THE MATHEMATICAL THEORY OF L SYSTEMS

G. Rozenberg
A. Salomaa

DAIMI PB-33
July 1974

Institute of Mathematics University of Aarhus
DEPARTMENT OF COMPUTER SCIENCE
Ny Munkegade - 8000 Aarhus C - Denmark
Phone 06-12 83 55

CONTENTS

CONTENTS (continued):

THE MATHEMATICAL THEORY OF L SYSTEMS

G. Rozenberg

Institute of Mathematics
Utrecht University
Utrecht- De Uithof
Holland

Department of Mathematics
University of Antwerp, UIA
Wilrijk
Belgium

and

A. Salomaa

Department of Computer Science
University of Aarhus
Aarhus
Denmark

3

[illegible]

Running head: same as title

List of symbols:

$\cap$ $\bigcup$ $\subseteq$ $\subsetneqq$ $\rightarrow$ $\Rightarrow$

Script letters: $\mathcal{P}$ $\mathcal{L}$ $\mathcal{G}$

Greek letters as typed.

All correspondence to:

A. Salomaa
Department of Computer Science
University of Aarhus
8000 Aarhus C
Denmark

Abstract

The paper gives a survey of the mathematical theory of developmental systems and languages. Many of the results have not yet appeared in print.

0. INTRODUCTION

The theory of L systems originated from the work of Lindenmayer [31]. The original aim of this theory was to provide mathematical models for the development of simple filamentous organisms. At the beginning L systems were defined as linear arrays of finite automata, later however they were reformulated into the more suitable framework of grammar-like constructs. From then on, the theory of L systems was developed essentially as a branch of formal language theory. In fact it constitutes today one of the most vigorously investigated areas of formal language theory.

In this paper we survey the mathematical theory of L systems. As to the biological aspects of the theory we refer the reader to an excellent paper by Lindenmayer ("Developmental systems and languages in their biological context", a contribution to the book Herman and Rozenberg [19]).

This paper is organized in such a way that it discusses several typical problem areas and the results obtained therein. The results quoted here may not always be the most important ones but they are quite representative for the direction of research in this theory. It is rather unfortunate that we have no space here to discuss the basic techniques for solving problems in this theory, but information about these can be found in the listed references. As the most complete source of readings on L system the volumes [19] and [51] are recommended to the reader. On the other hand, the present paper contains many results not contained in these two volumes. This reflects the very rapid development in the area.

In this paper we assume the reader to be familiar with basic formal language theory, e.g. within the scope of the book [56] by Salomaa. We shall also freely use standard formal language notation and terminology. (Perhaps the only unusual term used in this paper is "coding" which means a letter-to-letter homomorphism).

We also want to remark that this survey is of informal character, meaning that quite often concepts are introduced in a not entirely rigorous manner, and results are presented in a descriptive way rather than in a form of very precise mathematical statements.This was dictated by the limited size of the paper. We hope that this does not decrease the usefulness of this paper.

Finally, we would like to state that this survey is by no means exhaustive and the selection of topics and results presented reflects our personal point of view.

1. L SCHEMES AND L SYSTEMS

In this section we give definitions and examples of basic objects (the so called L schemes and L systems) to be discussed in this paper. We start with the most general class, the so-called TIL schemes and TIL systems. (They were introduced in [26].) TIL systems are intended to model the developemnt of multicellular filamentous organisms in the case when an interaction can take place among the cells and the environment can be subject to changes.

Definition 1.1. Let $k, l \in N$. An L scheme with tables and with $<k, l>$ interactions (abbreviated T $<k, l>$ L scheme) is a construct $S = <\Sigma, \mathcal{P}, g>$ where Σ is a finite nonempty set (the alphabet of S), g is a symbol which is not in Σ (the masker of S), $\mathcal{P}$ is a finite nonempty set, where each element of P (called a table of S) is a relation satisfying the following: $P \subseteq A \times \Sigma^*$ with

$$A = \bigcup_{\substack{i,j,m,n \geq 0 \\ i+j = k \\ m+n = l}} \{g^i\}\Sigma^j \times \Sigma \times \Sigma^m\{g^n\}$$

and for every $<\alpha, a, \beta>$ in A there exists a γ in Σ^* such that $<\alpha, a, \beta, \gamma> \in P$.

(Each element of P is called a production.)

Definition 1.2. Let $S = <\Sigma, \mathcal{P}, g>$ be a T $<k, l>$ L scheme. We say that S is:

1. an L scheme with $<k,l>$ interactions (abbreviated $<k,l>$ L scheme) if $\#\mathcal{P} = 1$.
2. an L scheme with tables and without interactions (abbreviated TOL

<u>scheme</u>) if $k = l = 0$.

3. an <u>L scheme without interactions</u> (abbreviated <u>0L scheme</u>) if both $\# \mathcal{P} = 1$ and $k = l = 0$.

<u>Definition 1.3.</u> A construct $S = \langle \Sigma, \mathcal{P}, g \rangle$ is called a <u>TIL scheme</u> (resp. IL scheme) if, for some $k, l \in N$, S is a $T \langle k, l \rangle L$ scheme (resp. $\langle k, l \rangle L$ scheme).

<u>Definition 1.4.</u> Let $S = \langle \Sigma, \mathcal{P}, g \rangle$ be a $T \langle k, l \rangle L$ scheme. Let $x = a_1 \dots a_n \in \Sigma^*$, with $a_1, \dots, a_n \in \Sigma$, and let $y \in \Sigma^*$. We say that <u>x directly derives y in S</u> (denoted as $x \underset{S}{\Rightarrow} y$ if $y = \gamma_1 \dots \gamma_n$ for some $\gamma_1, \dots, \gamma_n$ in Σ^* such that there exists a table P in $\mathcal{P}$ and for every i in $\{1, \dots, n\}$ P contains a production of the form $\langle \alpha_i, a_i, \beta_i, \gamma_i \rangle$ where α_i is the suffix of $g^k a_i \dots a_{i-1}$ of length k and β_i is the prefix of $a_{i+1} \dots a_n g^l$ of length l. (For $i = 1$, $g^k a_1 \dots a_{i-1}$ reads g^k. For $i = n$, $a_{i+1} \dots a_n g^l$ reads g^l.) The transitive and reflexive closure of the relation $\underset{S}{\Rightarrow}$ is denoted as $\overset{*}{\underset{S}{\Rightarrow}}$ (when $x \overset{*}{\underset{S}{\Rightarrow}} y$ then we say that <u>x derives y in S</u>).

<u>Definition 1.5.</u> A <u>TIL system</u> (resp. <u>IL system</u>) is an ordered pair $G = \langle S, \omega \rangle$ where S is a TIL scheme (resp. an IL scheme) and ω is a word over the alphabet of S. The scheme S is called the <u>underlying scheme of G</u> and is denoted as $S(G)$. G is called a <u>$T \langle k, l \rangle L$ system</u> (resp. a <u>$\langle k, l \rangle L$ system</u>, a <u>T0L system</u>, a <u>0L system</u>) if $S(G)$ is a $T \langle k, l \rangle L$ scheme (resp. a $\langle k, l \rangle$ scheme, a T0L scheme, a 0L scheme).

IL systems in restricted form originated from [31]; in the form discussed here they were introduced in [44, 45]. TOL systems were introduced in [40] and 0L systems were introduced in [32] and [48].

<u>Definition 1.6.</u> Let $G = \langle S, \omega \rangle$ be a TIL system. Let $x, y \in \Sigma^*$. We say that <u>x directly derives y in G</u>, denoted as $x \underset{G}{\Rightarrow} y$ (resp. x derives y in G, denoted as $x \overset{*}{\underset{G}{\Rightarrow}} y$) if $x \underset{S}{\Rightarrow} y$ (resp. $x \overset{*}{\underset{S}{\Rightarrow}} y$).

<u>Notation</u>. It is customary to omit the masker g from the specification of a TOL system. If S is an IL or a 0L scheme (system) such that $\# \mathcal{P} = 1$, say $\mathcal{P} = \{P\}$, then in the specification of S we put P rather than $\{P\}$. Also to avoid cumbersome notation in specifying a TIL system G we simply extend the n-tuple specifying S(G) to an (n+1)-tuple where the last element is the axiom of G. (In this sense we write, e.g., $G = \langle \Sigma, \mathcal{P}, g, \omega \rangle$ rather than $G = \langle\langle \Sigma, \mathcal{P}, g \rangle, \omega \rangle$). In specifying productions in a table of a given TIL system one often omits those which clearly cannot be used in any rewriting process which starts with the axiom of the system. If $\langle \alpha, a, \beta, \gamma \rangle$ is a production in a TIL scheme (system) then it is usually written in the form $\langle \alpha, a, \beta \rangle \to \gamma$ (where $\langle \alpha, a, \beta \rangle$ is called its left-hand side and γ is called its right-hand side). When the productions of a TOL scheme (system) are being specified, then we write $a \to \gamma$ rather than $\langle \Lambda, a, \Lambda \rangle \to \gamma$.

<u>Example 1.1.</u> Let $\Sigma = \{a, b\}$, $P_1 = \{\langle g, a, \Lambda \rangle \to a^3, \langle a, a, \Lambda \rangle \to a, \langle a, b, \Lambda \rangle \to b^2, \langle b, b, \Lambda \rangle \to b^2, \langle b, a, \Lambda \rangle \to a\}$, $P_2 = \{\langle g, a, \Lambda \rangle \to a^4, \langle a, a, \Lambda \rangle \to a, \langle a, b, \Lambda \rangle \to b^3, \langle b, b, \Lambda \rangle \to b^3, \langle b, a, \Lambda \rangle \to a\}$ and $\omega = a^5 b^6 a$. Then $G = \langle \Sigma, \{P_1, P_2\}, g, \omega \rangle$ is a $T\langle 1, 0 \rangle$ L system.

<u>Example 1.2.</u> Let $\Sigma = \{a,b\}$, $P = \{<a,a,\Lambda> \to a^2, <b,a,\Lambda> \to a^2, <g,a,\Lambda> \to a, <a,b,\Lambda> \to b^2, <b,b,\Lambda> \to b^2, <g,b,\Lambda> \to b^2, <g,b,\Lambda> \to ab^2\}$ and $\omega = ba$. Then $G = <\Sigma, P, g, \omega>$ is a $<1,0>$ L system.

<u>Example 1.3.</u> Let $\Sigma = \{a,b\}$, $P_1 = \{a \to a^2, b \to b^2\}$, $P_2 = \{a \to a^3, b \to b^3\}$ and $\omega = ab$. Then $G = <\Sigma, \{P_1, P_2\}, \omega>$ is a T0L system.

<u>Example 1.4.</u> Let $\Sigma = \{A, \overline{A}, a, B, \overline{B}, b, C, \overline{C}, c, F\}$, $P = \{A \to A\overline{A}, A \to a, B \to B\overline{B}, B \to b, C \to C\overline{C}, C \to c, \overline{A} \to \overline{A}, \overline{A} \to a, \overline{B} \to \overline{B}, \overline{B} \to b, \overline{C} \to \overline{C}, \overline{C} \to c, a \to F, b \to F, c \to F, F \to F\}$ and $\omega = ABC$. Then $G = <\Sigma, P, \omega>$ is a 0L system.

2. SQUEEZING LANGUAGES OUT OF L SYSTEMS

There are several ways which one can associate a language with a given word-generating device. In this section we shall discuss several ways of defining languages by L systems.

2.1 Exhaustive approach

Given an L system G (with alphabet Σ and axiom ω) it is most natural to define its language, denoted L(G), as the set of all words (axiom included) that can be derived from ω in G; hence L(G) = $\{x \in \Sigma^* : \omega \overset{*}{\underset{G}{\Rightarrow}} x\}$. This situation can be illustrated by the following scientific diagram.

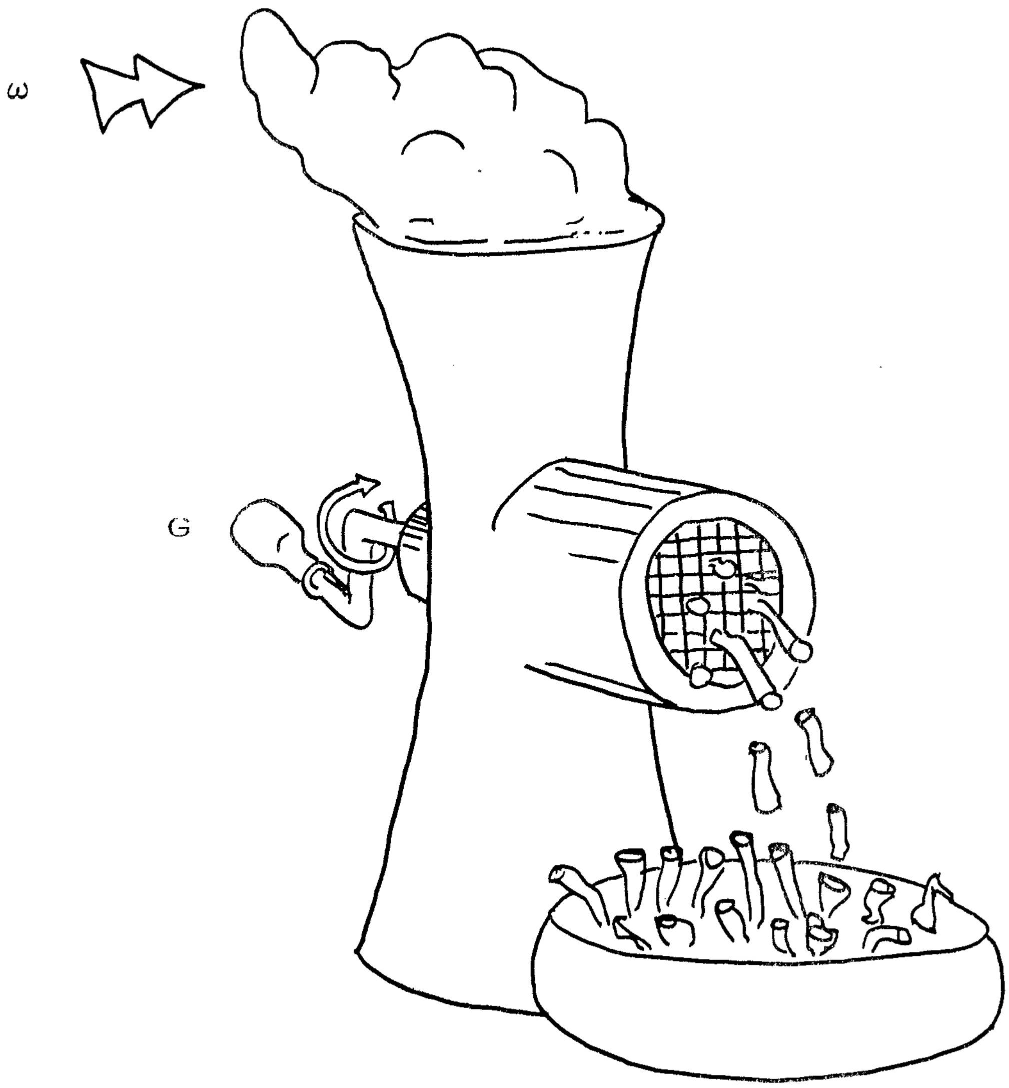


Fig. 1

Example 2.1.1. The language of the T <1,0> L system G from Example 1.1 is $\{a^{2n+3m}b^{2^n3^m}a : n,m \geq 1\}$. The language of the TOL system from Example 1.3 is $\{a^{2^n3^m}b^{2^n3^m} : n,m \geq 0\}$.

The languages obtained in this way from 0L, T0L, TIL and IL systems are called <u>0L</u>, <u>T0L</u>, <u>TIL</u> and <u>IL</u> <u>languages</u> respectively. (Their classes will be denoted by $\mathcal{L}$(0L), $\mathcal{L}$(T0L), $\mathcal{L}$(TIL), $\mathcal{L}$(IL) respectively). For k, l ≥ 0, a <u><k, l> L language</u> (resp. a <u>T<k, l> L language</u>) is a language generated by a <k, l> L system (resp. a T <k, l> L system).

One may notice here two major differences in generating languages by 0L and IL system on the one hand and context-free and type 0 grammars on the other. 0L and IL systems do not use nonterminal symbols while context-free and type 0 grammars use them. Rewriting in 0L and IL systems is absolutely parallel (all occurrences of all letters in a word are rewritten in a single derivation step) while rewriting in context-free and type 0 grammars is absolutely sequential (only one occurrence of one symbol is rewritten in a single derivation step).

2.2 Selective approaches

We now study various ways of squeezing a language out of a system such that all words obtained by grinding are not considered to be acceptable. Thus, we consider pairs (G,F), where G is an L system and F is a filter. The final language is obtained by applying F to the words in L(G). Again, the situation can be illustrated by the following scientific diagram.

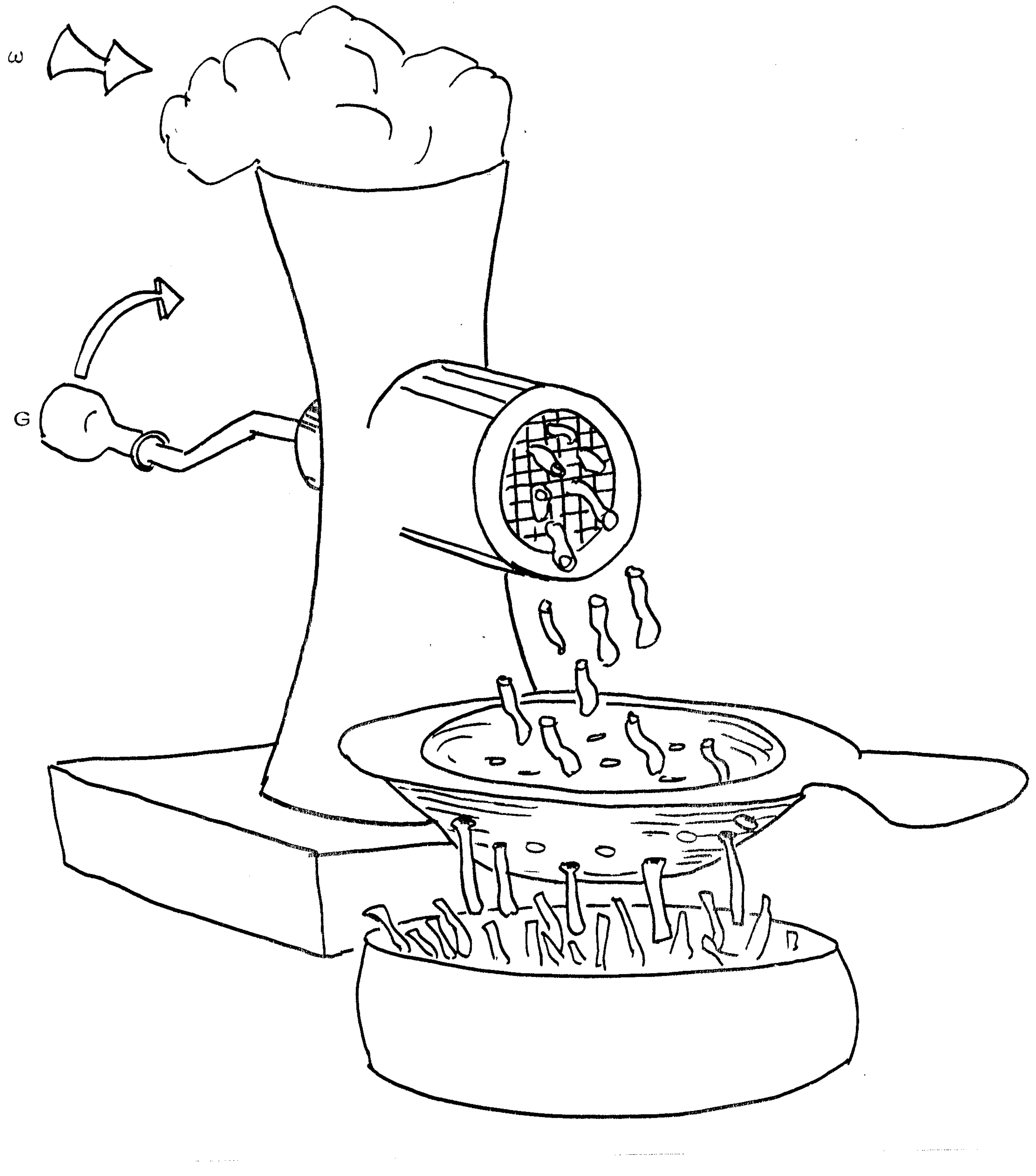


Fig. 2

2.2.1 Using nonterminals to define languages.

The standard step in formal language theory to define the language of a generating system is to consider not the set of all words generated by it but only those which are over some distinguished (usually called terminal) alphabet. In this way one gets the division of the alphabet of a given system into the set of terminal and nonterminal (sometimes also called auxiliary) symbols. In the case of L systems such an approach gives rise to the following classes of systems. Thus, filtering consists here of intersecting the language with the set of words over the terminal alphabet.

Definition 2.2.1.1. An extended 0L, (resp. T0L, IL, TIL) system, abbreviated E0L (resp. ET0L, EIL, ETIL) system, is a pair $G = \langle H, \Delta \rangle$, where H is a 0L (T0L, IL, TIL) system and Δ is a subalphabet of the alphabet of H (called the target alphabet of G).

Definition 2.2.1.2. The language of an E0L (ET0L , EIL , ETIL) system $G = \langle H, \Delta \rangle$, denoted as $L(G)$, is defined by $L(G) = L(H) \cap \Delta^*$.

An E0L (resp. ET0L, EIL, ETIL) system $G = \langle H, \Delta \rangle$ is usually specified as $\langle \Sigma, P, \omega, \Delta \rangle$ (resp. $\langle \Sigma, \mathcal{P}, \omega, \Delta \rangle$, $\langle \Sigma, P, g, \omega, \Delta \rangle$, $\langle \Sigma, \mathcal{P}, g, \omega, \Delta \rangle$ where $\langle \Sigma, P, \omega \rangle$ (resp. $\langle \Sigma, \mathcal{P}, \omega \rangle$, $\langle \Sigma, P, g, \omega \rangle$, $\langle \Sigma, \mathcal{P}, g, \omega \rangle$) is the specification of H itself.

Example 2.2.1.1. Let $G = \langle \Sigma, P, \omega, \Delta \rangle$, where Σ, P, ω are specified as in Example 1.4 and $\Delta = \{a, b, c\}$. Then $L(G) = \{a^n b^n c^n : n \geq 1\}$.

If K is the language of an E0L (resp. ET0L, EIL, ETIL) system, then it is called an E0L (resp. ET0L, EIL, ETIL) language. The classes of E0L languages, ET0L languages, EIL languages and ETIL languages are denoted by $\mathcal{L}$(E0L), $\mathcal{L}$(ET0L), $\mathcal{L}$(EIL), and $\mathcal{L}$(ETIL) respectively.

E0L systems and languages are discussed in [16]; ET0L systems and languages were introduced in [46]; EIL systems and languages are discussed e.g. in [6] and [44]; ETIL systems and languages were introduced in [26].

It is very instructive at this point to notice that, as far as generation of languages is concerned, the difference between E0L and EIL systems on one hand and context-free and type-0 grammars on the other hand is the absolutely parallel fashion of rewriting in E0L and EIL systems and the absolutely sequential fashion of rewriting in context-free and type-0 grammars.

2.2.2 Using codings to define languages

When we make observations of a particular organism and want to describe it by strings of symbols, we first associate a symbol to each particular cell. This is done by dividing cells into a number of types and associating the same symbol to all the cells of the same type. It is possible that the development of the organism can be described by a developmental system, but the actual system describing it uses a finer subdivision into types than the one we could observe. This is often experimentally unavoidable. In this case, the set of strings generated by a given developmental system is a coding of the "real" language of the organism which the given developmental system describes. The filtering device consists here of a coding table which associates with each letter its image under coding. This gives rise to the following classes of systems.

Definition 2.2.2.1. An OL (resp. TOL, IL, TIL) system with coding, abbreviated COL (resp. CTOL, CIL, CTIL) system, is a pair $G = \langle H, h \rangle$, where H is a OL (resp. TOL, IL, TIL) system and h is a coding.

Definition 2.2.2.2. The language of a COL (CTOL, CIL, CTIL) system $G = \langle H, h \rangle$, denoted as $L(G)$, is defined by $L(G) = h(L(H))$.

Example 2.2.2.1. Let $H = \langle \{a, b\}, \{a \to a^2, b \to b\}, ba \rangle$ and h be a coding from $\{a, b\}$ into $\{a, b\}$ such that $h(a) = h(b) = a$. Then $L(\langle H, h \rangle) = \{a^{2^n+1} : n \geq 0\}$.

If K is the language of a COL (resp. CTOL, CIL, CTIL) system, then it is called a COL, (resp. CTOL, CIL, CTIL) language. The classes of COL, CTOL, CIL, and CTIL languages are denoted by $\mathcal{L}(COL)$, $\mathcal{L}(CTOL)$, $\mathcal{L}(CIL)$ and $\mathcal{L}(CTIL)$ respectively.

Using codings to define languages of various classes of L systems was considered, e.g., in [5, 10, 11, 12, 36].

2.2.3. Adult languages of L systems

An interesting way of defining languages by L systems was proposed by A. Walker, [20] and [65]. Based on biological considerations concerning problems of regulation in organisms, one defines the adult language of an L system G, denoted as A(G), to be the set of all those words from L(G) which derive in G themselves and only themselves. Thus, the filtering device consists of a dynamical stability test. The notation $\mathcal{L}(AOL)$ is used for the family of adult OL languages. $\mathcal{L}(AIL)$, $\mathcal{L}(ATOL)$, $\mathcal{L}(ATIL)$ are corresponding notations for other adult families.

Example 2.2.3.1. Let $G = <\Sigma, P, \omega>$ be a 0L system such that $\Sigma = \{a, b\}$, $P = \{a \rightarrow \Lambda, a \rightarrow ab, b \rightarrow b\}$ and $\omega = a$. Then $A(G) = \{b^n : n \geqq 0\}$.

2.2.4 Fragmentation

Another filtering device which consists of taking subwords of a certain kind was recently introduced in [50] and [52]. The basic idea is the following. The right sides of the productions may contain occurrences of a special symbol g. This symbol induces a cut in the string under scan, and the derivation may continue from any of the parts obtained. Thus, if we apply the productions $a \rightarrow aga$, $b \rightarrow ba$, $c \rightarrow gb$ to the word abc, we obtain the words a, aba, and b. The biological significance of this device is that it provides us with a new formalism for blocking communication, splitting the developing filament and cell death.

Definition 2.2.4.1. Consider an alphabet Σ, let $g \in \Sigma$ and assume that $\Sigma_1 = \Sigma - \{g\}$ is not empty. A word x_1 over Σ_1 is a g-guarded subword of a word x over Σ if either $x_1 = x$ or else there are words y_1 and y_2 such that one of the following equations is satisfied:

$$x = y_1 g x_1 g y_2, \quad x = x_1 g y_2, \quad x = y_1 g x_1 .$$

Definition 2.2.4.2. Let $G = <\Sigma, P, \omega>$ be a 0L system, let $g \in \Sigma$, and assume that $g \rightarrow g$ is the only production for g in G. Then the fragmentation language of G, in symbols $J_g(G)$, consists of all g-guarded subwords of $L(G)$.

Example 2.2.4.1. Consider the 0L system

$G = \langle \{a,b,g\}, \{a \to a, b \to abagaba, g \to g\}, aba \rangle$. Then

$J_g(G) = \{aba^n : n \geq 1\} \cup \{a^n ba : n \geq 1\}$.

The family of all fragmentation 0L languages is denoted by $\mathcal{L}$(J0L). The families $\mathcal{L}$(JT0L) and $\mathcal{L}$(JIL) are defined in an analogous fashion.

It is of course possible to combine some of the filtering devices introduced above. Thus, one may apply a coding to a fragmentation 0L language, which gives rise to the family $\mathcal{L}$(CJ0L).

In the sequel we shall use the term L language to refer to any one of the types of languages introduced in this section. Similarly, we use the general terms "L system" and "L scheme".

2.3 Comparing the language generating power of various mechanisms for defining languages

Once several classes of language generating devices are introduced one is interested in comparing their language generating power. This is one of the most natural and most traditional topics investigated in formal language theory. In the case of L systems we have, for example, the following results.

Theorem 2.3.1. (see, e.g., Herman and Rozenberg [19]).

1) For X in {0L, T0L, IL, TIL}, $\mathcal{L}(X) \subsetneq \mathcal{L}(EX)$.

2) For X in {0L, T0L, IL, TIL}, $\mathcal{L}(X) \subsetneq \mathcal{L}(CX)$.

3) $\mathcal{L}$(0L) is incomparable with $\mathcal{L}$(A0L).

Theorem 2.3.2. (Ehrenfeucht, and Rozenberg, [10, 12], Herman and Walker, [20], Rozenberg, Ruohonen and Salomaa, [50]).

1) $\mathcal{L}(E0L) = \mathcal{L}(C0L)$ and $\mathcal{L}(ET0L) = \mathcal{L}(CT0L)$.

2) $\mathcal{L}(0L) \subsetneq \mathcal{L}(J0L) \subsetneq \mathcal{L}(E0L) = \mathcal{L}(CJ0L) \subsetneq \mathcal{L}(ET0L)$.

3) $\mathcal{L}(T0L) \subsetneq \mathcal{L}(JT0L) \subsetneq \mathcal{L}(ET0L) = \mathcal{L}(CJT0L)$.

4) $\mathcal{L}(T0L)$ and $\mathcal{L}(J0L)$ are incomparable, and so are $\mathcal{L}(A0L)$ and $\mathcal{L}(T0L)$, and $\mathcal{L}(A0L)$ and $\mathcal{L}(J0L)$.

3. FITTING CLASSES OF L LANGUAGES INTO KNOWN FORMAL LANGUAGE THEORETIC FRAMEWORK

The usual way of understanding the language generating power of a class of generative systems is by comparing them with the now classical Chomsky hierarchy. (One reason for this is that the Chomsky hierarchy has been the most intensively studied hierarchy in formal language theory.) In the area of L languages we have, for example, the following result. (In what follows $\mathcal{L}(RE)$ denotes the class of recursively enumerable languages, $\mathcal{L}(CS)$ denotes the class of context-sensitive languages, and $\mathcal{L}(CF)$ denotes the class of context-free languages.)

Theorem 3.1. (van Dalen [6], Rozenberg [46], Herman [16].
$\mathcal{L}(EIL) = \mathcal{L}(RE)$, $\mathcal{L}(ET0L) \subsetneq \mathcal{L}(CS)$ and $\mathcal{L}(CF) \subsetneq \mathcal{L}(E0L)$.

Note that this theorem compares classes of systems all of which use nonterminals for defining languages. Thus the only real difference (from the language generation point of view) between (the classes of) EIL, ET0L and E0L systems on the one hand and (the classes of) type-0, context-sensitive and context-free grammars respectively on the other hand is the parallel versus sequential way of rewriting strings. In this sense the above results tell us something about the role of parallel rewriting in generating languages by grammar-like devices. In the same direction we have another group of results of which the following two are quite representative.

<u>Theorem 3.2.</u> (Lindenmayer [32], Rozenberg and Doucet [48]).
A language is context-free if and only if it is the language of an EOL system $<\Sigma, P, \omega, \Delta>$ such that, for each a in Δ, the production $a \to a$ is in P.

<u>Theorem 3.3.</u> (Herman and Walker [20].)
A language is context-free if and only if it is the adult language of a 0L system.

As far as fitting some classes of L languages into the known formal language theoretic framework is concerned, results more detailed than those of Theorem 3.1 are available. For example we have the following results. Let $\mathcal{L}$(IND) denote the class of indexed languages and let $\mathcal{L}$(PROG) denote the class of Λ-free programmed languages.

<u>Theorem 3.4.</u> (Culik [3] and Rozenberg [46].)
$\mathcal{L}(\mathrm{ETOL}) \subseteq \mathcal{L}(\mathrm{IND})$ and $\mathcal{L}(\mathrm{ETOL}) \subsetneq \mathcal{L}(\mathrm{PROG})$.

Results like these can be helpful for getting either new properties or nice proofs of known properties of some classes of L languages. For example, the family $\mathcal{L}$(IND) possesses quite strong decidability properties which are then directly applicable to the class of ETOL languages. An example will be considered in section 8.

One of the most famous open problems in formal language theory, the LBA problem, has its counterpart also in the theory of L systems. An L system or scheme is propagating (abbreviated P) if the empty word does not occur on the right side of any production. This notion can be associated with any type of L system. Thus, we can speak of EPT0L systems and languages. An L system or L scheme is <u>deterministic</u> (abbreviated D) if whenever $<\alpha, a, \beta> \to \gamma_1$ and $<\alpha, a, \beta> \to \gamma_2$ are two productions (resp. two productions in the same table), then $\gamma_1 = \gamma_2$. Furthermore, we use the letter F to mean that, instead of one axiom, we may have a finite set of axioms in the system. The resulting language families are denoted using the letter $\mathcal{L}$ as before.

<u>Theorem 3.5.</u> (van Dalen [6], Vitanyi.)
The family $\mathcal{L}$(EPIL) equals the family of (Λ-free) context-sensitive languages. The family $\mathcal{L}$(EPDIL) equals the family of (Λ-free) deterministic context-sensitive languages.

The generative capacity of various mechanisms was already briefly discussed in Section 2.3. From the extensive literature in this area, we mention the following results, where also comparisons with the Chomsky hierarchy are taken into account. The results illustrate also the role of erasing, which will be further discussed in Section 5.

<u>Theorem 3.6.</u> (Vitanyi)
Every recursively enumerable language can be obtained from a language in $\mathcal{L}$(PD<1,0>L) by applying a homomorphism which maps each letter either to itself or to the empty word. On the other hand, the set of languages obtained from languages in $\mathcal{L}$(EPD<1,0>L) by applying nonerasing homomorphisms does not contain all regular languages.

<u>Theorem 3.7.</u> (Nielsen, Rozenberg, Salomaa and Skyum [36], Karhumaki [24].)

1) $\mathcal{L}(PD0L) \subsetneq \mathcal{L}(EPD0L) \subsetneq \mathcal{L}(CPD0L) \subsetneq \mathcal{L}(CD0L) \subsetneq \mathcal{L}(CDF0L) = \mathcal{L}(CPDF0L)$.

2) $\mathcal{L}(CF) \subsetneq \mathcal{L}(CPF0L)$.

The most interesting open problems in this area are whether or not the equations

$$\mathcal{L}(CPF0L) = \mathcal{L}(C0L) \; (= \mathcal{L}(E0L)), \quad \mathcal{L}(CPTF0L) = \mathcal{L}(CT0L) \; (= \mathcal{L}(ET0L))$$

hold true.

4. OTHER CHARACTERIZATIONS OF CLASSES OF L LANGUAGES WITHIN THE FRAMEWORK OF FORMAL LANGUAGE THEORY.

4.1 Closure properties.

A classical step toward achieving a mathematical characterization of a class of languages is to investigate its closure properties with respect to a number of operations. The next two results display the behaviour of some of the families of L languages with respect to the basic operations considered in AFL theory. There are essentially two reasons for considering these operations. One reason is that in this way we may contrast more sharply various families of L languages with traditional families of languages. The other reason is that we still know very little about what set of operations would be natural for families of L languages.

Theorem 4.1.1. (Rozenberg and Doucet [48], Rozenberg [40, 44], Rozenberg and Lee [26], Rozenberg, Ruohonen and Salomaa [50]).

None of the families of 0L, F0L, J0L, T0L, JT0L, IL, TIL languages is closed with respect to any of the following operations: $\cup$, $\cdot$, $*$, hom, hom^{-1}, $\cap_R$.

Theorem 4.1.2. (Rozenberg [46], van Dalen [6], Herman [16].)
The families of ET0L and EIL languages are closed with respect to all of the operations $\cup$, $\cdot$, $*$, hom, hom^{-1}, $\cap_R$. The family of E0L languages is closed with respect to the operations $\cup$, $\cdot$, $*$, hom and $\cap_R$ but it is not closed with respect to the hom^{-1} operation.

When we contrast the above two results with each other we see the role of nonterminals in defining languages of L systems. On the other hand contrasting the second result with the corresponding results for the classes of context-free and context-sensitive languages enables us to learn more about the nature of parallel rewriting in language generating systems.

4.2 Machine models.

It is customary in formal language theory to try to find, for a class of generative devices for languages, a class of recognition devices or acceptors which define the same collection of languages as the generative devices considered. Various classes of L languages have also been investigated from such a "programmer's point of view". Some of the machine models will be briefly mentioned in this chapter. It is obvious that a good machine model for a family gives a strong characterization of the family.

As regards machine models for L systems in general, the following point should be noticed. All reasonable machine models (at least in the classical sense) possess a finite-state control. Since the control can simulate a finite automaton, the family of acceptable languages will be closed under intersection with regular sets. If some L family does not have this closure property, one cannot expect to find for it a machine model reasonable in this sense.

We will now discuss the notion of a pre-set pushdown automaton. By presetting a pushdown automaton we mean that at the beginning of a computation a certain stack-square is allocated as the maximum location to which the stack may grow during the computation. The presetting remains unchanged in each particular computation. Formally, this is accomplished by letting a natural number n be one unchangeable item in instantaneous descriptions. The new structural feature is to let an overflow-indicator actively influence the computation. Formally, this means that there are two transition tables: δ (to be used when stack is not maximal) and δ_{top} (to be used when stack has reached the pre-set maximum). Moreover, empty locations in the stack should be preserved for counting purposes. An ordinary pushdown automaton is obtained as a special case where δ_{top} is empty. On the other hand, languages like $\{a^n b^n c^n \mid n \geq 1\}$ are easily shown acceptable by pre-set pushdown automata.

Two restrictions are now imposed on pre-set pushdown automata. The property of being locally finite means that there is a fixed bound on the length of local computations, i. e., computations with non-moving pointers. The finite return property means that there is a fixed bound on how many recursions there can be from a location.

Theorem 4. 2. 1. (van Leeuwen [27]).

The family of languages acceptable by locally finite pre-set pushdown automata with the finite return property equals the family $\mathcal{L}$(E0L). The family of languages acceptable by pre-set pushdown automata with the finite return property equals the smallest full AFL containing the family $\mathcal{L}$(0L).

A natural characterization for the family of languages accepted by unrestricted pre-set pushdown automata is obtained in Section 13. We conclude this Section 4. 2 by mentioning a few other machine models.

FM-and RM-automata introduced in [4] are machine models for the families mentioned in Theorem 4. 2. 1. Basically, they are pushdown automata writing on the stack pairs consisting of a letter and a natural number which indicates the "level" of the letter. Special restrictions of tally pushdown automata discussed in [60] can be used as acceptors for the families of 0L and <k, l>L languages. The point mentioned above concerning finite-state control is also discussed here in more detail. Finally, special cases of the PAC-machines (where PAC stands for the pushdown array of counters) introduced in [47] are acceptors for the families of E0L, F0L, and 0L languages.

4.3 Recurrence systems and recursion schemes.

In the following formal definition, we use the notion of an n-tuple standard function from [14, pp. 13-14].

Definition 4.3.1. A recurrence system is a construct $R = \langle \Sigma, f, \alpha \rangle$, where Σ is an alphabet, f is an n-tuple standard function over Σ and $\alpha = (\alpha_1, \ldots, \alpha_n)$ is an n-tuple of finite subsets of Σ^*. For $j \geq 0$, we define the n-tuple $\alpha^{(j)}$ of subsets of Σ^* by

$$\alpha^{(0)} = (\alpha_1^{(0)}, \ldots\ldots, \alpha_n^{(0)}) = \alpha,$$

$$\alpha^{(j+1)} = (\alpha_1^{(j+1)}, \ldots\ldots, \alpha_n^{(j+1)}) = f(\alpha^{(j)}).$$

The language L(R) of R is defined by $L(R) = \bigcup_{j=0}^{\infty} \alpha_1^{(j)}$.

Example 4.3.1. Let $R = \langle \{b, r, s, c, e, x\}, (f_1, f_2), (\phi, \{x\}) \rangle$ where $f_1(X_1, X_2) = br\, X_2\, s\, X_2\, c\, X_2\, e$ and $f_2(X_1, X_2) = x X_2$. Then $L(R) = \{ br\, x^j\, s\, x^j\, c\, x^j\, e : j > 0\}$.

Theorem 4.3.1. (Herman [17]). The family $\mathcal{L}$(E0L) equals the family of languages of recurrence systems.

Thus, the family of ALGOL-like languages (which equals $\mathcal{L}$(CF)) is a special case of recurrence languages, obtained by systems, where all components of α equal the empty set. Theorem 4.3.1 shows that $\mathcal{L}$(E0L) is a very natural extension of $\mathcal{L}$(CF) (and perhaps mathematically a more natural family altogether).

Recursion schemes introduced in [8] are systems of functional equations used to characterize $\mathcal{L}$(ET0L) and certain of its subclasses.

5. STRUCTURAL CONSTRAINTS ON L SYSTEMS

One of the possible ways of investigating the structure of any language generating device is to put particular restrictions directly on the definition of its various components and then to investigate the effect of these restrictions on the language generating power. Theorem Theorem 2.3.1 represents a result in this direction (it says for example that removing nonterminals from ETIL, EIL, ETOL and EOL systems decreases the language generating power of these classes of systems). Some results along these lines were already mentioned in section 3. Now we indicate some other results along the same lines.

The first of these results investigates the role of erasing productions in generating languages by the class of EOL systems.

<u>Theorem 5.1.</u> (Herman [17], van Leeuwen [27].)
A language K is an EOL language if and only if there exists an EPOL system G such that $K-\{\Lambda\} = L(G)$.

The result corresponding to Theorem 5.1 is valid also for ETOL systems.

Our next result discusses the need of "two-sided context" (more intuitively: "two-sided communication") in IL systems.

<u>Theorem 5.2.</u> (Rozenberg [44].)
There exists a language K such that K is a $<1,1>$ L language and for no $m \geq 0$ is K an $<m,0>$ L language or a $<0,m>$ L language.

Our last sample result in this line says that for the class of IL systems with two-sided context it is the amount of context available and not its distribution that matters as far as the language generating power is concerned.

Theorem 5.3. (Rozenberg [44].)

A language is an $<m,n>$ L language for some $m,n \geq 1$ if and only if it is a $<1,m+n-1>$ L language. For each $m \geq 1$ there exists a $<1,m+1>$ L language which is not a $<1,m>$ L language.

Definition 5.1. A language is a 1L language if it is an $<m,0>$ L language or a $<0,m>$ L language, for some $m \geqq 0$.

6. SQUEEZING SEQUENCES OUT OF L SYSTEMS

From a biological point of view the time-order of development is at least as interesting as the unordered set of morphological patterns which may develop. This leads to investigation of sequences of words rather than unordered sets of words (languages), which is a novel point in formal language theory. It turned out that investigation of sequences (of words) gives rise to a non-trivial and interesting mathematical theory (see, e.g., [19, 37, 38, 41, 53, 61, 63]).

The most natural way to talk about word sequences in the context of L systems is to consider deterministic L systems without tables and to take the exhaustive approach, which simply means to include in the sequence of a DIL system the ordered set of all words that the system generates (and in the order that these words are generated).

<u>Definition 6.1.</u> Let $G = \langle\Sigma, P, g, \omega\rangle$ be a DIL system. The <u>sequence of G</u>, denoted as $E(G)$, is defined by $E(G) = \omega_0, \omega_1, \ldots$ where $\omega_0 = \omega$ and for $i \geq 1$, $\omega_{i-1} \underset{G}{\Rightarrow} \omega_i$.

<u>Example 6.1.</u> Let $G = \langle\Sigma, P, g, \omega\rangle$ be a DIL system such that $\Sigma = \{a, b\}$, $\omega = baba^2$ and $P = \{\langle g, b, \Lambda\rangle \to ba, \langle a, b, \Lambda\rangle \to ba^2, \langle a, a, \Lambda\rangle \to a, \langle b, a, \Lambda\rangle \to a\}$. Then $E(G) = baba^2, ba^2ba^4, \ldots, ba^kba^{2k}, \ldots$.

<u>Definition 6.2.</u> Let s be a sequence of words. It is called a <u>DIL sequence</u> (resp. D0L sequence) if there exists a DIL system (resp. D0L system) G such that $s = E(G)$.

Obviously as in the case of L languages (see section 2) one can apply various mechanisms of squeezing sequences out of DIL systems.

Thus, in the obvious sense, we can talk about EDIL and EDOL sequences (when using nonterminals for defining sequences) or about CDIL and CDOL sequences (when using codings for defining sequences). Comparing the sequence generative power of these different mechanisms for sequence definition, we have, for example, the following result.

Theorem 6.1. (Nielsen, Rozenberg, Salomaa and Skyum [36].)
The family of DOL sequences is strictly included in the family of EDOL sequences, which in turn is strictly included in the family of CDOL sequences.

In the sequel we shall use the term L sequence to refer to any kind of sequences discussed in this section.

7. GROWTH FUNCTIONS

7.1. Definitions and basic problems

A particularly interesting aspect in the study of Lindenmayer systems is the theory of growth functions. The basic paper in this area is by Szilard [61]. In the theory of growth functions only the lengths of the words matter, no attention is paid to the words themselves. From the point of view of biology, this means making observations only about the number of cells a filamentous organism has. Growth functions fit into the framework of the theory of integral sequential word functions. The latter have been studied extensively in the past, e.g., in connection with probabilistic automata. This is also one of the areas where powerful mathematical tools (such as formal power series and difference equations) are applicable.

Consider any DOL, D1L, or DIL system defining a sequence of words $\omega_0, \omega_1, \omega_2, \ldots$, where ω_0 is the axiom. The function

$$f(n) = |\omega_n|$$

mapping the set of nonnegative integers into itself is termed the growth function associated with the system.

Example 7.1.1. We give a few examples of DOL growth functions. In each example, a is the axiom and the alphabet is implicitly defined by the productions. For the system determined by $a \to ab$, $b \to b$, we have $f(n) = (n+1)$. For the system determined by $a \to b$, $b \to ab$, we have $f(n)$ = n'th Fibonacci number. For the system defined by $a \to abc^2$, $b \to bc^2$, $c \to c$, we have $f(n) = (n+1)^2$. For the system defined by $a \to abd^6$, $b \to bcd^{11}$, $c \to cd^6$, $d \to d$, we have $f(n) = (n+1)^3$.

The following theorem is immediate by choosing t to be the length of the axiom and u to be the length of the longest right-hand side among the productions.

Theorem 7.1.1. For any L system, there are two constants u and t such that $f(n) \leqq tu^n$.

Thus, growth defined by an L system is at most exponential. Similarly, one sees that if propagating growth is not bounded by a constant, it has to be at least logarithmic. Consequently, functions like $\log\log n$ and 2^{2^n} cannot be growth functions associated with L systems. Whenever such growth is observed, it cannot be modelled by an L system.

We shall now introduce a classification of *growth types.* We use the numbers 3, 2, 1, 0 to denote what might be called the pure types. (In fact, we shall see in Section 7.2 that they are exactly the types possible in the DOL case.) In addition, we shall speak of the mixed types $2\frac{1}{2}$ and $1\frac{1}{2}$.

We say that a growth function $f(n)$ is of type 3 iff it is ultimately exponential, i.e., there are numbers $t > 1$ and n_0 such that $f(n) \geqq t^n$ for $n \geqq n_0$. (With biological connotations, exponential growth has sometimes been called "malignant".) A function $f(n)$ is of type 2 iff it is bounded from above by a polynomial and from below by a polynomial with positive leading coefficient of degree $\geqq 1$, of type 1 iff it is bounded from above by a constant and from below by a constant > 0, and of type 0 iff it becomes ultimately 0. A function $f(n)$ is of type $2\frac{1}{2}$ iff it is "between" the types 2 and 3, and of type $1\frac{1}{2}$ iff it is "between" the types 1 and 2. Of our examples above the second is of type 3, whereas the others are of type 2. The function $n^{\log n}$ is of type $2\frac{1}{2}$, and the functions $n^{\frac{1}{2}}$ and $\log n$ of type $1\frac{1}{2}$.

In addition to L systems we also consider L schemes. The *growth type combination* associated with a scheme S is the set of types of the growth functions associated with the systems $S(w)$ obtained from S by adding some axiom w. Thus, the DOL scheme determined by the productions $a \to a^2b$, $b \to bc$, $c \to cd$, $d \to \Lambda$ possesses growth type combination

3210, whereas the D0L scheme determined by the productions $a \to b$, $b \to ab$ possesses growth type combination 3.

We now list using standard automata theoretic terminology some basic problems concerning growth functions.

Analysis problem: Given a system, determine its growth function. A weaker variant is to determine only the type of the function. One may also ask for the growth type combination of a scheme.

Synthesis problem: Given a function f(n), determine if possible a system belonging to a given class of systems (say, D0L systems) whose growth function is f(n). An associated problem is the cell number minimization problem: realize a given growth function with a system (belonging to the class of systems considered) with minimal cardinality of the alphabet.

Growth equivalence problem: Given two systems (in the class of systems considered), determine whether their growth functions are the same.

As we shall see in the following sections, the basic problems have fairly nice solutions in the case of D0L systems but, for D1L and DIL systems, they are undecidable. In addition to these problems, various hierarchy problems have been studied. E.g., it may be the case that the set of growth functions generated by one class of systems is contained in the set of growth functions generated by another class of systems but the same does not hold true with the corresponding sets of sequences, [38].

7.2. DOL growth: equivalence, analysis, synthesis

To a DOL system $<\{a_1, \ldots, a_k\}, P, \omega>$, we associate the following matrices. The initial vector, π, is the k-dimensional row vector such that its i'th component equals the number of occurrences of the letter a_i in the axiom ω, for $i = 1, \ldots, k$. The final vector, η, is the k-dimensional column vector with all components equal to 1. The growth matrix, M, is the k-dimensional square matrix whose (i,j)'th entry equals the number of occurrences of a_j in the production for a_i, for $i, j = 1, \ldots, k$. These matrices are introduced because from the growth point of view the order of letters is immaterial.

Example 7.2.1. For the DOL system G_1 determined by the axiom a and the productions $a \rightarrow acde$, $b \rightarrow cde$, $c \rightarrow b^2d^2$, $d \rightarrow d^3$, $e \rightarrow bd$, we have

$$\pi = (10000),\ M = \begin{pmatrix} 10111 \\ 00111 \\ 02020 \\ 00030 \\ 01010 \end{pmatrix},\ \eta = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

Theorem 7.2.1. (Szilard [61], Paz and Salomaa [38].)
The growth function of a DOL system can be expressed in the form

$$f(n) = \pi M^n \eta.$$

Consequently, the generating function F(x) for the growth function can be expressed in the form

$$F(x) = \pi(I-Mx)^{-1}\eta,$$

where I is the identity matrix.

From the above expression, we can compute the generating function for the growth function of the system G_1 defined above:

$$F(x) = 1/(1-x)(1-3x).$$

Since the generating function is always a rational function, the DOL growth equivalence problem is reduced to deciding the identity of two polynomials:

Theorem 7.2.2. (Szilard [61], Paz and Salomaa [38].)
The growth equivalence problem for DOL systems is decidable.

Example 7.2.2. Let us consider another system G_2 with the axiom a and productions $a \to ab^3$, $b \to b^3$. Although it looks very different, it is growth equivalent to G_1 because we get exactly the same generating function.

In some sense, Theorem 7.2.1 also solves the analysis problem in the DOL case. However, more practical methods are obtained because the matrix representation gives rise to various strong mathematical characterizations. Some of the results are summarized in the following theorem.

Theorem 7.2.3. (Paz and Salomaa [38].)
The generating function for a DOL growth function is rational. Every DOL growth function satisfies a homogeneous difference equation with constant coefficients. The infinite Hankel matrix associated with a DOL growth function is of finite rank.

Considering the difference equation mentioned in Theorem 7.2.3, the following results are obtained.

Theorem 7.2.4. (Paz and Salomaa [38].)

Every D0L growth function is exponential, polynomial, or a combination of these. In particular, it is always of one of the types 3, 2, 1, 0. If f is a function such that for every integer n there are integers m and $i > n$ with the property

$$f(m+i) \neq f(m+n) = f(m+n-1) = \ldots = f(m)$$

then f is not a D0L growth function.

Theorem 7.2.4 can be used to construct examples of D1L growth functions which are not D0L growth functions. The best known among them is "Gabor's sloth" (due to Herman): A D1L system with the alphabet $\{a, b, c, d\}$, masker g, axiom ad, and productions given in the following table, where the row defines the left context and the column defines the symbol to be mapped.

	a	b	c	d
g	c	b	a	d
a	a	b	a	d
b	a	b	a	d
c	b	c	a	ad
d	a	b	a	d

The first few words in the sequence are

ad, cd, aad, cad, abd, cbd, acd, caad, abad,
cbad, acad, cabd, abbd, cbbd, acbd, cacd,
abaad, . . .

The lengths of the intervals in which the function stays constant not only grow beyond all bounds but they even grow exponentially. Examples like this show that the family of D1L growth functions properly contains the family of D0L growth functions.

As regards the synthesis of functions as D0L growth functions, efficient algorithms can be given for polynomials. A general algorithm is applicable whenever an upper bound for the Hankel rank of the function is known, [38].

Theorem 7.2.5. (Salomaa [53], Vitanyi [63].)
There is an algorithm for determining the growth type combination associated with a DOL scheme. In such a combination 2 never occurs without 1 but all other combinations are possible.

Theorem 7.2.6. (Pollul and Schütt [39].)
Assume that the generating function for a DOL growth function $f(n)$ equals $p(x)/q(x)$ where p and q are polynomials in least terms. Then f is of type 0 iff $q(x)$ is constant, and of type 3 iff $q(x)$ has a root of absolute value < 1.

As pointed out by Lindenmayer, [33], the biological significance of the mathematical results like those mentioned lies in the fact that for well-known growth functions, such as linear, square, cubic, exponential and S-shaped functions, one can study how classes of local cellular programs (DOL systems) can be found to realize such functions. Of course, the functions must be first digitized, i.e., some suitable discrete time steps must be chosen. There are usually valid biochemical or physiological grounds (e.g., diffusion and reaction rates, cell division rates) according to which length of time steps can be chosen. A consequence of Theorem 7.2.4 is that if a growth curve keeps rising without limit, but always slower and slower, then such a growth process cannot take place without interactions among the cells.

7.3. DIL growth

The existing rather comprehensive theory of D0L growth functions is due to the matrix representation of Theorem 7.2.1 and the resulting mathematical implications. No such theory exists for growth functions associated to systems with interactions. The situation is analogous to the corresponding difference between context-free and context-sensitive languages.

However, quite a number of specific examples have been constructed to yield general conclusions. As an over-all statement one can say that undecidability and great variety of possibilities are the characteristic features.

Theorem 7.3.1. (Vitanyi [64].)
The growth equivalence for PD1L systems is undecidable, and so is the problem of determining the type of a given PD1L growth function.

Theorem 7.3.2. (Karhumaki [22], Vitanyi [64].)
The types $1\frac{1}{2}$ and $2\frac{1}{2}$ are possible for PD1L growth.

Especially interesting is the example, [22], showing the existence of growth type $2\frac{1}{2}$. It is a complicated PD<1,1> L system realizing essentially the function $2^{n^{\frac{1}{2}}}$. By the results of [64], a PD1L system exists for this function.

It is obvious that any type combination among the numbers 3, 2, 1, 0 can occur for D1L schemes. The statement remains true even if $2\frac{1}{2}$ and $1\frac{1}{2}$ are added to the list, and even if a finer classification is introduced to replace type $1\frac{1}{2}$. Gabor's sloth is an example of logarithmic growth within the type $1\frac{1}{2}$. If you make the lengths of the constant intervals to grow in a linear (resp. quadratic) fashion, you get the growth function $n^{\frac{1}{2}}$ (resp. $n^{\frac{1}{3}}$) [30], [58]. Vitanyi [64] has generalized these results further to fractional powers. It is also possible, [23], to obtain growth directly corresponding to type $2\frac{1}{2}$, i.e., a growth function which is faster than logarithmic but slower than any fractional power.

Although all details around this matter have not yet been clarified, it seems reasonable to assume that the whole hierarchy of IL languages exemplified in Theorem 5.3 collapses as regards growth functions: D1L growth functions give you essentially everything.

7.4. Length sets

For a language L, we define its <u>length set</u>

$$\lg(L) = \{n \mid n = |w|, \text{ for some } w \in L\}.$$

Considering type X systems, we use the notation $\mathcal{L}\mathcal{G}(X)$ for the family of length sets associated with languages in $\mathcal{L}(X)$. It is well-known (e.g., consider regular and context-free languages) that a proper inclusion between two language families may reduce to the equality between the corresponding families of length sets. On the other hand, a proper inclusion between two length set families always implies a proper inclusion between the corresponding language families, provided inclusion has been shown by some other means. (This argument is used, e.g., in the proof of Theorem 3.7 (1).) To avoid trivial exceptions in the statement of the following results, we make the following definitional convention: Whenever a set S belongs to a family of length sets then also $S \cup \{0\}$ belongs to the same family.

<u>Theorem 7.4.1.</u> (Karhumaki)

$$\mathcal{L}\mathcal{G}(CF) \subsetneq \mathcal{L}\mathcal{G}(PD0L) \subsetneq \mathcal{L}\mathcal{G}(D0L) \subsetneq \mathcal{L}\mathcal{G}(DF0L) = \mathcal{L}\mathcal{G}(PDF0L) \subsetneq \mathcal{L}\mathcal{G}(P0L)$$
$$= \mathcal{L}\mathcal{G}(PF0L) \subseteq \mathcal{L}\mathcal{G}(0L) = \mathcal{L}\mathcal{G}(F0L) \subsetneq \mathcal{L}\mathcal{G}(DTF0L) = \mathcal{L}\mathcal{G}(DT0L) = \mathcal{L}\mathcal{G}(T0L)$$
$$= \mathcal{L}\mathcal{G}(TF0L) \subsetneq \mathcal{L}\mathcal{G}(CS)$$

Moreover, $\mathcal{L}\mathcal{G}(\mathrm{PF0L}) \subsetneq \mathcal{L}\mathcal{G}(\mathrm{PTF0L}) = \mathcal{L}\mathcal{G}(\mathrm{PT0L}) = \mathcal{L}\mathcal{G}(\mathrm{PDTF0L})$ $= \mathcal{L}\mathcal{G}(\mathrm{PDT0L}) \subseteq \mathcal{L}\mathcal{G}(\mathrm{T0L})$.

The most interesting open problem in this area is whether or not the inclusion

$$\mathcal{L}\mathcal{G}(\mathrm{P0L}) \subseteq \mathcal{L}\mathcal{G}(\mathrm{0L})$$

is proper.

A decidability problem concerning length sets will be briefly discussed in Section 8.

8. DECISION PROBLEMS

8.1. Some decidability and undecidability results

Decidability results for growth functions were already discussed in the previous section. We now extend this discussion to concern the standard problems studied in formal language theory.

Since the corresponding results hold for index languages, the following theorem is an immediate corollary of Theorem 3.4.

Theorem 8.1.1. Membership, emptiness and finiteness problems are decidable for the family $\mathcal{L}$(ETOL) (and, consequently, for all of its subfamilies), and so is the problem of deciding of an arbitrary language in the family and of an arbitrary word whether or not the word occurs (resp. occurs infinitely often) as a subword in a word in the language.

By an SF language we mean a language which equals the set of sentential forms of a context-free grammar. It is easy to see that the family of Λ-free linear SF languages is included in both of the families $\mathcal{L}$(POL) and $\mathcal{L}$(PDTOL).

Theorem 8.1.2. (Blattner [1], Rozenberg [42], Salomaa [54].) The equivalence problem for Λ-free linear SF languages is undecidable. Consequently, the equivalence problem for POL and PDTOL languages is undecidable.

Decision methods of an arithmetic nature can be obtained and normal form results, [59], concerning regular languages become useful in the case where the alphabet consists of one letter only. Such systems are called UL systems. In this case, it is obviously irrelevant whether the system is with or without interactions.

Theorem 8.1.3. (Herman, Lee, van Leeuwen and Rozenberg [18], Salomaa [55].)
The equivalence problem of UL languages is decidable. There is an algorithm for deciding whether a given UL language is regular and also whether a given regular language is UL.

Theorem 8.1.4. (Rozenberg)
It is undecidable whether an arbitrary IL (C0L, E0L, F0L) system generates a 0L language.

8.2. D0L equivalence problem

Without any doubt, the most intriguing open mathematical problem around L systems is the D0L equivalence problem. The first impression most people have had is that the undecidability result of Theorem 8.1.2 does not hold for D0L systems. However, the problem is still open. The following variations of the problem have been considered. Each of the variations can be stated for PD0L systems as well. (This is, of course, a special case of the variation stated.)

(i) The language equivalence problem for D0L systems (i.e., given two systems, one has to decide whether the generated languages coincide).
(ii) The sequence equivalence problem for D0L systems.
(iii) The Parikh language equivalence problem for D0L systems. (As usual, the Parikh vector associated to a word indicates the number of occurrences of each letter in the word. The Parikh language, resp. Parikh sequence, is the set, resp. the sequence of Parikh vectors associated to words in a language, resp. in a sequence.)

(iv) The Parikh sequence equivalence problem for D0L systems.

(v) The growth language equivalence problem for D0L systems (i.e., whether or not the ranges of the growth functions coincide).

(vi) The growth equivalence problem for D0L systems.

Considering the matrix representation for growth functions and making an appropriate change in the final vector η, it is seen that (iv) and (vi) are decidable.

Theorem 8.2.1. (Nielsen [34].)
Problem (iii) is decidable. Problem (i) is decidable if and only if problem (ii) is decidable.

The second sentence of Theorem 8.2.1 holds for PD0L systems as well, [34]. Problem (v) is open for the general case but has been shown decidable for PD0L systems, [34].

Thus, to solve the language equivalence problem, it suffices to solve the sequence equivalence problem. For the solution of the latter, it suffices to determine a constant k (depending, for instance, on the cardinality of the alphabet) such that if the sequences coincide with respect to first k words, they coincide altogether. However, finding such a constant is not easy even for PD0L sequences over two-letter alphabet. It does not suffice to choose k equal to the cardinality of the alphabet, as seen by considering the following example.

Example 8.2.1. Consider two D0L systems G_1 and G_2 over the alphabet $\{a,b\}$ and with the axiom ab. The productions for G_1 (resp. G_2)

are $a \to abb$ and $b \to aabba$ (resp. $a \to abbaabb$ and $b \to a$). The sequences coincide with respect to the first three words only.

A modification of this example shows that it is possible to construct two DOL systems over an alphabet with an even cardinality n such that their sequences coincide with respect to the first $3n/2$ words but not after that. These considerations may be contrasted with the following result for growth functions.

<u>Theorem 8.2.2.</u> (Karhumaki)
Assume that n_1 (resp. n_2) is the cardinality of the alphabet of a DOL system G_1 (resp. G_2). G_1 and G_2 are growth equivalent if and only if for each $i = 1, \ldots, n_1+n_2+1$, the i'th word in the sequence of G_1 is of the same length as the i'th word in the sequence of G_2.

<u>Theorem 8.2.3.</u> (Ehrenfeucht and Rozenberg)
The equivalence problem is decidable for DOL systems with growth type $\leqq 2$.

<u>Theorem 8.2.4.</u> (Johansen and Meiling [21].)
Assume that G_1 (resp. G_2) is a DOL system such that the free group generated by $L(G_1)$ (resp. $L(G_2)$) is finitely generated. Then it is decidable whether $L(G_1) = L(G_2)$.

We conjecture that the equivalence problem for DOL systems is decidable. Combining Theorem 8.2.3. with the results in [34], it suffices to consider conservative and growing sequences (i.e., each letter occurs in every word and the sequence of Parikh vectors is strictly growing) which, furthermore, are of growth type 3.

9. GLOBAL VERSUS LOCAL BEHAVIOUR OF L SYSTEMS

The topic discussed in this section, global versus local behaviour of L systems, is undoubtedly one of the most important in the theory of L systems. Roughly speaking, a global property of an L system is a property which can be expressed independently of the system itself (for example a property expressed in terms of its language or sequence). On the other hand a local property of an Lsystem is a property of its set of productions (for example a property of the "graph" of productions of a given system). In a sense the whole theory of L systems emerged from an effort to explain on the local (cellular) level global properties of development.

As an example of research in this direction we discuss the so-called locally catenative L systems and sequences (see [49]). Locally catenative L sequences are examples of **L** sequences in which the words themselves carry in some sense the history of their development.

<u>Definition 9.1.</u> An infinite sequence of words $\tau_0, \tau_1, \ldots$ is called <u>locally catenative</u> if there exist positive integers $m, n, i_1, \ldots, i_n$ with $n \geq 2$ such that for each $j \geq m$ we have $\tau_j = \tau_{j-i_1} \tau_{j-i_2} \cdots \tau_{j-i_n}$.

<u>Definition 9.2.</u> A DIL (or a D0L) system G is called <u>locally catenative</u> if $E(G)$ is locally catenative.

Very little is known about locally catenative DIL sequences. For locally catenative D0L sequences some interesting results are available. Our first result presents a property of a D0L sequence which is equivalent to the locally catenative property.

Let G be a DOL system such that $E(G) = \omega_0, \omega_1, \ldots$ is a doubly infinite sequence, meaning that the set of different words occurring in E(G) is infinite. We say that E(G) is covered by one of its words if there exist $k \geq 0$ and $j \geq k+2$ and a sequence s of occurrences of ω_k in (some of the) strings $\omega_{k+1}, \omega_{k+2}, \ldots, \omega_{j-1}$ such that ω_j is the catenation of the sequence of its subwords derived from respective elements of s.

Theorem 9.1. (Rozenberg and Lindenmayer [49].)
A DOL system G is locally catenative if and only if E(G) is covered by one of its words.

Our next theorem presents the result of an attempt to find a "structural" property of the set of productions of a DOL system such that its sequence is locally catenative. First we need some more notation and terminology.

If $G = <\Sigma, P, \omega>$ is a DOL system then the graph of G is the directed graph whose nodes are elements of Σ and for which a directed edge leads from the node a to the node b if and only if $a \rightarrow \alpha b \beta$ is in P for some words α, β over Σ.

Theorem 9.2. (Rozenberg and Lindenmayer [49].)
Let $G = <\Sigma, P, \omega>$ be a PDOL system such that L(G) is infinite, ω is in Σ and each letter from Σ occurs in a word in L(G). If there exists σ in Σ such that $\omega \overset{*}{\underset{G}{\Rightarrow}} \sigma$ and each cycle in the graph of G goes through the node σ then E(G) is locally catenative.

We may note that neither of the above results is true in the case of DIL sequences (systems).

10. DETERMINISTIC VERSUS NONDETERMINISTIC BEHAVIOUR OF L SYSTEMS

An L system is called deterministic if, roughly speaking, after one of its tables has been chosen, each word can be rewritten in exactly one way. Investigation of the role the deterministic restriction plays in L systems is an important and quite extensively studied topic in the theory of L systems (see, e.g., [7, 9, 25, 34, 38, 41, 53, 61]). First of all, some biologists claim that only deterministic behaviour should be studied. Secondly, studying deterministic L system especially when opposed to general (undeterministic) L systems, allows us to understand more clearly the structure of L systems. Finally, the notion of determinism studied in this theory differs from the usual one studied in formal language theory. One may say that they are dual to each other: "deterministic" in L systems means a deterministic process of generating strings, "deterministic" in the sense used in formal language theory means a deterministic process of parsing. Contrasting these notions may help us to understand some of the basic phenomena of formal language theory.

As an example of a research towards understanding deterministic restriction in L systems we shall discuss deterministic TOL systems.

It is not difficult to construct examples of languages which can be generated by a TOL system but cannot be generated by a DTOL system. One would like however to find a nontrivial (and hopefully interesting) property which would be inherent to the class of deterministic TOL languages. It turns out that observing the sets of all subwords generated by DTOL systems provides us with such a property. In fact the ability to generate an arbitrary number of subwords of an arbitrary length is a property of a TOL system which disappears when the deterministic restriction is introduced. More precisely, we have the following result. (In what follows $\pi_k(L)$ denotes the number of subwords of length k that occur in the words of L).

Theorem 10.1. (Ehrenfeucht and Rozenberg [9].)
Let Σ be a finite alphabet such that $\#\Sigma = n \geq 2$. If L is a language generated by a DTOL system, $L \subseteq \Sigma^*$, then $\lim\limits_{k\to\infty} \frac{\pi_k(L)}{n^k} = 0$.

Recently, results concerning the decomposition of some ETOL languages into EDTOL languages have been obtained. Along these lines, a rather intricate problem has been solved (see Theorem 3.4).

Theorem 10.2. (Ehrenfeucht and Rozenberg [13].)
The family of ETOL languages is contained properly in the family of indexed languages.

11. L TRANSFORMATIONS

An L system consists of an L scheme and of a fixed word (the axiom). An L scheme by itself represents a transformation (a mapping) from Σ^+ into Σ^* (where Σ is the alphabet of the L scheme). From the mathematical point of view, it is most natural to consider such transformations. This obviously may help to understand the nature of L systems. Although not much is known in this direction yet, some results about TOL transformations are already available.

Let a TOL scheme $G = <\Sigma, \mathcal{P}>$ be given. (Note that each table P of $\mathcal{P}$ is in fact a finite substitution, or a homomorphism in the case that $\mathcal{P}$ satisfies a deterministic restriction.) The basic situation under examination consists of being given two of the following three sets: a set L_1 of (start) words over Σ, a set L_2 of (target) words over Σ, and a (control) set ζ of finite sequences of applications of tables from $\mathcal{P}$. The problem is to ascertain information about the remaining set. (Note that we can consider a sequence of elements from $\mathcal{P}$ either as a word over $\mathcal{P}^*$, called a control word, or a mapping from Σ^* into Σ^*. We shall do both in the sequel but this should not lead to confusion.) The following are examples of known results concerning this problem.

<u>Theorem 11.1.</u> (Ginsburg and Rozenberg [15].)
If L_2 is a regular language, and L_1 an arbitrary language then the set ζ of control words leading from L_1 to L_2 is regular.

<u>Theorem 11.2.</u> (Ginsburg and Rozenberg [15].)
If L_2 is a regular language and ζ is an arbitrary set of control words then the set of all words mapped into L_2 by ζ is regular.

Theorem 11.3. (Ginsburg and Rozenberg [15].)
If L_1 is a regular language and ζ is a regular set of control words then the set of all words obtained from the words of L_1 by applying mappings from ζ is an ET0L language. Moreover each ET0L language can be obtained in this fashion.

Theorem 11.4. (Ginsburg and Rozenberg [15].)
There is no T0L scheme $S = <\Sigma, \mathcal{P}>$ such that $\mathcal{P}^*$ is the set of all finite nonempty substitutions on Σ^*. There is no T0L scheme $S = <\Sigma, \mathcal{P}>$ such that $\mathcal{P}^*$ is the set of all homomorphisms on Σ^*.

Theorem 11.5. The family of E0L languages is not closed under E0L transformations. The family of ET0L languages is closed under ET0L transformations.

As regards control devices, one can also take an approach different from Theorems 11.1-11.3 and study various kinds of control devices, as customary in the theory of formal grammars. Such a study was initiated in [43]. The most complete account on this topic is the recent paper [35]. Two sets of results in [35] can be outlined as follows: (i) The generative capacity due to various control mechanisms is in many cases reverse for L systems and grammars, i.e., a mechanism giving rise to weak generative capacity for L systems gives rise to strong capacity for grammars, and vice versa, and (ii) Some of the problems concerning "appearance checking", which are still open for grammars, have been solved for L systems.

In Section 2.2.3, we discussed adult languages of L systems. One may also consider adult languages of L schemes. They are naturally defined as fixed points of the corresponding L transformation. We mention the following result in this direction.

Theorem 11.6. (Walker)
The family of adult languages of IL schemes is properly contained in the family of regular languages.

12. GETTING DOWN TO PROPERTIES OF SINGLE L LANGUAGES OR SEQUENCES

One of the aims of the theory of L systems is to understand the structure of a single L language or sequence. Although some results in this direction are already available, there is not enough work done on this (rather difficult) topic.

It is in general difficult to prove results of the form that a certain language does not belong to a certain family because such a proof involves an investigation through the whole class of devices generating the family. This remark holds true also with respect to L families. It is, therefore, of special interest to prove results to the effect that all languages within a certain family possess some specific property. If the particular language under investigation does not have this property, it can then be immediately concluded that it is not in the family. We will mention some of such characterization results for the families of ET0L, EDT0L and E0L languages. The results do not give a full characterization of the families. However, in many interesting cases, they can be used to show that a particular language is not in the family.

For a set B of letters and a word x, we denote by $\#_B(x)$ the total number of occurrences of the letters belonging to B in x. If L is a language over an alphabet Σ and B is a nonempty subset of Σ, then

(i) B is called <u>nonfrequent</u> in L iff there exists a constant $C(B,L)$ such that for every x in L, $\#_B(x) < C(B,L)$,

(ii) B is called <u>rare</u> in L iff, for every natural number k, there exists a natural number n_k, such that whenever $n > n_k$ and a word x in L contains n occurrences of letters from B, then each two such occurrences lie at a distance $\geqq k$ from each other.

Theorem 12.1. (Ehrenfeucht and Rozenberg [11].)
If L is an ETOL language over an alphabet Σ and B is a nonempty subset of Σ which is rare in L, then B is nonfrequent in L.

For a word x, we define $\mu(x)$ as the minimal natural number n such that any two non-overlapping subwords of length n of x are different. A language L is called exponential iff there exists a natural number $C > 1$ such that whenever $x_1, x_2 \in L$ and $|x_1| > |x_2|$, then $|x_1| \geqq C|x_2|$.

Theorem 12.2. (Ehrenfeucht and Rozenberg [11].)
If L is an exponential language in the family EDTOL, then there exists a natural number C(L) such that every nonempty word x in L satisfies

$$|x|/\mu(x) < C(L).$$

Theorem 12.2 can be used to show that there is a 0L language which is not in $\mathcal{L}$(EDTOL), for instance, the language

$$\{x \in \{a,b\}^* \mid \text{ for some } n \geqq 0,\ |x| = 2^n\} - \{b\}.$$

Consider again a language L over an alphabet Σ and a nonempty subset B of Σ. Define

$$N(L,B) = \{n \mid \text{ for some } x \text{ in } L,\ \#_B(x) = n\}.$$

We say that B is numerically dispersed in L iff N(L,B) is infinite and, for every natural number k, there exists a natural number n_k such that whenever u_1 and u_2 are in N(L,B), and $u_1 > u_2 > n_k$, then $u_1 - u_2 > k$. B is clustered in L iff N(L,B) is infinite and there exist natural numbers k_1, k_2 both larger than 1 such that whenever a word x in L satisfies $\#_B(x) \geqq k_1$, then x contains at least two occurrences of letters from B which lie at a distance $< k_2$ from each other.

Theorem 12.3. (Ehrenfeucht and Rozenberg [11].)
Let L be an E0L language over Σ and B a nonempty subset of Σ. If B is numerically dispersed in L, then B is clustered in L.

For D0L sequences we have the following result. (In what follows if x is a word and k a positive integer then $\mathrm{Pref}_k(x)$ denotes either x itself if $k \geq |x|$ or the word consisting of the first k letters of x if $k < |x|$. Similarly $\mathrm{Suf}_k(x)$ denotes either x itself if $k \geq |x|$ or the word consisting of the last k letters of x if $k < |x|$.)

Theorem 12.4. (Rozenberg [41].)
For every D0L system G such that $E(G) = \omega_0, \omega_1, \dots$ is infinite there exists a constant C_G such that for every integer k the sequence $\mathrm{Pref}_k(\omega_0), \mathrm{Pref}_k(\omega_1), \dots$ (resp. $\mathrm{Suf}_k(\omega_0), \mathrm{Suf}_k(\omega_1), \dots$) is ultimately periodic with period C_G.

13. GENERALIZING L SYSTEM IDEAS: TOWARDS A UNIFORM FRAMEWORK

It was noticed already in the early papers on L systems (e.g., cf. [40]) that the underlying operation is that of iterated substitution. However, in the theory of L systems it occurs in a somewhat modified way. The notion of a K-iteration grammar introduced in this section generalizes this idea. Moreover, it can be used to illustrate the following point.

One of the first observations concerning L systems was that L families have very weak closure properties. In fact, many of the families are anti-AFL's, i.e., closed under none of the AFL operations. This phenomenon is due to the lack of a terminal alphabet rather than to parallelism which is the essential feature concerning L systems. The notion of a K-iteration grammar can be used to convert language families with weak closure properties into full AFL's in a rather natural way. Furthermore, it provides a uniform framework for discussing all context-free variations of 0L systems and shows the relation between 0L systems and (iterated) substitutions.

<u>Definition 13.1.</u> Let K be a family of languages. A <u>K-iteration</u> grammar is a quadruple $G = (V_N, V_T, S, U)$, where V_N and V_T are disjoint alphabets (of nonterminals and terminals), $S \in V^+$ with $V = V_N \cup V_T$ (initial word), and $U = \{\sigma_1, \ldots, \sigma_n\}$ is a finite set of K-substitutions defined on V and with the property that, for each i and each $a \in V$, $\sigma_i(a)$ is a language over V. The language generated by such a grammar is defined by

$$L(G) = \cup \ \sigma_{i_1} \ldots \sigma_{i_k}(S) \cap V_T^*$$

where the union is taken over all integers $k \geqq 1$ and over all k-tuples $(i_1, \ldots, i_k)$ with $1 \leqq i_j \leqq n$. The family of languages generated by K-iteration grammars is denoted by K_{iter}. For $t \geqq 1$, we denote by $K^{(t)}_{iter}$ the subfamily of K_{iter}, consisting of languages generated by such grammars where U consists of at most t elements.

The different variations of 0L families can now be easily characterized within this framework. Denote by F the family of finite languages. Clearly, $F^{(1)}_{iter} = \mathcal{L}(E0L)$ and $F_{iter} = \mathcal{L}(ET0L)$. The families with D and/or P are charaterized as follows. D means that the σ's are homomorphisms. P means that the σ's are Λ-free. Thus, $\mathcal{L}(EPDT0L)$ is the subfamily of F_{iter}, obtained by such grammars where all substitutions σ are Λ-free homomorphisms. If one wants to consider families without E (0L, T0L, etc.) , then one simply assumes that V_N is empty. Thus , all L systems without interactions find their counterpart in this formalism, which can be extended to cover IL languages, [66]. Note, however, that so far one has not considered in the theory of L systems cases more general than $K = F$.

In the next theorem we obtain a natural characterization for the family $R^{(1)}_{iter}$, where R is the family of regular languages.

<u>Theorem 13.1.</u> (van Leeuwen [28], Christensen [2].)
The family $R^{(1)}_{iter}$ equals the family of languages accepted by pre-set push-down automata. This family lies strictly between the families $\mathcal{L}(E0L)$ and $\mathcal{L}(ET0L)$.

Theorem 13.2. (van Leeuwen [27], Salomaa [57].)
Assume that the family K contains all regular languages and is closed under finite substitution and intersection with regular languages. Then the families K_{iter} and $K_{iter}^{(t)}$ are full AFL's.

Every cone satisfies the conditions required for K in the preceding theorem. The resulting full AFL's are naturally called Lindenmayer AFL's. Apart from the inclusion relations obvious by the definitions, very little is known about these AFL's.

A very natural notion in AFL-theory arising from L systems is the notion of a hyper-AFL, [28, 57]. By definition, a language family K is a hyper-AFL iff K satisfies the requirements of Theorem 13.2 and, moreover, $K_{iter} = K$.

Theorem 13.3. (Christensen [2])
The family $\mathcal{L}$(ET0L) is the smallest hyper-AFL.

This interesting mathematical property gives additional significance to problems concerning $\mathcal{L}$(ET0L). It might be mentioned that indexed languages form also a hyper-AFL.

Since K-iteration grammars provide a uniform framework, they can be used to generalize specific results. A number of such general results already exists, [30]. We mention here the result generalizing the important Theorem 2.3.2. (1).

Definition 13.2. Let K be a family of languages. A language L is hyper-sentential over K if there exist a K-iteration grammar G and a language $L_1 \in K$ such that

$$L = \{y : x \overset{*}{\underset{G}{\Rightarrow}} y \text{ for some } x \in L_1\}.$$

A language L is hyper-algebraic over K if there exist an alphabet Σ and a language L_1 hyper-sentential over K such that $L = L_1 \cap \Sigma^*$.

Theorem 13.4. (van Leeuwen and Wood, [30])
Assume that K is closed under K-substitution, marked K-substitution and the operation $\cap \Sigma^*$, and that K contains singleton sets. Then each language L hyper-algebraic over K is of form $L = h(L_1)$, where L_1 is a language hyper-sentential over K and h is a homomorphism.

Another generalization (in the direction of Ω-algebras) is presented in [62]. This might be a proper way of introducing a truly many-dimensional theory of L systems.

14. CONCLUSIONS

The following two concluding remarks are in order.

(1) In the first five years of its existence the mathematical theory of L systems has become each year more and more fruitful and popular. This is exemplified by exponential growth of the number of papers produced (per year), and a linear (with a decent coefficient) growth of both the number of results and the number of people joining the area.

(2) It may have already occurred to the reader (and it is certainly clear to the authors of this paper) that both formal language theory and the theory of L systems have benefited by the existence of the other.

ACKNOWLEDGEMENT

The authors are indebted to P. Vitanyi, A. Szilard and D. Wood for their useful comments on the first version of this paper.

REFERENCES.

[1] M. Blattner, The unsolvability of the equality problem for sentential forms of context-free grammars, Journal of Computer and System Sciences, 1973, v. 7, 463-468.

[2] P. A. Christensen, Hyper AFL's and ETOL systems, Masters' thesis, University of Aarhus, 1974.

[3] K. Culik II, On some families of languages related to developmental systems, University of Waterloo Technical Report CS-73-12.

[4] K. Culik II and J. Opatrný, Macro OL systems, submitted for publication.

[5] K. Culik II and J. Opatrný, Literal homomorphisms of OL languages, University of Waterloo Technical Report CS-73-25.

[6] D. van Dalen, A note on some systems of Lindenmayer, Mathematical Systems Theory, 1971, v. 5, 128-140.

[7] P. Doucet, On the membership question in some Lindenmayer systems, Indagationes Mathematicae, 1972, v. 34, 45-52.

[8] P. J. Downey, Developmental systems and recursion schemes, Proc. Conf. Biol. Motiv. Automata Theory, IEEE, 1974, 54-58.

[9] A. Ehrenfeucht and G. Rozenberg, A limit theorem for sets of subwords in deterministic TOL systems, Information Processing Letters, 1973, v. 2, 70-73.

[10] A. Ehrenfeucht and G. Rozenberg, The equality of EOL languages and codings of OL languages, to appear in International Journal of Computer Mathematics.

[11] A. Ehrenfeucht and G. Rozenberg, Three useful results concerning L languages without interactions, appears in [51].

[12] A. Ehrenfeucht and G. Rozenberg, Trade-off between the use of nonterminals, codings and homomorphisms in defining languages for some classes of rewriting systems.
Proc. 2nd Coll. Automata, Languages, Programming (Saarbrücken), Springer Lecture Notes in Computer Science.

[13] A. Ehrenfeucht and G. Rozenberg, On decomposing some ETOL languages into deterministic ETOL languages, submitted for publication.

[14] S. Ginsburg, The Mathematical Theory of Context-Free Languages, McGraw-Hill, 1966.

[15] S. Ginsburg and G. Rozenberg, TOL systems and control sets, submitted for publication.

[16] G. T. Herman, Closure properties of some families of languages associated with biological systems, Information and Control, 24, 1974, 101-121.

[17] G. T. Herman, A biologically motivated extension of Algol-like languages, Information and Control, 1973, v. 22, 487-502.

[18] G. T. Herman, K. P. Lee, J. van Leeuwen and G. Rozenberg, Characterization of unary developmental languages, Discrete Mathematics, 1973, v. 6, 235-247.

[19] G. T. Herman and G. Rozenberg, Developmental systems and languages, to be published by North-Holland Publishing Company.

[20] G. T. Herman and A. Walker, Context-free languages in biological systems, to appear in International Journal of Computer Mathematics.

[21] P. Johansen and E. Meiling, Free groups in Lindenmayer systems, appears in [51].

[22] J. Karhumäki, An example of a PD2L system with the growth type 2 1/2, Information Processing Letters, 1973, v. 2, 131-134.

[23] J. Karhumäki, Some growth functions of context-dependent L-systems, appears in [51].

[24] J. Karhumäki, The family of PDOL growth-sets is properly included in the family of DOL growth-sets, to appear in Ann. Acad. Sci. Fennicae.

[25] K. P. Lee and G. Rozenberg, The length sets of DOL languages are uniformly bounded, submitted for publication.

[26] K. P. Lee and G. Rozenberg, TIL systems and languages, submitted for publication.

[27] J. van Leeuwen, Pre-set pushdown automata and OL grammars, University of California, Berkeley, Computer Science Technical Report 10, 1973.

[28] J. van Leeuwen, Hyper AFL's and all preset PDA languages are indexed, memorandum, 1973.

[29] J. van Leeuwen, F-iteration languages, extended abstract, 1973.

[30] J. van Leeuwen and D. Wood, A decomposition theorem for hyper-algebraic extensions of language families, submitted for publication.

[31] A. Lindenmayer, Mathematical models for cellular interactions in development, Parts I and II, Journal of Theoretical Biology, 1968, v. 18, 280-315.

[32] A. Lindenmayer, Developmental systems without cellular interactions, their languages and grammars, Journal of Theoretical Biology, 1971, v. 30, 455-484.

[33] A. Lindenmayer, Growth functions of multicellular organisms and cellular programs, Abstracts of the 10th Symposium on Biomathematics and Computer Science in the Life Sciences, Houston, March, 1973.

[34] M. Nielsen, On the decidability of some equivalence problems for DOL systems, Information and Control, 25, 1974, 166-193.

[35] M. Nielsen, EOL and ETOL systems with control devices, submitted for publication.

[36] M. Nielsen, G. Rozenberg, A. Salomaa and S. Skyum, Nonterminals, homomorphisms and codings in different variations of OL systems. Parts I and II, submitted for publication.

[37] A. Paz, Similarity in DTOL and related problems, State University of New York at Stony Brook, Department of Computer Science, technical report No. 15, 1973.

[38] A. Paz and A. Salomaa, Integral sequential word functions and growth equivalence of Lindenmayer systems, Information and Control, 1973, v. 23, 313-343.

[39] W. Pollul and D. Schütt, Growth in DOL systems, submitted for publication.

[40] G. Rozenberg, TOL systems and languages, Information and Control, 1973, v. 23, 357-381.

[41] G. Rozenberg, DOL sequences, Discrete Mathematics, 1974, v. 7, 323-347.

[42] G. Rozenberg, Direct proofs of the undecidability of the equivalence problem for sentential forms of linear context-free grammars and the equivalence problem for OL systems, Information Processing Letters, 1972, v. 1, 233-235.

[43] G. Rozenberg, On OL-systems with restricted use of productions, to appear in Journal of Computer and System Sciences.

[44] G. Rozenberg, L-systems with interactions: the hierarchy, Departmental report 28-72, Department of Computer Science, State University of New York at Buffalo, 1972.

[45] G. Rozenberg, Propagating L-systems with interactions, Departmental report 29-72, Department of Computer Science, State University of New York at Buffalo, 1972.

[46] G. Rozenberg, Extensions of tabled OL-systems and languages, International Journal of Computer and Information Sciences, 1973, v. 2, 311-334.

[47] G. Rozenberg, On a family of acceptors for some classes of developmental languages, to appear in International Journal of Computer Mathematics.

[48] G. Rozenberg and P. Doucet, On OL-languages, Information and Control, 1971, v. 19, 302-318.

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

[50] G. Rozenberg, K. Ruohonen and A. Salomaa, Developmental systems with fragmentation, submitted for publication.

[51] G. Rozenberg and A. Salomaa (ed.), "L Systems", Springer Lecture Notes in Computer Science, Vol. 15, 1974.

[52] K. Ruohonen, Developmental systems with interactions and fragmentation, submitted for publication.

[53] A. Salomaa, On exponential growth in Lindenmayer systems, Indagationes Mathematicae, 1973, v. 35, 23-30.

[54] A. Salomaa, On sentential forms of context-free grammars, Acta Informatica, 1973, v. 2, 40-49.

[55] A. Salomaa, Solution of a decision problem concerning unary Lindenmayer systems, to appear in Discrete Mathematics.

[56] A. Salomaa, "Formal Languages", Academic Press, 1973.

[57] A. Salomaa, Macros, iterated substitution and Lindenmayer AFL's, University of Aarhus, Computer Science Department, technical report No. 18, 1973.

[58] A. Salomaa, On some recent problems concerning developmental languages, Proceedings of the Conference on Automata and Formal Languages, Bonn, 1973, Springer Lecture Notes in Computer Science, Vol. 2, 23-34.

[59] A. Salomaa, Theory of Automata, Pergamon Press, 1969.

[60] W. J. Savitch, Some characterizations of Lindenmayer systems in terms of Chomsky-type grammars and stack machines, submitted for publication.

[61] A. L. Szilard, Growth functions of Lindenmayer systems, University of Western Ontario, Computer Science Department, Technical Report No. 4, 1971.

[62] A. L. Szilard, On the algebraic foundations of developmental systems, Thesis, University of Western Ontario, 1974.

[63] P. Vitányi, Structure of growth in Lindenmayer systems, Indagationes Mathematicae, 1973, v. 35, 247-253.

[64] P. Vitányi, Growth of strings in context dependent Lindenmayer systems, appears in [51.].

[65] A. Walker, Dynamically stable strings in developmental systems, self repair, and the Chomsky hierarchy, Proc. Conf. Biol. Motiv. Automata Theory, IEEE, 1974, 46-49.

[66] D. Wood, A note on Lindenmayer systems, spectra and equivalence, McMaster Univ. Computer Science, Technical Report 74/1, 1974.

Since our presentation below (Definition 1. 1) begins with the most general type of an L system, we will describe here briefly and in very informal terms the basic notions about L systems. We feel such a description will be useful expecially for a reader who is less acquainted with formal language theory.

The essential feature about L systems, as opposed to grammars, is that the rewriting of a string happens in a parallel manner, contrary to the sequential rewriting in grammars, This means that at every step of the rewriting process according to an L system every letter has to be rewritten. One step of the rewriting process according to a grammar changes only some part of the string considered.

Let us consider a very simple example. Assume that we are dealing with a context-free grammar containing the production $S \rightarrow SS$. Then, starting from S , we get any string of the form S^n , where $n \geqq 1$. This follows because at one step of the rewriting process we can replace one occurrence of S by SS and leave the other occurrences unchanged. Assume next that we are dealing with an L system containing the production $S \rightarrow SS$. Then, starting from S , we get by this production only strings of the form S^{2^n} , $n \geqq 0$. This follows because we cannot leave occurrences of S unchanged. Thus, if we are rewriting the string SS , we obtain at one step the string $SSSS = S^4$, and not the string S^3 . On the other hand, if our L system contains also the production $S \rightarrow S$ then we can derive any string of the form S^n , $n \geqq 1$.

This parallelism in rewriting reflects the basic biological motivation behind L systems. We are trying to model the development of an organism. The development takes place in a parallel way, simultaneously everywhere in the organism. Sequential rewriting is not suitable for this modeling.

The simplest version of L systems assumes that the development of a cell is free of influence of other cells. This type of L systems is cus~~o~~tomarily called a <u>OL system</u>. ("O" stands for zero-sided communication between cells.) By definition, a OL-system is a triple $G = (\Sigma, \omega, P)$, where Σ is an alphabet, ω is a word over Σ , and P is a finite set of rewriting rules of the form

$$a \to x \quad , \quad a \in \Sigma \quad , \quad x \in \Sigma^*$$

(It is also assumed that P contains at least one rule for each letter of Σ .) The language of G consists of all words which can be derived from ω using rules of P in the parallel way. (The meaning of this should be clear enough. The formal definition in terms of the yield-relation will be given in Section 1.)

As an example, consider the OL-system

$$(\{a,b\}\ a,\ \{a \to b, b \to ab\}) \ .$$

The first few words in the generated language are

$$a, b, ab, bab, abbab, bababbab, abbabbababbab \ .$$

Since the system is <u>deterministic</u> (there is only one production for each letter), its language is generated as a <u>sequence</u> in a unique way. The mathematically minded reader will also notice that the lengths of the words in this sequence form the famous Fibonacci sequence. In fact, our OL system provides a very simple way to generate the Fibonacci sequence, when compared to other possible devices in automata and formal language theory.

In L systems with interactions, abbreviated IL systems, the productions have the form $(y, a, z) \to x$. Such a production can be applied to rewrite the letter a in the context yaz as x . If in all productions ~~by~~ the length of y (resp. z) equals k (resp. l), we speak of a system with $\langle k, l \rangle$ interactions. (From the biological point of view, this means that an individual cell may communicate with k of its left and l of its right neighbours.) Near the ends of the string, the missing neighbours are provided by a special letter g . For instance, the string aaa may be rewirtten as bbaba by the $\langle 1, 1 \rangle$ productions $(g, a, a) \to bb$, $(a, a, a) \to ab$, $(a, a, g) \to a$.

An L system with <u>tables</u> (abbreviated T) has several sets of rewriting rules instead of just one set. At one step of the rewriting process, rules belonging to the same set have to be applied. For an L system of any type, systems of the same type and with tables may be considered. The biological motivation for introducing tables is that one may want different rules to take care of different environmental conditions (heat, light, etc.) or of different stages of development.

When defining the language generated by an L system, we have so far considered only the exhaustive approach: all words derivable from the axiom by the rules in a parallel way belong to the language. In addition, various selective approaches will be considered in Section 2. Our presentation in Section 1 will begin with the most general type of an L system, namely, TIL system. The other types are then obtained as special cases. We will also consider L <u>schemes</u> differing from Lsystems in that the axiom ω is not specified.

As we already pointed out, we do not consider biological applications in this paper. At first sight, L systems appear to be too simple to deal with the complexity of real biological development. However, they have been successfully applied to the study of a variety of organisms, as well as a number of biological problems of a general nature. Among the organisms that have been investigated we find red algae, blue-green algae, fungi, mosses, snails and leaves, Among the theoretical problems that have been attacked we find problems associated with the notions of polarity, symmetry, regulation, synchronization, opical and banded patterns, branching patterns and positional information.

We are now ready to begin our survey on the mathematical theory of L systems. We would like to emphasize that the survey is by no means exhaustive and the selection of topics and results presented reflects our personal point of view.

14. SOME PROOF TECHNIQUES

In this section we will try to present to the reader some proof techniques used in the theory of L systems. As an example we will discuss proofs of some results concerning EOL systems and languages.

Because of the limitations on the size of this paper and also to ensure that the reader is not lost in technicalities we will sketch the proofs presented here rather than to provide them in all details.

Let us start with an auxiliary notation.

If G is an EOL system and x derives y in G in k steps, then we write $x \overset{k}{\underset{G}{\Rightarrow}} y$.

Also we state without a proof the following obvious result.

<u>Lemma 14.1</u>. For every EOL language K there exists an EOL system $G = \langle \Sigma, P, S, \Delta \rangle$ such that $L(G) = K$ and $S \in \Sigma - \Delta$.

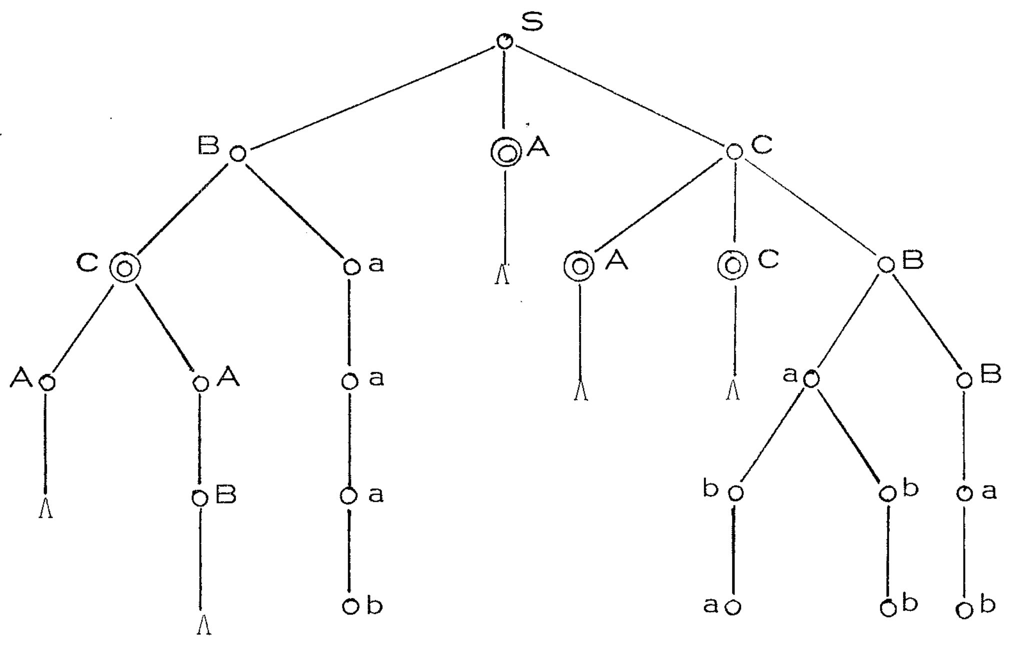
First we will present a proof of theorem 5.1: A language K is an EOL language if and only if there exists an EPOL system G such that $K - \{\Lambda\} = L(G)$.

If there exists an EPOL system G such that $K - \{\Lambda\} = L(G)$, then clearly K is an EOL language.

The more difficult part of the proof of this theorem is to show that if K is an EOL language then there exists an EPOL system G such that $K - \{\Lambda\} = L(G)$. Let $K = L(H)$ for an EOL system $H = \langle \Sigma, P, S, \Delta \rangle$ and let us assume that $L(H)$ is infinite and P contains erasing productions (otherwise the result holds trivially).

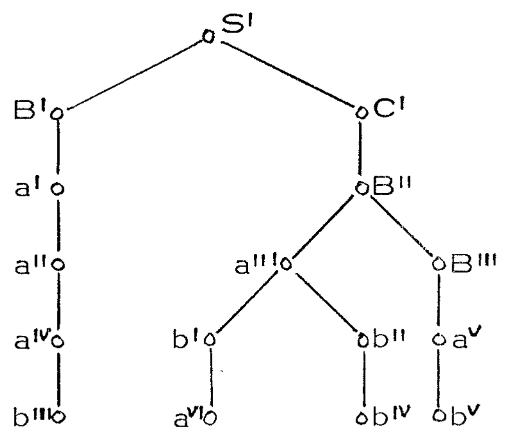
By Lemma 14.1, we assume that $S \in \Sigma-\Delta$. We also assume that $\Delta \subseteq \Sigma$. (This can be clearly done without loss of generality). The idea underlying our proof can be explained rather simply. We want to construct an EPOL system G which would simulate derivations in H in such a way that in corresponding derivation trees (in G) the occurrences of symbols which do not contribute anything to the final product (word) of a tree will not be introduced at all.

Let us assume that the following tree T is a derivation tree (for a word babb) in H :



In simulating this tree in G we want to avoid the situation in which we will be forced to apply an erasing production and so we want to delete every subtree which does not "contribute" to the final result babb . Hence we want to delete subtrees with double circled roots.

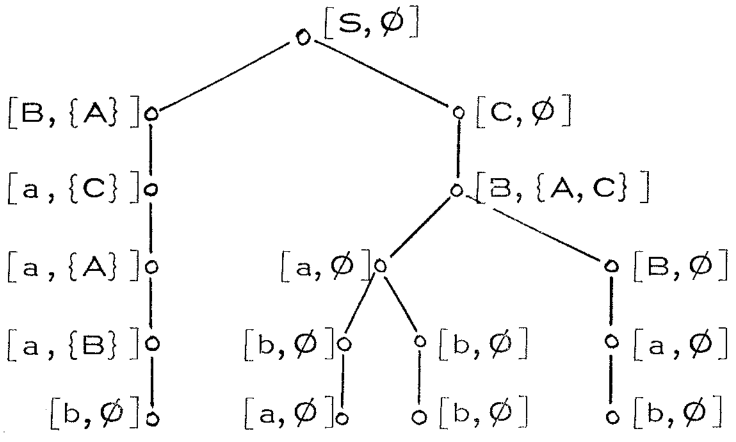
We would like then to be able to produce in G a derivation tree of this form



where S', B'–$B''', C', a', \dots, a^{VI}, b', \dots, b^{V}$ are some "representations" of symbols S, B, C, a, b .

In other words we are "killing" non-productive occurrences as early (going top-down) as possible. But, in general, there is no relation whatsoever between the level on which we delete (in G) a subtree at its root and the level (in H) on which this subtree really vanishes. Thus we have to carry along some information which would allow us to say (in G) at a certain moment: aha : the considered subtree vanishes (in H). Fortunately for this purpose we can carry a finite information only: it is enough to remember the minimal subalphabet Min(x) of a word x derived so far in the considered subtree rather than the word itself. We will carry this information as the second component in two-component letters of the form $[\sigma, z]$ where $\sigma \in \Sigma$ and $Z \subseteq \Sigma$.

Thus in our particular example we will have the following tree in G .



Now inspecting words on all levels of this tree we notice that only the last word

$$[b,\phi][a,\phi][b,\phi][b,\phi]$$

should be transformed to the terminal word (babb) because only on this level all subtrees that we have decided to delete (in G) really vanished (in H).

How to perform such a transformation within the system G itself?

To this aim we introduce a rather simple trick called "the synchronization".

For each letter of the form $[\sigma,\phi]$ with σ in Δ we introduce a production $[\sigma,\phi] \to \sigma$ and a production $\sigma \to F$ where F is a distinguished nonterminal symbol in G for which the only production in G is $F \to F$. Assuming that no other productions have terminal symbols on their right hand sides, this trick does the job. To see this observe that

$$[b,\phi][a,\phi][b,\phi][b,\phi] \underset{G}{\Rightarrow} babb \underset{G}{\Rightarrow} F^4 \underset{G}{\Rightarrow} F^4 \underset{G}{\Rightarrow} \cdots$$

but if we attempt to use these terminating productions too early then we fail to obtain a terminal word.

Now the reader should easily understand the following construction.

Let $G = \langle V, R, [S, \emptyset], \Delta \rangle$ be the EOL system defined by

1. $V = V_1 \cup \{F\} \cup \Delta$, where $V_1 = \{[\sigma, Z] : \sigma \in \Sigma$ and $Z \subseteq \Sigma\}$, and F is a new symbol.

2. R consists of the following productions:

2.1 If $A \to B_1 \ldots B_k$ is in P with $k \geq 2$, $B_1, \ldots, B_k \in \Sigma$ then, for every $Z \subseteq \Sigma$,

$[A, Z] \to [B_{i_1}, Z_{i_1}][B_{i_2}, Z_{i_2}] \ldots [B_{i_p}, Z_{i_p}]$ is in R if $1 \leq i_1 < i_2 < \ldots < i_p \leq k$ and,

for $2 \leq j \leq p-1$, $Z_{i_j} = \mathrm{Min}(B_{i_j+1} B_{i_j+2} \ldots B_{i_{j+1}-1})$,

$$Z_{i_1} = \mathrm{Min}(B_{i_1+1} B_{i_1+2} \ldots B_{i_2-1}) \cup \mathrm{Min}(B_1 B_2 \ldots B_{i_1-1}),$$

$$Z_{i_p} = \mathrm{Min}(B_{i_p+1} B_{i_p+2} \ldots B_k) \cup Z'$$

providing that $Z' \in \mathrm{Suc}_H(Z)$, where

$\mathrm{Suc}_H(Z) = \{U \subseteq \Sigma :$ there exist x, y in Σ^* with $\mathrm{Min}(x) = Z$, $\mathrm{Min}(y) = U$ where $x \underset{H}{\Rightarrow} y\}$.

2.2 If $A \to B$ is in P with B in Σ, then, for every $Z \subseteq \Sigma$ $[A, Z] \to [B, Z']$ is in R providing that $Z' \in \mathrm{Suc}_H(Z)$.

2.3 $[\sigma, \emptyset] \to \sigma$ is in R for all σ in Δ.

2.4 $F \to F$ is in R, and so is $\sigma \to F$ for all σ in Δ.

The reader should be able to convince himself that $L(G) = L(H) - \{\Lambda\}$, and this ends the proof of Theorem 5.1.

Now we present a proof for the first part of Theorem 2.3.2.1) (slightly extended). We denote by $\mathcal{L}(\mathrm{HOL})$ the family of homomorphic images of OL languages. We also make the definitional convention that whenever a language L belongs to some of the families considered, then also $L \cup \{\Lambda\}$ belongs to the same family.

Theorem 14.1. (Ehrenfeucht and Rozenberg [10])

$$\mathcal{L}(\mathrm{COL}) = \mathcal{L}(\mathrm{HOL}) = \mathcal{L}(\mathrm{EOL}).$$

Proof.

I) By definition $\mathcal{L}(\mathrm{COL}) \subseteq \mathcal{L}(\mathrm{HOL})$.

II) After having read the proof of Theorem 5.1 the reader should be able to produce easily the proof of the containment $\mathcal{L}(\mathrm{HOL}) \subseteq \mathcal{L}(\mathrm{EOL})$. If one considers only non-erasing homomorphisms then the synchronization trick presented in the proof of Theorem 5.1 suffices on its own. However if one considers an erasing homomorphism then this trick by itself does not work. But then one can construct an equivalent EOL system in which the symbols which are to be erased by the considered homomorphism will not be introduced at all. Technically it can be done in exactly the same way as avoiding (occurrences of) symbols which are to be erased in the given system.

III) Thus we have to prove now that $\mathcal{L}(\mathrm{EOL}) \subseteq \mathcal{L}(\mathrm{COL})$. By Theorem 5.1, it suffices to prove that $\mathcal{L}(\mathrm{EPOL}) \subseteq \mathcal{L}(\mathrm{COL})$. If A is an ultimately periodic set of non-negative integers then *thres* (A) denotes the smallest integer j for which there exists a positive integer q such that, for all $i \geq j$, i is in A if and only if $(i+q)$ is in A . The smallest positive integer q such that, for all $i \geq$ thres (A) , whenever i is in A also $i+q$ is in A , is denoted by *per* (A) .

If G is an EOL system with a terminal alphabet Δ and σ is a letter in the alphabet of G , then the *spectrum of* σ *in* G , denoted as $\mathrm{Spec}(\sigma, G)$, is defined by $\mathrm{Spec}(\sigma, G) = \{n \geq 0 : \sigma \overset{n}{\underset{G}{\Rightarrow}} w$ for some w in $\Delta^*\}$.

The spectra of letters in EOL systems satisfy the following important result.

Lemma 14.2. Let $G = \langle \Sigma, P, S, \Delta \rangle$ be an EOL system and let $\sigma \in \Sigma$. Then $\mathrm{Spec}(\sigma, G)$ is an ultimately periodic set.

Proof.

For a given σ in Σ let H_σ be a right-linear grammar such that

$H_\sigma = \langle V_N, V_T, R, \{\sigma\} \rangle$ where

$V_N = \{[Z] : Z \subseteq \Sigma\}$,

$V_T = \{a\}$ where a is a new symbol,

$R = \{[X] \to a[Y] : Y \in \mathrm{Suc}_G(X)\} \cup \{[X] \to \Lambda : X \subseteq \Delta\}$.

We leave to the reader the easy proof of the fact that $\{\sigma\} \overset{*}{\underset{H_\sigma}{\Longrightarrow}} a^k$ for some $k \geq 0$ if and only if there exists $w \in \Delta^*$ such that σ derives w in G in k steps.

But it is well known (see, e. g., [59]) that if a language K over a one letter alphabet, $K \subseteq \{a\}^*$ is generated by a right-linear grammar then $\{n : a^n \in K\}$ is an ultimately periodic set.

Thus the lemma holds.

Now we need some more terminilogy and notation.

Let $G = \langle \Sigma, P, S, \Delta \rangle$ be an EOL system and let $\sigma \in \Sigma$. We say that σ is vital, if for every $k > 0$ there exists an $l > k$ such that $a \overset{l}{\Rightarrow} w$ for some $w \in \Delta^*$. (We will use A_G to denote the set of all vital symbols from Σ).

Once we have noticed that each symbol in an EPOL system G contributes terminal subwords to terminal words in G in an ultimately periodic fashion we are trying to decompose G into a (finite) number of component systems in each of which one can consider only terminal contributions at the same moments of time.

Let $G = \langle \Sigma, P, S, \Delta \rangle$ be an EPOL system. We define the <u>uniform period</u> of G, denoted as m_G, to be the smallest positive integer such that

(i) for all $k \geq m_G$, if a is in $\Sigma - A_G$ and $a \overset{k}{\underset{G}{\Rightarrow}} w$, then $w \notin \Delta^*$,

(ii) for all a in A_G, $m_G >$ thres (Spec(G, a)) and per (Spec(G, a)) divides m_G.

Now our starting point is to consider all words that can be derived from S in m_G steps. (We will loose in this way all terminal words that can be derived in less than m_G steps from G but this is a finite set and, as we will see, easy to handle).

Then we will divide the words in this set into (not necessarily disjoint) subsets in each of which we can view all derivations going "according to the same clock" or, in more mathematical terms, conforming to the same (ultimately periodic) spectrum.

Here is thus our basic construction.

<u>Construction</u>. Let $0 \leq k < m_G$ and let $Ax(G,k) = \{w \in A_G^+ : S \overset{m_G}{\underset{G}{\Rightarrow}} w$ and, for all a in $Min(w)$, $m_G + k$ is in $Spec(G, a)\}$.
If $Ax(G,k) \neq \emptyset$, then, for all w in $Ax(G,k)$ define a OL system $G(k, w) = \langle \Sigma_{k,w}, R_{k,w}, w \rangle$ as follows:

(i) $\Sigma_{k,w} = \{a \in A_G : m_G + k$ is in $\mathrm{Spec}(G,a)$ and, for some $l \geq 0$, a is in $\mathrm{Min}(y)$ for some y such that $w \xRightarrow[G]{l \cdot m_G} y\}$,

(ii) $a \to \alpha$ is in $R_{k,w}$ if and only if $a \xRightarrow[G]{m_G} \alpha$ with $a \in \Sigma_{k,w}$ and $\alpha \in \Sigma^+_{k,w}$.

Hence we have the following situation. If $w \in \mathrm{Ax}(G,k)$ then the derivation in $G(k,w)$ goes as follows:

$$w \Rightarrow \underbrace{\overbrace{\dots}^{1 \text{ step in } G(k,w)}}_{m_G \text{ steps in } G} \Rightarrow w_1 \Rightarrow \underbrace{\overbrace{\dots}^{1 \text{ step in } G(k,w)}}_{m_G \text{ steps in } G} \Rightarrow w_2 \Rightarrow \dots$$

Now using the fact that all symbols appearing in words in $L(G(k,w))$ contain $m_G + k$ in their spectra we can squeeze the language from $G(k,w)$ in the following way.

Define $M(G(k,w)))$by

$M(G(k,w)) = \{x \in \Delta^* :$ there exists y in $L(G(k,w))$ such that $y \xRightarrow[G]{m_G + k} x\}$.

We shall now show that the union of the languages $M(G(k,w))$ over all $k < m_G$ and w in $\mathrm{Ax}(G,k)$ is identical (modulo a finite set) to $L(G)$.

Claim 1. $L(G) = \{w \in \Delta^+ : S \overset{l}{\underset{G}{\Rightarrow}} w$ for some $l < 2m_G\} \cup \bigcup_{k<m_G} \bigcup_{w \in \mathrm{Ax}(G,k)} M(G(k,w))$.

Proof.

Obviously $\{w \in \Delta^+ : S \overset{l}{\underset{G}{\Rightarrow}} w$ for some $l < 2m_G\} \cup \bigcup_{k<m_G} \bigcup_{w \in \mathrm{Ax}(G,k)} M(G(k,w)) \subseteq L(G)$.

Now let us assume that x is in $L(G)$.

(a) If x can be derived in less than $2m_G$ steps, then x is in the first set in union.

(b) If x is derived in at least $2m_G$ steps, then let $D = (S, x_1, \dots, x_{m_G}, \dots, x_p = x)$ be a derivation in G where $p = l_p \cdot m_G + k_p$ for some $l_p \geq 2$ and $0 \leq k_p < m_G$.

For all l, $1 \leq l < l_p$ and a in $\mathrm{Min}(x_{l \cdot m_G})$, we have

(i) a is vital, since $a \overset{t}{\underset{G}{\Rightarrow}} \bar{x}$ for some word $\bar{x}$ in Δ^+, where $t = (l_p \cdot m_G + k_p) - l \cdot m_G \geq m_G$.

(ii) $m_G + k_p$ is in $\mathrm{Spec}(G, a)$, since $(l_p - l)m_G + k_p$ is in $\mathrm{Spec}(G, a)$ and $\mathrm{Spec}(G, a)$ is an ultimately periodic set with period m_G and th~~r~~eshold smaller than m_G.

Therefore x is in $M(G(k_p, x_{m_G}))$ and hence x is in our union of languages on the right hand side of the equality from the claim.

Therefore Claim 1 holds.

Now the reader should note that we have already proved a quite significant result: each EPOL language is a result of a finite substitution on a OL language!

However we want to replace the finite substitution mapping by a coding. To this aim we shall prove now that each component language in the "union formula for $L(G)$" as given in the statement of Claim 1 is a finite union of codings of OL languages.

<u>Claim 2</u>. Assume that $Ax(G, k) \neq \emptyset$ and let w be in $Ax(g, k)$. Then there exist OL systems $H_1, \dots, H_f$ and a coding h such that $M(G(k, w)) = \bigcup_{i=1}^{f} h(L(H_i))$.

Proof.

Let $w = b_1 \dots b_t$ where b_i is in A_G for $1 \le i \le t$.

For all a in $\Sigma_{k,w}$ let $U(a,k) = \{x \in \Delta^+ : a \xRightarrow[G]{m_G+k} x\} =$
$= \{\alpha_{a,k,1}, \alpha_{a,k,2}, \dots, \alpha_{a,k,u(a,k)}\}$, say, and $\overline{\Sigma}_{k,w} = \{[a,b],$
$\overline{[a,b]} : a \in \Sigma_{k,w}$ and $b \in \Delta\}$.

Let $W(w) = \{[b_1, c_{11}][b_1, c_{12}] \dots \overline{[b_1, c_{1r_1}]}[b_2, c_{21}] \dots$
$\dots \overline{[b_2, c_{2r_2}]} \dots [b_t, c_{t1}] \dots \overline{[b_t, c_{tr_t}]} : c_{j1} \dots c_{jr_j} \in U(b_j, k)$ for
$1 \le j \le t\}$.

Let $\overline{R}_{k,w} = \{[a,b] \to \Lambda : a \in \Sigma_{k,w}, b \in \Delta\}$
$\cup \{\overline{[a,b]} \to [c_1, d_{11}][c_1, d_{12}] \dots \overline{[c_1, d_{1v_1}]} \dots [c_s, d_{s1}]$
$\dots \overline{[c_s, d_{sv_s}]} : b \in \Delta$, $a \to c_1 \dots c_s$ is in $R_{k,w}$ and
$d_{j_1} \dots d_{jv_j} \in U(c_j, k)$ for $1 \le j \le s\}$.

Let, for every z in $W(w)$, $G(k,w,z)$ be the OL system
$\langle \overline{\Sigma}_{k,w}, \overline{R}_{k,w}, z\rangle$ and let h be a coding from $\overline{\Sigma}_{k,w}$ into Δ such that
$h([a,b]) = h(\overline{[a,b]}) = b$.

We leave to the reader the obvious, but tedious proof of the fact that

$$M(G(k,w)) = \bigcup_{z \in W(w)} h(L(G(k,w,z))).$$

Thus Claim 2 holds.

Now we state two obvious results.

(I) If K is a finite language, then there exist a OL system G and a coding h such that $K = h(L(G))$.

(II) If $H_1, \dots, H_f$ are OL systems, $h_1, \dots, h_f$ are codings and $K = \bigcup_{i=1}^{f} h_i(L(H_i))$, then there exist a OL system G and a coding $\overline{h}$ such that $K = \overline{h}(L(G))$.

The above two simple observations allow us now to collect together all the component languages of G by means of one OL system and one coding.

This ends the proof of Theorem 14.1.