Prof. Dr. Jürgen Dassow
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik

F O R M A L L A N G U A G E S

A N D

B I O L O G I C A L P R O C E S S E S

Vorlesungsmanuskript

Magdeburg, April - July 2008

Introduction

In the end of the fifties as N. Chomsky has introduced the well-known classes of regular, context-free and context-sensitive languages the aim was to model the syntax of natural languages. Based on the Backus-Naur form for the description of the syntax of programming languages, in the beginning of the sixties S. Ginsburg and H.G. Rice noticed that the grammars introduced by Chomsky can be used for programming languages, too. Since that time until at least the middle of the seventies most investigations to formal languages followed this approach. The central feature of such grammars is a sequential process of rewriting of subwords.

On the other hand one has to mention that already since the fifties there exist some devices nearly related to formal languages which were motivated and/or applied to biological phenomena. The well-known Kleene Theorem on the description of regular languages by means of algebraic operations was discovered by S.C. Kleene as he represented the events in nerve nets. Furthermore, it was known that cellular automata are able to a self-replicating behaviour known from biological organisms or colonies of organisms. But in both cases, in order to model the biological processes finite automata or collections of finite automata have been used.

Since the seventies the situation changed completely. Motivated by biological processes new types of grammars have been introduced and their investigation dominated in a certain sense the development of the theory of formal languages.

In 1968 the first approach was initiated by A. Lindenmayer (see [16]. Cell divisions, changes of states of the cells, death of cells etc. were modelled by production as one uses in Chomsky grammars. However, the rewriting process by application of rules is a parallel one because cell divisions, changes of cell states etc. proceed in parallel. The large interest in these Lindenmayer systems originated from the biological motivation as well as by the interest in a comparison between sequential and parallel processes in computer science. The monograph [13] presents a summary of the state of the theory of developmental systems and languages in 1975 and considers intensively motivation from and application to bilogy, whereas the monograph [27] emphasizes the mathematical theory of such systems. Further summaries and material can be found in [26], [17], [28], [29], [15]. In [25] the authors use Lindenmayer systems to generate graphical representations of plants.

Although DNA sequences are twisted strands (in a 3-dimensional space) it is very natural to model them by (linear) strings/words. Mutations of DNA sequences, genes, chromosomes etc. caused by deletions, insertions, splicings, inversions etc. can be described by operations on words. Iterated applications of these operations model the evolution of molecules. Thus we have sequential process, again, however, the basic step is not a rewriting. After the first investigations in this direction by T. Head (see [11]) in the last

decade a lot of papers appeared studying the behaviour of formal languages under these operations. Moreover, one has to mention that these considerations are nearly related to some aspects of molecular computing (see [1], [18]). The book [23] is the first monograph on this topic, summaries are contained in [2], [12], [24], [7].

An approach – called membrane systems – to describe the behaviour of a single cell was startet by Gh. Păun in the paper [21]. A cell is considered as an object with membranes which define substructures of the cell, e.g. the kernel of the cell. Changes of the objects in the different regions of the cell are described by rules associated with the regions. However, the rules are not applied to words as in the two types of grammars mentioned above, the rules are applied to multisets since the objects in a region form a multiset. The books [22] and [2] summarize parts of the theory developed for these grammatical systems.

We mention that these three new types of grammars/languages are natural by their motivation from biology as well as by the fact that they allow nice characterizations of well-known classes of formal languages.

In this lecture we shall emphasize Lindenmayer systems, languages and systems using operations as splicing and membrane systems. We shall omit grammars with valuations (see [5]), eco-grammar systems (see [4]) and other language generating devices modelling aspects of biology.

Throughout this lecture we assume that the students/reader is familiar with the basic concepts of the theory of formal languages as usually presented in basic courses on Theoretical Computer Science and with some facts of mathematics (especially linear algebra, theory of difference equations, combinatorial formulae, etc). The notation, some definitions and results are summarized in the first chapter.

Jürgen Dassow April - July 2008

Contents

Chapter 1

Basics of Mathematics and Formal Languages

In this chapter we recall some basic knowledge of mathematics and the theory of formal languages which will be used in the following chapters. We emphasize those concepts and facts which we refer to, however, we add some definitions etc. which are useful to understand the following chapters.

1.1 Sets, Words, Multisets

If a set A is contained in a set B, then we write $A \subseteq B$. If the inclusion is proper, we write $A \subset B$.

By $\mathbf{N}$ we denote the set of all positive integers, i.e., $\mathbf{N} = \{1, 2, \ldots\}$. $\mathbf{N}_0$ denotes the set of all non-negative integers, i.e., $\mathbf{N}_0 = \mathbf{N} \cup \{0\} = \{0, 1, 2, \ldots\}$.

A permutation p of the set $M = \{1, 2, \ldots n\}$ is a one-to-one mapping of M onto itself. Obviously, p can be given as $(p(1), p(2), \ldots, p(n))$. Two elements $p(i)$ and $p(j)$ of p form an inversion if $p(i) > p(j)$ and $i < j$. By $I(p)$ we denote the number of inversions of p.

An *alphabet* is a non-empty finite set. Its elements are called letters. A *word* (over an alphabet V) is a sequence of letters (of V). By λ we denote the empty word which contains no letter. By V^* (and V^+, respectively) we designate the set of all (non-empty) words over V. The *product* (concatenation) of words is defined as the juxtaposition of the words. We say that v is a *subword* of w iff $w = x_1 v x_2$ for some $x_1, x_2 \in V^*$. The word v is called a *prefix* of w iff $w = vx$ for some $x \in V^*$, and v is called a *suffix* of w iff $w = xv$ for some $x \in V^*$.

By $\#_a(w)$ we denote the number of occurrences of a letter a in a word w. The *length* $|w|$ of a word w over V is defined as $|w| = \sum_{a \in V} \#_a(w)$.

Let $V = \{a_1, a_2, \ldots, a_n\}$ where $a_1, a_2, \ldots a_n$ is a fixed order of the elements of V. Then

$$\Psi_V(w) = (\#_{a_1}(w), \#_{a_2}(w), \ldots, \#_{a_n}(w))$$

is the *Parikh vector* of the word $w \in V^*$.

A *language* over V is a subset of V^*.

Convention: Two languages L_1 and L_2 are called equal (written as $L_1 = L_2$) if and only if L_1 and L_2 differ at most in the empty word, i.e., $L_1 \setminus \{\lambda\} = L_2 \setminus \{\lambda\}$.

For two languages L and K we define their *concatenation* as

$$L \cdot K = \{wv \mid w \in L,\ v \in K\}\,.$$

and the *Kleene closure* L^* (of L) by

$$\begin{aligned} L^0 &= \{\lambda\}\,, \\ L^{i+1} &= L^i \cdot L \quad \text{for} \quad i \geq 0\,, \\ L^* &= \bigcup_{i \geq 0} L^i\,. \end{aligned}$$

A *homomorphism* $h : X^* \to Y^*$ is a mapping where

$$h(wv) = h(w)h(v) \text{ for any two words } w, v \in X^*\,. \tag{1.1}$$

Obviously, a homomorphism can be given by the images $h(a)$ of the letters $a \in X$; an extension to words follow from the homomorphism property (1.1). We extend the homomorphism to languages by

$$h(L) = \{h(w) \mid w \in L\}\,.$$

If h is a homomorphism, then the inverse homomorphism h^{-1} applied to a language $K \subseteq Y^*$ is defined by

$$h^{-1}(K) = \{w \mid w \in X^*,\ h(w) \in K\}\,.$$

For a word $w = a_1 a_2 \ldots a_n$ with $n \geq 0$ and $a_i \in V$ for $1 \leq i \leq n$, we set $w^R = a_n a_{n-1} \ldots a_1$. It is obvious that $\lambda^R = \lambda$ and $(w_1 w_2)^R = w_2^R w_1^R$ for any two words w_1 and w_2. For a language L, we set $L^R = \{w^R \mid w \in L\}$.

A multiset M over V is a mapping of V^* into the set $\mathbf{N}$ of non-negative integers. $M(x)$ is called the multiplicity of x. The cardinality and the length of a multiset M are defined as

$$\#(M) = \sum_{x \in V^*} M(x) \text{ and } l(M) = \sum_{x \in V^*} M(x)|x|\,.$$

A multiset M is called finite iff there is a finite subset U of V^* such that $M(x) = 0$ for $x \notin U$. Then its cardinality is the sum of the multiplicities of the elements of U. A finite multiset M can be represented as a "set" where M contains $M(x)$ occurrences of x. Thus a finite multiset M in this representation consists of $\#(M)$ elements. For example, the multiset M over $V = \{a, b\}$ with $M(a) = M(b) = M(aba) = 1$, $M(ab) = M(ba) = 2$ and $M(x) = 0$ in all other cases can be represented as $M = [a, b, ab, ab, ba, ba, aba]$[1]. Obviously, as for sets, the order of the elements in the multiset M is not fixed and can be changed without changing the multiset. For a multiset $M = [w_1, w_2, \ldots, w_n]$ (in such a representation) we have $l(M) = |w_1 w_2 \ldots w_n|$. Moreover, for a multiset M over V and $a \in V$, we set $\#_a(M) = \#_a(w_1 w_2 \ldots w_n)$.

[1]We use the brackets [and] instead of { and } in order to distinguish multisets from sets.

1.2 Linear Algebra

A (m,n)-*matrix* is a scheme of $m \cdot n$ (real) numbers $a_{i,j}$, $1 \leq i \leq m$ and $1 \leq j \leq n$. The scheme consists of m rows where the i-th row consists of the elements $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$, $1 \leq i \leq m$. Equivalently, it is given by n columns where the j-th column is built by the numbers $a_{1,j}, a_{2,j}, \ldots, a_{m,j}$, $1 \leq j \leq n$. Thus we get

$$M = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \ldots & a_{2,n} \\ . & . & . & \ldots & . \\ a_{m,1} & a_{m,2} & a_{m,3} & \ldots & a_{m,n} \end{pmatrix}$$

We write $M = (a_{i,j})_{m,n}$ and omit the index m, n if the size of the matrix is known from the context.

Obviously, row vectors are $(1,n)$-matrices and column vectors are $(m,1)$-matrices. A matrix is called a *square* matrix, if it is an (n,n)-matrix for some n. Let $E_{n,n}$ be the square (n,n)-matrix with $a_{i,i} = 1$ for $1 \leq i \leq n$ and $a_{j,k} = 0$ for $j \neq k$ (again, we omit the index if the size is understood by the context); $E_{n,n}$ is called the *unity matrix*. By O we denote the *zero matrix* where all entries are the real number 0.

Let $M_1 = (a_{i,j})_{m,n}$ and $M_2 = (b_{k,l})_{r,s}$ be two matrices, and let d be a (real) number. Then the *product* $d \cdot M_1$ is defined by

$$d \cdot M_1 = (d \cdot a_{i,j})_{m,n} .$$

The *sum* $M_1 + M_2$ is defined iff $m = r$ and $n = s$ by setting

$$M_1 + M_2 = (a_{i,j} + b_{i,j})_{m,n} .$$

The *product* $M_1 \cdot M_2$ is defined iff $n = r$ by setting

$$M_1 \cdot M_2 = (\sum_{j=1}^{n} a_{i,j} b_{j,l})_{m,s} .$$

The *transposed matrix* $(M_1)^T$ is formed by interchanging the rows and columns, i.e.,

$$(M_1)^T = (a_{j,i})_{n,m} .$$

The *determinant* of an (n,n)-matrix M is defined by

$$det(M) = \sum_{p=(i_1,i_2,\ldots,i_n)} (-1)^{I(p)} a_{1,i_1} a_{2,i_2} \ldots a_{n,i_n}$$

where the sum is taken over all permutations of $1, 2, \ldots, n$. By definition, *det* maps matrices to reals.

The characteristic polynomial $\chi_A(x)$ of a (square) (n,n)-matrix A is defined as

$$\chi_A(x) = det(A - xE) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_2 x^2 + a_1 x + a_0 .$$

We note that $a_n = (-1)^n$ and $a_0 = det(A)$.

A complex number μ is called an *eigenvalue* of the square matrix A iff $det(A-\mu E) = 0$, i.e., iff μ is a root of χ_A. [2]

The following theorem is named after the English mathematicians CAYLEY and HAMILTON.

Theorem 1.1 *For any square matrix A, $\chi_A(A) = O$.* □

If we give a complete writing of the characteristic polynomial $\chi_A(A)$, then this means

$$\chi_A(A) = a_n A^n + a_{n-1}A^{n-1} + a_{n-2}A^{n-2} + \ldots + a_2 A^2 + a_1 A + a_0 E = O\,.$$

Theorem 1.2 *Let $a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_2 x^2 + a_1 x + a_0$ be a polynomial of degree n with the roots α_i of multiplicity t_i, $1 \le i \le s$, and $\sum_{i=1}^{s} t_i = n$. Then the linear difference equation*

$$a_n f(m+n) + a_{n-1} f(m+n-1) + \ldots + a_2 f(m+2) + a_1 f(m+1)x + a_0 f(m) = 0$$

for $m \ge 0$ has the solution

$$f(m) = \sum_{i=1}^{s} (\beta_{i,0} + \beta_{i,1} m + \beta_{i,2} m^2 + \ldots \beta_{i,t_i-1} m^{t_i-1}) \alpha_i^m$$

with certain constants $\beta_{i,j}$, $1 \le i \le s$, $0 \le j \le t_i - 1$. □

1.3 Formal Languages

A *phrase structure grammar* (or short grammar) is a quadruple $G = (N, T, P, S)$, where
– N is an alphabet,
– T is an alphabet,
– $V_G = N \cup T$, $N \cap T = \emptyset$,
– P is a finite subset of $(V_G^* \setminus T^*) \times V_G^*)$,
– S is an element of N.
The elements of N and T are called nonterminals and terminals, respectively. The elements of P are called rules and written as $\alpha \to \beta$ instead of (α, β). S is called the axiom or start word.

A *direct derivation* $x \Longrightarrow_G y$ is defined by the following conditions:
– $x = x_1 \alpha x_2$, $y = x_1 \beta x_2$,
– $\alpha \to \beta \in P$.
By $\Longrightarrow_G^*$ we denote the reflexive and transitive closure of $\Longrightarrow_G$. The *language* $L(G)$ *generated by* G is defined by

$$L(G) = \{z \mid z \in T^* \text{ and } S \Longrightarrow_G^* z\}\,.$$

A grammar G is called *monotone* if and only if every rule of P has the form $\alpha \to \beta$ with $|\alpha| \le |\beta|$.

[2] Here we have to consider complex numbers since the roots of polynomials are complex numbers in general.

A grammar G is called *context-sensitive* if and only if every rule of P has the form $uAv \to uwv$ with $A \in N$, $w \in V^+$, $u, v \in V^*$

A grammar G is called *context-free* if and only if every rule of P has the form $A \to w$ with $A \in N$ and $w \in V^*$,

A grammar G is called *regular* if and only if every rule of P has the form $A \to wB$ or $A \to w$ with $A, B \in N$ and $w \in T^*$,

By REG, CF, CS, MON and RE we denote the families of regular, context-free, context-sensitive, monotone and arbitrary (phrase structure) grammars.

A language L is called a regular, context-free, context-sensitive and monotone language if and only if $L = L(G)$ for some regular, context-free, context-sensitive and monotone grammar G, respectively. A language L is recursively enumerable iff $L = L(G)$ for some (phrase structure) grammar G. [3]

For a family X of grammars, by $\mathcal{L}(X)$ we denote the family of languages generated by grammars of X. $\mathcal{L}(FIN)$ designates the family of finite languages.

Theorem 1.3 $\mathcal{L}(FIN) \subset \mathcal{L}(REG) \subset \mathcal{L}(CF) \subset \mathcal{L}(CS) = \mathcal{L}(MON) \subset \mathcal{L}(RE)$ □

We say that a family $\mathcal{L}$ of languages is closed under the n-ary operation τ if, for any languages $L_1, L_2, \ldots, L_n$ of $\mathcal{L}$, $\tau(L_1, L_2, \ldots L_n) \in \mathcal{L}$.

The following theorem presents the closure properties of the families of the Chomsky hierarchy with respect to some important operations.

Theorem 1.4 *The table of Figure 1.1 holds. A + or - at the intersection of the row with operation τ and the column with X means that $\mathcal{L}(X)$ is closed or not closed under τ, respectively.*

	$\mathcal{L}(FIN)$	$\mathcal{L}(REG)$	$\mathcal{L}(CF)$	$\mathcal{L}(CS)$	$\mathcal{L}(RE)$
union	+	+	+	+	+
intersection	+	+	-	+	+
concatenation	+	+	+	+	+
Kleene-closure	+	+	+	+	+
homomorphisms	+	+	+	-	+
inverse homomorphisms	-	+	+	+	+
intersect with reg. sets	+	+	+	+	+

Figure 1.1: Closure properties of the families of the Chomsky hierarchy

We give some theorems which give characterizations of recursively enumerable languages.

Theorem 1.5 *For any recursively enumerable language L, there is a phrase structure grammar $G = (N, T, P, S)$ with $L = L(G)$ and all rules of P have one of the following four forms*

$$A \to B \text{ or } A \to a \text{ or } A \to \lambda \text{ or } AB \to CD \text{ with } A, B, C, D \in N \text{ and } a \in T\,.$$

[3]The notion "recursively enumerable" comes from the theory of computation and the theory of recursive function where the same set of languages occur.

For a proof we refer to [6].

Lemma 1.6 *For any recursively enumerable language L, there are context-free languages L_1 and L_2 such that $L = \{u \mid uv \in L_1 \text{ for some } v \in L_2\}$.*

For a proof we refer to [23], Theorem 3.13.

Lemma 1.7 *For any recursively enumarable language $L \subset V^*$, there is a context-sensitive language L' and letters c_1 and c_2 not contained in V such that $L' \subseteq L\{c_1\}\{c_2\}^*$ and, for any $w \in L$, there is a number $i \geq 1$ such that $wc_1c_2^i \in L'$.*

Proof. Let L be a recursively enumerable language, and let $G = (N, T, P, S)$ be a phrase structure grammar generating L. We construct the monotone grammar

$$G' = (N \cup \{C, S'\}, T \cup \{c_1, c_2\}, P', S')$$

where P' consists of all rules of the following forms:

- $S' \to Sc_1$
  (this rule introduces the start symbol of G and the additional symbol c_1),
- $\alpha \to \beta$ where $\alpha \to \beta \in P$ and $|\alpha| \leq |\beta|$,
  $\alpha \to \beta C^p$ where $\alpha \to \beta \in P$ and $|\alpha| - |\beta| = p > 0$
  (these monotone rules simulate the rules of P),
- $Ca \to aC$ for $a \in N \cup T \cup \{c_1\}$
  (by these rules, C can be shifted to the right),
- $C \to c_2$
  (terminating rules for C).

By the explanations added to the rules it is obvious that $v \in L(G')$ if and only if $v = c_2^{r_1}w_1c_2^{r_2}w_2 \ldots c_2^{r_k}w_kc_2^s$ where $r_i \geq 0$ for $1 \leq i \leq k$, $s \geq 0$ and $w_1w_2 \ldots w_n = wc_1$ for some $w \in L$. Since $L(G) \in \mathcal{L}(CS)$ (by Theorem 1.3) and $\mathcal{L}(CS)$ is closed under intersections (with regular sets), $L' = L(G') \cap T^*\{c_1\}\{c_2\}^*$ is a context-sensitive language, too. It is easy to see that L' has the properties required in the statement. □

For the definition of an (accepting) Turing machine and a proof of the following theorem we refer to [6].

Theorem 1.8 *A language L is recursively enumerable if and only if $L = T(M)$ for some (deterministic) Turing machine.*

Let $G = (N, T, P, S)$ be a phrase structure grammar. For a derivation

$$D : S \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \ldots \Longrightarrow w_r = w$$

of $w \in T^*$ in G, we define the *workspace of w by D* by

$$Ws_G(w, d) = max\{|w_i| \mid 1 \leq i \leq r\}$$

and the workspace of w by

$$Ws_G(w) = \min\{Ws_G(w, D) \mid D \text{ is a derivation of } w \text{ in } G\}\,.$$

Theorem 1.9 *If $G = (N, T, P, S)$ is a phrase structure grammar and k is a positive integer such that $Ws_G(w) \leq k|w|$ holds for any $w \in L(G)$, then $L(G)$ is a context-sensitive language.* □

For a proof we refer to [31].

We now present some properties of regular and context-free languages. For proofs we refer to [6].

Theorem 1.10 *For any regular language L there is a regular grammar $G = (N, T, P, S)$ where all rules of P have the form $A \to aB$ or $A \to a$ with $A, B \in N$ and $a \in T$ such that $L = L(G)$.*

Theorem 1.11 *For any context-free language L there is a context-free grammar $G = (N, T, P, S)$ where all rules of P have the form $A \to BC$ or $A \to a$ with $A, B, C \in N$ and $a \in T$ such that $L = L(G)$.*

Theorem 1.12 *Let L be a regular language. Then there is a constant k (which depends on L) such that, for any word w with $|w| > k$, there is a decomposition $w = xyz$ such that*
– $|x| < k$,
– $|y| > 0$,
– $xy^iz \in L$ for any integer $i \geq 0$.

Theorem 1.13 *Let L be a context-free language. Then there is a constant k (which depends on L) such that, for any word w with $|w| > k$, there is a decomposition $w = vwxyz$ such that*
– $|wxy| < k$,
– $|wy| > 0$,
– $vw^ixy^iz \in L$ for any integer $i \geq 0$.

A *finite nondeterministic automaton* $\mathcal{A}$ is specified as a quintuple $\mathcal{A} = (X, Z, z_0, F, \delta)$ where
– X is a finite non-empty set (the set of input symbols),
– Z is a finite non-empty set (the set of states),
– $z_0 \in Z$ and $\emptyset \subset F \subseteq Z$,
– δ is a mapping from $Z \times X$ into 2^Z.

We extend δ to a mapping from $Z \times X^*$ into 2^Z by the following settings:
– $\delta(z, \lambda) = \{z\}$,
– $\delta(z, wa) = \bigcup_{z \in \delta(z,w)} \delta(z, a)$.

The language $T(\mathcal{A})$ of words over X accepted by $\mathcal{A}$ is defined by

$$T(A) = \{w \mid \delta(z_0, w) \cap F \neq \emptyset\}.$$

A finite *deterministic* automaton is a finite nondeterministic automaton where any set $\delta(z, a)$, $z \in Z$, $a \in X$, contains at most one state.

Theorem 1.14 *The following statements are equivalent:*
– L is generated by a regular grammar.
– L is accepted by a finite nondeterministic automaton.
– L is accepted by a finite deterministic automaton.

For a proof we refer to [6].

Chapter 2

Lindenmayer Systems

2.1 The Basic Model – 0L Systems

2.1.1 Two Biological Examples

We start with two biological examples describing the development of an alga and a moss.

In Figure 2.1 the first 10 stages of the development of a red alga is shown.

Any small part represents a cell; thus stage a) is formed by one cell; stage b) consists of two cells and stage c) of four cells. Starting with stage d) we see a branching structure of the alga. Thus the first problem consists in the description of the branching structure. We choose a word over the alphabet consisting of the letters c, (and). c represents a cell and (and) are used to describe the branching. If we have a word $c^r(c^s)c^t$, then the central part of the alga is given by $c^r c^t$ and the subword c^s describes a branch. By this method we do not distinguish between branches to the left or to the right etc. Furthermore, we can iterate the process, i.e., if we have a word $c^n(c^r(c^s)c^t)c^m$, then $c^r c^t$ is a branch of $c^n c^m$ and c^s is a branch of the branch $c^r c^t$.

Then we can describe the stages given in Figure 2.1 as follows:

a) c
b) cc
c) $cccc$
d) $cc(c)cccc$
e) $cc(cc)cc(c)cccc$
f) $cc(ccc)cc(cc)cc(c)cccc$
g) $cc(cccc)cc(ccc)cc(cc)cc(c)cccc$
h) $cc(ccccc)cc(cccc)cc(ccc)cc(cc)cc(c)cccc$
i) $cc(cccccc)cc(ccccc)cc(cccc)cc(cccc)cc(cc)cc(c)cccc$
j) $cc(ccccccc)cc(cccccc)cc(ccccc)cc(cc(c)cccc)cc(cccc)cc(cc)cc(c)cccc$

The development from stage a) to stage b) can be considered as a division of the cell c resulting in cc. If we apply this division to both cells of stage b), again, then we get the four cells of stage c). But now we cannot continue in this way by two reasons: Stage d) does not consist of eight cells (which would be obtained from the division of four cells) and we cannot model the branching which occurs in stage d). In order to solve this problem one can introduce more rules for the cell or one makes a further differentiation of the cell

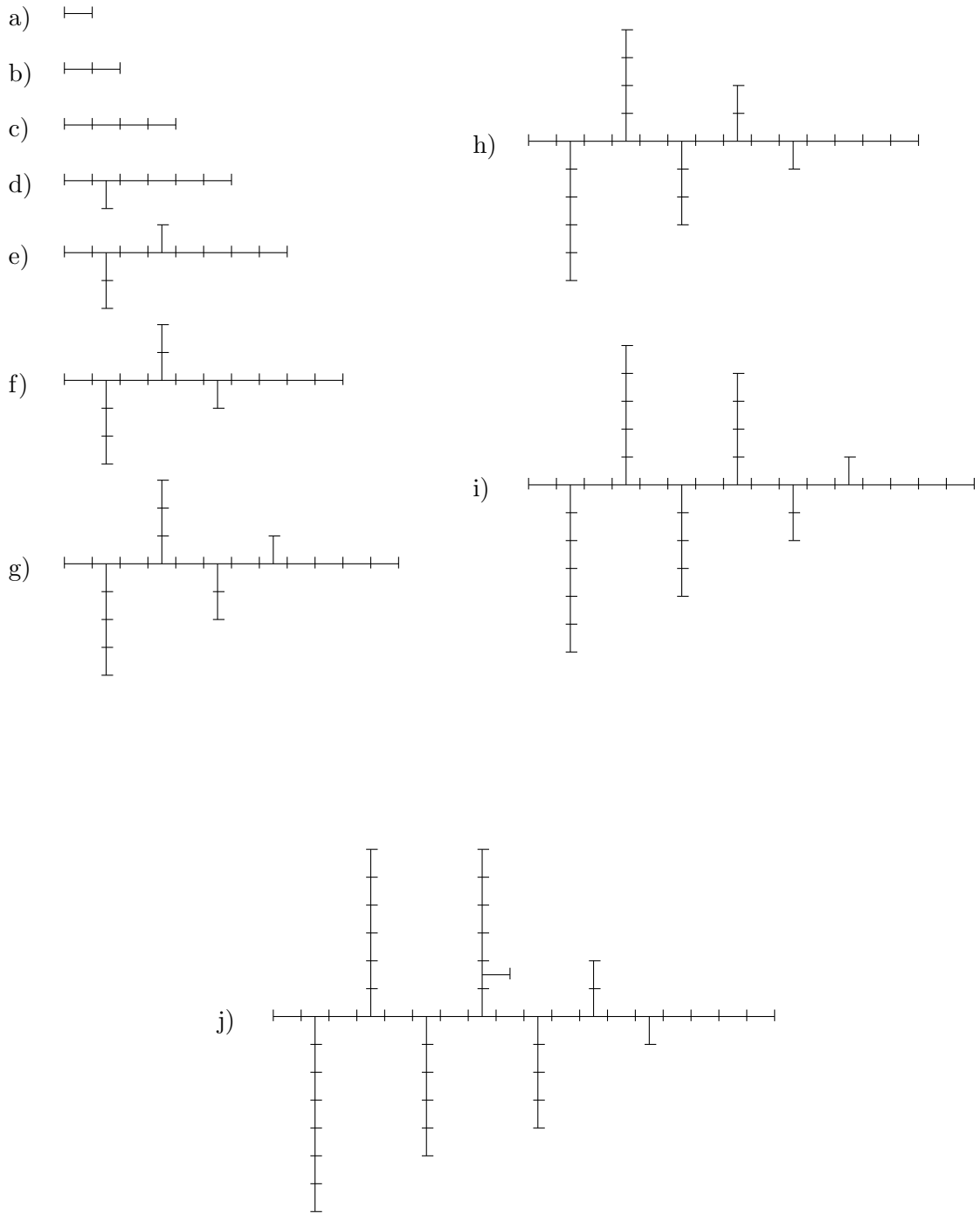

Figure 2.1: First stages of the development of a red alga

by introducing some states of the cell and different rules for different states.

We use the second approach and distinguish 10 states of cell c which we denote by the digits

$$0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9$$

of the decimal system. Moreover, we consider the rules

$$\begin{array}{llll} 0 \to 10 & 1 \to 32 & 2 \to 3(4) & 3 \to 3 \quad 4 \to 56 \\ 5 \to 37 & 6 \to 58 & 7 \to 3(9) & 8 \to 50 \quad 9 \to 39 \end{array}$$

for the states where the left hand side gives the state a of the cell and the right hand side gives the part which is obtained from a in one step of the development. The rules for 0 and 1 can be interpreted as divisions of one cell into two cells; the rules for 2 and 7 can be considered as the starting of a branch. The rule $3 \to 3$ can be omitted because it says that c in state 3 is not changed in the sequel. However, if we want to describe the development, then we have to tell what happens with each cell at every moment. Thus we add $3 \to 3$ in order to know what happens to cells in state 3.

Then we obtain the following description of the first stages of the development of the red alga and one sees that this corresponds to the stages given in Figure 2.1:

a) 4
b) 56
c) 3758
d) 33(9)3750
e) 33(39)33(9)3710
f) 33(339)33(39)33(9)3210
g) 33(3339)33(339)33(39)33(4)3210
h) 33(33339)33(3339)33(339)33(56)33(4)3210
i) 33(333339)33(33339)33(3339)33(3758)33(56)33(4)3210
j) 33(3333339)33(333339)33(33339)33(33(9)3750)33(3758)33(56)33(4)3210

We now consider the moss *Phascum cuspidatum*. A typical leaf of Phascum cuspidatum is shown in Figure 2.2. It consists of three types of cells: cells of type I are at the top of the leaf, cells of type II are along the margin of the leaf, and cells of type III form the inner part of the leaf.

The development of Phascum cuspidatum was already considered in 1845 by the Swiss biologist Carl Wilhelm von Nägeli (1817–1891). He noticed that essentially we have the developmental rules

$$I \to I + II,\ \ II \to II + II \ \text{ and } II \to II + III$$

and the rule $III \to III$ which says that cells of type III are not changed in the developmental process. However, as in the first example, in order to be precise one has to distinguish different states of the cells, because e.g.
– cells of type II do not changed according to one of the rules above in every step,
– cells of type I are changed in every step, however, they produce the cells of type II alternately to the right and to the left.

We describe a leaf as a square where the upper left corner corresponds to the top of the leaf. We use cells of type I_i and II_i^r where the lower index i is a number and reflects

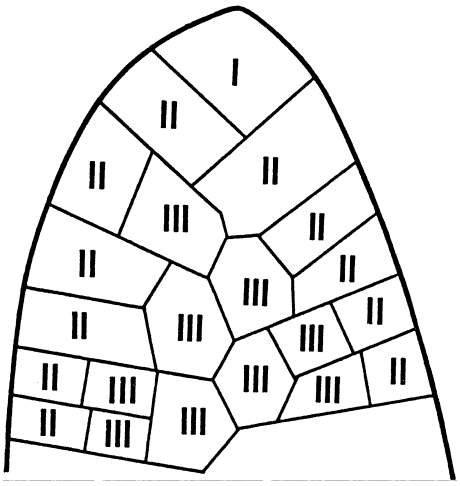

Figure 2.2: Leaf of the moss *Phascum cuspidatum*

the "age" of the cell and the upper index $r \in \{o, l\}$ gives the margin where the cell is (l stands for the left margin and o for the upper margin).

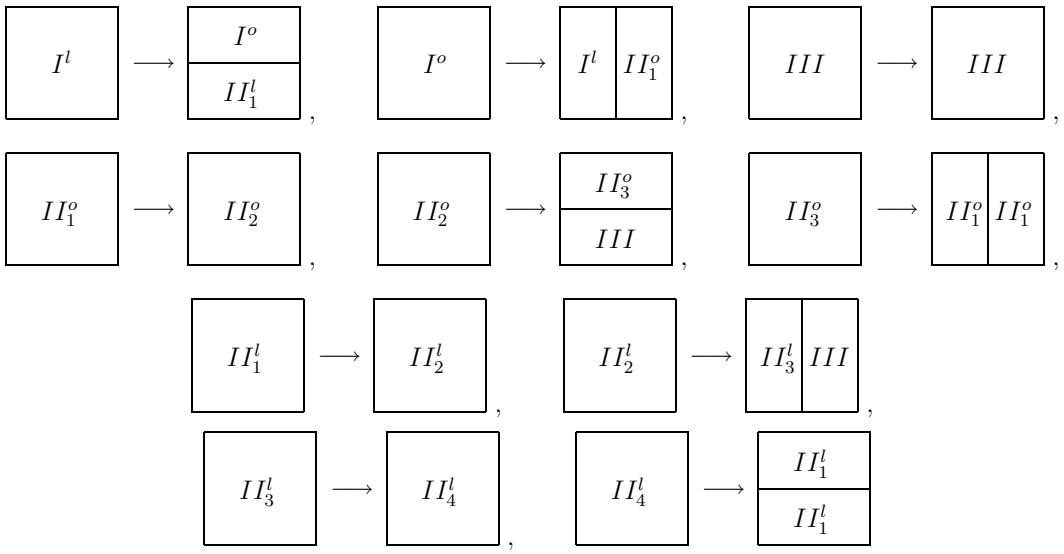

Figure 2.3: Rules for the development of the moss *Phascum cuspidatum*

Figure 2.3 gives the more detailed rules and in Figure 2.4 the first stages of the development according to these rules starting with a single cell of type I are shown. It is easy to see that the last stage corresponds to the leaf given in Figure 2.2.

2.1.2 Definitions and Examples

Looking on the examples presented in the preceding subsection we see that a formalization of them has to take into consideration the following aspects:

- in one step all cells or at least some of them are changed according to the rules in parallel, i.e., the rewriting is not a sequential process as in the case of phrase

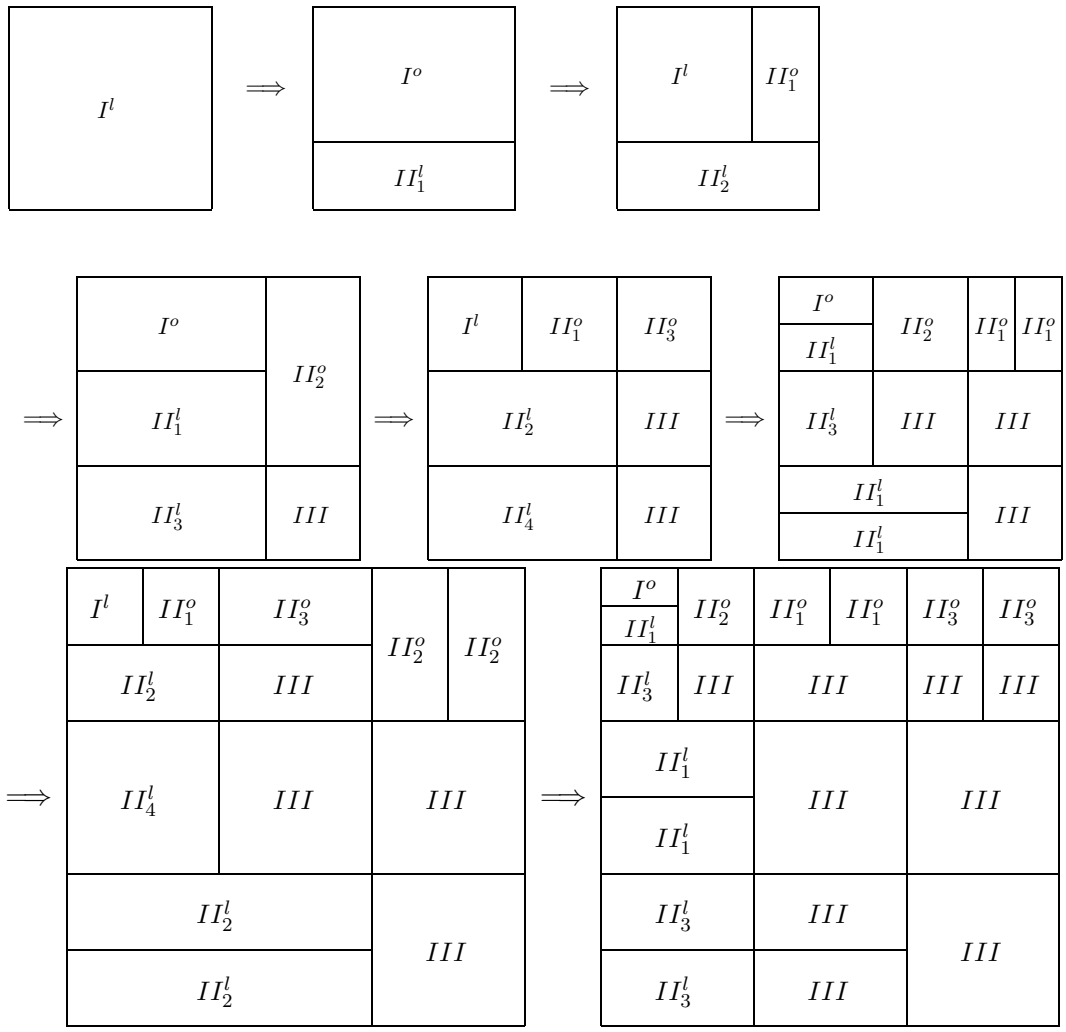

Figure 2.4: First stages of the development of the moss *Phascum cuspidatum*

structure grammars,

- in order to describe an organism we have to take into consideration all cells, independent of the fact whether there exist rules for the cells or the cells do not change in the further development, i.e., we do not distinguish between terminals and nonterminals as in phrase structure grammars.

We now introduce Lindenmayer systems as a new type of rewriting systems. We restrict to the case of words for simplicity. For approaches to multidimensional systems we refer to Section VI.5 of [27], [3] and parallel graph grammars (e.g., [14]). Moreover, we mention that by the method used in the description of the development of some red alga we are able to cover some multidimensional cases as branching structures by means of (linear) words.

Definition 2.1 *A Lindenmayer system without interaction (0L system, for short) is a triple* $G = (V, P, \omega)$ *where*

- V *is an alphabet,*

- *P is a finite complete set of productions over V, i.e., P is a finite subset of $V^+ \times V^*$ and, for any $a \in V$, there is a word w_a such that $(a, w_a) \in P$,*
- *$\omega \in V^+$.*

The elements of the alphabet represent the cells.

Any production of P is a description of a developmental rule. As usual, instead of (a, w) in P we write $a \to w$. Note that by the completeness condition we require that, for any letter or any cell, there is a developmental rule. Thus we have taken the rules $3 \to 3$ and $III \to III$ to describe the development of the red alga and Phascum cuspidatum in the preceding subsection which reflect that the cells are not changed in the further development. However, the set of rules for the red alga is not complete since we have no rules for the letters (and) which are used to model branches. In order to get a complete set one has to add ($\to$ (and) $\to$) which are clear from the biological motivation since the places of branchings do not move during the development.

The word ω represents the organism which we have in the first stage of the development. We call it the start word of the system. Obviously, it is not necessary that we start with a cell which requires that the start element has to be a (non-empty) word.

We now define the derivation process in a 0L system.

Definition 2.2 *Let $G = (V, P, \omega)$ be a 0L system. For two words $x \in V^+$ and $y \in V^*$, we say that x directly derives y in G (written as $x \Longrightarrow_G y$, or $x \Longrightarrow y$ if G is clear from the context) if and only if the following conditions are satisfied:*

- *$x = x_1 x_2 \dots x_n$ where $x_i \in V$ for $1 \leq i \leq n$,*
- *$y = y_1 y_2 \dots y_n$,*
- *$x_i \to y_i \in P$ for $1 \leq i \leq n$.*

Moreover, we sometimes use $\lambda \Longrightarrow_G \lambda$.

By this definition, in every derivation step we replace any letter of x according to rules of P. Thus we have a completely parallel derivation process.

The replacement of a letter x_i of x does not depend on the neighbouring letters x_{i-1} and x_{i+1}; we only have to use a rule of P. Thus there is no interaction between the letters of the word during a derivation. Hence one can say that we have a parallel context-free derivation process. The 0 (zero) in Definition 2.1 stands for no (or 0) interaction.

By $\Longrightarrow^*$ we denotes the reflexive and transitive closure of $\Longrightarrow$. Then $x \Longrightarrow^* y$ holds if and only if $x = y$ (reflexivity) or there are a natural number $r \geq 1$ and words $z_0, z_1, z_2, \dots, z_r$ such that

$$x = z_0 \Longrightarrow z_1 \Longrightarrow z_2 \Longrightarrow \dots \Longrightarrow z_{r-1} \Longrightarrow z_r = y$$

(transitivity).

Definition 2.3 *Let $G = (V, P, \omega)$ be a 0L system. The language $L(G)$ generated by G is defined as*

$$L(G) = \{z \mid \omega \Longrightarrow^* z\}.$$

By this definition, the language generated by a 0L system consists of all words which can be generated from the start element ω.

We set

$$\begin{aligned} L_0(G) &= \{\omega\}, \\ L_n(G) &= \{z \mid v \Longrightarrow z \text{ for some } v \in L_{n-1}(G)\} \quad \text{for} \quad n \geq 1. \end{aligned}$$

By induction (on n) it is easy to prove that $L_n(G)$ consists of all words y such that there is a derivation

$$\omega = z_0 \Longrightarrow z_1 \Longrightarrow z_2 \Longrightarrow \ldots \Longrightarrow z_{n-1} \Longrightarrow z_n = y\,.$$

Thus we get

$$L(G) = \bigcup_{n \geq 0} L_n(G)\,.$$

Before we give some examples we want to mention the differences between 0L systems and the phrase structure grammars.

- We have only one alphabet and no distinction between terminals and nonterminals.
- The language of a 0L system consists of all words generated by the systems, whereas the language generated by a phrase structure grammar only contains words over the terminal alphabet, which is a (proper) subset of all words generated by the grammar.
- In a derivation step of a 0L systems all letters of the current word are replaced, whereas in a derivation step of a phrase structure grammar subwords of a bounded length and in the case of a context-free grammar one letter is only replaced. This means that 0L systems are characterized by a purely parallel derivation process whereas context-free grammars are characterized by a purely sequential process.
- The derivation in a 0L system starts with a non-empty word over the underlying alphabet. In phrase structure grammars the derivation starts with a distinguished nonterminal.

Example 2.4 We consider the 0L system

$$G_1 = (\{a\}, \{a \to a^2\}, a)\,.$$

By induction, we prove that $L_n(G_1) = \{a^{2^n}\}$ for $n \geq 0$. By definition, $L_0(G_1) = \{a\}$ since a is the start word. Thus the basis of the induction is shown. Let $L_n(G_1) = \{a^{2^n}\}$. Because $L_{n+1}(G_1) = \{z \mid a^{2^n} \Longrightarrow z\}$ and $a^{2^n} \Longrightarrow (a^2)^{2^n} = a^{2^{n+1}}$ is the only derivation from a^{2^n}, we get $L_{n+1}(G_1) = \{a^{2^{n+1}}\}$. Therefore the induction step has been proved, too.

Hence we obtain

$$L(G_1) = \bigcup_{n \geq 0} \{a^{2^n}\} = \{a^{2^n} \mid n \geq 0\}\,.$$

Example 2.5 Let

$$G_2 = (\{a, b\}, \{a \to \lambda, b \to ab\}, aab)\,.$$

Then we only have the derivation

$$aab \Longrightarrow \lambda\lambda ab = ab \Longrightarrow \lambda ab = ab \Longrightarrow ab \Longrightarrow ab \Longrightarrow \ldots\,,$$

which results in

$$L(G_2) = \{aab, ab\}\,.$$

Example 2.6 We consider the 0L system

$$G_3 = (\{a\}, \{a \to a, a \to a^2\}, a)\,.$$

We show that

$$L(G_3) = \{a^n \mid n \geq 1\}\,.$$

This can be seen as follows. First, by induction, we prove $a^n \in L_{n-1}(G_3)$. By definition, we have $L_0(G) = \{a\}$. Further, applying $a \to a$ to the first $n-1$ occurrences of a in a^n and $a \to a^2$ to the last letter of a^n, we get $a^n = a^{n-1}a \Longrightarrow a^{n-1}a^2 = a^{n+1}$. Therefore $a^n \in L_{n-1}(G_3)$ implies $a^{n+1} \in L_n(G_3)$, and the induction step is performed. Thus we have

$$\{a^n \mid n \geq 1\} \subseteq \bigcup_{n\geq 0} L_n(G_3) = L(G_3)\,.$$

On the other hand, obviously from a word a^n we can only generate non-empty words over $\{a\}$ by application of $a \to a$ and $a \to a^2$. Hence

$$L(G_3) \subseteq \{a^n \mid n \geq 1\}\,.$$

Example 2.7 Let

$$G_4 = (\{a, b, c, d, e\}, \{a \to a, b \to ba, c \to cbb, d \to da, e \to cbbd\}, e)\,.$$

By definition, $L_0(G_4) = \{e\}$.

We now prove that, for $n \geq 1$,

$$L_n(G_4) = \{cbb(ba)^2(ba^2)^2 \ldots (ba^{n-1})^2 da^{n-1}\}\,.$$

Because there is only one production for e, we only have the derivation $e \Longrightarrow cbbd$. Therefore $L_1(G_4) = \{cbbd\}$ which proves the basis. Furthermore,

$$\begin{aligned} cbb(ba)^2(ba^2)^2 \ldots (ba^{n-1})^2 da^{n-1} &\Longrightarrow cbbbaba(baa)^2(baa^2)^2 \ldots (baa^{n-1})^2 daa^{n-1} \\ &= cbb(ba)^2(ba^2)^2 \ldots (ba^n)^2 da^n \end{aligned}$$

is the only one step derivation with left hand side $cbb(ba)^2(ba^2)^2 \ldots (ba^{n-1})^2 da^{n-1}$. Thus the induction step is shown, too.

Hence we get

$$L(G_4) = \{e\} \cup \{cbbbababa^2ba^2 \ldots ba^n ba^n da^n \mid n \geq 0\}\,.$$

Example 2.8 We consider the 0L system

$$G_5 = (\{a, b, c\}, \{a \to a^2, b \to ab, c \to bc, c \to c\}, abc)\,.$$

We now prove that

$$\begin{aligned} L(G_5) = {} & \{a^{2^n-1}ba^{2^{n_1}-1}ba^{2^{n_2}-1}b \ldots a^{2^{n_r}-1}bbc \mid n > n_1 > n_2 > \ldots n_r \geq 1, r > 0, n \geq 2\} \\ & \cup \{a^{2^n-1}ba^{2^{n_1}-1}ba^{2^{n_2}-1}b \ldots a^{2^{n_r}-1}bc \mid n > n_1 > n_2 > \ldots n_r \geq 1, r \geq 0, n \geq 1\}\,. \end{aligned}$$

Let

$$\begin{aligned} w_{n,n_1,n_2,\ldots,n_r} &= a^{2^n-1}ba^{2^{n_1}-1}ba^{2^{n_2}-1}b\ldots a^{2^{n_r}-1}bbc,\ n \geq 2, \\ w'_{n,n_1,n_2,\ldots,n_r} &= a^{2^n-1}ba^{2^{n_1}-1}ba^{2^{n_2}-1}b\ldots a^{2^{n_r}-1}bc,\ n \geq 1. \end{aligned}$$

Applying $c \to bc$ or $c \to c$, we only get the derivations

$$\begin{aligned} &w_{n,n_1,n_2,\ldots,n_r} \Longrightarrow w_{n+1,n_1+1,n_2+1,\ldots,n_r+1,1} \text{ and } w_{n,n_1,n_2,\ldots,n_r} \Longrightarrow w'_{n+1,n_1+1,n_2+1,\ldots,n_r+1,1}\,, \\ &w'_{n,n_1,n_2,\ldots,n_r} \Longrightarrow w_{n+1,n_1+1,n_2+1,\ldots,n_r+1} \text{ and } w'_{n,n_1,n_2,\ldots,n_r} \Longrightarrow w'_{n+1,n_1+1,n_2+1,\ldots,n_r+1}\,. \end{aligned}$$

Since the start word is w'_1, we can only generate words of the form $w_{n,n_1,n_2,\ldots,n_r}$ or $w'_{n,n_1,n_2,\ldots,n_r}$.

It remains to prove that we can obtain all these words. We prove this by induction on the sum $s = n + n_1 + n_2 + \ldots + n_r$. If $s = 1$, then we have to generate the start word $w'_1 = abc$. We consider two cases:

Case 1: $w_{n,n_1,n_2,\ldots,n_r}$, $n_r \geq 2$.
Then $w'_{n-1,n_1-1,n_2-1,\ldots,n_r-1} \in L(G_5)$ by induction and $w'_{n-1,n_1-1,n_2-1,\ldots,n_r-1} \Longrightarrow w_{n,n_1,n_2,\ldots,n_r}$. Therefore $w_{n,n_1,n_2,\ldots,n_r} \in L(G_5)$.

Case 2: $w_{n,n_1,n_2,\ldots,n_{r-1},1}$.
Then $n_{r-1} \geq 2$ and $w_{n-1,n_1-1,n_2-1,\ldots,n_{r-1}-1} \in L(G_5)$ by induction. Because we have the derivation $w_{n-1,n_1-1,n_2-1,\ldots,n_{r-1}-1} \Longrightarrow w_{n,n_1,n_2,\ldots,n_{r-1},1}$, we get $w_{n,n_1,n_2,\ldots,n_r} \in L(G_5)$.

Thus we can obtain all words of the form $w_{n,n_1,n_2,\ldots,n_r}$ with $r \geq 1$. Analogously, we can prove that all words of the forms $w'_{n,n_1,n_2,\ldots,n_r}$ with $r \geq 1$, w_n and w'_n can be generated.

Example 2.9 We consider the 0L system

$$G_6 = (\{a, b, c, d, e, f\},\ \{a \to dabc, a \to f, a \to e, b \to bc, c \to \lambda, d \to e, e \to e\},\ a)\,.$$

It is easy to see that

$$L(G_6) = \{a, e\} \cup \{e^{n-1}da(bc)^n \mid n \geq 1\} \cup \{e^{n+1}(bc)^n \mid n \geq 1\} \cup \{e^n f^{2^m}(bc)^n \mid n \geq 1, m \geq 0\}\,.$$

Giving the above definitions we followed the method to define phrase structure grammars and their languages. However, we can give a alternative definition of 0L systems based on algebraic concepts.

A mapping $\sigma : V^* \to 2^{W^*}$ is called a substitution if the following relations hold:

$$\begin{aligned} &\sigma(\lambda) = \{\lambda\}\,, \\ &\sigma(xy) = \sigma(x)\sigma(y) \text{ for } x, y \in V^*\,. \end{aligned}$$

In order to define a substitution it is sufficient to give the sets $\sigma(a)$ for any letter $a \in V$. Then we can determine $\sigma(a_1a_2\ldots a_n)$ for a word $a_1a_2\ldots a_n$ with $a_i \in V$ for $1 \leq i \leq n$ by

$$\sigma(a_1a_2\ldots a_n) = \sigma(a_1)\sigma(a_2)\ldots\sigma(a_n)$$

which is a generalization of the second relation in the definition of a substitution. Moreover, for a language L, we set

$$\sigma(L) = \bigcup_{x \in L} \sigma(x)\,.$$

Furthermore, we set

$$\begin{aligned} \sigma^0(x) &= \{x\}\,, \\ \sigma^n(x) &= \underbrace{\sigma(\sigma\ldots(\sigma(x))\ldots)}_{n \text{ times}} \text{ for } n \geq 1. \end{aligned}$$

Let $G = (V, P, \omega)$ be a 0L system. Then we define the substitution $\sigma_G : V^* \to 2^{V^*}$ by

$$\sigma(a) = \{w \mid a \to w \in P\}\,.$$

Then it follows that

$$x \Longrightarrow_G y \text{ if and only if } y \in \sigma_G(x)$$

because in both cases we replace all letters x_i of x by an element of $\sigma_G(x_i)$. Consequently we get

$$\begin{aligned} L_0(G) &= \{\omega\} = \sigma_G^0(\omega)\,, \\ L_1(G) &= \sigma_G(\omega) = \sigma_G^1(\omega)\,, \\ L_2(G) &= \sigma_G(L_1(G)) = \sigma_G(\sigma_G(\omega)) = \sigma_G^2(\omega) \end{aligned}$$

and, by induction,

$$L_n(G) = \sigma_G^n(\omega)\,.$$

This implies

$$L(G) = \bigcup_{n \geq 0} \sigma_G^n(\omega)\,.$$

We now define some special cases.

Definition 2.10 *i) A 0L system $G = (V, P, \omega)$ is called propagating (P0L system, for short) if $a \to w \in P$ implies $w \neq \lambda$.*

ii) A 0L system $G = (V, P, \omega)$ is called deterministic (D0L system, for short) if, for any $a \in V$, $a \to w \in P$ and $a \to v \in P$ imply $w = v$.

iii) A PD0L system is a 0L system which is propagating as well as deterministic.

In Figure 2.5 we summarize to which special cases the grammars of our examples belong.

Let $X \in \{0L,\ P0L, D0L, PD0L\}$. If L is a language such that $L = L(G)$ for some X system G, then we say that L is an X language. Moreover, by $\mathcal{L}(X)$ we denote the family of all languages generated by X systems. [1] Thus we get the families $\mathcal{L}(PD0L)$, $\mathcal{L}(D0L)$, $\mathcal{L}(P0L)$ and $\mathcal{L}(0L)$ of all PD0L, all D0L, all P0L and all 0L languages, respectively.

[1] In order to be precise we consider a countable set U and require that the underlying alphabet V of G is a (finite) subset of U. Hence we have in a family $\mathcal{L}(X)$ only languages over finite subsets of U. If we do not make such a restriction, it is not clear what are "all" languages.

grammar	PD0L	D0L	P0L
G_1	+	+	+
G_2	–	+	–
G_3	–	–	+
G_4	+	+	+
G_5	–	–	+
G_6	–	–	–

Figure 2.5: A + (a –, respectively) in the intersection of the row associated with G and the column associated with the case X if G is (not) a X system.

2.1.3 The Basic Hierarchy

We start with two lemmas which show that without loss of generality we can assume that derivations of the empty word have a bounded length and that the length of intermediate words in a derivation of x can be bounded linearly in the length of x.

Lemma 2.11 *Let $G = (V, P, \omega)$ be a 0L system with $n = \#(V)$. For $a \in V$, let $G_a = (V, P, a)$. If $\lambda \in L(G_a)$, then $\lambda \in L_m(G_a)$ for some $m \leq n$.*

Proof. We define L_r as the set of all letters $a \in V$ such that $\lambda \in L_m(G_a)$ for some $m \leq r$. Obviously, if $a \in L_r$ then $a \in L_{r+1}$, too. Thus we have $L_r \subseteq L_{r+1}$ for $r \geq 1$.

Let $L_r = L_{r+1}$. Further let $a \in L_{r+2}$. Then there is a derivation

$$a \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \ldots \Longrightarrow w_s = \lambda$$

with $s \leq r + 2$. If $s < r + 2$, then $a \in L_{r+1}$. Let $s = r + 2$. Then $b \in L_{r+1}$ for any letter b which occurs in w_1. By our assumption, $b \in L_r$ for any b occurring in w_1. Hence there is a derivation

$$a \Longrightarrow w_1 \Longrightarrow v_2 \Longrightarrow v_3 \Longrightarrow \ldots \Longrightarrow v_{r-1} \Longrightarrow \lambda\,.$$

This implies $a \in L_{r+1}$. Therefore in both cases we have shown that $a \in L_{r+1}$. This gives $L_{r+2} \subseteq L_{r+1}$ which proves $L_{r+2} = L_{r+1} = L_r$. By induction we can show that $L_{r+k} = L_r$ for all $k \geq 1$.

Thus there is a number $t \geq 1$ such that

$$L_1 \subset L_2 \subset L_3 \subset \ldots \subset L_{t-1} \subset L_t = L_{t+1} = L_{t+2} = \ldots\,.$$

Because L_t is a subset of V, t is smaller than the number of letters of V. Therefore $t \leq n$.

Now assume that $\lambda \in L(G_a)$, then $a \in L_t$ and thus $\lambda \in L_m(G_a)$ for some $m \leq t \leq n$. □

Lemma 2.12 *Let $G = (V, P, \omega)$ be a 0L system. Then there exists a constant C_G such that, for any word $x \in L(G)$, there is a derivation*

$$\omega = w_0 \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \ldots \Longrightarrow w_r = x$$

with $|w_i| \leq C_G \cdot (|x| + 1)$.

Proof. Let

$$\begin{aligned} n &= \#(V)\,, \\ k &= \max\{|w| \mid a \to w \in P\}\,, \\ l &= \max\{|z| \mid z \in L_m(G) \text{ for } m \leq n\}\,, \\ C_G &= \max\{k^n, l\}\,. \end{aligned}$$

Let $x \in L(G)$. Then $x \in L_r(G)$ for some $r \geq 0$. Let

$$\omega = w_0 \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \ldots \Longrightarrow w_r = x\,.$$

Assume that there is some letter $a \in V$ in the word w_j, $1 \leq j \leq r$, such that the subderivation from a yields the empty word. Then we substitute this subderivation by a derivation of λ which has at most n steps. Such a derivation exists by Lemma 2.11. This procedure is done as long the derivation contains subderivations of the empty word with more than n steps. As a result we obtain a derivation

$$\omega = v_0 \Longrightarrow v_1 \Longrightarrow v_2 \Longrightarrow \ldots \Longrightarrow v_s = x$$

with $s \leq r$. We now prove that $|v_i| \leq C_G(|x| + 1)$.

If $i \leq n$, then $v_i \in L_i(G)$ and therefore $|v_i| \leq l \leq C_G \leq C_G(|x| + 1)$ which proves the statement of the theorem.

If $i \geq n$, then we consider the word $v_{i-n} = u_1 u_2 \ldots u_t$ where $u_j \in V$ for $1 \leq j \leq t$. Then $x = u_1' u_2' \ldots u_t'$ where $u_j' \neq \lambda$ is obtained from u_j or $u_j' = \lambda$ (if from u_j a subderivation starts which yields the empty word). Let h be the number of letters u_j of v_{i-n} such that $u_j' \neq \lambda$. Then $h \leq |x|$. Moreover, since the subderivations giving λ are finished after n derivation steps by our construction v_i is build from the words u_j' which are obtained after n steps from u_j. By definition of k we have $|u_j'| \leq k^n$ and hence

$$|v_i| \leq hk^n \leq C_G(|x| + 1)\,.$$

This proves the theorem. □

We now compare the families generated by Lindenmayer systems with each other and with the families of the Chomsky hierarchy.

Theorem 2.13 *The diagram of Figure 2.6 holds.*

Proof. The part $\mathcal{L}(FIN) \subset \mathcal{L}(REG) \subset \mathcal{L}(CF) \subset \mathcal{L}(CS)$ is well-known as a part of the Chomsky hierarchy (see Theorem 1.3).

Since any PD0L system is a P0L system, too, it follows that $\mathcal{L}(PD0L) \subseteq \mathcal{L}(P0L)$. Analogously we obtain the other inclusions between $\mathcal{L}(PD0L)$, $\mathcal{L}(P0L)$, $\mathcal{L}(D0L)$ and $\mathcal{L}(0L)$.

In order to prove the strictness of the inclusions it is sufficient to prove the existence of languages L_1 and L_2 such that

$$L_1 \in \mathcal{L}(P0L),\ L_1 \notin \mathcal{L}(D0L) \text{ and } L_2 \in \mathcal{L}(D0L),\ L_2 \notin \mathcal{L}(P0L)\,.$$

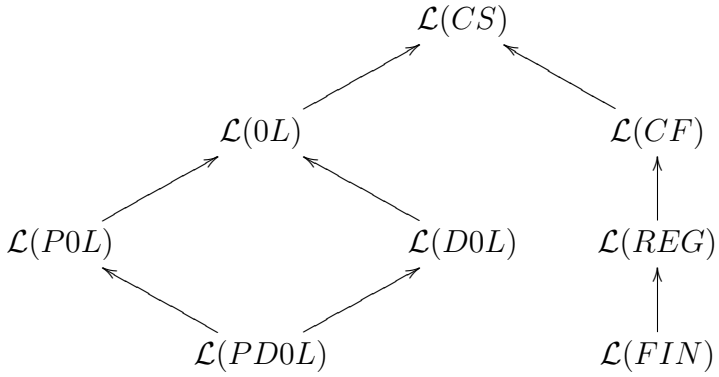

Figure 2.6: $\mathcal{L}_1 \to \mathcal{L}_2$ denotes a proper inclusion of $\mathcal{L}_1$ in $\mathcal{L}_2$. If two families are not connected by arrows, then they are incomparable.

Then $L_1 \in \mathcal{L}(P0L) \setminus \mathcal{L}(PD0L)$ and $L_1 \in \mathcal{L}(0L) \setminus \mathcal{L}(D0L)$ which proves the properness of two inclusions. L_2 can be used to show the strictnesses of the other two inclusions.

We consider $L_1 = \{a\}^+$. Because $L_1 = L(G_3)$ for the P0L system G_3 from Example 2.6, $L_1 \in \mathcal{L}(P0L)$ by definition. Let us assume that $L_1 \in \mathcal{L}(D0L)$. Then there is a D0L system $G = (\{a\}, \{a \to a^r\}, a^s)$ with $L(G) = L_1$. Since

$$a^s \Longrightarrow a^{sr} \Longrightarrow a^{sr^2} \Longrightarrow a^{sr^3} \Longrightarrow \ldots \Longrightarrow a^{sr^k} \Longrightarrow \ldots$$

is the only derivation in G, we get $L(G) = \{a^{sr^n} \mid n \geq 0\}$. Now it is easy to see that, for $r \geq 2$,

$$a^{s+1} \in L_1 \text{ and } a^{s+1} \notin \{a^{sr^n} \mid n \geq 0\}\,.$$

If $r = 1$, then $L(G) = \{a^s\}$. In both cases we have $L_1 \neq L(G)$ in contrast to the choice of G. Hence $L_1 \notin \mathcal{L}(D0L)$.

Let $L_2 = \{aab, ab\}$. By $L_2 = L(G_2)$ for the D0L system G_2 of Example 2.5, we have $L_2 \in \mathcal{L}(D0L)$. If $L_2 \in \mathcal{L}(P0L)$, then $L(G) = L_2$ for some P0L system $G' = (\{a, b\}, P, \omega)$. By the completeness we have rules $a \to w_a$ and $b \to w_b$ in P. Then we obtain $aab \Longrightarrow w_a w_a w_b$ and $w_a w_a w_b \in L(G') = L_2$. Since G' is propagating, w_a and w_b are non-empty words which implies that the length of $w_a w_a w_b$ is at least 3. Therefore $w_a = a$ and $w_b = b$. Thus $aab \Longrightarrow_{G'} aab$ and $ab \Longrightarrow_{G'} ab$ are the only direct derivation steps. This implies $L(G') = \{\omega\}$, i.e., $L(G')$ consists of one word, which contradicts $L(G') = L_2$ since L_2 contains two words. Hence $L_2 \notin \mathcal{L}(P0L)$.

Let $X \in \{DP0L, P0L, D0L, 0L\}$ and $Y \in \{FIN, REG, CF\}$. In order to prove that $\mathcal{L}(X)$ and $\mathcal{L}(Y)$ are incomparable, it is sufficient to present languages

$$L_3 \in \mathcal{L}(FIN),\ L_3 \notin \mathcal{L}(0L) \text{ and } L_4 \in \mathcal{L}(PD0L),\ L_4 \notin \mathcal{L}(CF)\,.$$

We choose $L_3 = \{a^2, a^4\}$. Obviously, $L_3 \in \mathcal{L}(FIN)$. If $L_3 \in \mathcal{L}(0L)$, then there is a 0L system $H = (\{a\}, P, \omega)$ with $L_3 = L(H)$. Let $a \to w_a \in P$. Then we get $a^4 \Longrightarrow (w_a)^4 \in L_3$ or $(w_a)^4 = \lambda$ (because $L_3 = L(H)$ means that both languages can differ in the empty word by our convention). In the former case, we have $w_a = a$, and in the latter case, we get $w_a = \lambda$. Thus we have to consider the following three cases:

Case 1. $P = \{a \to a\}$.
Then $\omega \Longrightarrow \omega$ holds which yields $L(H) = \{\omega\}$ in contrast to the choice of H such that $L(H) = L_3$.

Case 2. $P = \{a \to \lambda\}$.
Then $\omega \Longrightarrow \lambda$ holds which gives $L(H) = \{\omega, \lambda\}$ in contrast to the choice of H.

Case 3. $P = \{a \to a, a \to \lambda\}$.
Then $a^4 = aaaa \Longrightarrow aaa\lambda = a^3$. This implies $a^3 \in L(H)$ which contradicts $L(H) = L_3$.

Therefore $L_3 \notin \mathcal{L}(0L)$.

Let $L_4 = \{a^{2^n} \mid n \geq 2\}$. By Example 2.4, $L_4 = L(G_1)$ for the PD0L system G_1 and thus $L_4 \in \mathcal{L}(PD0L)$. On the other hand, it is well-known that $L_4 \notin \mathcal{L}(CF)$ (which can easily be proved by the use of a pumping lemma, see Theorem 1.13).

Let $G = (V, P, \omega)$ be a 0L system. We construct a phrase structure grammar $H = (N, V, P', S)$ with $L(H) = L(G)$ as follows. We set $N = \{A, B, C, D, E\}$ and define P' as the set of all rules of the following types:

a) $S \to AD\omega B$,
b) $AD \to AC$, $AD \to AE$,
c) $Ca \to wC$ for $a \to w \in P$ and $CB \to DB$,
d) $aD \to Da$ for $a \in V$,
e) $AEa \to aE$ and $Ea \to aE$ for $a \in V$, $AEB \to \lambda$, $EB \to \lambda$.

Any derivation in H starts with $S \Longrightarrow_H AD\omega B$. Let us now assume that we have generated a word $ADa_1a_2 \dots a_nB$ with $a_i \in V$ for $1 \leq i \leq n$. Now we can only apply the rules of type b) and we obtain $ACa_1a_2 \dots a_nB$ or $AEa_1a_2 \dots a_nB$.

In the former case we have to continue with rules of type c) which gives

$$\begin{aligned} ACa_1a_2 \dots a_nB &\Longrightarrow_H Aw_1Ca_2 \dots a_nB \Longrightarrow_H \dots \Longrightarrow_H Aw_1w_2 \dots w_nCB \\ &\Longrightarrow_H Aw_1w_2 \dots w_nDB \end{aligned}$$

where $a_i \to w_i \in P$ for $1 \leq i \leq n$. By rules of type d) we shift the letter D to the left and obtain $ADw_1w_2 \dots w_nB$. That is, we have obtained a word of the form we start with and – besides the nonterminals – we have simulated the derivation $a_1a_2 \dots a_n \Longrightarrow_G w_1w_2 \dots w_n$.

In the latter case, if $n = 0$, we get $AEB \Longrightarrow_H \lambda$, and if $n \geq 1$ we get the derivation

$$AEa_1a_2 \dots a_n \Longrightarrow_H a_1Ea_2 \dots a_nB \Longrightarrow_H \dots \Longrightarrow_H a_1a_2 \dots a_nEB \Longrightarrow_H a_1a_2 \dots a_n\,.$$

That is, we only delete the nonterminals. (Note that other derivations are not possible since the application of $Ea_1 \to a_1E$ would not delete the letter A which remains such that the derivation cannot terminate.)

Therefore, for any derivation

$$\omega \Longrightarrow_G v_1 \Longrightarrow_G v_2 \Longrightarrow_G \dots \Longrightarrow_G v_n$$

in G, there is a derivation

$$S \Longrightarrow_H AD\omega B \Longrightarrow_H^* ADv_1B \Longrightarrow_H^* ADv_2B \Longrightarrow_H^* \dots \Longrightarrow_H^* ADv_nB \Longrightarrow_H^* AEv_nB \Longrightarrow_H^* v_n$$

and conversely. Thus $L(G) = L(H)$.

Let x be a non-empty word of $L(G)$. By Lemma 2.12, there is a derivation of x such that any intermediate word has a length bounded by $C_G(|x|+1)$. Therefore all the intermediate words of the corresponding derivation of x in H have at most the length $C_G(|x|+1)+3 \leq C|x|$ for an appropriate constant C. By the workspace theorem (see Theorem 1.9), the language of all non-empty words of $L(H)$ is context-sensitive. Taking into consideration our convention concerning the equality of languages we have that the 0L language $L(G)$ is context-sensitive. □

2.1.4 Adult languages

The language generated by a 0L system consists of all words which can be derived from the start word. Thus it takes into consideration all phases of the development, e.g. in case of a flower the "green" phase, which produces the handle or stem and the leaves, as well as the "flowering" phase, where the blossom is build and the parts of the "green" phase are not changed. Especially, one is interested in the final stages or adult stages which are not changed (at least for a long time).

Modelling this aspect within the framework of 0L systems we are interested in those strings w which belong to the language and only allow the derivation $w \Longrightarrow w$. This leads to the following definition.

Definition 2.14 *The adult language $L_A(G)$ of a 0L system G is the set of all words $z \in L(G)$ such that, for any $v \in V^*$, $w \Longrightarrow v$ implies $w = v$.*

The adult alphabet $V_A(G)$ is set of all letters of V, which occur in words of $L_A(G)$.

Let $X \in \{0L, P0L, D0L, PD0L\}$. By $\mathcal{L}(AX)$ we denote the family of adult languages generated by X systems.

Example 2.15 We consider the 0L system G_6 from Example 2.9 Since

$$L(G_6) = \{a, e\} \cup \{e^{n-1}da(bc)^n \mid n \geq 1\} \cup \{e^n(bc)^n \mid n \geq 1\} \cup \{e^n f^{2^m}(bc)^n \mid n \geq 1, m \geq 0\}$$

we obtain the adult language

$$L_A(G_6) = \{e\} \cup \{e^n(bc)^n \mid n \geq 1\}.$$

Note that the adult language of G_6 is a context-free language whereas $L(G_6)$ is not context-free.

The aim of this section is to show that the fact seen in the example holds in general, i.e., any adult language of a 0L system is a context-free language, and conversely, any context-free language is an adult language of some 0L system.

Theorem 2.16 *For any context-free grammar G, there is a propagating 0L system G' such that $L_A(G') = L(G)$.*

Proof. Let $G = (N, T, P, S)$ be a context-free grammar. First we construct a context-free grammar $G'' = (N'', T, P'', S'')$ in Chomsky normal form such that $L(G'') = L(G)$ (see Theorem **??**. Then we define the 0L system G' by

$$G' = (N'' \cup T, P'' \cup \{a \to a \mid a \in N'' \cup T\}, S'') .$$

Since G'' is in Chomsky normal form, P'' contains no rule of the form $A \to \lambda$. Thus G' is propagating.

Let $x \Longrightarrow_{G''} y$. Then $x = u_1 A u_2$ and $y = u_1 w u_2$ for some rule $A \to w \in P''$. Then we also have $x \Longrightarrow_{G'} y$ by applying $a \to a$ to all letters of u_1 and u_2 and $A \to w \in P'$ to the distinguished occurrence of A in x.

Moreover, if $x \Longrightarrow_{G'} y$, then $x = x_1 x_2 \to x_n$, $y = y_1 y_2 \dots y_n$ and $x_i \to y_i \in P'$ for $1 \le i \le n$. Let $M = \{i_1, i_2, \dots, i_r\}$ be the subset of $\{1, 2, \dots, n\}$ such that $x_i \to y_i \neq x_i$ for $i \in M$ and $x_j \to y_j = x_j$ for $j \in \{1, 2, \dots, n\} \setminus M$. Then $i \in M$ implies $x_i \in N''$ and we have in G'' the derivation

$$\begin{aligned}
x &= u_1 x_{i_1} u_2 x_{i_2} \dots u_r x_{i_i} u_{r+1} \\
&\Longrightarrow_{G''} u_1 y_{i_1} u_2 x_{i_2} \dots u_r x_{i_i} u_{r+1} \\
&\Longrightarrow_{G''} u_1 y_{i_1} u_2 y_{i_2} u_3 x_{i_3} \dots u_r x_{i_i} u_{r+1} \\
&\quad \dots \\
&\Longrightarrow_{G''} u_1 y_{i_1} u_2 y_{i_2} \dots u_{r-1} y_{i_{r-1}} u_r x_{i_i} u_{r+1} \\
&\Longrightarrow_{G''} u_1 y_{i_1} u_2 y_{i_2} \dots u_{r-1} y_{i_{r-1}} u_r y_{i_i} u_{r+1} \\
&= y
\end{aligned}$$

This proves that $L(G')$ is the set of all sentential forms of G''.

If $x = u_1 A u_2$ is a sentential form of G'' with $A \in N''$, then we apply to all letters of u_1 and u_2 rules of the form $a \to a$ and to A a rule $A \to w$ with $w \neq A$ (such a rule exists since G'' is in Chomsky normal form) and obtain $u_1 w u_2 \neq x$. Hence x is not in the adult language of G'. On the other hand, if a sentential form x' only contains terminals, then we can only apply identity rules to the letters of x' which gives $x' \in L_A(G')$.

Therefore $L_A(G')$ consists of all sentential forms which only contain terminals, i.e., $L_A(G') = L(G'') = L(G)$. □

Lemma 2.17 *Let $G = (V, P, \omega)$ be a 0L system. Let $m = \#(V_A(G))$. For any letter $a \in V_A(G)$, let $G_a = (V, P, a)$. Then, for any $a \in V_A(G)$, $\lambda \in L_t(G_a)$ for some $t \le m$ or $L_n(G_a)$ contains exactly one word z_a and, for all words $v \in V^*$, $z_a \Longrightarrow v$ implies $v = z_a$.*

Proof. Let $a \in V_A(G)$. Then there is exactly one rule $a \to w_a$ in P. Assume the contrary, i.e., $a \to w_1 \in P$ and $a \to w_2 \in P$ with $w_1 \neq w_2$. For a word $x \in L_A(G)$ where a occurs in x, i.e., $x = x_1 a y_1$, we then have two derivation $x_1 a y_1 \Longrightarrow x_2 w_1 y_2$ and $x_1 a y_1 \Longrightarrow x_2 w_2 y_2$ with $x_2 w_1 y_2 \neq x_2 w_2 y_2$ in contrast to the property of the words of $L_A(G)$.

Since $x = x_1 a y_1 \Longrightarrow x_2 w_a y_2 = x \in L_A(G)$, all letters of w_a belong to the adult alphabet $V_A(G)$. Thus there is a unique derivation in G_a. Therefore, for any $t \ge 0$, $L_t(G)$ contains at most one word.

If w_a contains two occurrences of a, then we have a derivation

$$\begin{aligned} w_a &= u_1au_2au_3 \Longrightarrow u_1'w_au_2'w_au_3' \\ &= u_1'u_1au_2au_3u_2'u_1au_2au_3u_3' \Longrightarrow u_1''u_1'w_au_2'w_au_3'u_2''u_1'w_au_2'w_au_3'u_3'' \\ &= u_1''u_1'u_1au_2au_3u_2'u_1au_2au_3u_3'u_2''u_1'u_1au_2au_3u_2'u_1au_2au_3u_3'u_3'' \\ &\quad \ldots\ldots \end{aligned}$$

and therefore from w_a we can generate a word with an arbitrarily large number of occurrences of a. Therefore, from x we can also generate a word with an arbitrarily large number of occurrences of a. Again, we have a contradiction to the property defining the words of $L_A(G)$.

Now assume that w_a contains exactly one occurrence of a. Let $w_a = p_1aq_1$. Then we have the derivation

$$x = x_1ay_1 \Longrightarrow x_2p_1aq_1y_2 \Longrightarrow x_3p_2p_1aq_1q_2y_3 \Longrightarrow \ldots \Longrightarrow x_mp_{m-1}p_{m-2}\ldots p_1aq_1q_2\ldots q_{m-1}y_m\,.$$

Obviously, if $p_i \neq \lambda$ or $q_i \neq \lambda$ for $i \geq 1$, then we can generate arbitrarily long words from x in contrast to $x \in L_A(G)$. If λ can be generated from p_1 and q_1 then by Lemma 2.11 we have derivations

$$p_1 \Longrightarrow p_2 \Longrightarrow p_3 \Longrightarrow \ldots \Longrightarrow p_s \Longrightarrow \lambda \text{ and } q_1 \Longrightarrow q_2 \Longrightarrow q_3 \Longrightarrow \ldots \Longrightarrow q_t \Longrightarrow \lambda$$

with $s < m$ and $t < m$ and non-empty words $p_1, p_2, \ldots, p_s, q_1, q_2, \ldots, q_t$. Then we get from a in at most m steps the word $p_sp_{s-1}\ldots p_1aq_1q_2\ldots q_t$ which has the property

$$p_sp_{s-1}\ldots p_1aq_1q_2\ldots q_t \Longrightarrow p_sp_{s-1}\ldots p_1aq_1q_2\ldots q_t\,.$$

Thus $z_a = p_sp_{s-1}\ldots p_1aq_1q_2\ldots q_t$ has the properties required in the statement.

Now assume that w_a contains no occurrence of a. For $i \geq 1$, let z_i be the unique word in $L_i(G_a)$. We first prove that z_i has no occurrence of a for $i \geq 1$. Let x be a word of $L_A(G)$ containing a. Then $x = x_0ax_1ax_2\ldots ax_r$ with $r \geq 1$ and x_i does not contain a for $0 \leq i \leq r$. If u defined by $x_0 \Longrightarrow u$ contains an occurrence of a and z_j contains a for some $j \geq 1$, $x \Longrightarrow^j u_0z_ju_1z_ju_2z_j\ldots z_ju_r = x$. Because u_0 starts with u which contains a and any z_j contains an a, the generated word x contains at least $r+1$ occurrences of a which is impossible. Analogously we get a contradiction if u contains no a, i.e., all a's are contained in the word derived from $x_1ax_2a\ldots ax_r$.

Now we have $x = x_1ay_1 \Longrightarrow x_2z_1y_2 = x$. Because a does not occur in $z_1 = w_a$, a has to occur in at least one of the words x_2 and y_2. We only discuss the case that a occurs in x_2, the other case can be handled analogously. Then $x = v_1av_2z_1y_2$. Now we get $x \Longrightarrow v_1'z_1v_2'z_2y_2' = x$. Continuing in this way we get that x contains all the words $z_1, z_2, z_3, \ldots$ which is only possible if $z_i = \lambda$ for some i. From Lemma 2.11 we know that $\lambda \in L_t(G_a)$ for some $t \leq n$. □

Lemma 2.18 *For any 0L system G, there is a 0L system $G' = (V, P', S)$ such that $L_A(G') = L_A(G)$ and, for any $a \in V_A(G')$, the only production in P is $a \to a$.*

Proof. Let $G = (V, P, S)$. Without loss of generality we assume that the start word of G is a letter S not belonging to the adult alphabet $V_A(G)$ (if this is not the case, we add S to the alphabet and $S \to \omega$ to the set of productions, where ω is the original start word; these additions do not change the adult language). Let $n = \#(V_A(G))$. For any letter $a \in V_A(G)$, let $G_a = (V, P, a)$.

We define the homomorphism h by

$$\begin{aligned} &h(a) = \lambda \text{ if } a \in V_A(G) \text{ and } \lambda \in L_m(G_a) \text{ for some } m \leq n\,, \\ &h(a) = z_a \text{ if } a \in V_A(G) \text{ and } z_a \in L_n(G_a)\,, \\ &h(a) = a \text{ if } a \notin V_A(G)\,. \end{aligned}$$

By Lemma 2.17 the homomorphism is well defined. We set

$$\begin{aligned} G' &= (V, P', S)\,, \\ P' &= \{a \to h(w) \mid a \to w \in P \text{ and } a \notin V_A(G)\} \cup \{a \to a \mid a \in V_A(G)\}\,. \end{aligned}$$

Now one can easily prove by induction on the number i of derivation steps that

$$S \Longrightarrow_G^i x \text{ if and only if } S \Longrightarrow_{G'}^i h(x)\,. \tag{2.1}$$

Now let $x = x_1x_2 \dots x_m$ be a word of $L_A(G)$ with $x_i \in V_A(G)$ for $1 \leq i \leq m$. Then $x_1x_2 \dots x_m \Longrightarrow_{G'} x_1x_2 \dots x_m$ by the definition of P' and this is the only possible derivation of x. Therefore $x \in L_A(G')$. Therefore $L_A(G) \subseteq L_A(G')$.

Now let w be a word of $L_A(G')$. By (2.1) there is a word w' such that $w' \in L(G)$ and $w = h(w')$. We assume that $w' = w_0B_1w_1B_2w_2 \dots w_{t-1}B_tw_t$ with $t \geq 0$, $w_i \in V_A(G)^*$ for $0 \leq i \leq t$ and $B_j \notin V_A(G)$ for $1 \leq j \leq t$. For $0 \leq i \leq t$, we define z_i as follows. If $w_i = \lambda$, then we set $z_i = \lambda$. If $w_i \neq \lambda$, then w_i consists only of letters $V_A(G)$. Then we define z_i as the only word which can be generated from w_i in m steps in G. Therefore

$$h(w_i) = z_i \quad \text{and} \quad z_i \Longrightarrow_G z_i \tag{2.2}$$

for $1 \leq i \leq n$. Then

$$h(w') = z_0B_1z_1B_2z_2 \dots z_{t-1}B_tz_t = w\,.$$

For $0 \leq i \leq t$, by the definition of P', if $z_i \neq \lambda$,

$$z_i \Longrightarrow_{G'} z_i \tag{2.3}$$

holds, because all letters of z_i belong to $V_A(G)$. Let $B_rB_{r+1} \dots B_s$ be a subword of w only consisting of letters not in $V_A(G)$, i.e $z_r = z_r + 1 = \dots z_{s-1} = \lambda$, and the letters before B_r and after B_s in w – if they exist – are from $V_A(G)$. Since $w \in L_A(G')$, we have $w \Longrightarrow_{G'} w$. Taking into consideration (2.3) we get

$$B_rB_{r+1} \dots B_s \Longrightarrow_{G'} B_rB_{r+1} \dots B_s\,. \tag{2.4}$$

For $r \leq i \leq s$, if $B_i \to y_i$ in P, then $B_i \to h(y_i) \in P'$. Thus

$$B_rB_{r+1} \dots B_s \Longrightarrow_{G'} h(y_r)h(y_{r+1}) \dots h(y_s)\,.$$

By (2.4),

$$B_r B_{r+1} \dots B_s = h(y_r)h(y_{r+1}) \dots h(y_s)\,. \tag{2.5}$$

Therefore, for $r \leq i \leq s$, $h(y_i)$ contains only letters not in $V_A(G)$, which implies $h(y_i) = y_i$ by the definition of h (since letters of $V_A(G)$ occur in $h(y_i)$ otherwise). By (2.5) this yields

$$B_r B_{r+1} \dots B_s \Longrightarrow_G y_r y_{r+1} \dots y_s = h(y_r)h(y_{r+1}) \dots h(y_s) = B_r B_{r+1} \dots B_s\,.$$

If we combine this relation with (2.2) we get $w \Longrightarrow_G w$ is the only derivation for $w = h(w')$ which proves that $w \in L_A(G)$. Thus we obtain $L_A(G') \subseteq L_A(G)$. □

Theorem 2.19 *For any 0L system G, there is a context-free grammar G'' such that $L(G'') = L_A(G)$.*

Proof. First, for G, we consider the 0L system $G' = (V, P', S)$ according to in Lemma 2.18, i.e., $a \to a$ is the only rule for $a \in V_A(G')$ and $L_A(G') = L_A(G)$. Then we construct the context-free grammar

$$G'' = (V \setminus V_A(G'), V_A(G'), P' \setminus \{a \to a \mid a \in V_A(G')\}, S)\,.$$

It is easy to show that $L(G'') = L_A(G')$ □

Theorem 2.20 *i) $\mathcal{L}(A0L) = \mathcal{L}(AP0L) = \mathcal{L}(CF)$.*
ii) $\mathcal{L}(AD0L) = \mathcal{L}(APD0L) = \{\{w\} \mid w \in V^+\} \cup \{\emptyset\}$.

Proof. i) By Theorem 2.16, we get $\mathcal{L}(CF) \subseteq \mathcal{L}(AP0L)$. Obviously, $\mathcal{L}(AP0L) \subseteq \mathcal{L}(A0L)$. Furthermore, By Theorem 2.19, we have $\mathcal{L}(A0L) \subseteq \mathcal{L}(CF)$. Combining these relations we get the statement.

ii) If $G = (V, P, \omega)$ is a D0L system, then we have only one derivation $\omega = w_0 \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow w_3 \Longrightarrow \dots$ If there is an i such that $w_i = w_{i+1}$, then we have $L_A(G) = \{w_i\}$. If $w_i \neq w_{i+1}$ for any i, then $L_A(G) = \emptyset$. By our convention, the languages $\{\lambda\}$ and $\emptyset$ are equal, since they differ in the empty word only. This proves

$$\mathcal{L}(AD0L) \subseteq \{\{w\} \mid w \in V^+\} \cup \{\emptyset\}\,. \tag{2.6}$$

Let w be a non-empty word over some alphabet V. Then we consider the propagating D0L system $G = (V, \{a \to a \mid a \in V\}, w)$. Obviously, $w \Longrightarrow w \Longrightarrow w \Longrightarrow \dots$ is the only derivation in G. Thus $L_A(G) = \{w\}$. Furthermore, $L_A(G_1)$ is empty. Therefore

$$\{\{w\} \mid w \in V^+\} \cup \{\emptyset\} \subseteq \mathcal{L}(APD0L)\,. \tag{2.7}$$

If we combine (2.6) and (2.7) with $\mathcal{L}(APD0L) \subseteq \mathcal{L}(AD0L)$ (which holds by definition), then we obtain the statement.

□

2.1.5 Decision problems

In this section we want to discuss the decidability status of the classical decision problems considered in the theory of formal languages for interactionless Lindenmayer systems. For $X \in \{0L, P0L, D0L, PD0L\}$, we regard the following problems.

Membership problem:	Given X system $G = (V, P, \omega)$ and $w \in V^*$, decide whether or not $w \in L(G)$.
Emptiness problem:	Given X system $G = (V, P, \omega)$, decide whether or not $L(G)$ is empty,
Finiteness problem:	Given X system $G = (V, P, \omega)$, decide whether or not $L(G)$ is finite.
Equivalence problem:	Given X systems $G = (V, P, \omega)$ and $H = (V, P', \omega')$, decide whether or not $L(G) = L(H)$.

We mention that the membership problem and the finiteness problem have some biological relevance. Let us assume that we have a 0L system G which we want use as a model for the development of some filamentous organism or alga etc. Usually such a model is obtained by an analysis of the first step of the biological object. Now the membership problem is the question whether or not a later stage of the development can be got by the model G. The finiteness problem has an negative answer if and only if the development does not be finished after a certain time and at least one branch of the development produces larger and larger plants. By such an interpretation, there is an interest from a biological point of view in these questions. The equivalence problem is the test whether or not two given models describe the same development.

However, one has to note that in biology one is more interested in the sequences of the stages of the development instead of the set of all stages. In this lecture we shall only consider the language theoretic part, for a discussion of decidability for sequences instead of languages we refer to [13] and [27].

First we note that the emptiness problem is not of interest since any 0L system generates a non-empty language because the start word is in the language. Thus the answer to the emptiness problem is "no", and this answer can be given immediately, i.e., it can be given in constant time.

By Theorem 2.13, $\mathcal{L}(0L)$ is contained in $\mathcal{L}(CS)$. If we consider the proof we see that, for a given 0L system G, we can construct a context-sensitive grammar H with $L(G) = L(H)$. It is known that it is decidable, whether or not a given word w belongs to the language generated by a given context-sensitive grammar. Thus the membership for 0L systems is decidable, too. However, all known algorithms for the membership problem for context-sensitive grammar have exponential time complexity.

By Lemma 2.12, there is a natural algorithm. We construct all derivations which only contain words of length $C_G|w|$. If w is obtained the answer to the membership is "yes", otherwise $w \notin L(G)$ holds. However, this algorithm is also exponential in time because we have to consider exponentially many derivation steps.

In [19] it has been shown that the Cocke-Younger-Kasami algorithm known for context-free grammars can be translated to interactionless 0L systems. However, by the parallelism

in the derivation in 0L system we get the complexity $O(|w|^4)$ for a fixed 0L system (in the case of context-free grammars, the complexity is $O(|w|^3)$).

The following theorem gives this statement in a weaker version (we do not mention the degree of the polynomial). We omit the (long) proof.

Theorem 2.21 *For 0L systems, the membership problem is decidable in polynomial time.* □

Theorem 2.22 *For 0L systems, the finiteness problem is decidable in polynomial time.*

Proof. Let $G = (V, P, \omega)$ be a 0L system. Let $n = \#(V)$. For any letter $a \in V$, we set $G_a = (V, P, a)$.

We call a letter $a \in V$ *surviving* in G iff $L_i(G_a)$ is non-empty for any $m \geq 0$. Let $V_s(G)$ be the set of all surviving letters of G. (Note that $V = V_s(G)$ for a propagating 0L system.)

We construct the directed graph $H_G = (V, E)$ where the set of vertices coincides with the set of all letters of V and $(a, b) \in E$ if and only if there is a production $a \to x_1 b x_2 \in P$ with $x_1, x_2 \in V^*$.

Claim 1: a is surviving if and only if there is an infinite path in H_G which starts in a.

Let H_G contain an infinite path starting in a. Let b be the letter which is obtained by the beginning of length i of the this path. Then, by the definition of H_G, there is a word in $L_i(G)$ which contains b. This shows the non-emptiness of $L_m(G_a)$ for any $i \geq 0$.

Conversely, let us assume that a is surviving. Then $L_{n+2}(G_a)$ is nonempty. Thus $L_{n+1}(G_a)$ contains a non-empty word w. Let b be a letter of w. Then there is a derivation

$$a = a_0 \Longrightarrow u_1 a_1 v_1 \Longrightarrow u_2 a_2 v_2 \Longrightarrow \ldots \Longrightarrow u_{n+1} a_{n+1} v_{n+1} = w$$

such that $a_{n+1} = b$ and $(a_i, a_{i+1}) \in E$ for $0 \leq i \leq n$. Clearly, there are integers i and j, $0 \leq i < j \leq n+1$, such that $a_i = a_j$. Therefore the path

$$a_0 \to a_1 \to a_2 \to \ldots \to a_{n+1}$$

contains a cycle and can be continued to an infinite path.

Now we interpret H_G as the graph of a (nondeterministic) finite automaton where all edges are labelled by Z, the start state is a and all states are accepting states. Then the existence of an infinite path starting from a is equivalent to the infinity of the (regular) language accepted by this automaton. By Claim 1, there is an polynomial algorithm which decides whether or not $a \in V_s(G)$ holds. Thus we can algorithmically construct the set $V_s(G)$ in polynomial time.

Now we construct the directed graph $H'_G = (V_s(G), E')$ where E' is the restriction of E to $V_s(G) \times V_s(G)$. Further, we define a labelling of the edges of E' by the letters X and Y. We label $(a, b) \in E'$ by X if and only if there is a production $a \to x_1 b x_2 \in P$ where $x_1 x_2$ contains a letter of $V_s(G)$. Otherwise, we label (a, b) by Y.

Claim 2: $L(G)$ is infinite if and only there is an infinite path in H'_G starting in a letter occurring in ω and containing an infinite number of occurrences of edges labelled by X.

Assume that there is an infinite path $a_0 \to a_1 \to a_2 \to \dots$ starting from $a = a_0$ which occurs in ω. Let $j \geq 0$. Then a_j occurs in some word of w_j of $L_j(G)$. Let $w_j = x_0 b_1 x_1 b_2 x_2 \dots b_r x_r$ for some $r \geq 1$, some words $x_k \in (V \setminus V_s(G))^*$, $0 \leq k \leq r$ and some letters $b_l \in V_s(G)$, $1 \leq l \leq r$. Let $a_j = b_s$. Then we consider a derivation $w_j \Longrightarrow w_{j+1} = x'_0 y_1 x'_1 y_2 x'_2 \dots y_r x'_r$, where any subword y_l, $1 \leq l \leq r$ contains at least one letter of $V_s(G)$ (such rules exist for letters of $V_s(G)$ by definition) and a rule $a_j \to x_1 a_{j+1} x_2$ to $b_s = a_j$. If (a_j, a_{j+1}) is labelled by X, then $y_s = x_1 a_{j+1} x_2$ contains at least two occurrences of surviving letters. Therefore w_{j+1} contains at least $r+1$ occurrences of surviving letters. Continuing in this way we obtain words in $L(G)$ with an arbitrarily large number of occurrences of letters of $V_s(G)$. Thus $L(G)$ has to be infinite.

Conversely, let us assume that $L(G)$ is infinite. Then, for any number $i \geq 1$, $L(G)$ contains a word of length i. We can improve this statement to the following one: For any number $i \geq 1$, $L(G)$ contains a word with at least i occurrences of letters of $V_s(G)$. If we assume the contrary, then there is a number j such that any word of $L(G)$ contains at most j letters of $V_s(G)$. Then any word of $L(G)$ can be written as $W = x_0 b_1 x_1 b_2 x_2 \dots b_r x_r$ for some $r \geq 1$, some words $x_k \in (V \setminus V_s(G)^*$, $0 \leq k \leq r$ and some letters $b_l \in V_s(G)$, $1 \leq l \leq r$, where $r \leq j$. Since there is a number t such that $L_t(G_a) = \emptyset$ for all $a \notin V_s(G)$, we have $w \Longrightarrow_G^t z_1 z_2 \dots z_r$ where $b_i \Longrightarrow_G^t z_i$ for $1 \leq i \leq r$. If $K = \max\{w_a \mid a \to w_a \in P, a \in V$, then $|z_1 z_2 \dots z_r| \leq r K^t \leq j K^t$. This yields a bound for the length of the words in $L(G)$ in contrast to the infinity of $L(G)$.

Now let p be a sufficient large number, and let w be a word of $L(G)$ containing at least p occurrences of letters from $V_s(G)$. Then we have have a derivation

$$\omega = u_0 a_0 v_0 \Longrightarrow u_1 a_1 v_1 \Longrightarrow u_2 a_2 v_2 \Longrightarrow \dots \Longrightarrow u_m a_m v_m = w$$

such that $(a_i, a_{i+1}) \in E$ for $0 \leq i \leq m$ and the path

$$a_0 \to a_1 \to a_2 \to \dots \to a_m$$

contains at least $n+1$ edges labelled by X. For $1 \leq q \leq n+1$, let

$$a_{i_q} \to a_{i_q+1} \to a_{i_q+2} \to \dots a_{i_{q+1}} \to a_{i_{q+1}+1}$$

be the subpath where (a_{i_q}, a_{i_q+1}) and $(a_{i_{q+1}}, a_{i_{q+1}+1})$ are labelled by X and its remaining edges are labelled by Y. Then we set

$$M_q = \{a_{i_q+1}, a_{i_q+2}, \dots a_{i_{q+1}-1}\}$$

for $1 \leq q \leq n+1$. Obviously, there are g and h, $1 \leq g < h \leq n+1$, such that $M_g = M_h$. Again, this implies the existence of a cycle containing an edge labelled by X. Therefore the infinite path with the required property exists.

Again, we interpret H'_G as a nondeterministic finite automaton taking the above labelling of the edges. The existence of a path having the properties mentioned in Claim 2 is equivalent to the infinity of the language accepted by the automaton after application of the homomorphisms which maps X to X and Y to the empty word. Thus we can decide the infinity and therefore the finiteness of $L(G)$. □

Theorem 2.23 *i) For (P)0L systems, the equivalence problem is undecidable.*
ii) For (P)D0L systems, the equivalence problem is decidable.

Proof. We only prove i). The proof for ii) is omitted because it is long if it is based on the elementary knowledge and formal language theory or it is based on deep results of (mathematical) group theory.

We prove i) by reduction to the Post Correspondence Problem. Let

$$U = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$$

be a set of pairs of words with $u_i, v_i \in \{a, b\}$ for $1 \leq i \leq n$.

We consider the 0L systems

$$G_1 = (V, P, S) \quad \text{and} \quad G_2 = (V, P', S)$$

with

$$\begin{aligned}
V &= \{S, S', S'', S_u, S_r, S_l, S', a, b, c\}, \\
P &= \{S \to S', S \to S'', S_u \to c, S_l \to c, S_r \to c\} \\
&\quad \cup \bigcup_{x \in \{a,b\}} \{S' \to xS'x, S' \to xS_l, S' \to S_r x, S_l \to xS_l, S_r \to S_r x, S_u \to S_r x, S_u \to xS_l\} \\
&\quad \cup \{S' \to xS_u y : x, y \in \{a, b\}, x \neq y\} \cup \{S_u \to xS_u y : x, y \in \{a, b\}\} \\
&\quad \cup \bigcup_{i=1}^{n} \{S'' \to u_i S'' v_i^R\}, \\
P' &= P \cup \bigcup_{i=1}^{n} \{S'' \to u_i c v_i^R\}.
\end{aligned}$$

It is easy to show that

$$\begin{aligned}
L(G_1) &= \{S, S', S''\} \cup \{\alpha S' \alpha^R : \alpha \in \{a, b\}^+\} \\
&\quad \cup \{\alpha S_u \beta^R : \alpha, \beta \in \{a, b\}^+,\ |\alpha| = |\beta|,\ \alpha \neq \beta\} \\
&\quad \cup \{\alpha S_r \beta^R : \alpha, \beta \in \{a, b\}^+,\ |\alpha| < |\beta|\} \\
&\quad \cup \{\alpha S_l \beta^R : \alpha, \beta \in \{a, b\}^+,\ |\alpha| > |\beta|\} \\
&\quad \cup \{\alpha c \beta^R : \alpha, \beta \in \{a, b\}^+,\ \alpha \neq \beta\} \\
&\quad \cup \{u_{i_1} u_{i_2} \ldots u_{i_k} S'' v_{i_k} v_{i_{k-1}} \ldots v_{i_1} : k \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq k\}
\end{aligned}$$

and

$$L(G_2) = L(G_1) \cup \{u_{i_1} u_{i_2} \ldots u_{i_k} c v_{i_k} v_{i_{k-1}} \ldots v_{i_1} : k \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq k\}.$$

Obviously, $L(G_1) \subseteq L(G_2)$, and $L(G_1) = L(G_2)$ holds if and only the part added to $L(G_1)$ to obtain $L(G_2)$ is contained in $\{\alpha c \beta^R : \alpha, \beta \in \{a, b\}^+,\ \alpha \neq \beta\}$. Thus we get $L(G_1) = L(G_2)$ iff the Post Correspondence Problem has no solution. □

Note that Theorem 2.23 implies that the equivalence problem for PD0L systems is decidable and that the equivalence problem for 0L systems is undecidable.

Since the transformation of an 0L system G into a context-free grammar G' such that $L_A(G) = L(G')$ is constructive, we get the following results from the decidability results for context-free languages. The membership in the adult language and the emptiness and the finiteness of the adult languages of a 0L systems are decidable, whereas it is undecidable whether two P0L systems generate the same adult language. The equivalence of two D0L system with respect to adult languages is decidable, because we can first check whether both system generate a non-empty adult language (consisting of one word) and then we determine the adult languages and compare them.

2.1.6 Growth functions

A very important field in the study of the development of filamentous organisms and plants is the growth of the organism or plant. Usually, as a measure of the size of the plant one takes the number of cells which build it and the growth is measured by a function which associates with a given time moment the size of the plant at this moment.

We now formalize this concept. In order to get a function one has to ensure that at every moment only one organism exists, i.e. the Lindenmayer system has to generate exactly one word. Therefore we have to restrict to deterministic Lindenmayer systems. For a D0L system $G = (V, P, \omega)$, we have a uniquely determined derivation

$$\omega = w_0 \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \ldots \Longrightarrow w_m \Longrightarrow \ldots , \tag{2.8}$$

and thus$L_m(G)$ contains exactly one element w_m. Conversely, let H be a 0L system which, for some letter $a \in V$, has two rules $a \to w_1$ and $a \to w_2$ with $w_1 \neq w_2$ in its set of productions, and let a occur in some word w of $L(H)$ (otherwise we can omit a and its rules). Then we can generate two words from w because we have the derivations $w = x_1 a x_2 \Longrightarrow x_1' w_1 x_2'$ and $w = x_1 a x_2 \Longrightarrow x_1' w_2 x_2'$. Hence $L_m(H)$ for some $m \geq 1$ contains at least two words.

Definition 2.24 *The growth function* $f_G : \mathbf{N} \to \mathbf{N}$ *of a deterministic 0L system* G *is defined by*

$$f_G(m) = |w_m| .$$

Example 2.25 We consider the deterministic 0L systems

$$\begin{aligned} G_1 &= (\{a\}, \{a \to a^2\}, a) , \\ G_2 &= (\{a, b\}, \{a \to \lambda, b \to ab\}, aab) , \\ G_4 &= (\{a, b, c, d, e\}, \{a \to a, b \to ba, c \to cbb, d \to da, e \to cbbd\}, e) \end{aligned}$$

given in the Examples 2.4, 2.5 and 2.7.

In G_1, the only derivation is

$$a \Longrightarrow a^2 \Longrightarrow a^4 \Longrightarrow a^8 \Longrightarrow \ldots ,$$

which results in

$$f_{G_1}(m) = 2^m \quad \text{for} \quad m \geq 0 .$$

In G_2, we only have the derivation

$$aab \Longrightarrow \lambda\lambda ab = ab \Longrightarrow \lambda ab = ab \Longrightarrow ab \Longrightarrow ab \Longrightarrow \dots ,$$

which gives

$$f_{G_2}(0) = 3 \quad \text{and} \quad f_{G_2}(m) = 2 \text{ for } m \geq 1 .$$

In Example 2.7, we have shown that

$$\begin{aligned} L_0(G_4) &= \{e\} , \\ L_m(G_4) &= \{cbb(ba)^2(ba^2)^2 \dots (ba^{m-1})^2 da^{m-1}\} \text{ for } m \geq 1 . \end{aligned}$$

Thus we get $f_{G_4}(0) = |e| = 1$ and, for $m \geq 1$,

$$\begin{aligned} f_{G_4}(m) &= |cbb(ba)^2(ba^2)^2 \dots (ba^{m-1})^2 da^{m-1}| \\ &= 1 + 2 \cdot 1 + 2 \cdot 2 + 2 \cdot 3 + \dots + 2 \cdot m + 1 + (m-1) \\ &= m + 1 + 2 \cdot \sum_{i=1}^{m} i = m + 1 + 2 \cdot \frac{m(m+1)}{2} = m^2 + 2m + 1 \\ &= (m+1)^2 . \end{aligned}$$

This gives

$$f_{G_4}(m) = (m+1)^2 \quad \text{for} \quad m \geq 0 .$$

In the examples we have determined the growth function by a determination of the sequence of words generated by the system and obtained $f_G(m)$ as the length of $|w_m|$ according to the definition. However, in biology one is interested in a computation of $f_G(m)$ for arbitrary m, especially for large m, without a determination of w_m. Such a computation can easily be done if one has a formula for f_G. In the sequel we shall present some such formulae.

In the sequel we shall assume that the D0L system under consideration is given as

$$G = (\{a_1, a_2, \dots, a_n\}, \{a_1 \to v_1, a_2 \to v_2, \dots, a_n \to v_n\}, \omega) \tag{2.9}$$

and that its only derivation is given by (2.8).

Definition 2.26 *Let G be a D0L system as in (2.9). Then we define the growth matrix M_G of G as the (n,n)-matrix*

$$M_G = (a_{i,j}) = (\#_{a_j}(v_i)) .$$

Example 2.27 Again, we consider the systems G_1, G_2 and G_4. We get

$$M_{G_1} = (2), \; M_{G_2} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \; M_{G_4} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 \end{pmatrix}$$

Theorem 2.28 *Let G be a D0L system as in (2.9), and let M_G be its growth matrix. Then, for $m \geq 0$,*

$$f_G(m) = \Psi(\omega)(M_G)^m(1,1,\ldots,1)^T\,.$$

Proof. First we note that

$$\begin{aligned}\Psi(w_m)\cdot(1,1,\ldots,1)^T &= (\#_{a_1}(w_m),\#_{a_2}(w_m),\ldots,\#_{a_n}(w_m))(1,1,\ldots,1)^T\\ &= \sum_{i=1}^{n}\#_{a_i}(w_m) = |w_m|\\ &= f_G(m)\,.\end{aligned}$$

Thus it is sufficient to to prove that, for $m \geq 1$,

$$\Psi(w_m) = \Psi(\omega)M_G^m\,.$$

This will be done by induction on m.

$m = 0$. We have

$$\Psi(w_0) = \Psi(\omega) = \Psi(\omega)\cdot E = \Psi(\omega)M_G^0\,,$$

where E is the unit matrix. Thus the induction basis is shown.

$m > 0$. By induction hypothesis,

$$\Psi(\omega)M_G^m = \Psi(\omega)M_G^{m-1}M_G = \Psi(w_{m-1})M_G \tag{2.10}$$

Further, any occurrence of a letter a_i, $1 \leq i \leq n$, in w_{m-1} contributes $\#_{a_j}(v_i)$ occurrences of a_j, $1 \leq j \leq n$, in w_m. Thus

$$\#_{a_j}(w_m) = \sum_{i=1}^{n}\#_{a_i}(w_{m-1})\#_{a_j}(v_i)\,.$$

This implies

$$\Psi(w_m) = \Psi(w_{m-1})M_G\,.$$

Together with (2.10) we get

$$\Psi(w_m) = \Psi(\omega)M_G^m\,.$$

□

Let

$$\chi_{M_G}(x) = det(M_G - xE) = a_nx^n + a_{n-1}x^{n-1} + \ldots + a_1x + a_0$$

be the characteristic function of M_G. By the Cayley-Hamilton Theorem (see Theorem 1.1),

$$O = \chi_{M_G}(M_G) = a_nM_G^n + a_{n-1}M_G^{n-1} + \ldots + a_1M_G + a_0E\,.$$

For $k \geq 0$, by left and right multiplication with $\Psi(\omega)M_G^k$ and $(1,1,\ldots 1)^T$, respectively, and Theorem 2.28, we obtain

$$\begin{aligned}0 &= a_n\Psi(\omega)M_G^{k+n}(1,1,\ldots,1)^T + a_{n-1}\Psi(\omega)M_G^{k+n-1}(1,1,\ldots,1)^T + \ldots\\ &\qquad a_1\Psi(\omega)M_G^{k+1}(1,1,\ldots,1)^T + a_0\Psi(\omega)M_G^k(1,1,\ldots,1)^T\\ &= a_nf_G(k+n) + a_{n-1}f_G(k+n-1) + \ldots + a_1f_G(k+1) + a_0f_G(k)\,.\end{aligned}$$

Thus the growth function f_G satisfies the difference equation

$$0 = a_n h(k+n) + a_{n-1} h(k+n-1) + \ldots + a_1 h(k+1) + a_0 h(k)\,.$$

Using the theory of difference equation (see Chapter 1) we get

$$h(x) = \sum_{i=1}^{s} (\beta_{i,0} + \beta_{i,1} x + \beta_{i,2} x^2 + \ldots \beta_{i,t_i-1} x^{t_i-1}) \mu_i^x$$

where, for $1 \leq i \leq s$, μ_i is a root of multiplicity t_i of

$$g(y) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0\,,$$

$\sum_{i=1}^{s} t_i = n$ holds and $\beta_{i,j}$, $1 \leq i \leq s$, $0 \leq j \leq t_i - 1$ are n real constants, which are uniquely determined by the values $h(0), h(1), \ldots, h(n-1)$.

If we take into consideration that $\chi_{M_G} = g$ holds, then the roots β_i, $1 \leq i \leq s$ are the eigenvalues of M_G. Thus we obtain the following theorem.

Theorem 2.29 *Let G be a D0L system as in (2.9), and let M_G be its growth matrix. For $1 \leq i \leq s$, let μ_i be a eigenvalue of M_G of multiplicity t_i such that $\sum_{i=1}^{s} t_i = n$. Then*

$$f_G(m) = \sum_{i=1}^{s} (\beta_{i,0} + \beta_{i,1} m + \beta_{i,2} m^2 + \ldots \beta_{i,t_i-1} m^{t_i-1}) \mu_i^m$$

for certain constants $\beta_{i,j}$, $1 \leq i \leq s$, $0 \leq j \leq t_i - 1$. □

Example 2.30 We apply the theory developed up to this point to the our D0L systems G_1, G_2 and G_4. In order to simplify the notation, we shall sometimes only use the indexes 1, 2 and 4 to refer to G_1, G_2 and G_4, respectively.

Then we get

$$\chi_1(x) = det(M_{G_1} - xE) = det(2 - x) = 2 - x\,.$$

The only eigenvalue is $\mu_1 = 2$ of multiplicity 1. Thus we get $f_{G_1}(m) = \beta_0 2^m$. Since $f_{G_1}(0) = 1 = \beta_0 2^0 = \beta_0$ we obtain $\beta_0 = 1$ which yields $f_{G_1}(m) = 2^m$ for $m \geq 0$.

Considering G_2 we have

$$\chi_2(x) = det(M_{G_2} - xE) = det \begin{pmatrix} -x & 0 \\ 1 & 1-x \end{pmatrix} = -x(1-x)\,.$$

Therefore the eigenvalues of G_2 are $\mu_1 = 0$ and $\mu_2 = 1$ which both are of multiplicity 1. Thus

$$f_{G_2}(m) = \beta_{1,0} 0^m + \beta_{2,0} 1^m\,.$$

In the sequel we shall assume that $0^0 = 1$ (note that 0^0 is an indefinite expression whose value depends on the context). Then we have

$$\begin{aligned} \beta_{1,0} \quad +\beta_{2,0} &= 3 = f_{G_2}(0) \\ \beta_{2,0} &= 2 = f_{G_2}(1). \end{aligned}$$

The solutions of this system of linear equations are $\beta_{1,0} = 1$ and $\beta_{2,0} = 2$. Consequently,

$$f_{G_2}(0) = 3 \quad \text{and} \quad f_{G_2}(m) = 2 \text{ for } m \geq 1\,.$$

For G_4, we get

$$\chi_4(x) = det(M_{G_4} - xE) = det \begin{pmatrix} 1-x & 0 & 0 & 0 & 0 \\ 1 & 1-x & 0 & 0 & 0 \\ 0 & 2 & 1-x & 0 & 0 \\ 1 & 0 & 0 & 1-x & 0 \\ 0 & 2 & 1 & 1 & -x \end{pmatrix} = (1-x)^4(-x)\,.$$

Thus the eigenvalues of M_{G_4} are $\mu_1 = 1$ of multiplicity 4 and $\mu_2 = 0$ of multiplicity 1. Therefore we get

$$f_{G_4}(m) = \beta_{1,0}0^m + (\beta_{2,0} + \beta_{2,1}m + \beta_{2,2}m^2 + \beta_{2,3}m^3)1^m\,.$$

The constants $\beta_{1,0},\ \beta_{2,0},\ \beta_{2,1},\ \beta_{2,2},\ \beta_{2,3}$ can be determined as the solutions of the following of linear equations

$$\begin{array}{rrrrrr} \beta_{1,0} & +\beta_{2,0} & & & = & 1 = f_{G_4}(0) \\ & \beta_{2,0} & +\beta_{2,1} & +\beta_{2,2} \quad +\beta_{2,3} & = & 4 = f_{G_4}(1) \\ & \beta_{2,0} & +2\beta_{2,1} & +4\beta_{2,2} \quad +8\beta_{2,3} & = & 9 = f_{G_4}(2) \\ & \beta_{2,0} & +3\beta_{2,1} & +9\beta_{2,2} \quad +27\beta_{2,3} & = & 16 = f_{G_4}(3) \\ & \beta_{2,0} & +4\beta_{2,1} & +16\beta_{2,2} \quad +64\beta_{2,3} & = & 25 = f_{G_4}(4)\,. \end{array}$$

We obtain

$$\beta_{1,0} = 0,\ \beta_{2,0} = 1,\ \beta_{2,1} = 2,\ \beta_{2,2} = 1,\ \beta_{2,3} = 0\,,$$

which leads $f_{G_4}(m) = 1 + 2m + m^2 = (m+1)^2$ for $m \geq 1$.

We mention that the formulae given in Theorems 2.28 and 2.29 have some nice and some bad features. One formula uses the Parikh vector of ω and growth matrix, which both can directly be obtained from the given system, however, the computation of $f_G(m)$ requires the calculation of the m-th power of a matrix. By the other formula, it is easy to compute the value of the growth function for an arbitrarily given argument, however, we note that it is hard to compute the eigenvalues of a growth matrix since they can be complex and the degree of the characteristic function will be arbitrarily large for arbitrarily large alphabets.

Theorem 2.31 *Let G be a D0L system. Then the growth f_G satisfies one of the following conditions:*
a) there is a constant c such that $f_G(m) \leq c$ for all m
b) there are constants c_1, c_2 and p such that $c_1 m^p \leq f_G(m) \leq c_2 m^p$ for large m,
c) there are constants c_1 and c_2 such that $c_1^m \leq f_G(m) \leq c_2^m$ for large m.

Proof. We only give the proof for the case that the eigenvalues of M_G are non-negative real numbers, for a general proof (including complex roots) one has to consider the absolute values $|\mu|$ of the eigenvalues μ of M_G.

If G generates a finite language, then case a) holds obviously.

Let us now assume that G generates an infinite language. Let μ be the largest eigenvalue of M_G. Then

$$f_G(m) = (\beta_0 + \beta_1 m + \beta_2 m^2 + \ldots + \beta_{t-1} m^{t-1})\mu^m + R(m)$$

where t is the multiplicity of μ and $R(m)$ is asymptotically smaller than $(\beta_0 + \beta_1 m + \ldots + \beta_{t-1} m^{t-1})\mu^m$.

If $\mu > 1$ holds, then f_G is asymptotically equal to $(\beta_0 + \beta_1 m + \ldots + \beta_{t-1} m^{t-1})\mu^m$. We choose d such that

$$\beta_0 + \beta_1 m + \beta_2 m^2 + \ldots + \beta_{t-1} m^{t-1} \leq d^m$$

for large m. Then we get

$$f_G(m) \leq d^m \mu^m = (d\mu)^m = c_2^m$$

for large m. On the other hand $\beta_0 + \beta_1 m + \ldots + \beta_{t-1} m^{t-1} > 0$ for large m. Therefore, we also have

$$\mu^m = c_1^m \leq f_G(m)$$

for large m. Thus we have case c).

If $\mu = 1$ holds, then f_G is asymptotically equal to $\beta_0 + \beta_1 m + \beta_2 m^2 + \ldots + \beta_{t-1} m^{t-1}$. Then it is easy to see that case b) holds if $t \geq 2$ and that a) holds if $t = 1$.

If $\mu < 1$, then $(\beta_0 + \beta_1 m + \beta_2 m^2 + \ldots + \beta_{t-1} m^{t-1})\mu^m$ tends to zero and the same holds for $R(m)$. Since the range of f_G is the set of non-negative integers, this situation cannot occur (since we assume that the generated language is infinite). □

Theorem 2.31 says that we have only three different types of growth functions of D0L systems. Thus there is no DOL system G such that $f_G(m) = \log(m)$ holds. If we have seen that the real growth of a plant is logarithmic, then we cannot take a D0L system to model the development.

2.2 Lindenmayer systems with interaction

2.2.1 Definitions and examples

It is a well-known fact that in reality other growth function also occur, for example there are organisms with logarithmic growth. The development of such an organism cannot be modelled by D0L systems.

In order to obtain more powerful systems one can take into consideration the context of a cell, i.e., the rules for the development of a cell does not only depend on the cell itself, it also depends on neighbouring cells. This reflects the biological situation much better than the case without interaction considered in the preceding section.

Again, we model the cells by elements of an alphabet and the organisms by words. However, we assume that the development of a cell in an organism depends on its k left

Bibliography

[1] L. M. Adleman, Molecular computation of solutions to combinatorial problems. *Science* **226** (1994) 1021-1024.

[2] C. Calude and Gh. Păun, *Computing with Cells and Atoms*. Taylor and Francis, London, 2001.

[3] J.W. Carlyle, S. Greibach and A. Paz, A two-dimensional generating system modelling growth and binary cell division. In: *Proc. 15th Annual Symp. Switching Automata Theory*, 1–12, 1974.

[4] E. Csuhaj-Varju, J. Kelemen, A. Kelemenova and Gh. Păun, Eco(grammar)systems - a grammatical framework for lifelike interaction. *Artificial Life* **24** (1997) 1–28.

[5] J. Dassow, Grammars with valuations - a discrete model for self-organization of biopolymers. *Discr. Appl. Math.* **4** (1982) 161–174.

[6] J. Dassow, *Theoretische Informatik*, Vorlesungsskript, Otto-von-Geuricke-Universität Magdeburg, 2001.

[7] J. Dassow, V. Mitrana and A. Salomaa, Operations and language generating devices suggested by genome evolution. *Theor. Comp. Sci.* **270** (2002) 701–738.

[8] J. Dassow and Gh. Păun, *Regulated Rerwriting in Formal Language Theory*. Springer-Verlag, Berlin, 1989 and Akademie-Verlag, Berlin, 1989.

[9] J. Dassow and Gy. Vaszil, Multiset splicing systems. *BioSystems* **74** (2004) 1–7.

[10] R. Freund, L. Kari, M. Oswald and P. Sosik, Computationally universal P systems without priorities: two catalysts are sufficient. *Theor. Comp. Sci.* **330** (2005) 251–266.

[11] T. Head, Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. *Bull. Math. Biology* **49** (1987) 737–759.

[12] T. Head, Gh. Păun and D. Pixton, Language theory and molecular genetics. In: [30], Vol. II, Chapter 7, 295–360

[13] G.T. Herman and G. Rozenberg, *Developmental Systems and Languages*. North-Holland Publ. Co., Amsterdam, 1975.

[14] D. Janssens, G. Rozenberg and R. Verraedt, On sequential and parallel node-rewriting graph grammars. Part I, *Computer Graphics and Image Processing* **18** (1982) 279–304 and Part II, *Computer Vision, graphics and Image Processing* **23** (1983) 295–312.

[15] L. Kari, G. Rozenberg and A. Salomaa, L systems. In: [30], Vol. I, Chapter 5, 253–328.

[16] A. Lindenmayer, Mathematical models for cellular interaction in development I and II. *J. Theoret. Biol.* **18** (1968) 280–315.

[17] A. Lindenmayer and G. Rozenberg (eds.), *Automata, Languages, Development.* North-Holland Publ. Co., 1976.

[18] R. J. Lipton, Using DNA to solve NP-complete problems. *Science* **268** (1995) 542–545.

[19] J. Opatrny and K. Culik II, Time complexity of recognition and parsing of E0L languages. In [17], 243–250.

[20] A. Păun and Gh. Păun, The power of communication: P systems with symport and antiport. *New Generation Computing* **20** (2002) 295–306.

[21] Gh. Păun, Computing with membranes. *J. Comp. System Sci.* **61** (2000) 108–143.

[22] Gh. Păun, *Membrane Computing.* Springer-Verlag, Berlin, 2002.

[23] Gh. Păun, G. Rozenberg and A.Salomaa, *DNA Computing - New Computing Paradigms.* Springer-Verlag, Berlin, 1998.

[24] Gh. Păun and A.Salomaa, DNA computing based on the splicing operation. *Mathematica Japonica* **43** (1996) 607–632.

[25] P. Pruzinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants.* Springer-Verlag, Berlin, 1990.

[26] G. Rozenberg and A.Salomaa (eds.), *L Systems.* LNCS 15, Springer-Verlag, Berlin, 1974.

[27] G. Rozenberg and A.Salomaa, *The Mathematical Theory of L Systems.* Academic Press, New York, 1980.

[28] G. Rozenberg and A.Salomaa (eds.), *The Book of L.* Springer-Verlag, Berlin, 1985.

[29] G. Rozenberg and A.Salomaa (eds.), *Lindenmayer Systems.* Springer-Verlag, Berlin, 1992.

[30] G. Rozenberg and A.Salomaa (eds.), *Handbook of Formal Languages.* Vol I – III, Springer-Verlag, 1997.

[31] A. Salomaa, *Formal Languages*. Academic Press, New York, 1973 (german edition, Springer-Verlag, Berlin, 1978).

[32] A. Salomaa, *Jewels of Formal Language Theory*. Computer Science Press, Rockville, 1981.