areas designated (the area being rewritten). There are naturally many other grafting combinations possible; these are designed to show the differences in dominance between

the two types of tissue in the head region only. Each animal is assumed to have at most one graft border and possibly one transection (eg. a 12/1-F graft).

The alphabet was described above.

The production rules are:

```
<1,2,1>   ->  2    (87.4%)
<1',2',1'>  ->  2'H   (80%, p<.001)
<1,2,1'>    ->  2H    (56.7%, p<.001)
<1',2',1>   ->  2'    (84.6%)
<3,1,2>     ->  H1
<3',1',2'> ->  H'1'
<g,1,2>     ->  H1
<g,1',2'>   ->  H'1'
```

Therefore, if the following grafts were made, the results would be according to the production rules.

```
(1)  g121-Fg    =>  H121-F          (inhibit head formation)
(2)  g121'-F'g =>  H12H1'-F'        (repress inhibition)
(3)  g1'2'1'-F'g =>  H'1'2'H'1'-F'  (repress inhibition)
(4)  g1'2'1-Fg =>  H'1'2'1-F        (inhibit head formation)
```

This model represents a theory that nerveless tissue exhibits normal head-formation inhibition ability for transmitting inhibitory information to normal tissue (as seen by comparing 1 and 4) but it does not respond to inhibitory signals sent by normal tissue (as seen in 2 and 3).

CELIA has been used to test hypotheses of similar grafting experiments in hydra using normal tissue only (Herman and Schiff, 1974). In this hydra simulation, assumptions are made regarding the diffusion rates and concentration of inhibitory and activating substances, and how they interacted to induce or repress head formation. The simulated results were compared with experimental data. While

the results could not prove the hypothesis being tested, it could indicate if the hypothesis deviates enough from empirical data to warrant rejecting the hypothesis.

CELIA is based on the theory of L systems. It can be used both to predict pattern formation and for hypothesis testing; both are significant applications of the theory.

Bibliography

Apter, M.J., 1966. Cybernetics and Development. Pergamon Press, Oxford.

Baker, R. and G.T. Herman, 1972. Simulation of organisms using a developmental model, I: Basic Description; II: The Heterocyst Formation Problem in Blue-green Algae. Int. J. Bio-Med. Comput., 3:201-215 and 3:251-267.

Carlyle, J.W., S. Greibach and A. Paz, 1974. A two-dimensional generating system modeling growth by binary cell division. IEEE 15th Annual Symposium on Switching and Automata Theory, 15: 1-12.

Chiaraviglio, L., 1965. Coding machines. J. Theoret. Biol., 8: 130-140.

Culik, K. and J. Opatrny, 1974. Literal homomorphisms of 0L-languages. Proceedings of the 1974 Conference on Biologically Motivated Automata Theory. 50-53.

Dalen, D. van, 1971. A note on some systems of Lindenmayer. Math. Systems Theory, 5:128-140.

Downey, P.J., 1974. Developmental systems and recursion schemes. Proceedings of the 1974 Conference on Biologically Motivated Automata Theory. 54-58.

Frijters, D., 1974. Growth and flowering of Aster: basically a TIL system. Working Report, Theoretical Biology Group, Utrecht, Netherlands.

Herman, G.T., 1970. The role of environment in developmental models. J. Theoret. Biol. 29: 329-341.

Herman, G.T., A. Lindenmayer and G. Rozenberg, 1974. Description of developmental languages using recurrence systems. Math. Systems Theory, 8: 316-341.

Herman, G.T. and W.H. Liu, 1973. The daughter of CELIA, the French flag and the firing squad. Simulation, 21: 33.

Herman, G.T. and G. Rozenberg, 1975. Developmental Systems and Languages. North Holland/American Elsevier, New York.

Herman, G.T. and G. Schiff, 1974. Simulation of organisms based on L systems. Proceedings of the 1974 Conference on Biologically Motivated Automata Theory. 70-76.

Herman, G.T. and A. Walker, 1975. Context-free languages in biological systems. Internl. J. Comp. Math., 4: 369-392.

Leeuwen, J. van, 1974. The complexity of developmental languages. Proceedings of the 1974 Conference on Biologically Motivated Automata Theory. 63-64.

Lindenmayer, Aristid, 1968a. Mathematical models for cellular interactions in development, Part I. J. Theoret Biol., 18: 280-299.

Lindenmayer, Aristid, 1968b. Mathematical models for cellular interactions in development, Part II. J. Theoret. Biol., 18: 300-315.

Lindenmayer, A., 1971. Developmental systems without cellular interactions, their languages and grammars. J. Theoret. Biol., 30: 455-484.

Lindenmayer, A., 1974. L-systems in their biological context. Proceedings of the 1974 Conference on Biologically Motivated Automata Theory. 65-69.

Lindenmayer, A., 1975a. In: Developmental Systems and Languages (G.T. Herman and G. Rozenberg). North Holland/American Elsevier, New York.

Lindenmayer, A., 1975b. Developmental algorithms for multicellular organisms: a survey of L-systems. J. Theoret. Biol., 54: 3-22.

Lindenmayer, A. and G. Rozenberg, 1972. Developmental systems and languages. Proc. 4th Annual ACM Symp. on Theory of Computing. 214-221.

Lindenmayer, A. and G. Rozenberg, 1976. Automata, Languages, Development. North Holland, Amsterdam.

Liu, W., 1973. A special purpose simulator for developmental biology, Masters Thesis, Dept. of Computer Science, State University of New York at Buffalo.

Mayoh, B., 1974. Multidimensional Lindenmayer organisms. In: L Systems, Lecture Notes in Computer Science (Rozenberg and Salomaa, eds.), 15: 302-326.

Mayoh, B., 1976. Another model for the development of multidemensional organisms. In: Lindenmayer and Rozenberg (eds.), Automata, Languages, Development. 469-485.

Nagel, M., 1976. On a generalization of Lindenmayer systems to labelled graphs. In: Lindenmayer and Rozenberg (eds.), Automata, Languages, Development. 487-508.

Paz, A., 1976. Multidimensional parallel rewriting systems. In: Lindenmayer and Rozenberg (eds.), Automata, Languages, Development. 509-515.

Rosen, R., 1964. Abstract biological systems as sequential machines, I and II. Bull. Math. Biophys., 26: 103 and 239.

Rozenberg, G., 1972a. Direct proofs of the undecidability of the equivalence problem for sentential forms of linear context-free grammars and the equivalence problem for ØL systems. Inform. Processing Letters, 1: 233-235.

Rozenberg, G., 1972b. L-systems with interactions: the hierarchy. Dept. of Comp. Sci., SUNY Buffalo, Departmental Report 28-72.

Rozenberg, G., 1973a. TØL systems and languages. Inform. and Control, 23: 357-381.

Rozenberg, G., 1973b. Extension of tabled ØL systems and languages. Internl. J. Comput. Inform. Sci., 2: 311-334.

Rozenberg, G., 1974. Introduction to L systems. Proceedings of the 1974 Conference on Biologically Motivated Automata Theory. 29-37.

Rozenberg, G. and P. Doucet, 1971. On ØL languages. Information and Control, 19: 302-318.

Rozenberg, G. and A. Lindenmayer, 1973. Developmental systems with locally catenative formulas. Acta Informatica, 2: 214-248.

Rozenberg, G. and A. Salomaa (eds.), 1974. L Systems, Lecture Notes in Computer Science, vol. 15. Springer Verlag, Heidelberg.

Rozenberg, G. and A. Salomaa, 1980. The Mathematical Theory of L-Systems. Academic Press, New York.

Sacks, P.G., 1978. Developmental dominance in hydra: effects of manipulating nerve cell density. Ph.D. thesis, Syracuse University, Dept. of Biology.

Salomaa, A., 1973. Lindenmayer systems: parallel rewriting without terminals. In: Formal Languages, Part II. Academic Press, New York.

Salomaa, A., 1974. Recent results on L-systems. Proceedings of the 1974 Conference on Biologically Motivated Automata Theory. 38-45.

Stahl, W., 1965. Algorithmically unsolvable problems for a cell automaton. J. Theoret. Biol., 8: 371-394.

Stahl, W. and H. Goheen, 1963. Molecular algorithms. J. Theoret. Biol., 5:266-287.

Walker, A., 1974. Dynamically stable strings in developmental systems, self repair, and the Chomsky hierarchy. Proceedings of the 1974 Conference on Biologically Motivated Automata Theory. 46-49.

Notation

F	family of languages, e.g. F(ØL) = family of all ØL languages
A	adult language
D	deterministic
P	propagating
E	extended language
I	with interactions
T	having tables
x^i	x concatenated with itself i times
	the alphabet
|x|	the length of the string x
#X	the number of elements in set X
L(G)	the language generated by system G
E(H)	sequence of strings generated by system H, beginning with the axiom
ω	the axiom or initial word of any L system
O(H)	sequence of subsets of the alphabet such that each subset is the set of all different elements which appear in the corresponding E(H)
Δ	the target alphabet (set of "terminals") for any extended language generated by an L system
->	produces
=>	directly derives
$\overset{*}{\Rightarrow}$	derives in Ø or more steps
$\overset{n}{\Rightarrow}$	derives in n steps

Developmental descriptions based on sequential machines.

From Lindenmayer (1968a,b).

Definitions

d(p,q) = r : the next-state function
p = present state
q = input sequence
r = next state

g(p,q) = u : the output function
p and q are as above
u = output sequence

Filaments with one-sided inputs

Assume that the outputs are identical to the states (so g(p,q) = p). The input to any cell is the output of its left neighbor. Each cell may have one of two states, Ø or 1. Given the following next-state function:

		input Ø	1
state	Ø	Ø	1
	1	11	Ø

Assuming an initial state of 1 and a constant environmental input of Ø, the following sequence is obtained.

row	environ. input	filament
Ø	Ø	1
1	Ø	11
2	Ø	11Ø
3	Ø	11Ø1
4	Ø	11Ø111
5	Ø	11Ø11100
6	Ø	1101110010
7	Ø	11011100101111
8	Ø	1101110010111100
9	Ø	110111001011111000010
10	Ø	11011100101111100001000011

Despite the fact that the distribution of cell division (that is, next state of 11 in this model) does not exhibit a regular pattern, there is a highly regular pattern produced. This is reminiscent of a growing root tip or apical shoot. This model assumes that outputs are transmitted in only one direction along the filament. This is the case in transport of plant auxins.

Models for a branching filamentous organism

1. Filaments with no inputs (zero-sided input)

This same model is presented as a ØL system in chapter 1 and discussed in terms of recurrence formulas in chapter 2. This is presented using the same terminology as is used for one-sided input, but since the patterns that emerge are independent of input (including environmental input), this constitutes a model which requires no inputs from neighboring cells or the environment.

A verbal description and illustration of how this organism develops is found in chapter 1. This model actually describes the red alga Callithamnion roseum Harvey.

The states are {1,2,3,...,9}. The next-state function is represented by the following table. The additional notation of parentheses has been added; they represent the presence of a branch.

	present state							
	1	2	3	4	5	6	7	8	9
next state	23	2	24	25	65	7	8	9(3)	9

Regardless of environmental input, the following sequence would be obtained from a starting state of 1.

row	filament
1	1
2	23
3	224
4	2225
9	2229(24)9(3)8765
12	2229(22765)9(2265)9(225)9(24)9(3)8765

2. Filaments with two-sided inputs

A branching filamentous model may also be developed using two-sided input. The set of possible states is {Ø,1,2,3}. Ø is used for environmental input only. The output of a cell is considered identical to its state (as in one-sided input functions).

The next-state functions are:

present state = 1

left input \ right input	0	1	2	3
0	2	1	1	1
1	2	2	1	1
2	2	2	1	1
3	1	2	3	3

present state = 2

left input \ right input	0	1	2	3
0	11	1	1	1
1	11	2	2	2
2	11	1	3	1
3	11	3	3	3

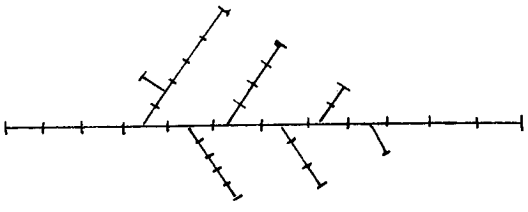
present state = 3

left input \ right input	0	1	2	3
0	2	1	1	1
1	1	2	3	3
2	1	2	3	3
3	1	3	3	1(1)

Given a cell in state 1 and constant environmental inputs of 0 from both sides, the following sequences will be obtained.

row
1 010
2 020
3 0110
4 0120
5 01110
6 01220
10 0122220
15 01223(2)33110
20 01212(3311)1(311)3(11)1(1)3220
24 01212(23(2)331)1(23311)2(3311)1(331)3(11)1(1)3220

Row 24 can be drawn as:



which is structurally similar to the model developed earlier.

Generating a PDØL system given any locally catenative sequence (Rozenberg and Lindenmayer, 1973).

Given a LCF <i1,...ik> with a cut p consistent with the formula and any sequence of integers $l_1,\ldots,l_p$ satisfying the following conditions:

$$l_p <= l_{p-1} <=\ldots<= l_1$$

$$l_{i1} + l_{i2} + \ldots + l_{ik} >= l_1$$

it is possible to construct a PDØL system G such that

$E(G) = a_0, a_1, a_2, \ldots$ satisfies the formula with cut p

and $|a_0| = l_p$, $|a_1| = l_{p-1}$,.... $|a_{p-1}| = l_1$

Let $d = l_p + l_{p-1} + \ldots + l_1$. Let $G = \langle \sum, P, \omega \rangle$ be a PDØL system such that:

$\#\sum = d$

$$\sum = \{ A_1^{(p)}, A_2^{(p)}, \ldots, A_{l_p}^{(p)}, A_1^{(p-1)}, \ldots, A_{l_{p-1}}^{(p-1)}, \ldots, A_1^{(1)}, \ldots, A_{l_1}^{(1)} \}$$

$$\text{axiom} = A_1^{(p)} \ldots A_{l_p}^{(p)}$$

P is defined as:

$$A^{(p)}_1 \to A^{(p-1)}_1 \,,\; A^{(p)}_2 \to A^{(p-1)}_2 \,,\ldots, A^{(p)}_l \to A^{(p-1)}_l \ldots A^{(p-1)}_{l_p}$$

$$A^{(p-1)}_1 \to A^{(p-2)}_1 \,,\; A^{(p-1)}_2 \to A^{(p-2)}_2 \,,\ldots, A^{(p-1)}_l \to A^{(p-2)}_l \ldots A^{(p-2)}_{l_{p-1}}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$A^{(2)}_1 \to A^{(1)}_1 \,,\; A^{(2)}_2 \to A^{(1)}_2 \,,\ldots, A^{(2)}_l \to A^{(1)}_l \ldots A^{(1)}_{l_2}$$

$$A^{(1)}_1 \to Z_1 \,,\; A^{(1)}_2 \to Z_2 \,,\ldots, A^{(1)}_l \to Z_{l_1} \ldots Z_g$$

where $g = l_{i1} + l_{i2} + \ldots + l_{ik}$ and

$$A^{(i1)}_1 A^{(i1)}_2 \ldots A^{(i1)}_{l_{i1}} \; A^{(i2)}_1 \ldots A^{(i2)}_{l_{i2}} \ldots A^{(ik)}_1 \ldots A^{(ik)}_{l_{ik}} =$$

$Z_1 Z_2 \ldots Z_g$ for $Z_1, \ldots, Z_g$ in $\sum$.

It has been defined that $E(G) = a_\emptyset, a_1, a_2, \ldots$ where

$$a_\emptyset = A^{(p)}_1 \ldots A^{(p)}_{l_p}$$

$$a_1 = A^{(p-1)}_1 \ldots A^{(p-1)}_{l_{p-1}}$$

$$a_{p-1} = A^{(1)}_1 \ldots A^{(1)}_{l_1}$$

$$a_p = Z_1 \ldots Z_g =$$

$$A_1^{(i1)} \ldots A_{i1}^{(i1)} \ldots A_1^{(ik)} \ldots A_{ik}^{(ik)} = a_{p-i1}\, a_{p-i2} \ldots a_{p-ik}$$

Therefore, by definition of a LCS, E(G) is
<i1,...ik> - locally catenative with cut p.

Appendix D

From Herman and Rozenberg (1975). Creating a context-sensitive grammar which corresponds to a given ØL system (see Chapter 1).

G = <Vn,Vt,P,S> where
Vn = {S,E.B,R,L,T}
Vt = {Ø,1,2,...,9,(,)}

Productions rules:
{ S -> 4, S -> 56, S -> 3758, S -> 33(9)375Ø,
S -> 33(BL)375E, BL -> 3BR, RE -> 1LE,
BR -> 9T, TE -> 1Ø,
R_t -> a_t R for any t in Vt where a_t denotes the
right hand side of the production rule for
t in the developmental model this is
simulating, eg. $a_Ø$ =1Ø, a_1 =32, etc.

tL -> Lt for any t in Vt
T_t -> a_t T for any t in Vt, a_t defined in production
rule above }

A typical derivation:

S
33(BL)375E
33(3BR)375E
33(3B)R375E
33(3B)3R75E
33(3B)33(9)R5E
33(3B)33(9)37RE
33(3B)33(9)371LE
33(3B)33(9)371LE
33(3B)33(9)3L71E
33(3B)33(9)L371E
33(3B)33(9L)371E
33(3B)33(L9)371E
33(3B)33L(9)371E
33(3B)3L3(9)371E
33(3B)L33(9)371E
33(3BL)33(9)371E
33(3BR)33(9)371E
33(339T)33(9)371E
33(339)T33(9)371E
33(339)3T3(9)371E
33(339)33T(9)371E
33(339)33(T9)371E
33(339)33(39T)371E
33(339)33(39)T371E
33(339)33(39)3T71E
33(339)33(39)33(9)T1E
33(339)33(39)33(9)32TE
33(339)33(39)33(9)3210
(which is w_6)

Appendix E

Constructing a CFG G such that L(G) = A(H) (see chapter 1, adult languages), (Walker, 1974).

G = <Vn, Vt, P, S> is a CFG. Assume without loss of generality that for each B in Vn B -> B is not in P and

$$S \overset{*}{\Rightarrow} qBr \overset{*}{\Rightarrow} qXr \text{ for some } q, X, r \text{ in } Vt^*.$$

Let V be the union of Vn and Vt.
Now construct from G a mapping d
V -> (finite subsets of V*) as follows.
For A in Vn, d(A) = {q | A -> q in P}
and for A in Vt, d(A) = {A}.
Extended to domain V* by $d(\Lambda) = \Lambda$
and d(Aq) = d(A)d(q).
H = <V,d,S> is a ØL system such that
L(G) = A(H).

Now construct G from H, given H = <V, d, S >. It can be shown that we can always effectively find the set $\sum \subset V$ of letters which appear in A(H). It can also be shown that for each b in $\sum$, there is a unique B in $\sum^*$ such that $d^m(b) = \{B\}$, where $m = \#\sum$, and d(B) = {B}. So we can define a mapping T:V -> V* by T(b) = b if b is in $V-\sum$, and $T(b) = d^m(b)$ if b is in $\sum$. We extend T to have as domain the subsets of V* in the obvious manner and then use T as follows. We construct from H = <V,d,S> a ØL system K = <V,k,S> where $k(a) = T^k(a)$ if a is in $V-\sum$ and k(a) = a if a is in $\sum$. It can be shown that K is such that A(H) = A(K). But K has the property that each symbol of its adult language derives itself only. Hence if we set G = < $V-\sum$, $\sum$, P, S > with

$$P = \{ A \rightarrow a \mid A \text{ is in } V-\sum \text{ and } a \text{ is in } k(A) \}$$

then G is a CFG for which A(K) = L(G). So A(H) = L(G).

Appendix F

Why the language $L=\{c\} \cup \{a^{2^x} \mid x\geq\emptyset\} \cup \{ba^{2^x+1} \mid x\geq\emptyset\}$ cannot be generated by a $\langle\emptyset,n\rangle$ system for any n. (Refer to one-sided IL systems in Chapter 4).

In order for this language to be generated by a right-sided IL system, the following must be true.

1. For some $f>1$, $\langle -, a, a^n \rangle \rightarrow a^f$ (as the sequence of a's must be able to change).

2. $\langle -, a\, a^n \rangle \rightarrow x$ implies that $x \in \{a\}^*$ (as no a is ever followed by a letter other than a).

3. All but a finite number of words of the form ba^{2^x+1} must be derived from words of the same form (as seen in 2).

4. $\langle -, b\, a^n \rangle \rightarrow a^k$ is not in P, for any k. Otherwise, let f be the positive integer indicated in 1 and let t be any integer such that $a^n \Rightarrow a^t$. Then, for any x such that $2^x \geq n$

$$ba^{2^x+1} \Rightarrow a^{m+(2^x+1)-n)f+t} \quad \text{and}$$

$$a^{2^n} \Rightarrow a^{(2^x-n)f+t}$$

The right-hand sides may be arbitrarily long, but the difference in their lengths is always the constant m+f, which is greater than $\emptyset$. This contradicts the fact that all elements in $\{a\}^* \cap L$ are of the form a^{2^n}.

5. From 4, it is apparent that all words of the form a^{2^x} are derived from words of the same form.

6. If $\langle -, a, a^{n} \rangle \rightarrow X$ and $\langle -, a, a^{n} \rangle \rightarrow Y$, then X=Y.

   Otherwise, for infinitely many X, both a^{x}_{2} and a^{x}_{2+c} are in L, which contradicts the definition of the language.

7. From 1 and 6 it follows that there exists m>=2 such that for every X, if $\langle -, a, a^{n} \rangle \rightarrow X$ then $X = a^{m}$.

8. From 5 and 7 it follows that for infinitely may x, $a^{x}_{2} \Rightarrow a^{u}_{2}$ and $ba^{x}_{2+1} \Rightarrow Ba^{u}_{m2}$ for some u>x, a fixed m>=2 and B such that $\langle -, b, a^{n} \rangle \rightarrow B$.

Therefore, there is not a language of the form $\langle \emptyset, n \rangle$ which can generate the language L described.

For every grammar G, there exists an E<1,Ø> system H such that L(G) = L(H). (Herman and Rozenberg, 1975).

Let V = Vt U Vn. Let G = <Vn,Vt,P,S> be a grammar, where P consists of the following n (n>=1) productions:

$$A_{11} \ldots A_{1m_1} \rightarrow \alpha_1$$

$$A_{21} \ldots A_{2m_2} \rightarrow \alpha_2$$

$$\vdots$$

$$A_{n1} \ldots A_{nm_n} \rightarrow \alpha_n$$

for some positive integers m_i $(1<=i<=n)$ and for some $A_{11},\ldots,A_{1m_1},\ldots,A_{nm_n}$ in V* .

Let H = $<\sum, R, g, S'>$ be a <1,Ø> system where

$$\sum = Vt \cup Vn \cup \sum_1 \cup \sum_2 \cup \sum_3 \cup \{S',D,E\}$$

$$\sum_1 = \{ a \mid a \text{ is in } V \}$$

$$\sum_2 = \{ a \mid a \text{ is in } V \}$$

$$\sum_3 = \{ A^{(k,j)} \mid k \in \{1,\ldots,n\} \text{ and } 1<=j<=m_k \}$$

Assume that all given subsets of $\sum$ are disjoint.
R consists of the following productions:

1) $<g,S',\Lambda> \rightarrow \Lambda$, if $S \rightarrow \Lambda$ is in P.

2) $<g,S',\Lambda> \rightarrow \overline{\alpha}E$, if $S \rightarrow \alpha$ is in P and $\alpha \neq \Lambda$.

3) $<x,\overline{a},\Lambda> \rightarrow \dot{a}$ for every a,x such that a is in V and x is in $\sum_2 \cup \{g\}$

4) $<x,\overline{a},\Lambda> \rightarrow \overline{a}$ for every a,x such that a is in V and

$$x \text{ is in } \sum_1 U \sum_2 U \{\varepsilon\} U \{A^{(k,m_k)} \mid k \text{ is in } \{1,..,n\}\}$$

5) $\langle x, \bar{a}, \Lambda \rangle \rightarrow a$ for every a,x such that a is in V and x is in V U $\{\varepsilon\}$.

6) $\langle A^{(k,j)}, a, \Lambda \rangle \rightarrow A^{(k,j+1)}$ for every a,k,j such that k is in $\{1,..,n\}$, $1 \leq j < m_k$, $a = A_{k(j+1)}$.

7) $\langle A^{(k,j)}, \bar{a}, \Lambda \rangle \rightarrow D$, for every a,k,j such that k is in $\{1,..,n\}$, $1 \leq j < m_k$, $a \neq A_{k(j+1)}$

8) $\langle x, \dot{a}, \Lambda \rangle \rightarrow \bar{a}$ for every a,x such that a is in V and x is in $\sum_1 U \{\varepsilon\}$

9) $\langle x, \dot{a}, \Lambda \rangle \rightarrow A^{(k,1)}$ for every a,x,k such that k is in $\{1,..,n\}$, x is in $\sum_1 U \{\varepsilon\}$ and $a = A_{k1}$

10) $\langle x, \dot{a}, \Lambda \rangle \rightarrow D$ for every a,x such that a is in V and x is not in $\sum_1 U \{\varepsilon\}$

11) $\langle x, A^{(k,j)}, \Lambda \rangle \rightarrow D$ for every x,k,j such that x is not in $\sum_1 U \{\varepsilon\}$, k is in $\{1,..,n\}$, $1 \leq j < m_k$.

12) $\langle x, A^{(k,j)}, \Lambda \rangle \rightarrow \Lambda$ for every x,k,j such that x is in $\sum_1 U \{\varepsilon\}$, k is in $\{1,..,n\}$, $1 \leq j < m_k$

13) $\langle x, A^{(k,m_k)}, \Lambda \rangle \rightarrow \bar{\alpha}_k$, for every x,k such that x is in $\sum_1 U \{\varepsilon\}$, k is in $\{1,..,n\}$, $1 \leq j < m_k$.

14) $\langle x, D, \Lambda \rangle \rightarrow D$ for every x in $\sum U \{\varepsilon\}$.

15) $\langle x, E, \Lambda \rangle \rightarrow E$ for every x in $\sum_1 \cup \sum_2 \cup \{A_k^{(k,m_k)} \mid k \text{ is in } \{1,\ldots,n\}\}$.

16) $\langle x, E, \Lambda \rangle \rightarrow \Lambda$ for every x in $V \cup \{g\}$.

17) $\langle A_k^{(k,j)}, E, \Lambda \rangle \rightarrow D$ for every k,j such that k is in $\{1,..,n\}$ and $1 \leq i < m_k$.

18) $\langle x, a, \Lambda \rangle \rightarrow a$, for every a,x such that a is in V, x is in $V \cup \{g\}$.

19) $\langle x, y, \Lambda \rangle \rightarrow D$ for every x,y in $\cup \{g\}$ such that no production with $\langle x, y, \Lambda \rangle$ on the left was specified before.

In the above, if α is a word over V, $\alpha = a_1 \ldots a_m$ for some $m \geq 1$, a_i in V for $1 \leq i \leq m$, then $\bar{\alpha}$ denotes the word $a_1 \ldots a_m$. If $\alpha = \Lambda$, then $\bar{\alpha} = \Lambda$.

Some helpful comments:

After the derivation step involving the axiom, the simulation of derivations in G is going on in H in such a way that one rewriting step in G is usually simulated by a number of steps in H.

At the left hand side of the string $\bar{\alpha}$, a dotted symbol from $\sum_2$ is generated and then it travels to the right (group 3 from R). Then it either goes away, being changed to an element of $\sum_1$ (group 8 from R), or, when it finds a symbol which is the first symbol of the left hand side of the kth production ($1 \leq k \leq n$) from P (group 9 from R), it can start checking (groups 6,12 and 13 from R) whether this symbol is the first letter of a subword which matches the

left hand side of the kth production. If the match is successful, then it results in the rewriting of the subword according to the right hand side of the kth production (group 13 from R). If the match is unsuccessful, then it results in the introduction of the "dead symbol" D (groups 7 and 17 from R) which is not an element of Vt and which is always rewritten as D (group 14 from R).

In this way, it is possible to simulate a rewriting using an arbitrary rule from P, and at each moment of time it is possible to start to rewrite a string of the form $\bar{\beta}E$ for some β in V*, to the string βE (groups 5 and 18 from R) and then to β (group 16 from R).