L systems

Lila Kari
Department of Mathematics, University of Western Ontario
London, Ontario, N6A 5B7 Canada

Grzegorz Rozenberg
Department of Computer Science, University of Leiden
P.O.Box 9512, NL-2300 RA Leiden, The Netherlands

Arto Salomaa
Department of Mathematics, University of Turku
20 500 Turku, Finland

1 Introduction

1.1 Parallel rewriting

L systems are *parallel* rewriting systems which were originally introduced in 1968 to model the development of multicellular organisms, [L1]. The basic ideas gave rise to an abundance of language-theoretic problems, both mathematically challenging and interesting from the point of view of diverse applications. After an exceptionally vigorous initial research period (roughly up to 1975; in the book [RSed2], published in 1985, the period up to 1975 is referred to, [RS2], as “when L was young”), some of the resulting language families, notably the families of D0L, 0L, DT0L, E0L and ET0L languages, had emerged as fundamental ones in the *parallel or L hierarchy.* Indeed, nowadays the fundamental L families constitute a similar testing ground as the Chomsky hierarchy when new devices (grammars, automata, etc.) and new phenomena are investigated in language theory.

L systems were introduced by Aristid Lindenmayer in 1968, [L1]. The original purpose was to model the development of simple filamentous organisms. The development happens in parallel everywhere in the organism. Therefore, *parallelism* is a built-in characteristic of L-systems. This means, from the point of view of rewriting, that everything has to be rewritten at each step of the rewriting process. This is to be contrasted to the “sequential” rewriting of phrase structure grammars: only a specific part of the word under scan is rewritten at each step. Of course, the effect of parallelism can be reached by several sequential steps in succession. However, the *synchronizing control mechanism* is missing from the customary sequential rewriting devices. Therefore, parallelism cannot be truly simulated by sequential rewriting.

Assume that your only rule is $a \longrightarrow a^2$ and you start with the word a^3. What do you get? If rewriting is sequential, you can replace one a at a time by a^2, obtaining eventually all words words a^i, $i \geq 3$. If rewriting is parallel, the word a^6 results in one step. It is not possible to obtain a^4 or a^5 from a^3. Altogether you get only the words $a^{3 \cdot 2^i}$, $i \geq 0$. Clearly, the language

$$\{a^{3 \cdot 2^i} | \ i \geq 0\}$$

is not obtainable by sequential context-free or interactionless rewriting: letters are rewritten independently of their neighbours. Observe also how the dummy rule $a \longrightarrow a$ behaves dramatically differently in parallel and sequential rewriting. In the latter it has no influence and can be omitted without losing anything. In parallel rewriting the rule $a \longrightarrow a$ makes the simulation of sequential rewriting possible: the "real" rule $a \longrightarrow a^2$ is applied to one occurrence of a, and the "dummy" rule $a \longrightarrow a$ to the remaining occurrences. Consequently, all words a^i, $i \geq 3$, are obtained by parallel rewriting from a^3 if both of the rules $a \longrightarrow a^2$ and $a \longrightarrow a$ are available.

The present survey on L systems can by no means be encyclopedic; we do not even claim that we can exhaust all the main trends. Of the huge bibliography concerning L systems we reference only items needed to illustrate or augment an isue in our presentation. We are fully aware that also some rather influential papers have not been referenced. In the early years of L systems it was customary to emphasize that the exponential function 2^n described the yearly growth in the number of papers in the area. [MRS], published in 1981, was the latest edition of a bibliography on L systems intended to be comprehensive. After that nobody has undertaken the task of compiling a comprehensive bibliography which by now would contain at least 5 000 items.

On the other hand, L systems will be discussed also elsewhere in this Handbook. They form the basic subject matter in the chapter of P.Prusinkiewicz in Volume III but are also covered, for instance, in the chapters by W.Kuich G.Paun – A.Salomaa in the present Volume I.

When we write "L systems" or "ET0L languages" without a hyphen, it is not intentionally vicious ortography. We only follow the practice developed during the years among the researchers in this field. The main reason for this practice was that in complicated contexts the hyphen was often misleading. We want to emphasize in this connection the notational uniformity followed by the researchers of L systems, quite exceptional in comparison with most areas of mathematical research. In particular, the different letters have a well-defined meaning in the names of the L language families. For instance, it is immediately understood by everyone in the field what HPDF0L languages are. We will return to the terminology in Section 2 below.

1.2 Callithamnion roseum, a primordial alga

We begin with a mathematical model of the development of a red alga, *Callithamnion roseum*. The attribute "primordial" can be associated to it because the model for its development appears in the original paper [L1] by Aristid Lindenmayer. The mathematical model is a PD0L system, according to the terminology developed later. Here 0 indicates that the interaction between individual cells in the development is *zero-sided*, that is, there is no interaction; rewriting is context-free. The letter D indicates *determinism*: there is only one possibility for each cell, that is, there is only one rule for each letter. Finally, the letter P stands for *propagating*: there is no cell death, no rule $a \longrightarrow \lambda$ indicating that a letter should be rewritten as the empty word. We will return to this terminology in Section 2 below.

Following the general plan of this Handbook, our present chapter deals exclusively with words, that is, with one-dimensional L systems. Two-dimensional systems (map L systems, graph L systems, etc.) were introduced quite early, see [RS1]. An interpretation mechanism is needed for one-dimensional systems: how to interpret words as pictures depicting stages of development? In the case of *filamentous organisms* this normally happens using *branching structures*. In the model below the matching brackets [,] indicate branches, drawn alternately on both sides of the stem.

We are now ready to present the mathematical model for Callithamnion roseum. The alphabet is $\Sigma = \{1,2,3,4,5,6,7,8,9,[,]\}$ and the rules for the letters: $1 \longrightarrow 23$, $2 \longrightarrow 2$, $3 \longrightarrow 24$, $4 \longrightarrow 25$, $5 \longrightarrow 65$, $6 \longrightarrow 7$, $7 \longrightarrow 8$, $8 \longrightarrow 9[3]$, $9 \longrightarrow 9$. Beginning with the word $w_0 = 1$, we get the following *developmental sequence*:

$$\begin{aligned}
w_0 &= 1\\
w_1 &= 23\\
w_2 &= 224\\
w_3 &= 2225\\
w_4 &= 22265\\
w_5 &= 222765\\
w_6 &= 2228765\\
w_7 &= 2229[3]8765\\
w_8 &= 2229[24]9[3]8765\\
w_9 &= 2229[225]9[24]9[3]8765\\
w_{10} &= 2229[2265]9[225]9[24]9[3]8765\\
w_{11} &= 2229[22765]9[2265]9[225]9[24]9[3]8765\\
w_{12} &= 2229[228765]9[22765]9[2265]9[225]9[24]9[3]8765\\
w_{13} &= 2229[229[3]8765]9[228765]9[22765]9[2265]9[225]9[24]9[3]8765
\end{aligned}$$

Selected developmental stages $(w_0, w_6, w_7, \ldots, w_{15})$ are shown in the following picture, [PK].

1.3 Life, real and artificial

We conclude this Introduction with some observations that, in our estimation, are rather important in predicting the future developments of L systems. L systems were originally introduced to model the development of multicellular organisms, that is, the development of some form of "real" life. However, there have been by now numerous applications in computer graphics, where L systems have been used to depict *imaginary* life forms, imaginary *gardens of L* and also non-living specimens ranging from toys and birthday cakes to real-estate ads (see, for instance, [PL]). Their utmost simplicity and flexibility to small changes tailor-made according to individual wishes make L systems very suitable to model phenomena of *artificial life.*

Artificial life is customarily understood as the study of man-made constructs that exhibit, in some sense or in some aspects, the behavior of existing living organisms. Artificial life extends the traditional biology that is concerned with the carbon-chain-type of life evolved on Earth. Artificial life tries to synthesize life-like behavior within computers and other man-made devices. It is also more extensive than *robotics.* Robots are constructed to do some specific tasks, whereas the "creatures" of artificial life are only observed. As often explained, artificial life paints the picture of *life-as-it-could-be*, contrasted to the picture of traditional biology about *life-as-we-know-it.*

It is very difficult to draw a strict border between living and nonliving, animate and inanimate. No definition of "life", satisfactory in a mathematical or common sense, has so far been given. Perhaps it is better to view the set of living beings as a fuzzy set rather than to try to define it in crisp mathematical terms. Another possibility is to give lists of properties typical for living beings as contrasted to inanimate objects. However, so far none of such lists seems satisfactory. Also many individual properties, such as growth, give rise to doubts. Although growth is typical for living beings, it can be observed elsewhere.

However, one feature very characteristic for the architecture of all living beings is that life is *fundamentally parallel.* A living system may consist of millions of parts, all having their own characteristic behavior. However, although a living system is highly distributed, it is massively parallel.

Thus, any model for artificial life must be capable of simulating parallelism – no other approach is likely to prove viable. Among all grammatical models, L systems are by their very essence the most suitable for modeling parallelism. L systems may turn out to be even more suitable for modeling artificial life than real life.

Indeed, the utmost simplicity of the basic components and the ease of affecting changes tailor-made for clarifying a specific issue render L systems ideal for modeling artificial life. A good example is the so-called French Flag Problem: does polarized behavior at the global level imply polarization (that is, nonsymmetric behavior) at the local level? The idea behind this problem (as described in [H2] and [HR]) comes from a worm with three parts (like the French Flag): head, middle, tail. If one of the ends is cut, the worm grows again the missing part, head for head and tail for tail. The behavior of the uncut part is polarized – the remaining organism knows which end it assists to grow. However, such a global behavior can be reached by a fully symmetric local behavior. It can be modeled by an L system, where the rules for the individual cells are fully symmetric – there is no distinction between right and left, [HR]. While such a construction does not in any way prove that the real-life phenomenon is locally symmetric – the cells of the worm used in experiments can very well be polarized – it certainly constitutes an important fact of artificial life. We can have living species with polarized global behavior but with fully symmetric behavior at the level of individual cells – and possibly having some other features we are interested in implementing.

Other grammatical models, or grammar-like models such as cellular automata, seem to lack the versatility and flexibility of L systems. It is not easy to affect growth of the interior parts using cellular automata, whereas L-filaments grow naturally everywhere. If you want to make a specific alteration in the species you are interested in, then you very often find a suitable L system to model the situation. On the other hand, it seems that still quite much post-editing is needed in graphical real-life modeling. The variations of L systems considered so far seem to be too simple to capture some important features of real-life phenomena. We now proceed to present the most common of these variations, beginning with the basic ones.

2 The world of L, an overview

2.1 Iterated morphisms and finite substitutions: D0L and 0L

We will now present the fundamentals of L systems and their basic properties. Only very few notions of language theory are needed for this purpose. We follow Section 2.1 of Chapter 1 in this Handbook as regards the core terminology about letters, words and languages. The most important language-theoretic notion needed below will be a *finite substitution* over an alphabet Σ, as well as its special cases.

Definition. A *finite substitution* σ over an alphabet Σ is a mapping of Σ^* into the set of all finite nonempty languages (possibly over an alphabet Δ different from Σ) defined as follows. For each letter $a \in \Sigma$, $\sigma(a)$ is a finite nonempty language, $\sigma(\lambda) = \lambda$ and, for all words $w_1, w_2 \in \Sigma^*$,

$$\sigma(w_1 w_2) = \sigma(w_1)\sigma(w_2).$$

If none of the languages $\sigma(a)$, $a \in \Sigma$, contains the empty word, the substitution σ is referred to as λ-*free* or *nonerasing.* If each $\sigma(a)$ consists of a single word, σ is called a *morphism.* We speak also of *nonerasing* and *letter-to-letter morphisms.*□

Some clarifying remarks are in order. Morphisms were earlier called *homomorphisms* – this fact is still reflected by the notations h and H used in connection with morphisms. Usually in our considerations the target alphabet Δ equals the basic alphabet Σ – this will be the case in the definition of D0L and 0L systems. Then we speak briefly of a finite substitution or morphism *on the alphabet* Σ. In the theory of L systems letter-to-letter morphisms are customarily called *codings*; *weak codings* are morphisms mapping each letter either to a letter or to the empty word λ. Codings in this sense should not be confused with codes discussed elsewhere in this Handbook, also in Section 6 below.

By the above definition a substitution σ is applied to a word w by *rewriting* every letter a of w as some word from $\sigma(a)$. Different occurrences of a may be rewritten as different words of $\sigma(a)$. However, if σ is a morphism then each $\sigma(a)$ consists of one word only, which makes the rewriting *deterministic.* It is convenient to specify finite substitutions by listing the *rewriting rules* or *productions* for each letter, for instance,

$$a \longrightarrow \lambda, \quad a \longrightarrow a^2, \quad a \longrightarrow a^7,$$

this writing being equivalent to

$$\sigma(a) = \{\lambda, a^2, a^7\}.$$

Now, for instance,

$$\sigma(a^2) = \sigma(a)\sigma(a) = \{\lambda, a^2, a^4, a^7, a^9, a^{14}\}.$$

Similarly, the substitution (in fact, a morphism) defined by

$$\sigma_1(a) = \{b\}, \ \ \sigma_1(b) = \{ab\}, \ \ \Sigma = \{a, b\},$$

can be specified by listing the rules

$$a \longrightarrow b, \quad b \longrightarrow ab.$$

In this case, for any word w, $\sigma_1(w)$ consists of a single word, for instance,

$$\sigma_1(a^2ba) = \{b^2ab^2\} = b^2ab^2,$$

where the latter equality indicates only that we often identify singleton sets with their elements.

The above results for $\sigma(a^2)$ and $\sigma_1(a^2ba)$ are obtained by applying the rules *in parallel*: every occurrence of every letter must be rewritten. Substitutions σ (and, hence, also morphisms) are in themselves *parallel* operations. An application of σ to a word w means that something happens everywhere in w. No part of w can remain idle except that, in the presence of the rule $a \longrightarrow a$, occurences of a may remain unchanged.

Before our next fundamental definition, we still want to extend applications of substitutions (and, hence, also morphisms) to concern also languages. This is done in the natural "additive" fashion. By definition, for all languages $L \subseteq \Sigma^*$,

$$\sigma(L) = \{u | \ u \in \sigma(w), \text{ for some } w \in L\}.$$

Definition. A *0L system* is a triple $G = (\Sigma, \sigma, w_0)$, where Σ is an alphabet, σ is a finite substitution on Σ, and w_0 (referred to as the *axiom*) is a word over Σ. The 0L system is *propagating* or a *P0L system* if σ is nonerasing. The 0L system G *generates the language*

$$L(G) = \{w_0\} \cup \sigma(w_0) \cup \sigma(\sigma(w_0)) \cup \ldots = \bigcup_{i \geq 0} \sigma^i(w_0). \ \ \square$$

Consider, for instance, the 0L system

$$\text{MISS}_3 = (\{a\}, \sigma, a) \text{ with } \sigma(a) = \{\lambda, a^2, a^5\}.$$

(Here and often in the sequel we express in the name of the system some characteristic property, rather than using abruptly an impersonal G– notation.) The system can be defined by simply listing the productions

$$a \longrightarrow \lambda, \ \ a \longrightarrow a^2, \ \ a \longrightarrow a^5$$

and telling that a is the axiom. Indeed, both the alphabet Σ and the substitution σ can be read from the productions. (We disregard the case, where Σ contains letters not appearing in the productions.) L systems are often in the sequel defined in this way, by listing the productions.

Going back to the system MISS$_3$, we obtain from the axiom a in one "derivation step" each of the words λ, a^2, a^5. Using customary language-theoretic notation, we denote this fact by

$$a \Longrightarrow \lambda, \quad a \Longrightarrow a^2, \quad a \Longrightarrow a^5.$$

A second derivation step gives nothing new from λ but gives the new words a^4, a^7, a^{10} from a^2 and the additional new words a^6, a^8, a^9, a^{11}, a^{12}, a^{13}, a^{14}, a^{15}, a^{16}, a^{17}, a^{19}, a^{20}, a^{22}, a^{25} from a^5. In fact,

$$a^5 \Longrightarrow a^k \quad \text{iff } k = 2i + 5j, \;\; i + j \leq 5, \;\; i \geq 0, \;\; j \geq 0.$$

(Thus, we use the notation $w \Longrightarrow u$ to mean that $u \in \sigma(w)$.) A third derivation step produces, in fact in many different ways, the missing words a^{18}, a^{21}, a^{23}, a^{24}. Indeed, it is straightforward to show by induction that

$$L(\text{MISS}_3) = \{a^i | \;\; i \neq 3\}.$$

Let us go back to the terminology used in defining L systems. The letter L comes from the name of Aristid Lindenmayer. The story goes that Aristid was so modest himself that he said that it comes from "languages". The number 0 in "0L system" indicates that interaction between individual cells in the development is zero-sided, the development is *without interaction*. In language-theoretic terms this means that rewriting is *context-free*. The system MISS$_3$ is not *propagating*, it is not P0L. However, it is a *unary* 0L system, abbreviated U0L system: the alphabet consists of one letter only. The following notion will be central in our discussions.

Definition. A 0L system (Σ, σ, w_0) is *deterministic* or a *D0L system* iff σ is a morphism. $\Box$

Thus, if we define a D0L system by listing the productions, there is exactly one production for each letter. This means that rewriting is completely deterministic. We use the term *propagating*, or a *PD0L system*, also here: the morphism is nonerasing. The L system used in Section 1.2 for modeling Callithamnion roseum was, in fact, a PD0L system.

D0L systems are the simplest among L systems. Although most simple, D0L systems give a clear insight into the basic ideas and techniques behind L systems and parallel rewriting in general. The first L systems used as models in developmental biology, as well as most of the later ones, were in fact D0L systems. From the point of view of artificial life, creatures modeled by D0L systems have been called, [S7], "Proletarians of Artificial Life", briefly PAL's. In spite of the utmost simplicity of the basic definition, the theory of D0L systems is by now very rich and diverse. Apart from providing tools for modeling real and artificial life, the theory has given rise to new deep insights into language theory (in general) and into the very basic mathematical notion of an endomorphism on a free monoid (in particular), [S5]. At present still a wealth of problems and mysteries remains concerning D0L systems.

Let $G = (\Sigma, h, w_0)$ be a D0L system – we use the notation h to indicate that we are dealing with a (homo)morphism. The system G generates its language $L(G)$ in a specific order, as a *sequence*:

$$w_0, \;\; w_1 = h(w_0), \;\; w_2 = h(w_1) = h^2(w_0), \;\; w_3, \ldots$$

We denote the sequence by $S(G)$. Thus, in connection with a D0L system G, we speak of its language $L(G)$ and sequence $S(G)$. Indeed, D0L systems were the first widely studied *grammatical devices generating sequences*. We now discuss five examples, paying special attention to sequences. All five D0L systems are propagating, that is, PD0L systems. The first one is also unary, that is, a UPD0L system.

Consider first the D0L system EXP$_2$ with the axiom a and rule $a \longrightarrow a^2$. It is immediate that the sequence $S(\text{EXP}_2)$ consists of the words a^{2^i}, $i \geq 0$, in the increasing length order. Secondly, consider the D0L system LIN with the axiom ab and rules $a \longrightarrow a$, $b \longrightarrow ab$. Now the sequence $S(\text{LIN})$ consists of the words $a^i b$, $i \geq 1$, again in increasing length order. The notation LIN refers to linear growth in word length: the j'th word in the sequence is of length $j + 2$. (Having in mind the notation w_0, w_1, w_2, we consider the axiom ab to be the 0th word etc.)

Our next D0L system FIB has the axiom a and rules $a \longrightarrow b$, $b \longrightarrow ab$. The first few words in the sequence $S(\text{FIB})$ are

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

From the word ab on, each word results by catenating the two preceding ones. Let us establish inductively this claim,

$$w_n = w_{n-2} w_{n-1}, \quad n \geq 2.$$

Denoting by h the morphism of the system FIB, we obtain using the definition of h:

$$\begin{gathered} w_n = h^n(a) = h^{n-1}(h(a)) = h^{n-1}(b) = h^{n-2}(h(b)) = \\ = h^{n-2}(ab) = h^{n-2}(a)h^{n-2}(b) = h^{n-2}(a)h^{n-2}(h(a)) = \\ = h^{n-2}(a)h^{n-1}(a) = w_{n-2}w_{n-1}. \end{gathered}$$

The claim established shows that the word lengths satisfy the equation

$$|w_n| = |w_{n-2}| + |w_{n-1}|, \;\; n \geq 2.$$

Hence, the length sequence is the well-known *Fibonacci sequence* 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

The D0L system SQUARES has the axiom a and rules

$$a \longrightarrow abc^2, \;\; b \longrightarrow bc^2, \;\; c \longrightarrow c.$$

The sequence S(SQUARES) begins with the words

$$a, abc^2, abc^2bc^2c^2, abc^2bc^2c^2bc^2c^2c^2.$$

Denoting again by w_i, $i \geq 0$, the words in the sequence, it is easy to verify inductively that

$$|w_{i+1}| = |w_i| + 2i + 3, \text{ for all } i \geq 0.$$

This shows that the word lengths consist of the consecutive sequence: $|w_i| = (i+1)^2$, for all $i \geq 0$. Similar considerations show that the D0L system CUBES with the axiom a and productions

$$a \longrightarrow abd^6, \;\; b \longrightarrow bcd^{11}, \;\; c \longrightarrow cd^6, \;\; d \longrightarrow d$$

satisfies $|w_i| = (i+1)^3$, for all $i \geq 0$.

Rewriting being deterministic gives rise to certain *periodicities* in the D0L sequence. Assume that some word occurs twice in a sequence: $w_i = w_{i+j}$, for some $i \geq 0$ and $j \geq 1$. Because of the determinism, the words following w_{i+j} coincide with those following w_i, in particular,

$$w_i = w_{i+j} = w_{i+2j} = w_{i+3j} = \ldots$$

Thus, after some "initial mess", the words start repeating periodically in the sequence. This means that the language of the system is finite. Conversely, if the language is finite, the sequence must have a repetition. This means that the occurrence of a repetition in $S(G)$ is a necessary and sufficient condition for $L(G)$ being finite. Some other basic periodicity results are given in the following theorem, for proofs see [HR], [Li2], [RS1]. The notation alph(w) means the minimal alphabet containing all letters occurring in w, pref$_k(w)$ stands for the prefix of w of length k (or for w itself if $|w| < k$), suf$_k(w)$ being similarly defined.

Theorem 2.1. Let w_i, $i \geq 0$, be the sequence of a D0L system $G = (\Sigma, h, w_0)$. Then the sets $\Sigma_i = \text{alph}(w_i)$, $i \geq 0$, form an ultimately periodic sequence, that is, there are numbers $p > 0$ and $q \geq 0$ such that $\Sigma_i = \Sigma_{i+p}$ holds for every $i \geq q$. Every letter occurring in some Σ_i occurs in some Σ_j with $j \leq \text{card}(\Sigma) - 1$. If $L(G)$ is infinite then there is a positive integer t such that, for every $k > 0$, there is an $n > 0$ such that, for all $i \geq n$ and $m \geq 0$,

$$\text{pref}_{k-1}(w_i) = \text{pref}_{k-1}(w_{i+mt}) \text{ and } \text{suf}(w_i) = \text{suf}_{k-1}(w_{i+mt}).\square$$

Thus, both prefixes and suffixes of any chosen length form an ultimately periodic sequence. Moreover, the period is independent of the length chosen; only the initial mess depends on it.

Definition. An infinite sequence of words w_i, $i \geq 0$, is *locally catenative* iff, for some positive integers k, $i_1, \ldots i_k$ and $q \geq \max(i_1, \ldots, i_k)$,

$$w_n = w_{n-i_1} \ldots w_{n-i_k} \text{ whenever } n \geq q.$$

A D0L system G is *locally catenative* iff the sequence $S(G)$ is locally catenative.

□

Locally catenative D0L systems are very important both historically and because of their central role in the theory of D0L systems: their study has opened up new branches of the theory. A very typical example of a locally catenative D0L system is the system FIB discussed above.

Also the system EXP_2 is locally catenative: the sequence $S(\text{EXP}_2)$ satisfies

$$w_n = w_{n-1}w_{n-1} \quad \text{for all } n \geq 1.$$

A celebrated problem, still open, is to decide whether or not a given D0L system is locally catenative. No general algorithm is known, although the problem has been settled in some special cases, for instance, when an upper bound is known for the integers $i_1, \ldots i_k$, as well as recently for binary alphabets, [Ch].

An intriguing problem is the avoidability or unavoidability of *cell death*: to what extent are rules $a \longrightarrow \lambda$ necessary in modeling certain phenomena? What is the real difference between D0L and PD0L systems and between 0L and P0L systems? There are some straightforward observations. The word length can never decrease in a PD0L sequence, and a P0L language cannot contain the empty word. We will see in the sequel, especially in connection with growth functions, that there are remarkable differences between D0L and PD0L systems. The theory of D0L systems is very rich and still in many respects poorly understood.

As an example of the necessity of cell death, consider the D0L system DEATH_b with the axiom ab^2a and rules

$$a \longrightarrow ab^2a, \quad b \longrightarrow \lambda.$$

The sequence $S(\text{DEATH}_b)$ consists of all words $(ab^2a)^{2^i}$, $i \geq 0$, in the increasing order of word length. We claim that the language L consisting of these words cannot be generated by any PD0L system G. Indeed, ab^2a would have to be the axiom of such a G, and $ab^2a \Longrightarrow ab^2aab^2a$ would have to be the first derivation step. This follows because no length-decrease is possible in $S(G)$. The two occurrences of a in ab^2a must produce the same subword in ab^2aab^2a. This happens only if the rule for a is one of the following: $a \longrightarrow \lambda$, $a \longrightarrow ab^2a$, $a \longrightarrow a$. The first two alternatives lead to non-propagating systems. But also the last alternative is impossible because no rule for b makes the step $b^2 \Longrightarrow b^2aab^2$ possible. We conclude that it is not possible to generate the language $L(\text{DEATH}_b)$ using a PD0L system.

A slight change in DEATH$_b$ makes such a generation possible. Consider the D0L system G_1 with the axiom aba and rules $a \longrightarrow aba$, $b \longrightarrow \lambda$. Now the PD0L system G_2 with the axiom aba and rules $a \longrightarrow a$, $b \longrightarrow baab$ generates the same sequence, $S(G_1) = S(G_2)$.

We say that two D0L systems G_1 and G_2 are *sequence equivalent* iff $S(G_1) = S(G_2)$. They are *language equivalent*, briefly *equivalent*, iff $L(G_1) = L(G_2)$. Instead of D0L systems, these notions can be defined analogously for any other class of L systems – sequence equivalence of course only for systems generating sequences. The two systems G_1 and G_2 described in the preceding paragraph are both sequence and language equivalent. Clearly, sequence equivalence implies language equivalence but the reverse implication is not valid. Two D0L systems may be (language) equivalent without being sequence equivalent, they can generate the same language in a different order. A simple example is provided by the two systems

$$(\{a,b\}, \{a \longrightarrow b^2, b \longrightarrow a\}, b) \text{ and } (\{a,b\}, \{a \longrightarrow b, b \longrightarrow a^2\}, a).$$

Among the most intriguing mathematical problems about L systems is the *D0L equivalence problem*: construct an algorithm for deciding whether or not two given D0L systems are equivalent. Equivalence problem is a fundamental decision for any family of generative devices: decide whether or not two given devices in the family generate the same language. For D0L systems one can consider, in addition, the *sequence equivalence problem*: Is $S(G_1) = S(G_2)$, given D0L systems G_1 and G_2? The D0L equivalence problems were celebrated open problems for most of the 70's. They were often referred to as the most simply stated problems with an open decidability status. We will return to them and related material in Section 4. It was known quite early, [N], that a solution to the D0L sequence equivalence problem yields a solution to the D0L language equivalence problem, and vice versa.

2.2 Auxiliary letters and other auxiliary modifications

A feature very characteristic for D0L systems and 0L systems is that you have to accept everything produced by the machinery. You have the axiom and the rules, and you want to model some phenomenon. You might want to exclude something that comes out of from the axiom by the rules because it is alien to the phenomenon, does not fit it. This is not possible, you have to include everything. There is no way of hiding unsuitable words. Your D0L or 0L models have no filtering mechanism.

It is customary in formal language theory to use various *filtering mechanisms*. Not all words obtained in derivations are taken to the language but the terminal language is somehow "squeezed" from the set of all derived words. The most typical among such filtering mechanisms, quite essential in grammars in the Chomsky hierarchy, is the use of *nonterminal letters*. An occurrence of a nonterminal in a word means that the word is not (maybe not yet) acceptable. The generated language contains only words without nonterminals. Thus, the language directly obtained is *filtered* by intersecting it with Σ_T^*, where Σ_T is the alphabet consisting of terminal letters. This gives the possibility of rejecting (at least some) unsuitable words.

The same mechanism, as well as many other filters, can be used also with L systems. However, a word of *warning* is in order. The original goal was to model the development of a species, be it real or artificial. Each word generated is supposed to represent some stage in the development. It would be rather unnatural to exclude some words and say that they do not represent proper stages of the development! This is exactly what filtering with nonterminals does.

However, this objection can be overruled because the following justification can be given. Some other squeezing mechanisms, notably *codings* (that is, letter-to-letter morphisms), can be justified from the point of view of developmental models: more careful experiments or observations can change the interpretation of individual cells, after which the cells are assigned new names. This amounts to applying a letter-to-letter morphism to the language. A rather amazing result concerning parallel rewriting, discussed below in more detail, is that coding is in important cases, [ER1], equivalent to the use of nonterminals in the sense that the same language is obtained, and the transition from one squeezing mechanism to the other is effective. By this result, also the use of nonterminals is well-motivated.

The letter E ("extended") in the name of an L system means that the use of nonterminals is allowed. Thus, an *E0L system* is a 0L system, where the alphabet is divided into two disjoint parts, *nonterminals* and *terminals*. E0L and 0L systems work in the same way but only words over the terminal alphabet are in the language of an E0L system. Thus, an E0L system G can be also viewed as a 0L system, where a subalphabet Σ_T is specified and the language of the 0L system is intersected with Σ_T^* to get the language of the E0L system.

The following E0L system SYNCHRO is very instructive. In our notation, *capital* letters are nonterminals and *small* letters terminals. The axiom of SYNCHRO is ABC, and the rules as follows:

$$\begin{array}{lllll}
A \longrightarrow AA' & A \longrightarrow a & A' \longrightarrow A' & A' \longrightarrow a & a \longrightarrow F \\
B \longrightarrow BB' & B \longrightarrow b & B' \longrightarrow B' & B' \longrightarrow b & b \longrightarrow F \\
C \longrightarrow CC' & C \longrightarrow c & C' \longrightarrow C' & C' \longrightarrow c & c \longrightarrow F \\
F \longrightarrow F & & & &
\end{array}$$

It is easy to verify that

$$L(\text{SYNCHRO}) = \{a^n b^n c^n \mid \ n \geq 1\}.$$

This follows because our E0L system is *synchronized* in the sense that all terminals must be reached simultaneously. Otherwise, the failure symbol F necessarily comes to the word and can never be eliminated. The language obtained is a classical example in language theory: a context-sensitive language that is not context-free.

Filtering mechanisms provide families of L languages with a feature very desirable both language-theoretically and mathematically: *closure* under various operations. Without filtering, the "pure" families, such as the families of 0L and D0L languages, have very weak closure properties: most of the customary language-theoretic operations may produce languages outside the family, starting with languages in the family. For instance, $L = \{a, a^3\}$ is the union of two 0L languages. However, L itself is not 0L, as seen by a quick exhaustive search over the possible axioms and rules.

We refer the reader to the preceding chapter in this Handbook for a more detailed discussion concerning the following definition. In particular, the six operations listed are not arbitrary but exhaust the "rational" operations in language theory.

Definition. A family $\mathcal{L}$ of languages is termed a *full AFL* ("abstract family of languages") iff $\mathcal{L}$ is closed under each of the following operations: union, catenation, Kleene star, morphism, inverse morphism, intersection with regular languages. The family $\mathcal{L}$ is termed and *anti-AFL* iff it is closed under none of the operations above. $\square$

Most of the following theorem can be established by exhibiting suitable examples. [RS1] should be consulted, especially as regards the nonclosure of E0L languages under inverse morphisms.

Theorem 2.2. The family of 0L languages is an anti-AFL, and so is the family of D0L languages. The family of E0L languages is closed under all AFL operations except inverse morphism. $\square$

Theorem 2.2 shows clearly the power of the E-mechanism in transforming a family of little structure (in the sense of weak closure properties) into a structurally strong family. The power varies from L family to another. The difference between D0L and ED0L languages is not so big. The periodicity result of Theorem 2.1 concerning alphabets holds also for ED0L sequences and, thus, the words that are filtered out occur periodically in the sequence, which is a considerable restriction.

We now mention other filtering mechanisms. The *H-mechanism* means taking a morphic image of the original language. Consider the case that the original language is a 0L language. Thus, let $G = (\Sigma, \sigma, w_0)$ be a 0L system and $h : \Sigma^* \longrightarrow \Delta^*$ a morphism. (It is possible that the target alphabet Δ equals Σ.) Then $h(L(G))$ is an *H0L language*. The *HD0L languages* are defined analogously.

The *N-mechanism* refers similarly to *nonerasing* morphisms. Thus, N0L languages are of the form $h(L(G))$ above, with the additional assumption that h is nonerasing. The *C-mechanisms* refers to *codings* and *W-mechanism* to *weak codings*.

A further variation of L systems consists of having a *finite set of axioms* – instead of only one axiom as we have had in our definitions so far. This variation is denoted by including the *letter F* in the name of the system. Thus, every finite language L is an F0L language: we just let L be the set of axioms in an F0L system, where the substitution is the identity.

When speaking of language families, we denote the family of E0L languages simply by E0L, and similarly with other families. Consider the family D0L. We have introduced five filtering mechanisms: E, H, N, C, W. This gives six possibilities – either some filtering or the pure family. For each of the six possibilities, we may still add one or both of the letters P and F, indicating that we are dealing with the propagating or finite-axiom variant. This gives altogether 24 families. The following remarkable theorem gives an exhaustive characterization of the mutual relashionship between these 24 families. That such a complicated hierarchy is completely understood is a rather rare situation in language theory. Many of the proofs are rather involved – we return to a tricky question in Section 3. Most of the results are originally from [NRSS], see also [RS1]. In comparing the families, we follow the λ-convention: two languages are considered equal if they differ by the empty word only. Otherwise, propagating families would be automatically different from nonpropagating ones.

Theorem 2.3. The following diagram characterizes mutual relations between deterministic L families. Arrows denote strict inclusion. Families not connected by a path are mutually incomparable.

$$HD0L = WD0L =$$
$$= HPD0L = WPD0L =$$
$$= HDF0L = WDF0L =$$
$$= HPDF0L = WPDF0L$$
$$= NDF0L = CDF0L$$
$$= NPDF0L = CPDF0L$$

$$\begin{tikzcd}
 & ND0L = NPD0L = CD0L & & EDF0L & \\
CPD0L & & ED0L & DF0L & EPDF0L \\
 & EPD0L & D0L & & PDF0L \\
 & & PD0L & &
\end{tikzcd}$$

In the nondeterministic case there is much more collapse in the hierarchy because the C-mechanism has in most cases the same generative capacity as the E-mechanism. (In the deterministic case the former is much stronger.) A rather surprising fact in the nondeterministic case is that, although E0L = C0L, CP0L is properly contained in EP0L, which is the opposite what one would expect knowing the deterministic case. The key results are given in the next theorem, [NRSS] and [RS1].

Theorem 2.4. Each of the following families equals E0L:

$$\begin{aligned} E0L = \quad & C0L = N0L = W0L = H0L = NP0L = EP0L = WP0L = \\ & HP0L = EF0L = CF0L = NF0L = WF0L = HF0L = EPF0L = \\ & NPF0L = WPF0L = HPF0L. \end{aligned}$$

The family E0L lies strictly between context-free and context-sensitive languages and contains properly the mutually incomparable families CP0L and F0L. □

Thus, the family E0L contains properly the family of context-free languages. This fact follows because a context-free grammar can be transformed into an E0L system, without affecting the language, by adding the production $x \longrightarrow x$ for each letter x. That the containment is proper can be seen by considering the language generated by the E0L system SYNCHRO. This fact should be contrasted with the fact that most finite languages are outside the family of 0L languages.

It is customary in language theory to try to reduce grammars into *normal forms*, that is, to show that every grammar can be replaced by an equivalent (generating the same language) grammar possessing some desirable properties. The following theorem, [RS1], is an illustration of such a reduction for L systems. Observe that the special property of the E0L system SYNCHRO concerning terminals is, in fact, a general property of E0L languages.

Theorem 2.5. Every E0L language is generated by an E0L system satisfying each of the following conditions: (i) The only production for each terminal letter a is $a \longrightarrow F$, where F is a nonterminal having $F \longrightarrow F$ as the only production. (ii) The axiom is a single nonterminal not occurring on the right side of any production. (iii) The right side of every production is either a terminal word or consists only of nonterminals. (iv) A terminal word is reachable from every nonterminal apart from F (and the axiom if the language is empty).□

Usually it is difficult to show that a given language is *not* in a given family, because, in principle, one has to go through all the devices defining languages in the family. E0L languages possess certain combinatorial properties and, consequently, a language not having those properties cannot be an E0L language, [RS1]. For instance, the language

$$\{a^m b^n a^m \mid\ 1 \le m \le n\}$$

is not an E0L language. It is very instructive to notice that the languages

$$\{a^m b^n a^m \mid\ 1 \le n \le m\} \text{ and } \{a^m b^n a^m \mid\ m, n \ge 1\}$$

are E0L languages. Finally, the language

$$\{a^{3^n} \mid \; n \geq 1\} \cup \{b^n c^n d^n \mid \; n \geq 1\}$$

is an EP0L language but not a CP0L language.

2.3 Tables, interactions, adults, fragmentation

A feature very characteristic for parallel rewriting is the use of *tables*, [R1], [R2]. A table is simply a set of rewriting rules. A system has several tables, always finitely many. It is essential that, at each step of the rewriting process, always rules from the same table must be used. This reflects the following state of affairs in modeling the development of organisms, real or artificial. There may be different conditions of the environment (day and night, varying heat, varying light, pollution, etc.) or different developmental stages, where it is important to use different rules. Then we consider all sets of rules, tables, obtained in this fashion. Observe that tables do not make sense in sequential rewriting. Because only one rule is used at each derivation step, it suffices to consider the total set of rules. We now define the variations of 0L and D0L systems, resulting by augmenting the system with tables.

Definition. A *T0L system* is a triple $G = (\Sigma, S, w_0)$, where S is a finite set of finite substitutions such that, for each $\sigma \in S$, the triple (Σ, σ, w_0) is a 0L system. The language of the T0L system, $L(G)$, consists of w_0 and of all words in all languages $\sigma_1 \ldots \sigma_k(w_0)$, where $k \geq 1$ and each σ_i belongs to S – some of the σ_i's may also coincide. If all substitutions in S are, in fact, morphisms then G is *deterministic* or a *DT0L system.* □

Thus, D indicates that all substitutions (all tables) are deterministic. However, according to the definition above, there is no control in the use of the tables – the tables may be used in an arbitrary order and multitude. Thus, a DT0L language is not generated in a sequence. A definite sequence results only if the order in the use of the tables is specified in a unique way.

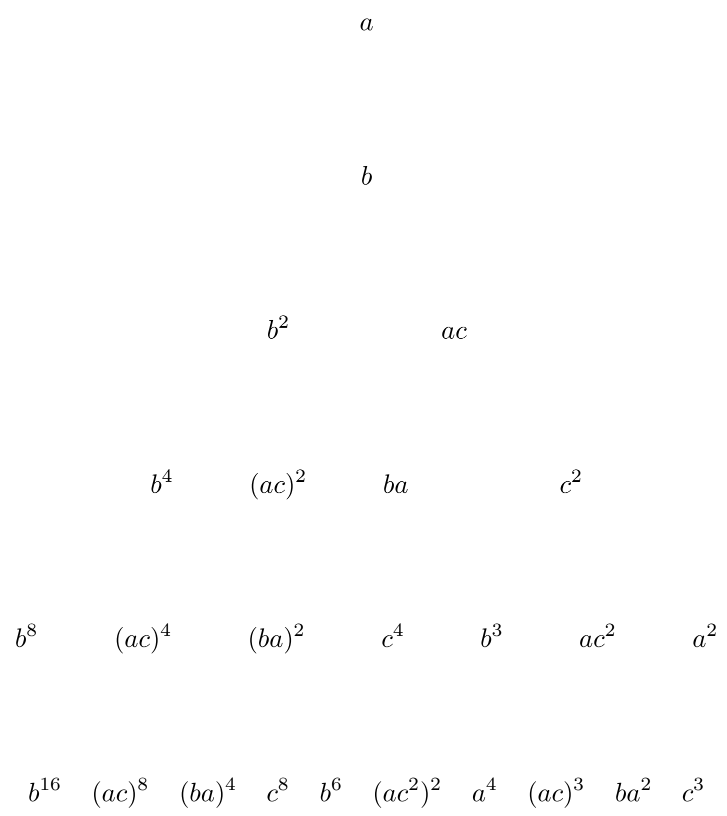
The letter F has the same meaning as before: finitely many axioms instead of only one axiom. Also the filtering mechanisms E, H, C, N, W are the same as before – we will use them below without further explanations.

Following our earlier practice, we will define a T0L system by specifying the axiom and each table, a table being a set of productions included in brackets to indicate that they belong together. There are no restrictions, the same production may appear in several tables.

As an example consider the DT0L system PAL with the axiom a and two tables

$$T_d = [a \longrightarrow b, b \longrightarrow b^2, c \longrightarrow a] \text{ and } T_n = [a \longrightarrow c, b \longrightarrow ac, c \longrightarrow c].$$

(The indices d and n come from "day" and "night", the meaning will become clear below.) Instead of a linear sequence, derivations can be represented as a tree:


a
b
b^2 ac
b^4 $(ac)^2$ ba c^2
b^8 $(ac)^4$ $(ba)^2$ c^4 b^3 ac^2 a^2
b^{16} $(ac)^8$ $(ba)^4$ c^8 b^6 $(ac^2)^2$ a^4 $(ac)^3$ ba^2 c^3

Here the branch indicates which of the tables was used: the left descendant results from an application of T_d, the right descendant from an application of T_n. If a descendant is not marked down (like a and c on the third level), it indicates that it occurrs already at an earlier level. The continuation can, therefore, be ignored if one is only interested in determining the language. The left extremity of the tree contains the powers b^{2^i}, $i \geq 0$. However, all powers b^i, $i \geq 1$, occur somewhere in the tree and so do all powers a^i and c^i, $i \geq 1$. This is seen as follows. Observe first that, for all $i \geq 0$, ba^{i+1} results from ba^i by applying first T_n, then T_d. Now from ba^i the word b^{i+2} results by applying T_d, the word c^{i+2} by applying T_n twice, and the word a^{i+2} from c^{i+2} by T_d.

Although seemingly simple, PAL has a rich structure. An interested reader should have no difficulties in specifying $L(\text{PAL})$ explicitly.) If the order of the application of the tables is given, a unique sequence of words results. One might visualize the two tables as giving rules for the day and night. The alternation is the natural order of application: $T_dT_nT_dT_nT_d\ldots$ (we agree that we begin with T_d). Another possibility is to consider "eternal daylight" (only T_d is used) or "eternal night". Let us still make free the choice of the axiom: instead of the axiom a, we have an arbitrary nonempty word w over $\{a, b, c\}$ as the axiom. (Sometimes the term *L scheme*, instead of an L system, is used to indicate that the axiom is not specified.) Denote by DAY-AND-NIGHT-PAL(w), DAY-PAL(w), NIGHT-PAL(w) the modifications of PAL thus obtained. Each of them generates a specific *sequence* of words. In fact, DAY-PAL(w) and NIGHT-PAL(w) are PD0L systems, whereas DAY-AND-NIGHT-PAL(w) can be viewed as a CPD0L system. For $w = abc$, the sequences look as follows.

$$\begin{aligned}\text{DAY-NIGHT}: &\quad abc, b^3a, (ac)^3c, (ba)^3a, (ac^2)^3c, (ba^2)^3a,\\ &\quad (ac^3)^3c, (ba^3)^3a, (ac^4)^3c, (ba^4)^3a, \ldots\\ \text{DAY}: &\quad abc, b^3a, b^7, b^{14}, b^{28}, \ldots\\ \text{NIGHT}: &\quad abc, cac^2, c^4, c^4, \ldots\end{aligned}$$

Definition. For an infinite sequence w_i, $i \geq 0$, of words, the function $f(n) = |w_n|$ (mapping the set of nonnegative integers into itself) is termed the *growth function* of the sequence.□

Thus, for NIGHT-PAL(abc), $f(0) = 3$, $f(n) = 4$ for $n \geq 1$. For DAY-PAL(abc), $f(0) = 3$, $f(1) = 4$, $f(i + 2) = 2^{7 \cdot 2^i}$ for $i \geq 0$. Finally, for DAY-NIGHT-PAL(abc), $f(0) = 3$, $f(1) = 4$, $f(2i) = f(2i + 1) = 3i + 1$ for $i \geq 1$.

Thus, the three growth functions possess, respectively, the property of being bounded from above by a constant, exponential or linear. In fact, the growth function of any NIGHT-PAL(w) (resp. DAY-PAL(w), DAY-NIGHT-PAL(w)) is bounded from above by a constant (resp. exponential, linear). We will return to this matter in Section 5, where growth functions will be discussed.

The following two theorems summarize results concerning mutual relashionships between "table families", [NRSS], [RS1]. The first of the theorems deals with deterministic families, and the second with nondeterministic families.

Theorem 2.6. The following inclusions and equalities hold:
DT0L $\subset$ CDT0L= NDT0L= EDT0L= WDT0L= HDT0L,

PDT0L $\subset$ CPDT0L $\subseteq$ NPDT0L $\subseteq$ EPDT0L= WPDT0L= HPDT0L,

DTF0L $\subset$ CDTF0L= NDTF0L= EDTF0L= WDTF0L= HDTF0L,

PDTF0L $\subset$ CPDTF0L $\subseteq$ NPDTF0L $\subseteq$ EPDTF0L= WDPTF0L= HPDTF0L.

The "pure" families (without any filtering) satisfy the following inclusion diagram:

$$\begin{tikzcd}
 & \mathrm{DTF0L} & \\
\mathrm{DT0L} & & \mathrm{PDTF0L} \\
 & \mathrm{PDT0L} &
\end{tikzcd}$$

□

Theorem 2.7. Each of the following families equals ET0L:
ET0L= CT0L= NT0L= WT0L= HT0L=
= NPT0L= EPT0L= WPT0L= HPT0L=
= CTF0L= NTF0L= ETF0L= WTF0L= HTF0L
= NPTF0L= EPTF0L= WPTF0L= HPTF0L
The families E0L, TF0L and CPT0L= CPTF0L are all stricly included in ET0L. □

The family ET0L is the largest widely studied L family, where rewriting is context-free (no cell interactions are present). It is also very pleasing mathematically and has strong closure properties. (It is, however, not closed under the shuffle operation.) It was observed already in the early 70's that ET0L is contained in the family of indexed languages (see the preceding chapter of this Handbook for a description of indexed languages and see [En] for more general hierarchies) and, consequently, facts concerning indexed languages hold also for ET0L languages. The facts in the following theorem can be established rather easily.

Theorem 2.8. The family ET0L is a full AFL, whereas the family T0L is an anti-AFL. Every ET0L language is generated by an ET0L system with two tables. Every ET0L language is generated by an ET0L system such that $a \longrightarrow F$ is the only rule in every table for every terminal a, and $F \longrightarrow F$ is the only rule in every table for the nonterminal F. □

The two normal form results stated in Theorem 2.8, the two-table condition and synchronization, cannot always be reached simultaneously. A number of deep combinatorial results, [RS1], can be established for EDT0L and ET0L languages. Using such results, relatively simple examples can be given of languages *not* in the families. Let Σ contain at least two letters and $k \geq 2$ be a fixed integer. Then neither of the languages

$$\{w \in \Sigma^*|\ \ |w| = k^n, n \geq 0\} \text{ and } \{w \in \Sigma^*|\ \ |w| = n^k, n \geq 0\}$$

is in EDT0L. The 0L system with the axiom a and rules

$$a \longrightarrow ab, b \longrightarrow bc, b \longrightarrow bd, c \longrightarrow c, d \longrightarrow d$$

generates a language not in EDT0L. None of the languages

$$\{(ab^n)^m|\ \ 1 \leq m \leq n\}, \quad \{(ab^n)^m|\ \ 1 \leq n \leq m\},$$

$$\{(ab^n)^m|\ \ 1 \leq m = n\}, \quad \{w \in \{a,b\}^*|\ \ |w|_b = 2^{|w|_a}\}$$

is in ET0L. Here $|w|_x$ denotes the number of occurrences of the letter x in w.

In the remainder of this subsection 2.3 we survey briefly some areas in the theory of L systems that are important both historically and language-theoretically in the sense that they have built bridges between sequential and parallel rewriting. Our exposition will be informal. More details can be found in [RS1], [HWa], [RRS].

In all L systems discussed so far the individual cells develop without any interaction with their neighbours. In language-theoretic terms, rewriting has been context-free. We now discuss *L systems with interactions, IL sysetms.* First some terminology.

In an *(m,n)L system*, $m, n \geq 0$, the rules look like

$$(\alpha, a, \beta) \longrightarrow w, \ \ |\alpha| = m, |\beta| = n.$$

This means that, between the words α and β, the letter a can be rewritten as the word w. Also now, *parallelism* applies: all letters must be rewritten simultaneously. In order to get sufficiently many letters on both sides of any given letter, the whole rewriting takes place between "environment symbols" #, m of them to the left and n to the right. The rules have to be provided also for the case that some prefix of α or suffix of β consists of #'s.

The *IL system* is a collective name for all (m,n)L systems. (1, 0)L and (0, 1)L systems are referred to as *1L systems*: cell interaction is 1-sided, the rewriting of a letter depends always on its left neighbour only (or always on its right neighbour).

The use of letters such as D, E, P is the same as before. In particular, *determinism* means that, for each configuration consisting of a letter and an (m,n)-neighbourhood, there is exactly one rule. The following D(1, 0)L system is known as *Gabor's Sloth*, due to Gabor Herman. It is very important in the theory of growth functions. The alphabet is $\{a,b,c,d\}$ and the axiom ab. The rules are defined by the following table, where the row indicates the left neighbour and the column the letter to be rewritten:

	a	b	c	d
#	c	b	a	d
a	a	b	a	d
b	a	b	a	d
c	b	c	a	ad
d	a	b	a	d

Here again, because of determinism, the language is produced in a *sequence*, the beginning of which is:

$$ad, cd, aad, cad, abd, cbd, acd, caad, abad, cbad,$$

$$acad, cabd, abbd, cbbd, acbd, cacd, abaad, \ldots$$

Such a growth in word length is not possible for D0L sequences. The intervals in which the growth function stays constant grow beyond all limits – they even do so exponentially in terms of the constant mentioned. The entire growth is logarithmic.

It is obvious that the generative capacity increases if the interaction becomes more extensive: $(m+1, n)$L systems generate more than (m, n)L systems. A similar result concerns also the right context, leading to an infinite hierarchy of language families. It is, however, very interesting that only the *amount of context matters*, not its distribution. All the following families coincide:

$$(4,1)\mathrm{L} = (3,2)\mathrm{L} = (2,3)\mathrm{L} = (1,4)\mathrm{L}.$$

The families $(5,0)$L and $(5,0)$L are mutually incomparable, contained strictly in $(4,1)$L, and incomparable with $(3,1)$L. Analogous results hold true in general.

Since already E$(1,0)$L systems (as well as E$(0,1)$L systems) generate all recursively enumerable languages, further modifications such as tables are studied for some special purposes only. EPIL systems produce, as one would expect, the family of context-sensitive languages. There are many surprises in the deterministic case. For instance, there are nonrecursive languages in D1L (showing that the family is big), whereas ED1L does not contain all regular languages (showing that even the extended family is small).

At some stage also the organisms modeled by L systems are expected to become *adults*. It has become customary in the theory of L systems to define adults as follows. A word belongs to the *adult language* of a system exactly in case it derives no words but itself. For instance, assume that

$$a \longrightarrow ab, \quad b \longrightarrow c, \quad c \longrightarrow \lambda, \quad d \longrightarrow dc$$

are the only rules for a, b, c, d in a 0L system. Then all words of the form $(abc)^i(dc)^j$ belong to the adult language. Adult languages of 0L systems are called A0L *languages*. Of course, A can be used in connection with other types of L systems as well.

We have presented adult languages following their customary definition. It is maybe not a proper way to model adults in artificial life, perhaps it models better some kind of "stagnant stability". It is, however, a very interesting language-theoretic fact that A0L equals the family of context-free languages. Similarly, A$(1,0)$L equals the family of recursively enumerable languages.

L systems with fragmentation, JL systems, should be quite useful for modeling artificial life. The mechanism of fragmentation provides us with a new formalism for blocking communication, splitting the developing filament and also for cell death. (The letter J comes from the Finnish word JAKAUTUA, to fragment. At the time of its introduction, all letters coming from suitable English words already had some other meaning for L systems!).

The basic idea behind the fragmentation mechanism is the following. The right-hand sides of the rules may contain occurrences of a special symbol q. The symbol induces a *cut* in the word under scan, and the derivation may continue from any of the parts obtained. Thus, if we apply the rules

$$a \longrightarrow aqa, \quad b \longrightarrow ba, \quad c \longrightarrow qb$$

to the word abc, we obtain the words a, aba and b. The derivation may continue from any of them. The J0L system with the axiom aba and rules $a \longrightarrow a$, $b \longrightarrow abaqaba$ generates the language

$$\{aba^n | \ n \geq 1\} \cup \{a^n ba | \ n \geq 1\}.$$

An interesting language-theoretic fact is that E0L languages, in addition to numerous other closure properties, are closed under fragmentation:

$$\text{EJ0L} = \text{JE0L} = \text{E0L}.$$

J0L systems have been used recently, [KRS], for obtaining a compact representation of certain regular trees.

We have already referred to the quite unusual uniformity in the basic terminology about L systems, in particular, the letters used for naming systems. The following summarizing *glossary* is intended to assist the reader. It is not exhaustive – also other letters have been used but without reaching a similar uniformity.

A. adult, adult word, adult language
C. coding, letter-to-letter morphism, image under such morphism
D. deterministic, only one choice, only one choice in each table
E. extended, intersection with a terminal vocabulary is taken
F. finite set of axioms, rather than only one axiom
H. homomorphism, morphism, image under morphism
I. interactions, neighbouring cells affect the development of the cell
J. fragmentation, the mechanism of inducing cuts
L. Lindenmayer, appears in the name of all developmental systems

N. nonerasing morphism, image under such a morphism
O. actually number 0 but often read as the letter, information 0-sided no interaction, rewriting context-free
P. propagating, no cell death, empty word never on the right-hand side of a rule
T. tables, sets of rules, diversification of developmental instructions
U. unary, alphabet consisting of one letter
W. weak coding, a morphism mapping each letter to a letter or the empty word, image under such a morphism

3 Sample L techniques: avoiding cell death if possible

It is clear that, due to space restrictions, we are only able to state results, not to give proofs or even outline proofs. The purpose of this section is to give some scattered samples of the techniques used. The study of L systems has brought many new methods, both for general language theory and for applications to modeling, notably in computer graphics. Therefore, we feel it proper to include some examples of the methods also in this Handbook. For computer graphics, we refer to the contribution of P.Prusinkiewicz in this Handbook.

A very tricky problem, both from the point of view of the theory and the phenomenon modeled, is *cell death*. Are rules of the form $a \longrightarrow \lambda$ really necessary? Here you have to be very specific. Consider the D0L system DEATH$_b$ defined in subsection 2.1. We observed that $S(\mathrm{DEATH}_b)$ cannot be generated by any PD0L system. In this sense cell death is necessary. However, the situation is different if we are interested only in the sequence of word lengths. The very simple PD0L system with the axiom a^4 and the rule $a \longrightarrow a^2$ possesses the same growth function as DEATH$_b$.

In general, the growth function of every PD0L system is monotonously increasing. There can be no decrease in word length, because in a PD0L system every letter produces at least one letter. Assume that you have a monotonously strictly increasing D0L growth function f, that is, $f(n) < f(n+1)$ holds for all n. Can such an f always be generated by a PD0L system? It is quite surprising that the answer is negative, [K3]. There are monotonously strictly increasing D0L growth functions that cannot be generated by any PD0L system. For some types of D0L growth cell death is necessary, although it cannot be observed from the length sequence.

If one wants to construct a D0L-type model without cell death, one has to pay some price. The productions $a \longrightarrow \lambda$ certainly add to the generative capacity and cannot be omitted without compensation. The amount of the price depends on what features one wishes to preserve. We are now interested in languages. We want to preserve the language of the system but omit productions $a \longrightarrow \lambda$.

We can achieve this by taking finitely many axioms and images under codings, that is, letter-to-letter morphisms. In other words, the C- and F-features are together able to compensate the effect of the productions $a \longrightarrow \lambda$. *If we add C and F to the name of the system, we may also add P.* Specifically, we want to prove that

(*) $$\text{D0L} \subseteq \text{CPDF0L}.$$

The inclusion

$$\text{CD0L} \subseteq \text{CPDF0L}$$

is an immediate consequence of (*). (In fact, since the composition of two codings is again a coding, we can essentially multiply (*) from the left by C.) Moreover, (*) is an important building block in establishing the hierarchy presented in Theorem 2.3, in particular, in showing the high position of some P-families. The earliest (but very compactly written) proof of (*) appears in [NRSS].

We now begin the proof of (*). Let

$$G = (\Sigma, h, w_0)$$

be a given D0L system. We have to show that $L = L(G)$ is a CPDF0L language, that is, L is a letter-to-letter morphic image of a language generated by a PD0L system with finitely many axioms. This is obvious if L is finite. In this case we take L itself to be the finite axiom set of our PDF0L system. The identity rule $a \longrightarrow a$ will be the only rule for every letter, and the coding morphism is the identity as well. If L contains the empty word λ, then L is necessarily finite. This follows because, whenever λ appears in the sequence, no non-empty word can appear in the sequence afterwards. This observation implies that, for the proof of (*), it is not necessary to make the λ-convention considered in connection with Theorem 2.3. If the empty word appears in the given D0L language, it can be taken as an axiom of the PDF0L system.

Thus, from now on we assume that the given D0L language $L = L(G)$ is *infinite*. We use our customary notation w_i, $i \geq 0$, for the sequence $S(G)$. Our argumentation will make use of *periodicities* in $S(G)$, already mentioned in Theorem 2.1 (However, we will not make use of Theorem 2.1, our argumentation will be self-contained.) We will first illustrate periodicities by a simple case needed only for a minor detail. Consider the minimal alphabets of the words w_i,

$$\Sigma_i = \text{alph}(w_i), \quad i \geq 0.$$

Each Σ_i is a subset of the finite set Σ. Hence, there are only finitely many alphabets Σ_i and, consequently, numbers $q \geq 0$ and $p \geq 1$ such that $\Sigma_q = \Sigma_{q+p}$. (We may take q to be the smallest number such that Σ_q occurs later and p the smallest number such that $\Sigma_q = \Sigma_{q+p}$. This choice defines q and p unambiguously.)

We now use the obvious fact that

$$\mathrm{alph}(w) = \mathrm{alph}(w') \text{ implies } \mathrm{alph}(h(w)) = \mathrm{alph}(h(w')),$$

for all words w and w'. Consequently, $\Sigma_{q+1} = \Sigma_{q+p+1}$ and, in general,

$$\Sigma_{q+i} = \Sigma_{q+p+i}, \quad \text{for all } i \geq 0.$$

Thus, the alphabets form the ultimately periodic sequence

$$\underbrace{\Sigma_0, \ldots, \Sigma_{q-1},}_{\text{initial mess}} \underbrace{\Sigma_q, \ldots, \Sigma_{q+p-1},}_{\text{period}} \quad \Sigma_q, \ldots, \Sigma_{q+p-1}, \ldots$$

From the point of view of a developmental model, we can visualize this as follows. In the early history, there may be some primaeval cell types which vanish and never come back again. Strange phenomena are possible, for instance, one can have in the words of the initial mess any length sequence of positive integers one wants just by using sufficiently many cell types (letters). After the early history, p different phenomena have been merged together. (The merged phenomena should not be too different, for instance, the growth order of the p subsequences must be the same. Section 5 explains this further.)

The following example might illustrate the situation. Consider the PD0L system with the axiom $d_1 d_2^4$ and productions

$$\begin{array}{llll} d_1 \longrightarrow d_3 d_4^2, & d_2 \longrightarrow d_4, & d_3 \longrightarrow ad, & d_4 \longrightarrow \lambda, \\ a \longrightarrow a_1, & b \longrightarrow b_1, & c \longrightarrow c_1, & d \longrightarrow d, \\ a_1 \longrightarrow abc^2, & b_1 \longrightarrow bc^2, & c_1 \longrightarrow c. & \end{array}$$

The beginning of the word sequence is

$$d_1 d_2^4, d_3 d_4^6, ad, a_1 d, abc^2 d, a_1 b_1 c_1^2 d,$$

$$abc^2 bc^4 d, a_1 b_1 c_1^2 b_1 c_1^4 d, abc^2 bc^4 bc^6 d, \ldots,$$

the corresponding part of the length sequence being

$$5, 7, 2, 2, 5, 5, 10, 10, 17, \ldots.$$

In this case the lengths of the initial mess and period are: $q = 4$, $p = 2$. The reader might have already noticed that our old friend SQUARES is lurking in this D0L system and, consequently, the growth functon satisfies

$$f(0) = 5, f(1) = 7, f(2i) = f(2i+1) = i^2 + 1, \text{ for } i \geq 1.$$

The initial mess is used to generate two exceptional values initially, and the period of length 2 serves the purpose of creating an idle step after each new square has been produced.

We now go back to the proof of (*), considering again the D0L system $G = (\Sigma, h, w_0)$ generating an infinite language $L = L(G)$. For each $a \in \Sigma$, we denote by u_i^a, $i \geq 0$, the word sequence of the D0L system with the axiom a and morphism h. Thus, $u_i^a = h^i(a)$. We divide Σ into two disjoint parts, $\Sigma = \Sigma^{fin} \cup \Sigma^{inf}$ by defining

$$\Sigma^{fin} = \{a \in \Sigma |\ \{u_i^a |\ i \geq 0\} \text{ is finite }\},\ \ \Sigma^{inf} = \Sigma \setminus \Sigma^{fin}.$$

The set Σ^{fin} consists of those letters that generate a finite language. In other words, the sequence u_i^a, $i \geq 0$, is ultimately periodic, with the period p_a and threshold q_a. Of course, the letters a with the rule $a \longrightarrow \lambda$ are in Σ^{fin}. Since the language L is infinite, the set Σ^{inf} must be nonempty.

We now choose a *uniform period* P satisfying

$$u_P^a = u_{iP}^a, \quad \text{for all } i \geq 0 \text{ and } a \in \Sigma^{fin}.$$

Conditions sufficient for such a choice are that P exceeds all thresholds q_a and is divisible by all periods p_a, a rude estimate for P followed in [NRSS] is to take the product of all periods and thresholds. Further, let M be the maximum length of any word derived in G in P steps from a single letter:

$$M = \max\ \{|u_P^a|\ \mid\ a \in \Sigma\}.$$

Our true period will be a multiple of P, PQ. Specifically, $Q \geq 2$ is an integer such that

$$|u_{P(Q+1)}^a| > M, \text{ for all } a \in \Sigma^{inf}.$$

Observe that by definition all sequences $\{u_i^a\}$, $a \in \Sigma^{inf}$, contain arbitrarily long words. However, also length decrease occurs in the sequences. It might look questionable that we can find a number Q as required. Since D0L sequences are full of surprises, let us prove this in detail. We prove a somewhat stronger statement which is of interest also on its own: *For any D0L sequence $\{v_i\}$ containing infinitely many different words and any number t, there is a bound i_0 such that*

$$|v_i| > t \textit{ whenever } i \geq i_0.$$

This claim follows by considering the sequence of the minimal alphabets of the words v_i. As seen above, this sequence is ultimately periodic, with period p and threshold q. Each of the D0L sequences

$$\{\bar{v}_i^k = v_{ip+q+k} |\ i \geq 0\}, \quad 0 \leq k < p$$

is *conservative*, that is, the minimal alphabet of all words in the sequence is the same. The morphism defining these sequences is the pth power of the original morphism. This means that all p sequences $\{\bar{v}_i^k\}$ are monotonously length-increasing. Thus, to exceed the length t, we only have to watch that the word length in each of the p sequences, merged together to obtain $\{v_i\}$, exceeds t.

Intuitively, the statement just established says that there is some over-all growth in the sequence because all of the subsequences are growing.

There are three key ideas in the following formal proof. *(i)* The effect of the λ-rules can be eliminated in long enough derivation steps. The new productions will simulate steps of length PQ in G. This means that we must have PQ starting points. But this will be no problem because we have the F-feature available. Our first new system G' may still have λ-rules but they can be applied to the axioms only. Thus, a straightforward modification of augmenting the axiom set by words derivable in one step in G gives a second new system G'' without λ-rules. *(ii)* Each letter develops according to G in the same way in a specific number of steps, quite independently of the location of the steps. Our simulation uses the interchange between the steps from 0 to PQ and from P to $P(Q+1)$. *(iii)* In our new system G' an occurrence of a letter, say b, is replaced by information concerning where this particular b comes from P steps earlier in G. The letter b has a definite ancestor on each preceding level. Say c is the ancestor P steps earlier. We cannot recover b if we know only c. However, c generates at most M (occurrences of) letters in P steps. Hence, we should also tell the number i, $1 \le i \le M$, indicating that the particular occurrence of b is the ith letter (counting from left to right) generated by c in P steps. Altogether b is replaced by the pair (c, i); the alphabet of G' includes pairs of this form. The letter b is recovered for the final language by the coding g:

$$g((c, i)) = b.$$

The procedure enables us to get one "free" step (of length P) in the derivations. The situation can be depicted as follows:

past $\ldots\ldots c \ldots\ldots$

P steps Replace b by (c, i);

in G Coding back:

$g((c, i)) = b$

present $\ldots \underbrace{\ldots}_{i-1 \text{ letters}} b \ldots\ldots$

We are now ready for the formal details. Define first a DF0L system G'. The alphabet is

$$\Sigma \cup \Sigma_{place}, \text{ where } \Sigma_{place} = \Sigma \times \{1, 2, \ldots, M\}.$$

Thus, letters of Σ_{place} are pairs (a, i), where $a \in \Sigma$ and $1 \le i \le M$. The axiom set of G' is $\{w_i | \ 0 \le i < PQ\}$. (The words w_i are from the sequence of the original G.) For each $a \in \Sigma$, the right-hand side of its rule in G' is a word over Σ_{place}, obtained from u^a_{PQ} by replacing any occurrence of a letter, say b, with the letter $(c, i) \in \Sigma_{place}$ iff the particular b is the ith letter (from left to right) generated, in P steps, from an occurrence of the letter c in $u^a_{P(Q-1)}$. If $u^a_{PQ} = \lambda$ then $a \longrightarrow \lambda$ is the production for a in G'.

To complete the definition of G', we still have to define the rules for the letters of Σ_{place}. Each right-hand side will be a *nonempty* word over Σ_{place}. Consider first a letter (a, i) for which $i < |u^a_P|$. The rule is $(a, i) \longrightarrow (b, j)$ iff the ith letter of $u^a_{P(Q+1)}$ is derived as the jth letter from an occurrence of the letter b in u^a_{PQ}. Consider next the letter $(a, |u^a_P|)$. The right-hand side of its rule is the following nonempty (by the definition of Q) word over Σ_{place} of length

$$|u^a_{P(Q+1)}| - |u^a_P| + 1.$$

The ith letter in this word is defined as the letter (b, j) for which the $(|u^a_P| - 1 + i)$th letter of $u^a_{P(Q+1)}$ is derived as the jth letter from an occurrence of the letter b in u^a_{PQ}. (The letters (a, i) with $|u^a_P| < i \leq M$ are useless in the sense that they never occur in $L(G')$. We may take the identity production $(a, i) \longrightarrow (a, i)$ for such letters. It is also irrelevant how the coding g is defined for such letters.)

We now define the letter-to-letter morphism g as follows:

If $a \in \Sigma$ then $g(a) = a$,

If $(a, i) \in \Sigma_{place}$ and $1 \leq i \leq |u^a_P|$ then $g((a, i))$ equals the ith letter of u^a_P.

It now follows by our construction that $g(L(G')) = L(G)$. The explanations given before the construction should be helpful in establishing this fact. G' is a DF0L system surely having λ-rules if G has them. However, the λ-rules of G' can be applied only to the axioms. After the first derivation step λ-rules are no longer applicable because all words are over the alphabet Σ_{place} for which all rules are propagating. Let w'_i, $0 \leq i < PQ$, be the word derived in G' from w_i in one step.

We now transform G' to a PDF0L system G'' by augmenting the axiom set of G' with the set $\{w'_i | \ 0 \leq i < PQ\}$ and replacing the rules of G' for letters $a \in \Sigma$ by the identity rules $a \longrightarrow a$. In this way all λ-rules are removed. The effect of the rules of G' for letters of Σ, that is, the introduction of the letters in Σ_{place} is in G'' taken care of by the additional axioms w'_i, $0 \leq i < PQ$. The original axioms of G' generate themselves in G''. Thus, the PDF0L system G'' satisfies $L(G'') = L(G')$. We may use the coding g defined above to obtain the final result

$$L = L(G) = g(L(G'')).$$

This shows that L is a CPDF0L language, which completes the proof of (*). □

The construction presented above is quite complicated. This is understandable because the result itself is perhaps the most sophisticated one about the elimination of λ-rules in the theory of L-systems. The analogous construction is much simpler, [SS], if one only wants to preserve length sequences, not the language, as we have preserved.

We mention finally that, to eliminate λ-rules, we need *both* of the features C and F. It is seen from Theorem 2.3 that neither the family CPD0L nor the family PDF0L contains the family D0L.

4 L decisions

4.1 General. Sequences versus languages

We will now present results concerning *decision problems* in the theory of L systems. This is a rich area – our overview will be focused on the celebrated *D0L equivalence problem* and variations. More information is contained especially in [CK2] and [RS3]. We will begin with a general overview.

Customary decision problems investigated in language theory are the *membership, emptiness, finiteness, equivalence* and *inclusion problems.* The problems are stated for a specific *language family*, sometimes in a *comparative* sense between two families, [S4]. When we say that the equivalence problem between 0L and D0L languages is decidable, this means the existence of an algorithm that receives as its input a pair (G, G') consisting of a 0L and a D0L system and tells whether or not $L(G) = L(G')$. "Equivalence" without further specifications always refers to language equivalence, in case of deterministic systems we also speak of *sequence equivalence.* Given two language families $\mathcal{L}$ and $\mathcal{L}'$, we also speak of the $\mathcal{L}'$ -ness problem for $\mathcal{L}$, for instance, of the D0L-ness (resp. regularity) problem for 0L languages. The meaning of this should be clear: we are looking for an algorithm that decides of a given 0L system G whether or not $L(G)$ is a D0L language (resp. a regular language).

We now present some basic decidability and undecidability results, dating mostly already from the early 70's, see [Li1], [CS1], [La1], [La2], [RS1], [RS3], [S2], [S5] for further information. It was known already in 1973 that ET0L is included in the family of indexed languages, for which membership, emptiness and finiteness were known to be decidable already in the 60's. For many subfamilies of ET0L (recall that ET0L is the largest L family without interactions) much simpler direct algorithms have been developed.

Theorem 4.1. Membership, emptiness and finiteness are decidable for ET0L languages, and so are the regularity and context-freeness problem for D0L languages and the D0L-ness problem for context-free languages. Also the equivalence problem between D0L and context-free languages is decidable, and so is the problem of whether or not an E0L language is included in a regular language. □

Theorem 4.2. The equivalence problem is undecidable for P0L languages and so are the context-freeness, regularity and 0L-ness problems for E0L languages. □

Because even simple subfamilies of IL contain nonrecursive languages (for instance, the family D1L), it is to be expected that most problems are undecidable for the L families with interactions. For instance, language and sequence equivalence are undecidable for PD1L systems.

For systems without interactions, many undecidable problems become decidable in the case of a one-letter alphabet. Decision problems can be reduced to arithmetical ones in the *unary* case. Sample results are given in the following theorem, [La1], [La2], [S2]. TU0L and U0L are written briefly TUL and UL.

Theorem 4.3. The equivalence problem between TUL and UL languages, as well as the equivalence problem between TUL languages and regular languages are decidable. the regularity and UL-ness problems are decidable for TUL languages, and so is the TUL-ness problem for regular languages. □

The cardinality $d_G(n_)$ of the language generated by an L system G using *exactly n derivation steps* has given rise to many decision problems. The earliest result is due to [Da1], where the undecidability of the equation $d_{G_1}(n) = d_{G_2}(n)$ for two DT0L systems is shown. The undecidability holds for 0L systems as well. However, it is decidable, [Ni], whether or not a given 0L system G is *derivation-slender*, that is, there is a constant c such that $d_G(n) \leq c$ holds for all n. All derivation-slender 0L languages are *slender* in the sense that there is a constant k such that the language has at most k words of any given length, [NiS]. The converse does not hold true: the language $\{b^iab^i | \ i \geq 0\} \cup \{b^{2i+1} | \ i \geq 0\}$ is a slender 0L language not generated by any derivation-slender 0L system. Further results concerning slender 0L languages are presented in [NiS]. However, at the time of this writing, the general problem of deciding the slenderness of a given 0L language remains open.

We now go into different variations of *D0L equivalence problems*, this discussion will be continued in the next subsection. This problem area has given rise to many important new notions and techniques, of interest far beyond the theory of L systems. We denote briefly by *LE-D0L* and *SE-D0L* the language and sequence equivalence problems of D0L systems. It was known already in the early 70's, [N], that the two problems are "equivalent" in the sense that any algorithm for solving one of them can be transformed into an algorithm for solving the other. We will now prove this result. Our proof follows [S6] and is another illustration of the techniques used in L proofs.

Theorem 4.4. SE-D0L and LE-D0L are equivalent.

Proof from LE-D0L to SE-D0L.

We show how any algorithm for solving LE-D0L can be used to solve SE-D0L. Given two D0L systems

$$G = (\Sigma, g, u_0) \text{ and } H = (\Sigma, h, v_0),$$

we define two new D0L systems

$$G_b = (\Sigma \cup \{b\}, g_b, bu_0), \quad H_b = (\Sigma \cup \{b\}, h_b, bv_0), b \notin \Sigma,$$

where $g_b(b) = h_b(b) = b^2$, and $g_b(a) = g(a)$, $h_b(a) = h(a)$, for all $a \in \Sigma$. Clearly, $S(G) = S(H)$ iff $L(G_b) = L(H_b)$. □

Proof from SE-D0L to LE-D0L.

We assume without loss of generality that the given languages $L(G)$ and $L(H)$ are infinite. Their finiteness is easily decidable and so is their equality if one of them is finite. The tools we will be using are *decompositions of D0L systems* and *Parikh vectors.* Decompositions refer to *periodicities* already discussed in Subsection 3.1 above. Given a D0L system $G = (\Sigma, g, u_0)$ and integers $p \geq 1$, $q \geq 0$ (period and initial mess), we define the D0L system

$$G(p, q) = (\Sigma, g^p, g^q(u_0)).$$

The sequence $S(G(p, q))$ is obtained from $S(G)$ by taking every pth word after the initial mess.

For $w \in \Sigma^*$, the *Parikh vector* $\Psi(w)$ is a card(Σ)-dimensional vector of nonnegative integers whose components indicate the number of occurrences of each letter in w. For two words w and w', the *notation* $w \leq_p w'$ (resp. $w <_p w'$) means that the Parikh vectors satisfy $\Psi(w) \leq \Psi(w')$ (resp. $\Psi(w) < \Psi(w')$). (The ordering of vectors is taken componentwise, two vectors can be incomparable.)

We use customary notations for the given D0L systems:

$$G = (\Sigma, g, u_0), \quad H = (\Sigma, h, v_0).$$

(We may assume that the alphabets coincide.) The notation w_i refers to words in one of the two D0L sequences u_i and v_i we are considering. We omit the proofs of the following properties (i) – (iv). They are straightforward, apart from (iv) which depends on the theory of growth functions discussed in the following section.

(i) There are words such that $w_i <_p w_j$.

(ii) We cannot have both $i < j$ and $w_j \leq_p w_i$.

(iii) Whenever $w_i <_p w_j$, then $w_{i+n} <_p w_{j+n}$ for all n.

(iv) Assume that x_i and y_i, $i \geq 0$, are D0L sequences over an alphabet with n letters such that $\Psi(x_i) = \Psi(y_i)$, $0 \leq i \leq n$. Then $\Psi(x_i) = \Psi(y_i)$ for all i. (We call x_i and y_i *Parikh equivalent.*)

We now give an algorithm for deciding whether or not $L(G) = L(H)$. The algorithm uses a parameter m (intuitively, the length of the discarded initial mess). Originally we set $m = 0$.

Step 1. Find the smallest integer $q_1 > m$ for which there exists an integer p such that

$$u_{q_1 - p} <_p u_{q_1}, \quad 1 \leq p \leq q_1 - m.$$

Let p_1 be the smallest among such integers p. Determine in the same way integers q_2 and p_2 for the system $H(1, m)$. (Step 1 can be accomplished by property (i).)

Step 2. If the two finite languages

$$\{u_i | \ m \leq i < q_1\} \text{ and } \{v_i | \ m \leq i < q_2\}$$

are different, stop with the conclusion $L(G) \neq L(H)$. (Otherwise, $q_1 = q_2$.)

Step 3. Apply the algorithm for SE-DOL. Check whether or not $p_1 = p_2$ and there is a permutation Π of the set of indices $\{0, 1, \ldots, p_1 - 1\}$ such that

$$S(G(p_1, q_1 + j)) = S(H(p_1, q_1 + \Pi(j))), \text{ for all } j = 0, \ldots, p_1 - 1.$$

If "yes", stop with the conclusion $L(G) = L(H)$. If "no", take q_1 as the new value of m and go back to Step 1.

Having completed the definition of the algorithm, we establish its correctness and termination.

Correctness. We show first that the conclusion in Step 2 is correct. When entering Step 2, we must have $\{u_i |\ i < m\} = \{v_i |\ i < m\}$. Indeed, this holds vacuously for $m = 0$, from which the claim follows inductively. If some word w belongs to the first and not to the second of the finite languages in Step 2 and still $L(G) = L(H)$, then $w = v_i$ for some $i \geq q_2$. (We cannot have $i < m$ because then w would occur twice in the u-sequence.) By (iii), for some $j \geq m$, $v_j <_p w$. By the choice of q_1 and (ii), $v_j \notin L(G)$, which is a contradiction.

Also the conclusion in Step 3 is correct. When entering Step 3, we know that

$$\{u_i |\ 0 \leq i < q_1\} = \{v_i |\ 0 \leq i < q_1\}.$$

The test performed in Step 3 shows that also

$$\{u_i |\ i \geq q_1\} = \{v_i |\ i \geq q_1\}.$$

Termination. If $L(G) \neq L(H)$, a word belonging to the difference of the two languages is eventually detected in Step 2 because the parameter m becomes arbitrarily large. Assuming that $L(G) = L(H)$, we show that the equality is detected during some visit to Step 3. It follows by property (iii) that neither p_1 nor p_2 is increased during successive visits to Step 1. Thus, we only have to show that the procedure cannot loop by producing always the same pair (p_1, p_2). If n is the cardinality of the alphabet, there cannot be $n + 2$ such consecutive visits to Step 1.

Assume the contrary: the same pair (p_1, p_2) is defined at $n + 2$ consecutive visits. We must have $p_1 = p_2$. Otherwise, the larger of the two numbers has to be decreased at the next visit to Step 1, because of property (iii) and the fact that Step 2 was passed after the preceding visit. The same argument shows that p_1 must assume the maximal value $q_1 - m$. If m is the initial mess at the first of our $n + 2$ visits, the sequences $S(G)$ and $S(H)$ contain $n + 1$ segments of length p_1, beginning with u_m and v_m, such that each segment in $S(H)$ is a permutation of the corresponding segment in $S(G)$. Moreover, it is always the same permutation because otherwise, by property (iii), the value of p_1 will be decreased at the next visit to Step 1. This implies, by property (iv), that for some permutation Π and all $j = 0, \ldots, p_1 - 1$, the sequences

$$S(G(p_1, m + j)) \text{ and } S(H(p_1, m + \Pi(j)))$$

are Parikh equivalent. Since termination did not occur in Step 3, there is a j such that the two sequences are not equivalent. Thus, for some $u \in L(G)$ and $v \in L(H)$, we have $\Psi(u) = \Psi(v)$ but $u \neq v$. Since $L(G) = L(H)$, we have also $v \in L(G)$. Hence, $S(G)$ has two words with the same Parikh vector, which contradicts property (ii). □

4.2 D0L sequence equivalence problem and variations

The decidability of SE-D0L was first shown in [CF]. A much simpler proof based on *elementary morphisms*, [ER2], was given in [ER3], a detailed exposition of it appears in [RS1]. Our argument below follows [CK2] and uses two results, the correctness of the so-called *Ehrenfeucht Conjecture* and *Makanin's Algorithm.* We refer the reader to [RS1] and [CK2] for further information concerning the history of SE-D0L and related problems.

SE-D0L has given rise to many important new notions that have been used widely in language theory: *morphic equivalence, morphic forcing, test set, elementary morphism, bounded balance, equality set.* All these notions are related to the problem of studying the equation $g(x) = h(x)$ for a word x and morphisms g and h.

We say that the morphisms g and h defined on Σ are *equivalent* on a language $L \subseteq \Sigma^*$, in symbols $g \equiv_L h$, iff $g(x) = h(x)$ holds for all $x \in L$. The *morphic equivalence problem* for a family $\mathcal{L}$ of languages, [CS1], consists of deciding, given a language L in $\mathcal{L}$ and morphisms g and h, whether or not $g \equiv_L h$. Observe that SE-D0L is a special case of the morphic equivalence problem for the family of D0L languages: two D0L systems G and H with morphisms g and h are sequence equivalent iff g and h are equivalent on $L(G)$ (or on $L(H)$). So we have the special case where $L(G)$ is generated by one of the morphisms whose equivalence on $L(G)$ is to be tested.

Let us go back to the morphic equivalence problem. It would be desirable to have a *finite* set $F \subseteq L$ such that, to test $g \equiv_L h$, it suffices to test $g \equiv_F h$. Formally, we say that a language L is *morphically forced* by its subset L_1 iff, whenever two morphisms are equivalent on L_1, they are equivalent on L. A *finite* subset F of a language L is termed a *test set* for L iff L is morphically forced by F. The *Ehrenfeucht Conjecture* claims that every language possesses a test set. It was shown correct in the middle 80's. [AL] is usually quoted as the first proof but there were several independent ones about the same time – see [CK2], [RS3] for details.

The notion of an *equality set* occurs at least implicitly in earlier algorithms for SE-D0L, [CF], [ER3]. The equality set (also called *equality language*) of two morphisms is defined by

$$E(g, h) = \{x \in \Sigma^* | \ g(x) = h(x)\}.$$

SE-D0L amounts to deciding whether or not one of the languages, say $L(G)$, is contained in $E(g, h)$. This again amounts to deciding the emptines of the intersection between $L(G)$ and the complement of $E(g, h)$. If $E(g, h)$ is regular (implying that also its complement is regular), the latter problem becomes decidable because D0L is contained in E0L for which the emptiness is decidable, and E0L is closed under intersection with regular languages. Therefore, one should aim towards the case that $E(g, h)$ is actually regular. This situation was reached in the proof of [ER3] by using *elementary morphisms*. Essentially, a morphism being elementary means that it cannot be "simplified" by presenting it as a composition going via a smaller alphabet, [ER2].

Assume that the morphisms g and h are equivalent on L. We say that the pair (g, h) has a *bounded balance* on L iff there is a constant C such that

$$||g(x)| - |h(x)|| \leq C$$

holds for all prefixes x of words in L. (We know that $g(w) = h(w)$ holds for words $w \in L$. Therefore, if x is a prefix of w, then one of $g(x)$ and $h(x)$ is a prefix of the other. Having bounded balance means that the amount by which one of the morphisms "runs faster" is bounded by a constant.) For testing morphic equivalence, the property of having bounded balance gives the possibility of storing all necessary information in a finite buffer. Thus, the situation is similar to the equality set being regular.

Consider two disjoint alphabets Σ and N (N is the alphabet of *nonterminals* or *variables*). An *equation* over Σ with unknowns in N is a pair (u, v), usually written $u = v$, where u and v are words over $\Sigma \cup N$. A set T (possibly infinite) of equations is referred to as a *system of equations*. A *solution* to a system T is a morphism $h : (\Sigma \cup N)^* \longrightarrow \Sigma^*$ such that $h(u) = h(v)$, for all $(u, v) \in T$, and $h(a) = a$ for all $a \in \Sigma$. Thus, solutions can be viewed as card(N)-tuples of words over Σ. For instance, the morphism defined by

$$x \longrightarrow a, \quad y \longrightarrow ba, \quad z \longrightarrow ab$$

is a solution of the equation $xy = zx$ over $\{a, b\}$. This solution can also be represented as the triple (a, ba, ab).

Makanin, [Ma], has presented an algorithm for deciding whether or not a given finite system of equations possesses a solution. It is shown in [CK1] how any such algorithm can be extended to concern finite systems of equations and inequalities $u \neq v$. From this follows easily the decidability of the *equivalence problem for two finite systems of equations*, that is, whether or not the systems have the same sets of solutions. This, in turn, leads directly to the decidability of the problem whether or not a *subset of a finite language F is a test set for F*. We are now ready to establish the following main result.

Theorem 4.5. The morphic equivalence problem is decidable for D0L languages. Consequently, SE-D0L is decidable.

Proof. Because the Ehrenfeucht Conjecture is correct, we know that the language $L(G)$ generated by a given D0L system $G = (\Sigma, h, w_0)$ possesses a test set F. We only have to find F effectively. Define the finite languages $L_i, i \geq 0$, by

$$L_0 = \{w_0\}, \quad L_{i+1} = L_i \cup h(L_i).$$

We now determine an integer i_0 such that L_{i_0} is a test set for L_{i_0+1}. Such an integer surely exists because eventually the whole F is contained in some L_i. The integer i_0 can be found because we can decide whether a subset of a finite language is a test set. We claim that L_{i_0} is a test set for $L(G)$, which completes the proof. Indeed, since L_{i_0} is a test set for L_{i_0+1}, also $L_{i_0+1} \setminus \{w_0\}$ is a test set for $L_{i_0+2} \setminus \{w\}$. (Obviously, whenever F' is a test set for L', then $h(F')$ is a test set for $h(L')$.) Consequently, L_{i_0+1} is a test set for L_{i_0+2}. Since "being a test set for" is obviously transitive, we conclude that L_{i_0} is a test set for L_{i_0+2}, from which our claim follows inductively. □

If the alphabet consists of two letters, it suffices to test the first four words in the given D0L sequences in order to decide sequence equivalence. This result is optimal, as shown by the example

$$w_0 = ab, \quad g(a) = abbaabb, g(b) = a, h(a) = abb, h(b) = aabba.$$

This has given rise to the *2n-conjecture*: in order to decide SE-D0L, it suffices to test the first $2n$ words in the sequences, where n is the cardinality of the alphabet. No counterexamples to this conjecture have been found, although the only known bound of this kind is really huge, [ER4].

Theorems 4.4 and 4.5 yield immediately

Theorem 4.6. LE-D0L is decidable.

We are able to establish now also the following very strong result.

Theorem 4.7. The HD0L sequence equivalence problem is decidable.

Proof. Without loss of generality, we assume that the two given HD0L systems are defined by the morphisms f_1 and f_2 and D0L systems $H = (\Sigma, h, w_0)$ and $G = (\Sigma, g, w_0)$, respectively. We consider a "barred copy" $\overline{\Sigma}$ of the alphabet Σ: $\overline{\Sigma} = \{\overline{a} | \ a \in \Sigma\}$. The "barred version" $\overline{w}$ of a word $w \in \Sigma^*$ is obtained by barring each letter. We define three new morphisms, using the descriptive notations $\overline{f_1}$, $\overline{f_2}$ and $h \cup \overline{g}$, as follows:

$$\overline{f_1}(a) = f_1(a), \overline{f_2}(a) = \lambda, (h \cup \overline{g})(a) = h(a) \text{ for } a \in \Sigma,$$

$$\overline{f_1}(\overline{a}) = \lambda, \overline{f_2}(\overline{a}) = f_2(a), (h \cup \overline{g})(\overline{a}) = \overline{g(a)} \text{ for } a \in \Sigma.$$

Clearly, the original HD0L sequences are equivalent iff the morphisms $\overline{f_1}$ and $\overline{f_2}$ are equivalent on the D0L language defined by the morphism $h \cup \overline{g}$ and axiom $w_0\overline{w_0}$. Thus, Theorem 4.7 follows by Theorem 4.5. □

The decidability of the HD0L sequence equivalence is a very nontrivial generalization of the decidability of SE-D0L. For instance, no techniques based on bounded balance can be used because two morphisms may be equivalent on a D0L language L without having bounded balance on L. Our simple argument for Theorems 4.5 and 4.7 is no miraculous "deus ex machina". Two very strong tools (Ehrenfeucht and Makanin) have been used. (We want to mention in this connection that, in spite of their efforts, the editors failed in including in this Handbook a reasonably detailed exposition of Makanin's theory.) For a classroom proof, where all tools are developed from the beginning, the proof based on elementary morphisms, presented in [RS1] or [S5], is still to be recommended.

Some related results are collected in our last theorem. See also [CK2], [Ru2], [Ru3], [Ru4].

Theorem 4.8. The morphic equivalence is decidable for HDT0L languages, and so is the equivalence between D0L and F0L and between D0L and DT0L, as well as the inclusion problem for D0L. The equivalence is undecidable for DT0L, and so is the PD1L sequence equivalence. □

5 L Growth

5.1 Communication and commutativity

This section will discuss *growth functions* associated to sequences generated by L systems. The theory will be presented in an unconventional way, as a discussion between a mathematician, a language theorist and a wise layman. We have two reasons for this unconventional approach. First, there already exist good detailed expositions of the theory in a book form, [SS], [RS1]. In a conventional exposition, we could not do much better here than just repeat, even in a condensed form, what has been already said in [SS] or [RS1]. Secondly, growth functions require considerably more factual knowledge in mathematics than do other parts of the L system theory. Since L systems in general are of interest to a wide audience not otherwise working with mathematics, conventional mathematical formulation might scare away many of the readers of this chapter. While we are convinced that this is less likely to happen with our exposition below, we also hope that our presentation will open new perspectives also for a mathematically initiated reader.

Dramatis personae

Bolgani, a formal language theorist,
Emmy, a pure mathematician,
Tarzan, a lonesome traveller.

Emmy. I often wonder why we mathematicians always write texts incomprehensible practically to everybody. Of course every science has its own terminology and idioms, but our pages laden with ugly-looking formulas are still different. I have the feeling that in many cases the same message could be conveyed much more efficiently using less Chinese-resembling text.

Bolgani. Just think how Fermat got his message across, in a passage that has influenced the development of modern mathematics perhaps more than any passage of comparable length. *Cubum autem in duos cubos, aut quadratoquadra-*

tum in duos quadratoquadratos, et generaliter nullam in infinitum ultra quadratum potestatem in duos ejusdem nominis fas est dividere: cujus rei demonstrationem mirabilem sane detexi. Hanc marginis exiguitas non caperet[1]. Isn't it like beautiful poetry? And think of the philosophical implications. That something is correct is expressed by saying "fas est" – follows the divine law. Not even the gods among themselves can break mathematical laws.

Emmy. It still remains a problem whether some "demonstratio mirabilis" exists for Fermat's Last Theorem. One can hardly imagine that Fermat had in mind anything even remotely resembling the fancy methods of Andrew Wiles. But coming back to poetry: the words sound beautiful but one has to know something in order to appreciate the beauty.

Tarzan. Some basic knowledge is always necessary but in many cases rather little will suffice. Take a passage from Horace, also with references to divine activities.*Pone me pigris ubi nulla campis arbor aestiva recreatur aura, quod latus mundi nebulae malusque Juppiter urget. Pone sub curru nimium propinqui solis, in terra domibus negata: dulce ridentem Lalagen amabo, dulce loquentem*[2]. Certainly it sings much better than Fermat. But still you have to have some background in order to fully appreciate it, a mere understanding of the language is not enough.

Bolgani. I have an idea. Tarzan already read about the basics of L systems. (This refers to Sections 1 and 2 above.) Let us try to explain to him the theory of growth functions, using as few formulas as possible. This might open some interesting perspectives, even if no poetry would come up.

Tarzan. I already know something about the topic. Take, for instance, a D0L system. It generates a sequence of words. One considers the length of each word, as a function of the position of the word in the sequence. This function will be the *growth function* of our D0L system. One starts counting the positions from 0, the axiom having the position 0. This means that every growth function maps the set of nonnegative integers into the same set. Some things are obvious here. The growth function of a PD0L system is monotonously increasing, not necessarily strictly increasing at every step, whereas decrease is possible in the D0L case. Whenever 0 appears as a value of a D0L growth function, then all values afterwards, that is values for greater arguments, are also equal to 0. Whenever we have the empty word in the sequence, then only the empty word appears afterwards. Once dead, always dead. No resurrection is possible in this model of life, be it real or artificial. However, this is the only restriction I have.

[1] "It is improper (by divine right) to divide a cube into two cubes, a fourth power into two fourth powers and, generally, any power beyond a square into two powers with the same exponent as the original. I really found a miraculous proof for this fact. However, the margin is too small to contain it."

[2] "Place me where never summer breeze/ Unbinds the glebe, or warms the trees/ Where ever-lowering clouds appear,/ And angry Jove deforms th'inclement year:// Place me beneath the burning ray,/ Where rolls the rapid car of day;/ Love and the nymph shall charm my toils,/ The nymph, who sweetly speaks and sweetly smiles." (Ode XXII, tr. by P.Francis *Horace, Vol.I*, London, printed by A.J.Valpy, M.A., 1831)

If I can take as many letters as I want, I can get the first million, billion or any number of values exactly as I like: 351, 17, 1934, 1, 1964, 15, 5, 1942,... I just introduce new letters at every step, getting the lengths as I like. Since my alphabet is altogether finite, at some stage there will be repetitions of letters and, accordingly, patterns in the growth sequence. The over-all growth cannot be more than exponential. The length of the ith word is bounded from above by c^{i+1}, where the constant c equals the maximum of the lengths of the axiom and any right-hand side of the rules. This bound is valid for DIL systems as well.

Bolgani. We are dealing here with issues concerning *communication and commutativity.* I think we have already agreed that we can communicate with each other in a less formal manner – even about technically complicated topics. *Commutativity is the key to D0L growth functions.* It also opens the possibility of using strong mathematical tools. Why do we have commutativity? Because the order of letters is completely irrelevant from the point of view of growth. We only have to keep track of the number of occurrences of each letter or, as the saying goes, of the *Parikh vector* of the word. Only the number is significant, we can commute the letters as we like. But we have to know the Parikh vector, it is not sufficient to know the total length of the word. Two different letters may grow very differently.

Emmy. All information we need can be stored in the *growth matrix* of the D0L system. It is a square matrix, with nonnegative integer entries and dimension equal to the cardinality of the alphabet, say n. We have a fixed ordering of the letters in mind, and the rows and columns of the matrix are associated to the letters according to this ordering. The third entry in the second row indicates how many times the third letter occurs on the right side of the rule for the second letter. In this way the matrix gives all the information we need for studying growth.

Take our old friend SQUARES as an example. The rules were $a \longrightarrow abc^2$, $b \longrightarrow bc^2$, $c \longrightarrow c$. The growth matrix is, accordingly,

$$\begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

See that? The first entry in the second row is 0, because there are no a's on the right side of the rule for b. To complete the information, we still need the Parikh vector of the axiom. In this case it is $\Pi = (1, 0, 0)$ since the axiom is a.

So far we have been able to store the information in matrix form. We now come to the point which makes the play with matrices really worthwhile. *One derivation step amounts to multiplication by the growth matrix* if we are only interested in the length sequence. This is a direct consequence of the rule for matrix multiplication. Let us go back to SQUARES and take the word abc^2bc^4 in the sequence, two steps after the axiom. Its Parikh vector is $(1, 2, 6)$. Compute the matrix product $(1, 2, 6)M$. It is another Parikh vector, obtained as follows. You take a column of M, multiply its entries by the corresponding entry of $(1, 2, 6)$, and sum up the products. If your chosen column is the first one, this gives the result $1 \cdot 1+\ 2 \cdot 0+\ 6 \cdot 0 = 1$, and $1 \cdot 1+\ 2 \cdot 1+\ 6 \cdot 0 = 3$ and $1 \cdot 2+\ 2 \cdot 2+\ 6 \cdot 1 = 12$ in the other two cases. Altogether we get the new Parikh vector $(1, 3, 12)$. What does, for instance, the last entry tell us? The third column

$$\begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$$

indicates how many c's each of the letters a, b, c produces. The numbers of occurrences of the letters were $1, 2, 6$ to start with. Thus, the last entry $12 = 1 \cdot 2+\ 2 \cdot 2+\ 6 \cdot 1$ tells how many c's we have after an additional derivation step. Similarly the entries 1 and 3 tell how many a's and b's we have.

There is nothing in our conclusion pertaining to the particular example SQUARES. We have shown that after i derivation steps we have the Parikh vector $\Pi \ \ M^i$. (This holds true also for $i = 0$.) Let η be the n-dimensional column vector with all components equal to 1. The values $f(i)$ of the growth function are obtained by summing up the components of the Parikh vector $\Pi \ \ M^i$,

$$f(i) = \ \Pi \ \ M^i \ \ \eta, \quad i \geq 0.$$

We have derived the basic interconnection with the theory of matrices, and have now all the classical results available, [MT], [HW].

Tarzan. I know some of them. For instance, the Cayley-Hamilton Theorem says that every matrix satisfies its own characteristic equation. If we have n letters in the alphabet, we can express M^n in terms of the smaller powers:

$$M^n = c_1 M^{n-1} + \ldots + c_{n-1} M^1 + c_n M^0,$$

in this case with integer constants c_j. By the matrix representation for the growth function, we can use the same formula to express each growth function value in terms of values for smaller arguments:

$$f(i + n) = c_1 f(i + n - 1) + \ldots + c_n f(i), \ \ i \geq 0.$$

I see it now quite clearly. I can do my arbitrary tricks at the beginning. But once the number of steps reaches the size of the alphabet, I am stuck with this recursion formula.

Bolgani. We will see that many mysteries remain in spite of the formula. But the formula surely gives rise to certain regularities. For instance, the so-called "Lemma of Long Constant Intervals". Assume that f is a D0L growth function and, for every k, there is an i such that

$$f(i) = f(i + 1) = \ldots = f(i + k).$$

Then f is ultimately constant: there is an i_0 such that $f(i) = f(i_0)$ for all $i \geq i_0$.

The matrix representation gives also a very simple method for deciding *growth equivalence*: two D0L systems G and G', with alphabet cardinalities n and n', possess the same growth function iff the lengths of the first $n + n'$ terms are the same in both sequences. The bound is the best possibe. This algorithm is very simple, both as regards the proof of its validity and as regards the resulting procedure. This striking contrast to the difficulty of the D0L sequence equivalence problem is again due to commutativity. Consider an example of language-theoretic interest. Two D0L systems have both the axiom a. The rules of the first system are $a \longrightarrow ab^3$, $b \longrightarrow b^3$, and the rules of the second system:

$$a \longrightarrow acde,\ b \longrightarrow cde,\ c \longrightarrow b^2d^2,\ d \longrightarrow d^3,\ e \longrightarrow bd.$$

You can verify that the first seven numbers in both length sequences are 1, 4, 13, 40, 121, 364, 1093. Hence, the systems are growth equivalent.

Tarzan. The D0L system DEATH$_b$ had the axiom ab^2a and the rules $a \longrightarrow ab^2a$, $b \longrightarrow \lambda$. We noticed that cell death is necessary to generate its language. However, the very simple system with the axiom a^4 and rule $a \longrightarrow a^2$ is growth equivalent, and is without cell death. What is really the role of cell death, that is rules $a \longrightarrow \lambda$, in D0L growth? Does it consist only of making decreases possible? What is the difference between D0L and PD0L growth? If I have a non-decreasing D0L growth function, can I always get the same function using a PD0L system, perhaps at the cost of taking vastly many new letters?

Emmy. That would be too good to be true. The effects of cell death are deeper, also from the growth point of view. The answer to your question is "no", and we still come to the difference between D0L and PD0L growth. At the moment, I would like to dwell on more immediate matters.

The interconnection with matrices, in particular the recurssion formula you are stuck with, open the possibility of using a lot of classical results from the theory of matrices and difference equations. The values $f(i)$ can be expressed as "exponential polynomials" in i, that is, finite sums of terms of the form

$$(\alpha_0 + \alpha_1 i + \ldots + \alpha_{t-1} i^{t-1})\rho^i.$$

Indeed, here ρ is a root, with multiplicity t, of the characteristic equation of M. Maybe you like an example. Take the system with the axiom abc and the rules

$$a \longrightarrow a^2,\ \ b \longrightarrow a^5b,\ \ c \longrightarrow b^3c,$$

yielding

$$M = \begin{pmatrix} 2 & 0 & 0 \\ 5 & 1 & 0 \\ 0 & 3 & 1 \end{pmatrix}$$

Since the matrix is in lower diagonal form, the roots of the characteristic equation are seen immediately: 2 and 1 (double). This results in the expression

$$f(i) = (\alpha_0 + \alpha_1 i) \cdot 1^i + \alpha_2 \cdot 2^i$$

for the growth function. Finally, the coefficients α are determined by considering the first few values (here first three) in the length sequence. Observe that here the axiom becomes important – so far we did not use it at all! The final result is

$$f(i) = 21 \cdot 2^i - 12i - 18.$$

This solves the growth *analysis problem* for any system: we can write down the exponential polynomial representing the growth function of the system. Although the values $f(i)$ are nonnegative integers, the numbers ρ and α are complex numbers, sometimes not even expressible by radicals. Since the growth function is always an exponential polynomial, some growth types can never occur as D0L growth. D0L growth cannot be logarithmic: whenever the growth function is not bounded by a constant, the growth is at least linear. Similarly, nothing faster than exponential or between polynomially bounded and exponential, something like the $2^{\sqrt{n}}$, is possible.

Bolgani. Such in-between growth orders are possible in the DIL case, [K1]. As Tarzan already observed, super-exponential growth is impossible also in the DIL case.

Emmy. Still a few things. The theory says a lot also about the converse problem, *growth synthesis.* We have in mind a function, obtained experimentally or otherwise, and we want to construct a D0L system having the function as its growth function. For instance, we can synthesize any polynomial $p(i)$ with integer coefficients and assuming integer values for all integers $i \geq 0$. ($p(i)$ may assume negative values for non-integral values of i.) We cannot synthesize $i^2 - 4i + 4$ because it represents death at $i = 2$, but $i^2 - 41 + 5$ is OK. It is the growth function of the D0L system with the axiom $a_1^3 a_2 a_3$ and rules

$$a_1 \longrightarrow \lambda, a_2 \longrightarrow a_4, a_3 \longrightarrow a_5, a_4 \longrightarrow a_6, a_5 \longrightarrow \lambda,$$

$$a_6 \longrightarrow ad, a \longrightarrow abc^2, b \longrightarrow bc^2, c \longrightarrow c, d \longrightarrow d.$$

Observe that our old friend SQUARES is lurking in the stomach of this new D0L creature, where Tarzan's technique of disposable letters is also visible.

Making use of periodicities, we can also synthesize several polynomials in a single D0L system. Explicitly, this technique of *merging* means the following. Assume that we have some synthesizable polynomials, say $p_0(i)$, $p_1(i)$, $p_2(i)$, of the same *degree.* Then we can construct a D0L system whose growth function satisfies, for all i,

$$f(3i) = p_0(i), \quad f(3i+1) = p_1(i), \quad f(3i+2) = p_2(i).$$

It is essential that the polynomials are of the *same degree*; in general, the quotient of two mergeable functions should be bounded from above by a constant.

Recall the matrix representation of D0L growth functions, $f(i) = \Pi\ M^i\ \eta$. Functions of this form, where M is a square matrix and Π and η are row and column vectors of the same dimension, all with *integer* entries, are termed *Z-rational*. (This terminology is quite natural, see [SS].) If the entries are *nonnegative* integers, the function is termed *N-rational*. *D0L growth functions* are a special case of N-rational functions: it is required that all entries in ρ are equal to 1. *PD0L growth functions* are a further special case: it is also required that every row of M has a positive entry. The part of the matrix theory that comes into use here is customarily referred to as the Perron-Frobenius theory.

5.2 Merging and stages of death

Bolgani. We now face the challenging task of discussing the differences between the four classes of functions Emmy just introduced. By definition, the classes constitute an increasing hierarchy from PD0L growth functions (smallest class) to Z-rational functions (biggest class). That the inclusions between the classes are strict is also obvious. N-rational functions cannot assume negative values, as Z-rational functions clearly can. 0, 1, 1,... is an N-rational but not a D0L length sequence. Decrease is possible in D0L but not in PD0L length sequences. Such examples can be viewed as trivial, forced by definitions. However, in each case we can show the strictness of the inclusion also by a nontrivial example. Moreover, mathematically very nice characterizations are known for each of the four classes of functions. Emmy should say more about it, let me just give the following rough idea.

The difference between Z-rational and N-rational functions stems from the idea of *merging* or *mergeability*. An N-rational sequence can always be obtained by merging sequences with a "clean dominant term", whereas this is not necessarily possible for Z-rational sequences. By a "clean dominant term" I mean that the expression for an individual member of the sequence has a term $\alpha i^k \rho^i$, where α, ρ are constants and k is a nonnegative integer, completely determining the growth, that is, the member is asymptotically equal to this term. One can say that the *growth order* of the term, as a function of its position i, equals $i^k \rho^i$. As you recall, ρ is a root of the characteristic equation and can in general be a complex number but is real in a clean dominant term.

The difference between D0L and PD0L growth functions can be said to happen at the *primary stage of death*: death affected by the rewriting model itself. We will see that mathematically this difference amounts again to the difference between Z-rational and N-rational functions. Finally, the difference between N-rational and D0L comes from the fact that a *secondary stage of death* is possible for the former. Because some entries of the final vector ρ may equal 0, letters can be erased after the actual D0L sequence has been created. We can also speak of death occurring in the system, contrasted with death caused by outside forces.

Tarzan. I can see the difference between N-rational and D0L very clearly. The secondary stage of death can be viewed also as the *invisibility* of some letters: the letters themselves cannot be seen, although they sometimes produce visible offsprings. Take the very simple D0L system with the axiom ac and rules

$$a \longrightarrow b, \ \ b \longrightarrow a, \ \ c \longrightarrow d^2, \ \ d \longrightarrow c.$$

Take

$$\eta = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

so only a and d are kept visible. See? In the underlying D0L sequence a and b alternate, similarly c and d, and the latter also affect exponential growth. But because of death caused by outside forces, the exponential growth is visible only at even stages, the length sequence being 1, 2, 1, 4, 1, 8, ... This is obtained by merging a constant and an exponential sequence. Only sequences with the same growth order can be merged in a D0L system. This follows because clearly $\frac{f(i+1)}{f(i)}$ is bounded from above by a constant, an obvious choice being the maximal length of the right sides of the rules. It is clear by definition that N-rational functions coincide with HD0L growth functions. It is also easy to prove directly that if all entries of η are *positive* then the N-rational function is, in fact, a D0L growth function: a D0L system can be constructed by use of a dummy "filling letter" that takes care of the components of η exceeding 1. This means that the *secondary stage of death*, caused by the entries 0 in η, is really the essential characteristic of N-rational functions, when contrasted with D0L growth functions. The secondary stage of death can be used to eliminate the primary stage, HPD0L growth functions are the same as HD0L ones. This should be easy enough to establish directly; it is of course also a consequence of the characterization result, Theorem 2.3.

Emmy. Essentially everything has already been said. I still want to elaborate the fundamental issues. Take first the role of N-rational functions among Z-rational ones; what Bolgani talked about clean dominant terms. As we have seen, an individual member of a Z-rational sequence can be expressed as an exponential polynomial. Such an exponential polynomial has a part determining its growth order: in the asymptotic behavior only this part is significant. This part is determined by terms $i^t \rho^i$, where ρ has the greatest possible modulus and t is the greatest (for this modulus). Thus, among the roots of the characteristic equation of the growth matrix, we are interested in those with the greatest modulus. There may be many of them since in general the ρ's are complex numbers.

Tarzan. What Bolgani spoke about clean dominant terms means of course that in the decomposition, in the parts to be merged, the dominant terms are real numbers, implying that the original ρ's are real numbers multiplied by roots of unity. This is equivalent to saying that some integral power of ρ is real. I have seen that the term *rational in degrees* is used for such ρ's: the argument of ρ is a rational multiple of π – now the classical π, not your initial vector!

Emmy. You have seen many things during your trips, also from our fields. It was shown in [Be] that the phenomenon described by you happens for ρ's in the N-rational case, the converse result being due to [So2]. Parts of this theory go back to Kronecker. What would be an example? Here it is good that you know some classical analysis. Consider a number α such that

$$\cos 2\pi\alpha = 3/5, \quad \sin 2\pi\alpha = 4/5.$$

Then α must be irrational: the imaginary part of $(3+4i)^t$ is always congruent to 4 modulo 5. Thus, we have here irrationality in degrees. Along these lines one can show that, for instance, the function assuming positive values, defined by

$$f(2m) = 30^m, \;\; f(2m+1) = 25^m \cos^2 2\pi m\alpha$$

is Z-rational but not N-rational. It is an interesting detail that Z-rational and N-rational functions coincide in the polynomially bounded case, [SS].

The difference between Z-rational and N-rational functions turns out to be decisive also in the characterization of PD0L growth functions among D0L growth functions, [So1]. For any D0L growth function $f(m)$, the differences $d(m) = f(m+1) - f(m)$ constitute a Z-rational function. The function $f(m)$ is a PD0L growth function iff the differences $d(m)$ constitute an N-rational function. (We exclude here the empty word as the axiom.) This result gives the possibility of using any Z-rational function $g(m)$ assuming nonnegative values and being not N-rational to obtain a non-decreasing D0L growth function that is not a PD0L growth function. The idea is to merge the sequences R^m and $R^m + g(m)$, where R is a large enough integer. If you do not like the cosine present in the preceding example, I will give you another, still closely related example:

$$f(2m) = 10^m, \;\; f(2m+1) = 10^m + m \cdot 5^m + (3+4i)^m + (3-4i)^m.$$

Thus, although the total size of this D0L creature keeps increasing all the time, it must have cells which die immediately! This holds independently of the number of cell types (letters) we are using.

Bolgani. The D0L system of this example has an estimated 200 letters. You are not likely to find such examples just by playing around with D0L systems. The method works here in a backward manner. You begin with some roots ρ, algebraic numbers not rational in degrees. You then construct your characteristic equation, and then the matrix. When you have expressed the resulting Z-rational sequence as the difference of two PD0L sequences, you start to be in business and can work with L systems from that point on. These new aspects have brought with themselves several questions and decision problems not normally asked and studied in classical mathematics. For instance, how to decide of a given polynomial equation with integer coefficients whether or not it has a root outside the unit circle? This information can be used to decide whether or not growth is exponential.

Emmy. I failed to mention that every Z-rational function can be expressed as the difference of two PD0L growth functions. This result, with PD0L replaced by N-rational, was known already before the theory of L systems, and the strengthening is an application of the merging techniques, [SS]. We have clarified just about everything concerning the four classes of functions. We still have to be explicit about the relation between D0L growth functions and N-rational functions. Tarzan already observed that $\frac{f(i+1)}{f(i)}$ is bounded from above by a constant whenever f is a D0L growth function. Also the converse holds: whenever f is an N-rational function assuming positive values and there is a constant c such that $\frac{f(i+1)}{f(i)} \leq c$ holds for all i, then f is a D0L growth function. The proof of this converse, if presented in detail, is the most involved among the proofs of the results concerning growth functions we have been talking about. The most elegant approach is the use of generating functions, originally due to [KOE] and presented in detail in [RS1]. Another consequence of this converse is that a finite number of D0L growth functions with the same growth order can always be merged into one D0L growth function.

Bolgani. Although we have a basic understanding of D0L growth functions, a lot of open problems still remains. We should still talk about them. D0L systems have quite amazing features and capabilities behind their very simple and innocent-looking outer appearance. Let me mention an example.

Recall the Lemma of Long Constant Intervals. If a function stays constant in arbitrarily long intervals then it must be ultimately constant in order to qualify as a D0L growth function. A D0L length sequence can stagnate only for as many consecutive steps as the alphabet size indicates. If we see only the length sequence, we do not know the alphabet size but we know it must have some specific size. It is very natural to expect that a statement analogous to the Lemma of Long Constant Intervals holds true also for intervals, where the length increases strictly at every step. Indeed, when I was first asked this question, I felt sure when I gave a positive answer, thinking that it was only a matter of straightforward matrix calculation to prove the result. But not at all! The opposite result holds true, as shown in [K3]. There is a D0L system SURPRISE whose growth function f satisfies $f(i) < f(i-1)$ for infinitely many values of i. Thus, SURPRISE shrinks every now and then, and this continues forever. Thinking about the periodicities in D0L sequences, you would not expect SURPRISE to have arbitrarily long phases of strict growth. But it does. For each k, there is an i such that

$$f(i) < f(i+1) < f(i+2) < \ldots < f(i+k).$$

No matter how long a phase you need for SURPRISE to become bigger and stronger during the whole phase, a proper timing gives you such a phase!

5.3 Stagnation and malignancy

Tarzan. Cancer is sometimes identified with exponential growth. I would like to call growth in a D0L system *malignant* if, for some rational $t > 1$ and $i_0 \geq 0$,

$$f(i) > t^i \text{ whenever } i \geq i_0.$$

We know that, for D0L growth functions but not in the DIL case, this is equivalent to saying that there is no polynomial p such that $f(i) \leq p(i)$ holds for all i.

Suppose I want to play a doctor for creatures of artificial life, modeled by D0L systems. One of them comes to me and I have to tell whether or not he/she will eventually grow cancer. How can I do it? I know how he/she looked as an infant (the axiom) and the rules for the cell development. It is clear to me that if there is some letter that derives in some number of steps two copies of itself and possibly some other letters as well, then the growth is malignant. Here I take into account only letters actually occurring in the sequence, a condition that can be easily tested. So the occurrence of such an *expanding* letter is sufficient for malignancy but is it also necessary? Could the creature develop cancer without having any expanding cell? Probably not. If the condition is also necessary and if I can test of any letter whether or not it is expanding, then I can diagnose all creatures and tell each of them either good or bad news!

Bolgani. You sure can. The necessity of your condition is fairly easy to establish by purely language-theoretic means. If there are no expanding letters, a growth-equivalent system can be constructed having letters $a_1, \ldots, a_n$ such that the right side of the rule for a_i has at most one a_i and no letters a_j with $j < i$. The growth in this new system is bounded by a polynomial of degree $n - 1$. The system SQUARES is of this type with $n = 3$. In the early years of L systems D0L systems of this type were referred to as systems with "rank". Essentially, the rank r is equivalent with polynomial growth of degree $r - 1$.

How to test whether a given letter a is expanding? The following algorithm works. Studying the rules, form a sequence of sets $S_0, S_1, S_2, \ldots$ Each set S_i will be a subset of a finite set T, consisting of all letters of the given D0L system, as well as of all unordered pairs of letters, pairs (b, b) being included. Explicitly, S_0 consists of the pair (a, a). For all i, S_{i+1} is the union of S_i and S'_i, where S'_i is obtained as follows. Take an element y of S_i. If y is a single letter c appearing on the right side of the rule for b, include b to S'_i. Assume next that y is a pair (b, c). Then we include the letter d to S'_i if both b and c (or b twice if $b = c$) occur on the right side of the rule for d. Finally, we include the pair (d, e) to S'_i if the right sides of the rules for d and e, put together, contain both b and c.

The sets S_i form an increasing sequence of subsets of a finite set T. There must be an index j such that $S_{j+1} = S_j$. The letter a is expanding exactly in case it belongs to S_j. Indeed, we begin in S_0 with the situation where two a's occur, and investigate backwards all possibilities where they might have come from. In S_j we have reached the point where there are no more possibilities. Since $S_{j+1} = S_j$, nothing new can come up later. So it is decisive whether we have reached a single a latest by S_j. Whenever we reach a single a, we stop. This is bad news for our creature. If we reach S_j without seeing a single a, it is not yet good news. Every other letter has to be tested in the same way.

Tarzan. In the system DEATH$_b$, a is already in S_1, so it is immediately bad news. If you diagnose b in FIB, you get both (a, a) and (a, b) in S_1 and, hence, you get b in S_2. The letter a is also expanding but you have to go up to S_4 to reach bad news. As you described D0L systems with rank, it seems that for them no diagnosis is needed: it follows by the very definition that the system has no expanding letters. I can immediately tell the good news to SQUARES.

Emmy. The diagnosis can also be based on powers of the growth matrix, [K2]. You compute powers M^i, $i = 1, 2, \ldots$. Whenever you find a diagonal element greater than 1, you diagnose malignancy. If you have reached the value $i = 2^n + n - 1$, where n is the dimension, and have found only 0's and 1's in the main diagonal, you may tell the good news.

Tarzan. Recall our friends DAY-PAL and NIGHT-PAL. The growth matrix of the former has the diagonal entry 2, corresponding to the letter b. The letter b is reachable from any nonempty axiom, in fact all letters eventually become b's. Thus, it is bad news for any DAY-PAL(w).

I would like to talk about another notion. Bolgani used the term "stagnation" in connection with the Lemma of Long Constant Intervals. Let us say that a D0L sequence *stagnates* if it has two words w_t and w_{t+p}, $p > 0$, with the same Parikh vector. Clearly, this implies that

$$f(i) = f(i + p) \text{ for all } i \geq t,$$

that is, the lengths start repeating periodically with the period p after the threshold t. The condition is certainly decidable. For instance, we decide whether the language is finite. Every NIGHT-PAL(w) stagnates. Any axiom becomes a power of c in two steps.

What about a much weaker requirement? A D0L sequence *stagnates momentarily* if $f(t) = f(t+1)$ holds for some t. I can decide momentary stagnation for PD0L sequences. I just look at the subalphabet Σ' consisting of letters mapped to a letter by the morphism. Momentary stagnation is equivalent to the fact that some subset of Σ' appears in the sequence of minimal alphabets. Since the latter sequence is ultimately periodic, I can decide the fact. But things are different in the general D0L case, and momentary stagnation can take place in many ways. It sure must be decidable but how can I decide it?

Bolgani. You got me. This happens to be a celebrated open problem. Given a D0L growth function f, one has to decide whether or not $f(t) = f(t + 1)$ holds for some t. The term "constant level" had also been used, instead of your “momentary stagnation”. The following formulation is equivalent in the sense that an algorithm for either one of the problems can be converted into an algorithm for the other problem. Given two PD0L growth functions f and g, decide whether or not $f(t) = g(t)$ holds for some t. An algorithm is known in both cases, [SS], if “for some t” is replaced by “for infinitely many values of t”.

Emmy. Also the following is an equivalent formulation for the problem of momentary stagnation. Given a square matrix M with integer entries, decide whether or not the number 0 appears in the upper right-hand corner of some power of M. So far all attempts to determine an upper bound for the exponents to be tested have failed. The problem is decidable for $2\times$ 2 matrices. Examples are known of 3×3 matrices where you have to go to the exponent 50 to find the first 0. Again, this problem is decidable if you ask for infinitely many 0s instead of just one 0.

Bolgani. The following problems of *proper thinness* and *continuing growth* are more involved than the problem of momentary stagnation, in the sense that an algorithm for either one of them yields an algorithm for the problem of momentary stagnation but not necessarily vice versa, [PaS], [KS]. Decide whether or not a given D0L language has two words of the same length. (A language is called *properly thin* if all its words are of different lengths. This is a special case of slenderness.) Decide whether or not a given D0L growth function f satisfies the inequality $f(i) \leq f(i+1)$ for all i.

Tarzan. The whole theory of growth functions can apparently be extended to concern DT0L systems as well. Take the system PAL, with some specific axiom. Once we have fixed an order of the tables T_n and T_d, that is a word over the alphabet $\{n, d\}$, we end up with a unique word. From the growth point of view we get a mapping f of the set of words over the alphabet of tables, in the example $\{n, d\}$, into the set of nonnegative integers. In the D0L case the alphabet of tables consists of one letter only. In the DT0L case we can ask basically the same questions as before. Growth equivalence makes sense only if the table alphabets are of the same cardinality. Momentary stagnation takes place if there is a word x and letter a such that $f(xa) = f(x)$. We are now dealing with several growth matrices, one for each letter of the table alphabet; M_n and M_d for PAL. We get the expression $\Pi M_d M_n M_d M_n \rho$ for the length of the DAY-AND-NIGHT-PAL after 48 hours – you understand what I mean. One can also start with a fixed DT0L system, for instance PAL, and investigate what kind of growth phenomena are possible or likely to happen.

Emmy. One can also introduce Z-rational and N-rational functions as before, now the domain of the functions will be some Σ^* instead of N. From the point of view of power series, [SS], [KS], this means several noncommuting variables. Several undecidability results are obtained by a reduction to Hilbert’s Tenth Problem, for instance, the undecidability of the problems of momentary stagnation and continuing growth. On the other hand, growth equivalence can be shown decidable by a nice argument using linear spaces, [RS1].

Tarzan. It seems to me that you can be proud of many nice and certainly nontrivial results. Many challenging problems remain open. After some progress in areas such as artificial life you surely will ask entirely new kinds of questions about L growth. I have to go.

6 L codes, number systems, immigration

6.1 Morphisms applied in the way of a fugue

In the remainder of this chapter, we will present some recent work dealing with L systems. Two problem areas will be presented in more detail in Sections 6 and 7, whereas the final Section 8 will run through results in different areas in a rather telegraphic way. Unavoidably, the choice of material reflects at least to some extent our personal tastes. However, we have tried to choose material significant beyond L systems, as well as to present techniques showing the power of language-theoretic methods in completely different areas, a typical example being the results concerning *number systems* presented below.

The purpose of this Section 6 is to present some recent work in L systems, dealing with several developmental processes that have started at different times. One can visualize the different processes as chasing one another like the tunes in a fugue. We will restrict our attention to D0L processes. In fact, apart from some suggestions, very little work concerning other types of L systems has been done in this problem area, although most questions can be readily generalized to concern also other types of L systems. The issues discussed below will concern also generalized number systems, combinatoricss on words, codes and cryptography.

We consider again morphisms $h : \Sigma^* \longrightarrow \Sigma^*$. The morphisms can be used as an *encoding* in the natural way: words w over Σ are encoded as $h(w)$. If h is *injective*, *decoding* will always be unique. This is not the case for the morphism h defined by

$$h(a) = ab \ \ , h(b) = ba, \ \ h(c) = a.$$

The word aba can be decoded both as ac and cb because h is not injective. Because of unique decodability, injective morphisms are referred to as *codes*. There is a chapter below in this Handbook dealing with the theory of codes. Usually codes are defined as sets of words rather than morphisms. For finite codes, our definition is equivalent to the customary definition in the following sense. A morphism h is a code iff the set $\{h(a) | \ a \in \Sigma\}$ is a code, provided h is *non-identifying*, that is, $a \neq b$ implies $h(a) \neq h(b)$ for all letters a and b. We still repeat the decodability aspect in terms of cryptography. The morphism h being a code means that every "cryptotext" w' can be "decrypted" in at most one way, that is, there is at most one "plaintext" w that is "encrypted" by h into w': $h(w) = w'$. Most "monoalphabetic" cryptosystems in classical cryptography are codes in this sense.

We now come back to the morphism $h : \Sigma^* \longrightarrow \Sigma^*$ (not necessarily injective) and apply it in the "fugue way". This means that h is applied to the first letter, h^2 to the second, h^3 to the third, and so on, the results being catenated. This gives rise to a mapping $\overline{h} : \Sigma^* \longrightarrow \Sigma^*$, referred to as the *L associate* of h. We now give the formal definition.

Definition. Given a morphism $h : \Sigma^* \longrightarrow \Sigma^*$, its *L associate* $\overline{h}$ is defined to be the mapping of Σ^* into Σ^* such that always

$$\overline{h}(a_1 a_2 \dots a_n) = h(a_1)h^2(a_2)\dots h^n(a_n),$$

where the a's are (not necessarily distinct) letters of Σ. By definition, $\overline{h}(\lambda) = \lambda$. The morphism h is termed an *L code* iff its L associate is injective, that is, there are no distinct words w_1 and w_2 such that $\overline{h}(w_1) = \overline{h}(w_2)$. $\Box$

The L associate $\overline{h}$ is rather seldom a morphism itself. In fact, $\overline{h}$ is a morphism exactly in case h is idempotent, that is, $h^2 = h$. Also the equation $h\overline{h} = \overline{h}h$ is not valid in general, which makes many problems concerning L codes rather involved. Consider the classical "Caesar cipher" h affecting a circular permutation of the English alphabet:

$$h(a) = b, h(b) = c, \dots, h(y) = z, h(z) = a.$$

For the L associate we obtain, for instance,

$$\overline{h}(aaaa) = bcde, \;\; \overline{h}(dcba) = eeee.$$

The L associate is not a morphism but h is both a code and an L code, satisfying the equation $h\overline{h} = \overline{h}h$. The proof of the next result is straightforward.

Theorem 6.1. Every code is an L code but not vice versa. $\Box$

A significant class of L codes that are not codes results by considering *unary* morphisms. By definition, a morphism h is *unary* iff there is a specific letter a such that $h(b)$ is a power of a for every letter b. (Unary morphisms should not be confused with unary L systems; for the former, the alphabet Σ may still contain several letters.) Consider the unary morphism h defined by

(*) $$h(a) = a^2, \;\; h(b) = a.$$

Clearly, h is not a code, for instance, $h(bb) = h(a)$. However, h is an L code. Indeed, this follows by considering *dyadic representations* of integers, with the digits 1 and 2. Compute the exponents of a in $\overline{h}(aaa)$, $\overline{h}(baaba)$, $\overline{h}(bbbb)$:

$$2\cdot 2^0 + 2\cdot 2^1 + 2\cdot 2^2, \;\; 1\cdot 2^0 + 2\cdot 2^1 + 2\cdot 2^2 + 1\cdot 2^3 + 2\cdot 2^4, \;\; 1\cdot 2^0 + 1\cdot 2^1 + 1\cdot 2^2 + 1\cdot 2^3.$$

Observe that the original letters b and a behave exactly as the digits 1 and 2, respectively, in the *dyadic representation* of positive integers. For instance, *baaba* represents the number 53, written in dyadic notation $21221 = 2\cdot 2^4 + 1\cdot 2^3 + 2\cdot 2^2 + 2\cdot 2^1 + 1\cdot 2^0$.

See [S1] for an exposition concerning *n-adic* and *n-ary* number systems. In both cases n is the base of the representation, but the digits are 1, 2, ...,n in the n-adic and 0, 1,..., $n-1$ in the n-ary representation; 1, 2 in dyadic, and 0,1 in binary representation. The presence of 0 among the digits renders n-ary systems ambiguous, because an arbitrary number of initial 0's can be added to any word without changing the represented number. n-adic representation is unambiguous. There is a one-to-one correspondence between positive integers and nonempty words over $\{1, 2\}$ when the latter are viewed as dyadic representations. Normally in number systems the leftmost digit is defined to be the most significant one, and we will follow this practice below. The definition of an L associate, which is the natural one because words are normally read from left to right, makes the rightmost letter most significant because there the morphism is iterated most. Thus, actually the mirror image 12212 of the dyadic word 21221 corresponds to the original word *baaba*. This technical inconvenience (of going from words to their mirror images) is irrelevant as regards ambiguity. The morphism h defined by (*) is an L code. If there were two distinct words w and w' such that $\overline{h}(w) = \overline{h}(w')$, then the mirror images of w and w' would be (interpreting b as the digit 1 and a as the digit 2) two distinct dyadic representations for the same number, which is impossible.

Similarly, one can associate a number system to any unary morphism. The morphism is an L code iff the number system is unambiguous. This quite remarkable interconnection between number systems and L systems will now be presented more formally.

Definition. A *number system* is a $(v + 1)$-tuple

$$N = (n, m_1, \ldots, m_v)$$

of positive integers such that $v \geq 1$, $n \geq 2$, and $1 \leq m_1 < m_2 < \ldots < m_v$. The number n is referred to as the *base* and the numbers m_i as *digits.* A nonempty word $m_{i_k} m_{i_{k-1}} \ldots m_{i_1} m_{i_0}$, $1 \leq i_j \leq v$ over the alphabet $\{m_1, \ldots, m_v\}$ is said to *represent* the integer

$$[m_{i_k} \ldots m_{i_0}] = m_{i_0} + m_{i_1} n + m_{i_2} n^2 + \ldots + m_{i_k} n^k.$$

The set of all represented integers is denoted by $S(N)$. A set of positive integers is said to be *representable by a number system* (briefly, *RNS*), if it equals the set $S(N)$ for some number system N. Two number systems N_1 and N_2 are called *equivalent* if $S(N_1) = S(N_2)$. A number system N is called *complete* if $S(N)$ equals the set of all positive integers, and *almost complete* if there are only finitely many positive integers not belonging to $S(N)$. A number system is termed *ambiguous* if there are two distinct words w_1 and w_2 over the alphabet $\{m_1, \ldots, m_v\}$ such that $[w_1] = [w_2]$. Otherwise, N is termed *unambiguous.* An RNS set is termed *unambiguous* if it equals $S(N)$, for some unambiguous number system N. Otherwise, it is termed *inherently ambiguous.*□

Number systems as defined above are often referred to as "generalized number systems". They are more general than the customary notion because the number and size of the digits is independent of the base. Some integers may also have several representations or none at all. On the other hand, we have excluded the digit 0 (ordinary binary and decimal systems do not qualify) because it immediately induces ambiguity. For each $n \geq 2$, the number system $N = (n, 1, 2, \ldots, n)$ is complete and unambiguous. It is customarily referred to as the *n-adic* number system.

Let $h : \Sigma^* \longrightarrow \Sigma^*$ be a unary morphism. Thus, there is a letter $a \in \Sigma$ such that all h-values are powers of a. Assume that $h(a) = a^n$, $n \geq 2$. The *associated number system* is defined by

$$N(h) = (n, m_1, \ldots, m_v),$$

where the m_i's are the exponents, arranged in increasing order, in the equations $h(b) = a^{m_i}$ where b ranges over the letters of Σ. (We assume that h is non-erasing and non-identifying. The assumptions about h are needed to ensure that $N(h)$ is indeed a number system. On the other hand, if $h(a) = a$ or h is erasing or identifying, it is not an L code.) Observe that the base is always among the digits in a number system $N(h)$. By the above discussion, the following theorem is an immediate consequence of the definitions.

Theorem 6.2. A unary morphism is an L code iff the associated number system is unambiguous. $\square$

L codes were introduced in [MSW2], where also the basic interconnections with number systems were discussed. The ambiguity problem for number systems was shown decidable in [Ho1]. The theory of generalized number systems was initiated in [CS2]. We focus here our attention to basic issues and issues connected with L codes. The reader is referred to [Ho2] – [Ho9] for related problems and some quite sophisticated results (concerning, for instance, regularity, degrees of ambiguity, changes of the base and negative digits). The next theorem summarizes some basic results from [MSW2] and [CS2].

Theorem 6.3. A number system $N = (n, m_1, \ldots, m_v)$ is ambiguous if $v > n$. N is unambiguous if the digits m_j lie in different residue classes modulo n. No finite set is RNS, whereas every cofinite set is RNS, Every union of some residue classes modulo n, $n \geq 2$, is RNS. Consequently, both even and odd numbers form RNS sets. The former is unambiguous, whereas the latter is inherently ambiguous. There is an RNS set possessing representations with different bases m and n such that it has an unambiguous representation with m, whereas every representation with the base n is ambiguous. There are a 0L systems G and a DT0L system G_1, both with the alphabet $\{a, b\}$ such that

$$L(G) = L(G_1) = \{b\} \cup \{ba^i | \ i \in S(N)\}.\square$$

Equivalence is undecidable for 0L systems, whereas it is decidable for U0L systems (see Theorems 4.2 and 4.3). The last sentence of Theorem 6.3 gives a somewhat stronger decidability result. This follows by the decidability results for number systems, discussed still in Section 6.2.

Is an arbitrary given morphism an L code? At the time of this writing, the decidability of this problem is open in its general form, There is every reason to believe the problem to be decidable, especially because the remaining class of morphisms with an open decidability status seems rather small. For unary morphisms, the decidability follows by Theorem 6.2. The decidability was established for *permutation-free* morphisms in [Ho4], that is, for morphisms h such that h permutes no subalphabet Σ_1 of Σ. The next theorem gives the decidability result in its strongest known form, [Ho7].

Given a morphism $h : \Sigma^* \longrightarrow \Sigma^*$, we call a letter a *bounded* (with respect to h) if there is a constant k such that $|h^n(a)| \leq k$ holds for all n. Otherwise, a is said to be *growing.* A letter a is *pumping* if $h^n(a) = uav$ holds for some n and words u and v such that uv is nonempty but contains only bounded letters.

Theorem 6.4. It is decidable whether or not a morphism for which no letter is pumping is an L code. $\square$

6.2 An excursion into number systems

This subsection deals exclusively with number systems. We feel that such a brief excursion is appropriate to show the versatility of L systems. The work dealing with L codes has led to problems dealing with the representation of positive integers in arbitrary number systems. Typical questions concern the equivalence and ambiguity of number systems. In spite of their fundamental number-theoretic nature and also in spite of the fact that the representation of integers is fundamental in the theory of computing, very little was known about the solution of such problems before the interconnection with L codes was discovered. Many of the results below are interesting also withing a general language-theoretic setup: no other than a language-theoretic proof is known for these number-theoretic results. Our exposition will be illustrated by some examples.

Consider first the number system $N_1 = (2, 2, 3, 4)$. We claim that $S(N_1)$ consists of all positive integers that are not of the form $2^k - 3$, for some $k \geq 2$. (Thus, 1, 5, 13, 29, 61 are the first few numbers missed.) To show that no number of this form is in $S(N_1)$ we proceed indirectly. Let $x = 2^k - 3$ be the smallest such number in $S(N_1)$, and consider the representation $[a_1 \dots a_m] = x$. We must have $m \geq 2$ and $a_m = 3$ because, otherwise, the represented number is even. But now obviously $[a_1 \dots a_{m-1}] = 2^{k-1} - 3$, contradicting the choice of x. On the other hand, for any $k \geq 1$, an arbitrary integer x satisfying $2^{k+1} - 2 \leq x \leq 2^{k+2} - 4$ is represented by some word of length k, the upper and lower bounds being represented by 2^k and 4^k, respectively. Thus, our claim concerning $S(N_1)$ follows. The system N_1 is ambiguous: $[32] = [24] = 8$. The first sentence of Theorem 6.3 can be used to show that $S(N_1)$ is inherently ambiguous. In the dyadic number system (2, 1, 2), $S(N_1)$ is represented by all words over $\{1, 2\}$ that are not of the form 2^i1, for some $i \geq 0$. (Thus, a regular expression can be given for the set of words representing $S(N_1)$ in dyadic notation. A general statement of this fact is contained in the Translation Lemma below.)

The number system $N_2 = (2, 1, 4)$ is unambiguous, by the second sentence of Theorem 6.3. We claim that $S(N_2)$ equals the set of numbers incongruent to 2 modulo 3. Indeed, all numbers in $S(N_2)$ are of this type. This is clearly true of numbers represented by words of length 1 over the digit alphabet. Whenever x is congruent to 0 (resp.1) modulo 3, then both $2x+1$ and $2x+4$ are congruent to 1 (resp.0) modulo 3. Hence, by induction, every number in $S(N_2)$ is incongruent to 2 modulo 3. That all such numbers are in $S(N_2)$ is again seen indirectly. If $x = 3k$ or $x = 3k + 1$ is the smallest such number outside $S(N_2)$, we can construct by a simple case analysis a still smaller number outside $S(N_2)$.

Very simple number systems can lead to tricky situations not yet clearly understood. Consider, for $k \geq 3$, the number system $N(k) = (2, 2, k)$. When is $N(k)$ unambiguous? This happens if k is odd or if $k = 2m$ with an even m. The remaining case is not so clear. The first odd values of m yielding an unambiguous $N(K)$ are: 11, 19, 23, 27, 35, 37, 39, 43, 45, 47, 51, 53, 55, 59, 67, 69, 71, 75, 77, 79, 83, 87, 89, 91, 93, 95, 99. [MSW2] contains a more comprehensive theory about this example.

We now introduce a tool basic for decidability results. It consists of viewing the sets $S(N)$ as regular languages.

Translation Lemma. Given a number system $N = (n, m_1, \ldots, m_v)$, a regular expression $\alpha(N)$ over the alphabet $\{1, \ldots, n\}$ can be effectively constructed such that the set of words in the regular language denoted by $\alpha(N)$, when the words are viewed as n-adic numbers, equals $S(N)$. $\Box$

The reader is referred to [CS2] for details of the proof. The argument is a very typical one about regular languages and finite automata. It is easy to get a rough idea. A finite automaton (sequential machine) translates words $w\#$, where $w \in \{m_1, \ldots, m_v\}^+$, into words over $\{1, \ldots, n\}$. The word w is viewed as a number represented according to N in reverse notation, and the translate will be the representation of the same number in reverse n-adic notation. The "carry" (due to digits exceeding the base) is remembered by the state of the automaton. (A reader not familiar with this terminology is referred to Chapter 2 of this Handbook.) It turns out that 2 $\max(n, m_v)$ states will suffice. In one step, when reading the letter j in the state i, the automaton outputs j' and goes to the state i', where i' and j' are unique integers satisfying

$$i + j = j' + i'n, \quad 1 \leq j' \leq n.$$

When reading the boundary marker $\#$ in the state i, the automaton produces the output i in reverse n-adic notation. This is the only step, where the output may consist of more than one letter.

As an example, consider the number system $N = (2, 13, 22)$. We use the notation $a = 13, b = 22$ to avoid confusion. The computation of the automaton for the input $abaa\#$ looks as follows:

state	0	6	13	12	12
input	a	b	a	a	#
output	1	2	2	1	212
new state	6	13	12	12	

The mirror image $aaba$ of the input $abaa$ represents in N the number 213:

$$[aaba] = 13 \cdot 2^3 + 13 \cdot 2^2 + 22 \cdot 2 + 13 = 213.$$

The mirror image 2121221 of the output is the dyadic representation of 213. We have already pointed out the notational inconvenience caused by the fact that in the number-system notation the leftmost digit is the most significant one.

The claims in the following theorem are either immediate consequences of the Translation Lemma, or can be inferred from it using decidability results concerning regular languages.

Theorem 6.5. It is decidable whether or not a given number system is ambiguous, complete or almost complete. The equivalence problem is decidable for number systems. It is undecidable whether or not a given recursively enumerable set is RNS. It is also undecidable, given a recursively enumerable set S and an integer $n \geq 2$, whether or not there is a number system N with base n such that $S = S(N)$. It is decidable of a given number system N and an integer $n \geq 2$ whether or not there exists an unambiguous number system N' with base n satisying $S(N) = S(N')$. $\square$

For the number system $N = (3, 1, 3, 4, 6, 7)$ and $n = 2$, the decision method of the last sentence produces the number system $N' = N_2 = (2, 1, 4)$ already discussed above. On the other hand, every representation of $S(N)$ with base 3 is ambiguous. Thus, although the property characterizing $S(N)$ is intimately connected with the number 3, one has to choose a different base in order to get an unambiguous representation of $S(N)$!

Using more sophisticated methods, dealing with recognizable sets and formal power series, the decidability of the equivalence can be extended to concern number systems with arbitrary integer digits, [Ho5]. Also the final theorem in this subsection, [Ho9], results from an application of such methods. A weaker version of the theorem was established in [Ho6].

Theorem 6.6. Given a number system, it is decidable whether or not there exists an equivalent unambiguous number system. $\square$

6.3 Bounded delay and immigration

We now return to codes and L codes. For easier reference, we use certain bold letters to denote classes of morphisms: **C** stands for the class of codes and **L** for the class of L codes. The notation **P** refers to *prefix codes*, that is, morphisms h for which there are no distinct letters a and b for which $h(a)$ is a prefix of $h(b)$. (It is clear that morphisms satisfying this condition are codes.) We now present the idea of *bounded delay.*

With cryptographic connotations in mind, let us refer to the argument w as *plaintext* and to the encoded version $h(w)$ or $\overline{h}(w)$ as *cryptotext.* The idea behind the *bounded delay* is the following. We do not have to read the whole cryptotext on order to start writing the plaintext but rather always a certain prefix of the cryptotext determines the beginning of the plaintext.

The notation $\text{pref}_k(w)$ for the prefix of w of length $k \geq 1$ was already introduced in Section 2. The notation $\text{first}(w)$ stands for the first letter of a nonempty word w. A morphism h is of *bounded delay* k if, for all words u and w, the equation

$$\text{pref}_k(h(u)) = \text{pref}_k(h(w))$$

implies the equation $\text{first}(u) = \text{first}(w)$. The morphism h is of *bounded delay,* in the class $\mathbf{B}$, if it is of bounded delay k, for some k. It is again easy to see that the property of bounded delay implies the property of being a code. The morphism h defined by

$$h(a) = aa, \quad h(b) = ba, \quad h(c) = b$$

is a code but not of bounded delay. Indeed, one might have to read the whole cryptotext in order to determine whether the first plaintext letter is b or c. Different notions of bounded delay are compared in [Br] and [BeP]. The same class $\mathbf{B}$ is obtained no matter which definition is chosen, but different definitions may lead to different minimal values of k.

The idea of bounded delay is the same for codes and L codes: first k letters of the cryptotext determine the first plaintext letter. For codes the situation remains unaltered after handling the first letter a, because the cryptotext is still of the form $h(w)$ when $h(a)$ has been removed from the beginning. However, for L codes, the remainder of the cryptotext equals $h\overline{h}(w)$ rather than $\overline{h}(w)$. This means that we obtain different notions of bounded delay, depending on whether we are interested in finding only the first plaintext letter (*weak* notion, $\mathbf{W}$), or the first letter at each stage of decryption (*strong* or *medium strong* notion). The difference between the two latter notions depends on the way of bounding the delay: is the bound kept constant (*strong* notion $\mathbf{S}$), or is it allowed to grow according to the stage of decryption (*medium strong* notion, $\mathbf{M}$). We now give the formal definition.

Definition. A morphism h is of *weakly bounded delay* $k \geq 1$ if, for all words u and w, the equation

$$\text{pref}_k(\overline{h}(u)) = \text{pref}_k(\overline{h}(w))$$

implies the equation $\text{first}(u) = \text{first}(w)$. If for all $i \geq 0$ and all u and w, the equation

$$\text{pref}_k(h^i\overline{h}(u)) = \text{pref}_k(h^i\overline{h}(w))$$

implies the equation $\text{first}(u) = \text{first}(w)$, then h is of *strongly bounded delay* k. In general, h is of weakly or strongly bounded delay if it is so for some k. The notations **W** and **S** are used for the corresponding classes of morphisms. Finally, h is of *medium bounded delay* (notation **M**) if, for some recursive function f and all $i \geq 0$, u and w, the equation

$$\mathrm{pref}_{f(i)}(h^i\overline{h}(u)) = \mathrm{pref}_{f(i)}(h^i\overline{h}(w))$$

implies the equation $\mathrm{first}(u) = \mathrm{first}(w)$. □

Observe that we do not require h to be an L code in these definitions. The situation is analogous to that concerning ordinary codes. However, a morphism being in **B** implies that it is in **C**, whereas **L** and **W** are incomparable. All inclusion relations between the classes introduced are presented in the following theorem [MSW3].

Theorem 6.7. The mutual inclusion relations between the families introduced are given by the following diagram, where an arrow denotes strict inclusion and two families are incomparable if they are not connected by a path:

$$\begin{tikzcd}
 & & & \mathrm{C} & \mathrm{L} \\
\mathrm{S} & \mathrm{P} & \mathrm{B} & & \\
 & & & \mathrm{M} & \mathrm{W}
\end{tikzcd}$$

□

Medium bounded delay can be viewed in the theory of L codes as the most natural counterpart of bounded delay codes. It is natural to require that only a bounded amount of lookahead at each stage of the decryption process is needed. If the amount of lookahead remains the same throughout the process, the resulting notion is a very restricted one, as will be seen below. On the other hand, the drawback in the definition of **M** is that, in general, the construction of the sequence of values $f(i) = k_i$, $i = 0, 1, 2, \ldots$, seems to be an infinitary task. The following theorem, [HoS], is pleasing because it shows that it suffices to construct values only up to $\mathrm{card}(\Sigma) - 2$.

We say that a morphism $h : \Sigma^* \longrightarrow \Sigma^*$ is an the set **M'** if, for some $k > 0$ and all i with $0 \leq i \leq \mathrm{card}(\Sigma) - 2$, the equation

$$\mathrm{pref}_k(h^i\overline{h}(u)) = \mathrm{pref}_k(h^i\overline{h}(w))$$

always implies the equation $\mathrm{first}(u) = \mathrm{first}(w)$. Thus, we consider the sequence $f(i) = k_i$ only up to $\mathrm{card}(\Sigma) - 2$, and take the maximum of the resulting numbers.

Theorem 6.8. M' = M. □

We now present a simple characterization for the family **S**.

Theorem 6.9. A morphism h is in **S** iff, for any distinct letters a and b, $\mathrm{first}(h(a)) \neq \mathrm{first}(h(b))$.

Proof. Consider the "if"-part. The assumption means that there is a permutation Π of the alphabet Σ such that, for all a,

$$\mathrm{first}(h(a)) = \Pi(a).$$

Consequently, for all $i \geq 0$ and a,

$$\text{first}(h^i(a)) = \Pi^i(a).$$

Therefore,

$$\text{pref}_1(h^i\overline{h}(a)) = \text{first}(h^{i+1}(a)) = \Pi^{i+1}(a)$$

uniquely determines a, that is, h is of strongly bounded delay 1.

For the "only if" part, we need two auxiliary results that are simple exercises concerning morphisms. Growing letters were defined at the end of Subsection 6.1.

Claim 1. Assume that h is a code and a is a letter. Then either a is growing, or else $|h^i(a)| = 1$, for all i.

Claim 2. If h is a prefix code and a is growing, then first$(h(a))$ is growing.

Let h be in **S**. By Theorem 6.7, h is a prefix code. We proceed indirectly and assume that there are two distinct letters a and b such that first$(h(a))$ = first$(h(b))$. Since h is a prefix code, we may write

$$h(a) = cxdy \text{ and } h(b) = cxez,$$

where x, y, z are (possibly empty) words and c, d, e are letters such that $d \neq e$. By Claim 1, a and b are growing. By Claim 2, also c is growing. Hence, for every k, there is an i such that

$$\text{pref}_k(h^i\overline{h}(a)) = \text{pref}_k(h^{i+1}(a)) = \text{pref}_k(h^{i+1}(b)),$$

which contradicts the assumption that h is in **S**. $\square$

At the time of this writing, no general decision method is known for testing membership in **M** or **W**.

Some more sophisticated decidability results are obtained, [Ho7], by considering *ambiguity sets* closely connected with the theory of L codes. Essentially, a morphism being an L code means that the ambiguity set is empty. It can be shown that in most cases the ambiguity set is regular, which leads to decidability.

Variations in degrees of iteration constitutes the basic idea behind L codes. One can view the definition also as developmental processes started at different times. The same idea has led to the notion of *L systems with immigration.* One begins with a finite set of words consisting, intuitively, of "beginnings", "origins", "atoms", "founding fathers" that start to develop in an environment. An L system is used to model the development in the environment. The more time has elapsed since immigration, the more steps have been taken in the developmental process.

For any types of L systems, the corresponding system with immigration can be defined. However, so far only D0L systems have been investigated in this respect. We conclude with a few remarks about this area. We begin with a definition.

Definition. A *D0L system with immigration* (shortly, ImD0L system) is a triple $G = (\Sigma, h, B)$ where Σ is an alphabet, $h : \Sigma^* \longrightarrow \Sigma^*$ a morphism and B is a finite subset of Σ^*. Its language is defined by

$$L(G) = \{b_0 h(b_1) \dots h^n(b_n) | \ n \geq 0, b_i \in B\}.$$

An ImD0L system is *growing* if each word of B contains a growing letter. $\Box$

As a construct, an ImD0L system is a DF0L system. However, the definition of its language is entirely different. Intuitively, the words of B describe the various possibilities of immigration to the population, and the words of $L(G)$ describe various developmental stages of the immigrants. Mathematically ImD0L systems constitute a very natural generalization of D0L languages. Especially results concerning *subword complexity*, with some best-possible bounds, are of interest, [Ho8]. We conclude with some further results from [Ho8].

Theorem 6.10. Every ImD0L language is an HDT0L language and possesses effectively a test set. Regularity is decidable for languages generated by growing ImD0L systems. $\Box$

7 Parallel insertions and deletions

L systems are based on the language-theoretic notion of *finite substitution* over an alphabet Σ. So far, a substitution has been defined as an operation on an alphabet. A substitution is never applied to λ (except for the convention that λ is always mapped into λ). The work on parallel insertions, [Ka1], [Ka2], [KMPS], can be viewed as an attempt to understand the substitution on the empty word.

Let L_1, L_2 be two languages over an alphabet Σ. The operation of *parallel insertion* of a language L_2 into a language L_1, can be viewed as a nonstandard modification of the notion of substitution. It maps all letters of Σ into themselves and the empty letter into L_2, with the following additional convention. For each word w, between all the letters and also at the extremities, only one λ occurs. The effect of the substitution applied to L_1 will be the insertion of words belonging to L_2 between all letters and also at the extremities of words belonging to L_1.

The exact effect of the classical substitution that maps all letters into themselves and λ into a language L_2 would be the insertion of arbitrary many words of L_2 between letters and at the extremities of words in L_1. According to the definitions mentioned above, this would amount to the parallel insertion of L_2^* into L_1.

While preserving the type of parallelism characteristic to L systems, parallel insertion has a greater degree of nondeterminism: words of L_2 are indiscriminately inserted between all letters of the target words in L_1. One way to regulate the process of insertion is by introducing the notion of *controlled parallel insertion*: each letter determines what can be inserted after it.

Another way to look at operations of controlled insertion is the following. Such an operation can be viewed as a *production* of the form $a \longrightarrow aw$, where the word w comes from the language to be inserted next to a. The mode of controlled insertion determines how the productions are going to be applied. The controlled parallel insertion resembles, thus, the rewriting process of 0L systems. However, it gives rise to something different from 0L systems because the productions are always of the special form and, on the other hand, there are infinitely many productions.

Formally, if L_1, L_2 are languages over the alphabet Σ, the parallel insertion of L_2 into L_1 is defined as:

$$L_1 \Longleftarrow L_2 = \bigcup_{u \in L_1} (u \Longleftarrow L_2),$$

where

$$u \Longleftarrow L_2 = \{v_0 a_1 v_1 a_2 v_2 \dots a_k v_k | \ k \geq 0, a_j \in \Sigma, 1 \leq j \leq k,$$
$$v_i \in L_2, 0 \leq i \leq k \text{ and } u = a_1 a_2 \dots a_k\}.$$

The case $k = 0$ corresponds to the situation $u = \lambda$, when only one word $v_0 \in L_2$ is inserted.

As parallel insertion is associative, it induces a monoid structure on $\mathcal{P}(\Sigma^*)$, the set of all subsets of Σ^*, with $\{\lambda\}$ as the neutral element. The monoid is not commutative. For example, $a \Longleftarrow b = \{bab\}$ whereas $b \Longleftarrow a = \{aba\}$. The families of regular, context-free and context-sensitive languages are closed under parallel insertion. Indeed, for L_1, L_2 languages over Σ we have that $L_1 \Longleftarrow L_2 = L_2 s(L_1)$, where s is the λ-free substitution defined by

$$s: \ \Sigma^* \longrightarrow 2^{\Sigma^*}, \ \ s(a) = aL_2, \text{ for every } a \in \Sigma.$$

The assertion above now follows as the families of regular, context-free and context-sensitive languages are closed under catenation and λ–free substitutions.

The parallel insertion amounts thus to the application of a single substitution. If, as in the case of L systems, we consider iterated applications of the substitution, the obtained operation is much more powerful than the parallel insertion: starting with two one-letter words the iterated parallel insertion can produce a non-context-free language. Formally, if L_1, L_2 are languages over Σ, the parallel insertion of order n of L_2 into L_1 is inductively defined by the equations:

$$\begin{array}{lcl} L_1 \Longleftarrow^0 L_2 & = & L_1, \\ L_1 \Longleftarrow^{k+1} L_2 & = & (L_1 \Longleftarrow^i L_2) \Longleftarrow L_2, i \geq 0. \end{array}$$

The *iterated parallel insertion* of L_2 into L_1 is then defined as

$$L_1 \Longleftarrow^* L_2 = \bigcup_{n=0}^{\infty} (L_1 \Longleftarrow^n L_2).$$

The iterated parallel insertion is not commutative: the word *bbb* belongs to $\lambda \Longleftarrow^* b$ but not to $b \Longleftarrow^* \lambda$. It is not associative either as the word *cbcab* belongs to $a \Longleftarrow^* (b \Longleftarrow^* c)$ but not to $(a \Longleftarrow^* b) \Longleftarrow^* c$.

The iterated parallel insertion of the letter b into itself is

$$b \Longleftarrow^* b = \{b^{2^k-1} | \ k > 0\},$$

which proves that the families of regular and context-free languages are not closed under iterated parallel insertion. However, the family of context-sensitive languages is still closed under it. Indeed, given two context-sensitive grammars G_1 and G_2 generating the languages L_1 and L_2, a grammar G generating $L_1 \Longleftarrow^* L_2$ can be obtained as follows. G contains the rules of G_2, the rules of G_1, and the rules of G_1 modified in such a way that next to each letter, the axiom of G_2 is attached.

An interesting variation on the theme of parallel insertion is to combine it with the commutative closure. The *commutative closure* of a language L is the smallest commutative language containing L . The *commutative closure* can be viewed as a unary operation associating to every language L its commutative closure com(L). The *permuted parallel insertion* of the word v into u consists thus of the parallel insertion into u of all words which are letter-equivalent to v (Two words are letter-equivalent if one of them is obtained by permuting the letters of the other.)

More precisely, $L_1 \Longleftarrow_p L_2 = \bigcup_{u \in L_1, v \in L_2} (u \Longleftarrow \mathrm{com}(v))$. Obviously, the permuted parallel insertion can be expressed as $L_1 \Longleftarrow_p L_2 = L_1 \Longleftarrow \mathrm{com}(L_2)$. As expected , the fact that the families of regular and context-free languages are not closed under the commutative closure implies that they are not closed under permuted parallel insertion. On the other hand, being closed under both parallel insertion and commutative closure, the family of context-sensitive languages is closed under permuted parallel insertion.

We have dealt so far with operations where the same language is inserted in parallel between all the letters and at the extremities of a given word. The process resembles more the type of rewriting characteristic to L systems if every letter determines what can be inserted after it. Let L be a language over Σ and $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ a so-called *control function* satisfying $\Delta(a) \neq \emptyset$, $\forall a \in \Sigma$. The Δ–*controlled parallel insertion* into L (shortly controlled parallel insertion) is defined as:

$$L \Longleftarrow_c \Delta = \bigcup_{u \in L} (u \Longleftarrow_c \Delta),$$

where

$$u \Longleftarrow_c \Delta = \{a_1 v_1 a_2 v_2 \ldots a_k v_k | \ u = a_1 \ldots a_k, k \geq 1, a_i \in \Sigma.$$

$$\text{and } v_i \in \Delta(a_i), 1 \leq i \leq k\}.$$

Note that in the above definition, the control function cannot have the empty set as its value. This condition has been introduced because of the follwing reason.

If there would exist a letter $a \in \Sigma$ such that $\Delta(a) = \emptyset$, then all the words $u \in L$ which contain a would give $u \Longleftarrow_c \Delta = \emptyset$. This means that these words would not contribute to the result of the controlled parallel insertion. Consequently we can introduce, without loss of generality, the condition $\Delta(a) \neq \emptyset$, $\forall a \in \Sigma$.

If we impose the restriction that for a distinguished letter $b \in \Sigma$ we have $\Delta(b) = L_2$, $L_2 \subseteq \Sigma^*$, and $\Delta(a) = \lambda$ for every letter $a \neq b$, we obtain a particular case of controlled parallel insertion: *parallel insertion next to the letter b*. It is a binary language operation, whereas the arity of the Δ-controlled parallel insertion equals $\mathrm{card}(\Sigma) + 1$.

The families of regular, context-free and context-sensitive languages are closed under controlled parallel insertion. This follows as the result of the controlled parallel insertion $L \Longleftarrow_c \Delta$, where $L \subseteq \Sigma^*$ and $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$, $\Delta(a) \neq \emptyset$, $\forall a \in \Sigma$, can be accomplished by the λ-free substitution

$$\sigma : \Sigma^* \longrightarrow 2^{\Sigma^*}, \ \ \sigma(a) = a\Delta(a), \forall a \in \Sigma.$$

For each of the above mentioned variants of insertion, a dual deletion operation can be defined. Take, for example, the *parallel deletion*. Given words u and v, the parallel deletion of v from u consists of the words obtained by simultaneously erasing from u all the nonoverlapping occurrences of v. The definition is extended to languages in the natural way. Given a word u and a language L_2, the parallel deletion $u \Longrightarrow L_2$ consists of the words obtained by erasing from u all the nonoverlapping occurrences of words in L_2:

$$\begin{aligned} u \Longrightarrow L_2 = \ \ & \{u_1u_2\ldots u_ku_{k+1} |\ \ k \geq 1, u_i \in \Sigma^*, 1 \leq i \leq k+1 \text{ and} \\ & \exists v_i \in L_2, 1 \leq i \leq k \text{ such that } u = u_1v_1\ldots u_kv_ku_{k+1}, \\ & \text{ where } \{u_i\} \cap [\Sigma^*(L_2 \setminus \{\lambda\})\Sigma^*] = \emptyset, 1 \leq i \leq k+1\}. \end{aligned}$$

Note that, besides the fact that the parallel deletion erases from u nonoverlapping occurrences of words from L_2, a supplementary condition has to be fulfilled: between two occurrences of words of L_2 to be erased, no nonempty word from L_2 appears as a subword. This assures that *all* occurrences of words from L_2 have been erased from u, and is taken care of by the last line of the definition. The reason why λ had to be excluded from L_2 is clear. If this wouldn't be the case and λ would belong to L_2, the condition $\{u_i\} \cap \Sigma^*L_2\Sigma^* = \emptyset$ would imply $\{u_i\} \cap \Sigma^* = \emptyset$ – a contradiction. Note that words from L_2 can still appear as subwords in $u \Longrightarrow L_2$, as the result of catenating the remaining pieces of u. If L_1, L_2 are languages over Σ, we can define now $L_1 \Longrightarrow L_2 = \cup_{u \in L_1}(u \Longrightarrow L_2)$.

If L, R are languages over the alphabet Σ, L a λ-free language and R a regular one, then there exists, [Ka1], a rational transducer g, a morphism h and a regular language R' such that $L \Longrightarrow R = h(g(L) \cap R')$. As a corollary, the families of regular and context-free languages are closed under parallel deletion with regular languages. On the other hand, the family of context-free languages is not in general closed under parallel deletion. For example, consider $\Sigma = \{a, b\}$

and the context-free languages:

$$L_1 = \#\{a^i b^{2i} |\ i > 0\}^*,$$

$$L_2 = \#a\{b^i a^i |\ i > 0\}^*.$$

If the language $L_1 \Longrightarrow L_2$ would be context-free, then also the language

$$(L_1 \Longrightarrow L_2) \cap b^+ = \{b^{2^n} |\ n > 0\},$$

would be context-free, which is a contradiction.

A similar nonclosure situation happens in the case of context-sensitive languages: there exists a context-sensitive language L_1 and a word w over an alphabet Σ such that $L_1 \Longrightarrow w$ is not a context-sensitive language. Indeed, let L be a recursively enumerable language (which is not context-sensitive) over an alphabet Σ and let a, b be two letters which do not belong to Σ. Then there exists a context-sensitive language L_1 such that (see [S1]):

(i) L_1 consists of words of the form $a^i b\alpha$, where $i \geq 0$ and $\alpha \in L$;

(ii) For every $\alpha \in L$, there is an $i \geq 0$ such that $a^i b\alpha \in L_1$.

It is easy to see that $aL_1 \Longrightarrow \{a\} = bL$, which is not a context-sensitive language. We have catenated a to the left of L_1 in order to avoid the case $i = 0$, when the corresponding words from L would have been lost.

In a similar way we have defined the iterated parallel insertion, we can define the *iterated parallel deletion*. Despite of the simplicity of the languages involved, it is an open problem whether or not the family of regular languages is closed under iterated parallel deletion. Not only that, but it is still an open problem whether the iterated parallel deletion of a singleton word from a regular language still belongs to the family of regular languages. The answer to both of the previous questions is negative in the case of context-free and context-sensitive languages. For example, if we consider the alphabet $\Sigma = \{a, b\}$, the context-free language $L = \{a^i \# b^{2i} |\ i > 0\}^*$, and the word $w = ba$ then

$$(L \Longrightarrow^* ba) \cap a\#^+ b^+ = \{a\#^n b^{2^n} |\ n > 0\},$$

which implies that $L \Longrightarrow^* ba$ cannot be context-free.

The *permuted parallel deletion* of a language L_2 from L_1 is obtained by erasing from words in L_1 words that are letter-equivalent to words in L_2:

$$L_1 \Longrightarrow_p L_2 = \bigcup_{u \in L_1, v \in L_2} (u \Longrightarrow_p v),$$

where $u \Longrightarrow_p v = u \Longrightarrow \mathrm{com}(v)$. If $\Sigma = \{a, b\}$ and $L_1 = \$a^* b^* \#\# a^* b^* \$$ and $L_2 = \#\$(ab)^*$ then the permuted parallel deletion of L_2 from L_1 is

$$\begin{aligned} L_1 \Longrightarrow_p L_2 = \ & \{\$a^n b^m \# |\ m, n \geq 0, m \neq n\} \cup \\ & \{\#a^n b^m \$ |\ m, n \geq 0, m \neq n\} \cup \{\lambda\}, \end{aligned}$$

which shows that the result of permuted parallel deletion between two regular languages is not necessarily regular. If L_1 is the context-free language

$$L_1 = \{a_1^n b_1^m c_1^l \# c_2^l b_2^m a_2^n \# |\ n, m, l \geq 0\}$$

and L_2 is the regular language $L_2 = \#\#(a_2 b_2 c_2)^*$ then

$$L_1 \Longrightarrow_p L_2 = \{a_1^n b_1^n c_1^n |\ \ n \geq 0\}.$$

This shows that the family of context-free languages is not closed under permuted parallel deletion with regular languages. By using an argument similar to the one used for parallel deletion, one can show that the family of context-sensitive languages is not closed under permuted parallel deletions with singleton languages.

A *controlled* variant of the *parallel deletion* can be defined as follows. Let $u \in \Sigma^*$ be a word and $\Delta : \Sigma \longrightarrow 2^{\Sigma^*}$ be a control function which does not have $\emptyset$ as its value. The set $u \Longrightarrow_c \Delta$ is obtained by finding all the nonoverlapping occurrences of av_a, $v_a \in \Delta(a)$, in u, and by deleting v_a from them. Between any two ocurrences of words of the type av_a, $v_a \in \Delta(a)$, in u, no other words of this type may remain.

If one imposes the restriction that for a distinguished letter $b \in \Sigma$ we have $\Delta(b) = L_2$, and $\Delta(a) = \lambda$ for any letter $a \neq b$, a special case of controlled parallel deletion is obtained: *parallel deletion next to the letter b*. The parallel deletion next to the letter b is denoted by $\stackrel{b}{\Longrightarrow}$. Let us examine the set $u \stackrel{b}{\Longrightarrow} L_2$, where u is a nonempty word and L_2 is a language over an alphabet Σ. If $u = b^k$, $k > 0$, and no word of the form bv, $v \in L_2$ occurs as a subword in u, the set $u \stackrel{b}{\Longrightarrow} L_2$ equals the empty set. If u contains at least one letter different from b, u is retained in the result as we can erase λ near that letter. The other words in $u \stackrel{b}{\Longrightarrow} L_2$ are obtained by finding all the nonoverlapping occurrences of words of the type bv_i, $v_i \in L_2$ in u, and deleting v_i from them. There may exist more than one possibility of finding such a decomposition of u into subwords.

One can use similar techniques as for the parallel deletion to show that the family of regular languages is closed under controlled parallel deletion, while the families of context-free and context-sensitive languages are not (except when the languages to be deleted are regular).

Besides studying closure properties, one can get more insight into the power of parallel insertion and deletion operations by investigating classes of languages that contain simple languages and are closed under some of the operations. Such a class should be closed under a parallel insertion operation, a parallel deletion one and an iterated parallel insertion one. Particular controlled operations will be chosen in order to allow an increase as well a decrease of the length of the words in the operands. The iterative operation has been included to provide an infinite growth of the strings. Finally, we have to introduce the mirror image and the union with λ for technical reasons.

Thus, let $\mathcal{P}$ be the smallest class of languages which contains the empty set, the language $\{\lambda\}$, the singleton letters and is closed under mirror image, union with $\{\lambda\}$, controlled parallel insertion, iterated controlled parallel insertion and controlled parallel deletion with singletons. Then, [Ka2], $\mathcal{P}$ is contained in the family of context-sensitive languages and properly contains the family of regular languages. If we replace in $\mathcal{P}$ the controlled parallel deletion with singletons by unrestricted controlled parallel deletion, the new family $\mathcal{P}'$ is a Boolean algebra properly containing the family of regular languages.

The study of parallel insertion and deletion operations is closely connected to the study of equations involving them. If $\diamond$ denotes a parallel insertion or deletion operation, the simplest equations to consider are of the form $L \diamond X = R$, $Y \diamond L = R$, where R is a regular language, L is an arbitrary language, and X, Y are the unknown languages.

If $\diamond$ denotes parallel insertion, permuted parallel insertion or parallel insertion next to a letter, the problem whether there exists a singleton solution $X = \{w\}$ to the equation $L \diamond X = R$ is decidable for regular languages L and R. Indeed, if $\diamond$ denotes parallel insertion, let L, R be regular languages and m be the length of the shortest word in R. If there exists a word w such that $L \Longleftarrow w = R$ then it must satisfy the condition $\lg(w) \leq m$. As the family of regular languages is closed under parallel insertion, our problem is decidable. The algorithm for deciding it will consist of checking whether or not $L_1 \Longleftarrow w = R$ for all words w with $\lg(w) \leq m$. The answer is YES if such a word exists and NO otherwise. A similar proof can be used to show that, for $\diamond$ denoting parallel insertion, controlled parallel insertion and parallel insertion next to a letter, the existence of a singleton solution Y to the equation $Y \diamond L = R$ is decidable for regular languages L and R.

In contrast, given context-free languages L and regular languages R, the existence of both a solution and a singleton solution X, to the equation $L \diamond X = R$ is undecidable for $\diamond$ denoting parallel insertion, iterated parallel insertion, permuted parallel insertion or parallel insertion next to a letter. Let $\#$ be a letter which does not belong to Σ. We can actually prove a stronger statement: there exists a fixed regular language $R = \Sigma^*\#$ such that the problem above is undecidable for context-free languages L_1. This follows by noticing that the equation $(L_1\#) \Longleftarrow X = \Sigma^*\#$ holds for languages L_1, X over Σ exactly in case $L_1 = \Sigma^*$ and $X = \lambda$. Hence, if we could decide our problem, we would be deciding the problem "Is $L_1 = \Sigma^*$?" for context-free languages L_1, which is impossible. A similar proof, taking $R = \Sigma^* \cup \{\#\}$ and $L_1 = L \cup \{\#\}$, can be used to show that the existence of both a solution and singleton solution Y to the equation $Y \Longleftarrow L = R$ is undecidable for context-free languages L and regular languages R. Analogously it can be shown that the existence of both a solution and singleton solution to the equation $Y \Longleftarrow \Delta = R$ is undecidable for context-free control functions and regular languages R.

If we consider the same problems for parallel deletion operations, special attention has to be paid to the fact that, while the result of parallel insertion is always a word or set of words, the result of parallel deletion can be the empty set. If $\diamond$ denotes a binary deletion operation and L is a given language, a word y is called *right-useful with respect to* L *and* $\diamond$ if there exists $x \in L$ such that $x \diamond y \neq \emptyset$. A language X is called right-useful with respect to L and $\diamond$ if it consists only of right-useful words with respect to L and $\diamond$. The notion of *left useful* can be similarly defined.

In the following, when we state the undecidability of the existence of a solution to a certain equation, we will mean in fact that the existence of a *useful* solution is undecidable. For example, if $\diamond$ denotes the parallel deletion, permuted parallel deletion or iterated parallel deletion, the existence of a solution to the equation $L \diamond X = R$ is undecidable for context-free languages L and regular languages R. Let us consider the case of parallel deletion. If $\$, \#$ are letters that do not occur in Σ, there exists a regular language $R = \#\Sigma^+\# \cup \$\Sigma^*\$$ such that our problem is undecidable for context-free languages L. Indeed, for a context-free language L, consider the language $L_1 = \#\Sigma^+\# \cup \$L\$$. For all languages $L_2 \subseteq \Sigma^*$, the equation

$$\#\Sigma^+\# \cup \$L\$ \Longrightarrow X = \#\Sigma^+\# \cup \$\Sigma^*\$$$

holds if and only if $X = \{\lambda\}$ and $L = \Sigma^*$. If we could decide our problem, we could decide whether for given context-free languages L, the equation $L = \Sigma^*$ holds, which is impossible.

The problem remains undecidable even if instead of considering a binary operation, like parallel deletion, we consider a card(Σ)+1 – ary operation like controllel parallel deletion. It can be namely shown that the existence of a singleton control function with the property $L \Longrightarrow_c \Delta = R$ is undecidable for context-free languages L and regular languages R. The proof is similar to the preceding one and is based on the fact that the equation

$$L_1\$ \cup \{\#a\$ |\ a \in \Sigma\} \Longrightarrow_c \Delta = \Sigma^+ \cup \{\#a |\ a \in \Sigma\}$$

holds for $L_1 \subseteq \Sigma^*$ and singleton control functions Δ if and only if

$$\Delta(\#) = \Delta(\$) = \lambda, \Delta(a) = \$, a \in \Sigma \text{ and } L_1 = \Sigma^+.$$

The operations of parallel insertion and deletion are associated with several notions rather basic in the combinatorics of words. A *parallel deletion set* is a set of the form $w \Longrightarrow L$, $w \in \Sigma^*$, $L \subseteq \Sigma^*$. Parallel deletions sets are universal in the sense that every finite language can be viewed as a parallel deletion set (see [KMPS]). If we fix the nonempty finite set F in the equation

$$w \Longrightarrow L = F,$$

we can ask for an algorithm deciding for a given context-free language L whether or not a solution w to the equation $w \Longrightarrow L = F$ exists. If such an algorithm exists, we say that F is *CF-decidable*, otherwise *CF-undecidable*. F is called

CF-universal if for any (nonempty) context-free language L, there is a word w such that $w \Longrightarrow L = F$. We have that the set $\{\lambda\}$ is CF-universal and it is the only CF-universal set.

In spite of the fact that parallel deletion sets coincide with finite sets, every finite nonempty set $F \neq \emptyset$ is CF-undecidable, ([KMPS]).

The *parallel deletion number*, [KMPS], associated to a word w equals the cardinality of the largest parallel deletion set arising from w, that is

$$pd(w) = \max\{\text{card}(w \Longrightarrow L) |\ \ L \subseteq \Sigma^*\}.$$

For the alphabet with only one element, $pd(w)$ can be computed, but for the general case the question seems not to be simple at all. Indeed, one can show that if $w = a^n$, $n \geq 1$, then $pd(w) = n$. In the case of arbitrary alphabets with at least two symbols the following surprising result, [KMPS], is obtained. If $\text{card}(\Sigma) \geq 2$, then there is no polynomial f such that for every $w \in \Sigma^*$ we have $pd(w) \leq f(|w|)$.

Let us prove this result. It suffices to show that, given a polynomial f (in one variable), there are strings w such that $pd(w) > f(|w|)$.

Take a polynomial f of degree $n \geq 1$ and consider the strings

$$w_{n,m} = (a^m b^m)^n.$$

Moreover, take

$$L_m = \{a^i b^j |\ \ 1 \leq i, j \leq m-1\}.$$

and evaluate the cardinality of $w_{n,m} \Longrightarrow L_m$.

As each string in L_m contains at least one occurrence of a and one occurrence of b, we can delete from $w_{n,m}$ exactly n strings of L_m, which implies

$$w_{n,m} \Longrightarrow L_m = \{a^{m-i_1} b^{m-j_1} a^{m-i_2} b^{m-j_2} \dots a^{m-i_n} b^{m-j_n} |$$

$$1 \leq i_s, j_s \leq m-1, 1 \leq s \leq n\}.$$

Consequently,

$$\text{card}(w_{n,m} \Longrightarrow L_m) = (m-1)^{2n}.$$

Clearly, because $2n$ is a constant, for large enough m we have

$$pd(w_{n,m}) \geq (m-1)^{2n} > f(2nm) = f(|w_{n,m}|),$$

which completes the proof.

We observed that, for every word w, $w \Longrightarrow \Sigma^* = \{\lambda\}$. We can express this by saying that every word *collapses* to the empty word when subjected to parallel deletion with respect to Σ^*. We speak also of the *collapse set* of Σ^*. Thus, the collapse set of Σ^* equals Σ^*. In general, we define the *collapse set* of a nonempty language $L \subseteq \Sigma^*$ by

$$cs(L) = \{w \in \Sigma^* |\ \ w \Longrightarrow L = \{\lambda\}\}.$$

This language is always nonempty because it contains each of the shortest words in L. For example, $cs(\{a^nb^n|\ n \geq 1\}) = (ab)^+$ and $cs(\{a, bb\}) = a^*bb(a^+bb)^*a^* \cup a^+$, hence $cs(L)$ can be infinite for finite L. On the other hand, $cs(\{ab\} \cup \{a^nb^ma^p|\ n, m, p \geq 1\}) = \{ab\}$, hence $cs(L)$ can be finite for infinite L. We have that, [KMPS], there is a linear language L such that $cs(L)$ is not context-free. Indeed, take

$$L = \{dda^nb^mc^n|\ n, m \geq 1\} \cup \{da^nb^mc^p|\ n, m, p \geq 1, m \geq p\}.$$

While it is clear that L is linear,

$$cs(L) \cap d^2a^+b^+c^+ = \{d^2a^nb^mc^n|\ 1 \leq m \leq n\},$$

which, being non-context-free, shows that $cs(L)$ cannot be a context-free language. In fact, the collapse set of a language can be characterized as follows. If $L \subseteq \Sigma^*$ then

$$cs(L) = L^+ - M,$$

where

$$M = (\Sigma^*L \cup \{\lambda\})(\Sigma^+ \setminus \Sigma^*L\Sigma^*)(L\Sigma^* \cup \{\lambda\}).$$

As a corollary, we deduce that if L is a regular (context-sensitive) language, then $cs(L)$ is also a regular (context-sensitive) language.

Another natural way to define a parallel deletion operation is to remove exactly k strings, for a given k. Namely, for $w \in \Sigma^*$, $L \subseteq \Sigma^*$, $k \geq 1$, write

$$\begin{aligned} w \Longrightarrow_k L = \ & \{u_1u_2 \ldots u_{k+1}|\ u_i \in \Sigma^*, 1 \leq i \leq k+1, \\ & w = u_1v_1u_2v_2 \ldots u_kv_ku_{k+1}, \text{ for } v_i \in L, 1 \leq i \leq k\}. \end{aligned}$$

Sets of this form will be referred to as *k-deletion sets*; for given $k \geq 1$ we denote by E_k the family of k-deletion sets. For all $k \geq 1$, $E_k \subset E_{k+1}$, strict inclusion. Moreover, for every finite set F, there is a k such that $F \in E_k$, and given a finite set F and a natural number k, it is decidable whether $F \in E_k$ or not.

The operations of insertion and deletion seem to occur time and again in modeling natural phenomena. We have seen how L systems model the growth of multicellular organisms. The related notions of parallel insertion and deletion have recently become of interest in connection with the topic of DNA computing.

The area of DNA computing was born in 1994 when Adleman, [Ad1], succeeded in solving an instance of the Directed Hamiltonian Path solely by manipulating DNA strands. This marked the first instance where a mathematical problem could be solved by biological means and gave rise to a couple of interesting problems: a) can *any* algorithm be simulated by means of DNA manipulation, and b) is it possible, at least in theory, to design a programmable DNA computer?

To answer these questions, various models of DNA computation have been proposed, and it has been proven that these models have the full power of a

Turing machine. The models based on the bio-operations proposed by Adleman can be already implemented in laboratory conditions, but the fact that most bio-operations rely on mainly manual handling of tubes prevents the large scale automatization of the process.

There are two ways to overcome this obstacle and to further the research in DNA computing. The first approach is to try to speed-up and automatize the existing operations.

The second one is to try to design a model based entirely on operations that can be carried out by enzymes (cutting and pasting DNA strands at certain places). Indeed, already in [Ad2] the idea was advanced to "design a molecular computer (perhaps based on entirely different "primitives" than those used here) which would accomplish its task by purely chemical means inside of a single tube".

As DNA strands can be viewed as strings over the four letter alphabet $\Sigma = \{A, C, G, T\}$ (Adenine, Cytosine, Thymine and Guanine), it is natural to pursue this second approach within the frame of formal language theory (see the chapter in this Handbook written by Head, Paun, and Pixton for a formal language model of DNA computing, based on splicing). The cut and paste operations performed by enzymes can then be modeled as controlled parallel insertions and deletions. Further study of the power of these operations could assist in proving the claim in [SmS] that " only a finite collection of rewrite rules which insert or delete single U's in specified contexts of an mRNA sequence will suffice to get universality."

For attaining this goal, and with biological motivations in mind, the notion of control needs strenghtening: a word can be inserted into a string only if certain *contexts* are present. Formally, given a set $C \in \Sigma^* \times \Sigma^*$ called a *context set*, the *parallel contextual insertion*, [KT], of a word v into the word u is

$$u \Longleftarrow_C v = \{u_1 x_1 v y_1 u_2 x_2 v y_2 \dots u_k x_k v y_k u_{k+1} |\ k \geq 0,$$
$$u = u_1 u_2 \dots u_k u_{k+1}, (x_i, y_i) \in C, 1 \leq i \leq k\}.$$

The proof that controlled parallel insertions and deletions are enough to simulate the action of a Turing machine, would thus open a possible way for designing a molecular computer with all the operations carried out by enzymes.

8 Scattered views from the L path

We conclude this chapter with some scattered remarks, mostly about recent work. We have already emphasized that, in such a vast field as L systems, one cannot hope to be encyclopedic in a chapter of this size, especially if one tries to survey also the methods as we have done. Some of the areas neglected by us could have deserved even a section of their own.

Inductive inference certainly constitutes one such area. How much knowledge of the language or sequence is needed to infer the L system behind it, or one of the many possible L systems? This is relevant, for instance, in modeling experimental data. Inductive inference is a part of the theory of *learning*: one tries to get a complete picture from scattered information. Here the setup may vary. New data may come independently of the observer, or the latter may want some specific information, for instance, whether or not a specific word is in the language. Inductive inference (or *syntactic inference* as it is sometimes called) has been studied for L systems from the very beginning, [HR], [RSed1], [H1]. The reader is referred to [Yo2] and its references for some recent aspects.

There is a special chapter in this Handbook about *complexity*. [Ha], [JS], [vL] are early papers on the complexity of the membership problem for systems with tables, [vL] being especially interesting in that it provides a very nice classroom example of a reduction to an NP-complete problem. [Ke], [SWY], [LaSch] are examples of various areas of current interest. Most of the complexity issues studied in connection with L systems have been on the *metalevel*: complexity of questions *about* the system (membership, equivalence) rather than things happening *within* the system (length of derivations, workspace). This reflects perhaps the lack of really suitable machine models for L systems, [RS1].

If an L system is viewed as an *L form*, then it produces a family of languages rather than just a single language.The original L system is used as a propotype giving rise to a collection of systems resembling it. The study of L forms was originated in [MSW1]; they are also discussed in a later chapter in this Handbook.

[Sz], [V], [K2], [PS], [S3] represent different aspects of the early work on *growth functions*. A related more recent approach has been the application of L operations to power series, [Ku], [KS]. This approach will be dealt with in the chapter by W.Kuich in this Handbook. Also the recent very intensively investigated questions about *splicing* and DNA, [He], will be discussed elsewhere in this Handbook.

Many recent invetigations concerning length have dealt with *slenderness*, [APDS], [DPS1]. *Restricted parallelism* has been the object of many recent studies; the reader is referred to [Fe] and the references given therein.

Since the L families have weak closure properties, we face the following decision problem. Assume, for instance, that a family $\mathcal{L}$ is not closed under union. Given two languages from $\mathcal{L}$, their union may or may not belong to $\mathcal{L}$. Can we decide which alternative holds? The problem is decidable for PD0L languages and for U0L languages but undecidable for P0L languages, [DPS2]. For D0L languages the problem is open, due to difficulties discussed above in connection with growth functions.

It is quite common in language theory that an undecidable problem becomes decidable if sharper restrictions are imposed on the phenomena under study. For instance, equivalence is undecidable for context-free grammars, whereas *structural equivalence*, where also the derivation structures of the two grammars have to be the same, is decidable. The situation is exactly the same for E0L systems, [SY]. The papers [OW], [Nie], [SWY] present related results.

For *stochastic* variants of 0L systems, the reader is referred to [JM] and the references there. Stochastic variants have been useful in picture generation, [PL]. The area of *cooperating grammars and L systems*, [Pa], is discussed elsewhere in this Handbook.

[Yo1] and [Da2] introduce *control mechanisms* for 0L and DT0L systems, respectively. For instance, a table can be used if some other table has not been used. [IT], [Lan], [Har1], [Har2], [Ko] represent various D0L-related studies. In [Ko] conditions are deduced for D0L-simulation of DIL systems. In the *piecewise D0L systems* (or PD0L systems) of [Har2], the set Σ^* is partitioned, and the morphism depends on which part of the partition the word derived so far belongs to. A sequence of words still results. "Dynamical properties" such as finiteness and periodicity are decidable if the sets in the partition are regular languages.

The monographs [HR] and [RS1] and the collections of papers [RSed1] – [RSed4] contain many further references. The collections [RSed1] – [RSed3] also reflect the state of the art after roughly ten-year intervals. Of special value are the survey articles [L3], [L4], [LJ], where Aristid Lindenmayer was the author or coauthor.

References

[Ad1] L.Adleman, Molecular computation of solutions to combinatorial problems. *Science* v.266, Nov.1994, 1021 – 1024.

[Ad2] L.Adleman, On constructing a molecular computer. Manuscript in circulation.

[AL] M.Albert and J.Lawrence, A proof of Ehrenfeucht's conjecture. *Theoret. Comput. Sci.* 41 (1985) 121 – 123.

[APDS] M.Andrasiu, G.Paun, J.Dassow and A.Salomaa, Language-theoretic problems arising from Richelieu cryptosystems. *Theoret. Comput. Sci.* 116 (1993) 339 – 357.

[Be] J.Berstel, Sur les pôles et le quotient de Hadamard de séries N-rationelles, *C.R. Acad.Sci., Sér.A* 272 (1971) 1079 – 1081.

[BeP] J.Berstel and D.Perrin, *Theory of codes.* Academic Press, New York (1985).

[Br] V.Bruyere, Codes prefixes. Codes a delai dechiffrage borne. Nouvelle thèse, Université de Mons (1989).

[Ch] C.Choffrut, Iterated substitutions and locally catenative systems: a decidability result in the binary case. In [RSed3], 49 – 92.

[CF] K.Culik II and I.Fris, The decidability of the equivalence problem for D0L systems. *Inform. and Control* 35 (1977) 20 – 39.

[CK1] K.Culik II and J.Karhumäki, Systems of equations over a free monoid and Ehrenfeucht's conjecture. *Discrete Math.* 43 (1983) 139 – 153.

[CK2] K.Culik II and J.Karhumäki, A new proof for the D0L sequence equivalence problem and its implications. In [RSed2], 63 – 74.

[CS1] K.Culik II and A.Salomaa, On the decidability of homomorphism equivalence for languages. *J.Comput.Systems Sci.* 17 (1978) 163 – 175.

[CS2] K.Culik II and A.Salomaa, Ambiguity and decision problems concerning number systems. *Inform. and Control* 56 (1983) 139 – 153.

[Da1] J.Dassow, Eine Neue Funktion für Lindenmayer-Systeme. *Elektron.Informationsverarb.Kybernet.* 12 (1976) 515 – 521.

[Da2] J.Dassow, On compound Lindenmayer systems. In [RSed2], 75 – 86.

[DPS1] J.Dassow, G.Paun and A.Salomaa, On thinness and slenderness of L languages. *EATCS Bull.* 49 (1993) 152 – 158.

[DPS2] J.Dassow, G. Paun and A.Salomaa, On the union of 0L languages. *Inform. Process. Lett.* 47 (1993) 59 – 63.

[ER1] A.Ehrenfeucht and G.Rozenberg, The equality of E0L languages and codings of 0L languages. *Internat.J.Comput.Math.* 4 (1974) 95 – 104.

[ER2] A.Ehrenfeucht and G.Rozenberg, Simplifications of homomorphisms, *Inform. and Control* 38 (1978) 298 – 309.

[ER3] A.Ehrenfeucht and G.Rozenberg, Elementary homomorphisms and a solution of the D0L sequence equivalence problem. *Theoret. Comput. Sci.* 7 (1978) 169 – 183.

[ER4] A.Ehrenfeucht and G.Rozenberg, On a bound for the D0L sequence equivalence problem. *Theoret. Comput. Sci.* 12 (1980) 339 – 342.

[En] J.Engelfriet, The ET0L hierarchy is in the 0I hierarchy. In [RSed2], 100 – 110.

[Fe] H.Fernau, Remarks on adult languages of propagating systems with restricted parallelism. In [RSed4], 90 – 101.

[HW] G.Hardy and E.M.Wright, *An introduction to the Theory of Numbers.* Oxford Univ.Press, London and New York, (1954).

[Ha] T.Harju, A polynomial recognition algorithm for the EDT0L languages, *Elektron. Informationsverarb. Kybernet..* 13 (1977) 169 – 177.

[Har1] J.Harrison, Morphic congruences and D0L languages. *Theoret. Comput. Sci.* 134 (1994) 537 – 544.

[Har2] J.Harrison, Dynamical properties of PWD0L systems. *Theoret. Comput. Sci.* 14 (1995) 269 – 284.

[He] T.Head, Splicing schemes and DNA. In [RSed3], 371 – 384.

[H1] G.T.Herman, The computing ability of a developmental model for filamentous organisms. *J.Theoret.Biol.* 25 (1969) 421 – 435.

[H2] G.T.Herman, Models for cellular interactions in development withough polarity of individual cells. *Internat. J. System Sci.* 2 (1971) 271 –289; 3 (1972) 149 – 175.

[HR] G.T.Herman and G.Rozenberg, *Developmental Systems and Languages* North-Holland Publ., Amsterdam (1975).

[HWa] G.T.Herman and A.Walker, Context-free languages in biological systems. *Internat. J. Comput. Math.* 4 (1975) 369 – 391.

[Ho1] J.Honkala, Unique representation in number systems and L codes. *Discrete Appl. Math.* 4 (1982) 229 – 232.

[Ho2] J.Honkala, Bases and ambiguity of number systems. *Theoret. Comput. Sci.* 31 (1984) 61 – 71.

[Ho3] J.Honkala, A decision method for the recognizability of sets defined by number systems. *RAIRO* 20 (1986) 395 – 403.

[Ho4] J.Honkala, It is decidable whether or not a permutation-free morphism is an L code. *Internat. J. Computer Math.* 22 (1987) 1 – 11.

[Ho5] J.Honkala, On number systems with negative digits. *Annales Academiae Scientiarum Fennicae, Series A.I.Mathematica* 14 (1989) 149 – 156.

[Ho6] J.Honkala, On unambiguous number systems with a prime power base. *Acta Cybern.* 10 (1992) 155 – 163.

[Ho7] J.Honkala, Regularity properties of L ambiguities of morphisms. In [RSed3], 25 – 47.

[Ho8] J.Honkala, On D0L systems with immigration. *Theoret. Comput. Sci.* 120 (1993) 229 –245.

[Ho9] J.Honkala, A decision method for the unambiguity of sets defined by number systems. *Journal of Univ. Comput. Sci,* to appear.

[HoS] J.Honkala and A.Salomaa, Characterization results about L codes. *RAIRO* 26 (1992) 287 – 301.

[IT] M.Ito and G.Thierrin, D0L schemes and recurrent words. In [RSed2], 157 – 166.

[JS] N.Jones and S.Skyum, Complexity of some problems concerning L systems. *Lecture Notes in Computer Science* 52 (1977) 301 – 308.

[JM] H.Jürgensen and D.Matthews, Stochastic 0L systems and formal power series. In [RSed2], 167 – 178.

[K1] J.Karhumäki, An example of a PD2L system with the growth type 2 1/2. *Inform. Process. Lett.* 2 (1974) 131 – 134.

[K2] J.Karhumäki, On Length Sets of L Systems, Licentiate thesis, Univ. of Turku (1974).

[K3] J.Karhumäki, Two theorems concerning recognizable N-subsets of σ^*. *Theoret. Comput. Sci.* 1 (1976) 317 – 323.

[Ka1] L.Kari, On insertions and deletions in formal languages. Ph.D. thesis, University of Turku, Finland, 1991.

[Ka2] L.Kari, Power of controlled insertion and deletion. *Lecture Notes in Computer Science*, 812 (1994), 197 – 212.

[KMPS] L.Kari, A.Mateescu, G.Paun, A.Salomaa. On parallel deletions applied to a word, *RAIRO – Theoretical Informations and Applications*, vol.29, 2(1995), 129 – 144.

[KRS] L.Kari, G.Rozenberg and A.Salomaa, Generalized D0L trees. *Acta Cybern.*, 12 (1995) 1 – 9.

[KT] L.Kari, G.Thierrin, Contextual insertions and deletions. Submitted.

[KOE] T.Katayama, M.Okamoto and H.Enomoto, Characterization of the structure-generating functions of regular sets and the D0L growth functions. *Inform. and Control* 36 (1978) 85 – 101.

[Ke] A.Kelemenová, Complexity of 0L systems. In [RSed2], 179 – 192.

[Ko] Y.Kobuchi, Interaction strength of DIL systems. In [RSed3], 107 – 114.

[Ku] W.Kuich, Lindenmayer systems generalized to formal power series and their growth functions. In [RSed4], 171 – 178.

[KS] W.Kuich and A.Salomaa, *Semirings, Automata, Languages.* Springer-Verlag, Berlin, Heidelberg, New York (1986).

[Lan] B.Lando, Periodicity and ultimate periodicity of D0L systems. *Theoret. Comput. Sci.* 82 (1991) 19 – 33.

[LaSch] K.J.Lange and M.Schudy, The complexity of the emptiness problem for E0L systems. In [RSed3], 167 – 176.

[La1] M.Latteux, Sur les T0L systémes unaires. *RAIRO* 9 (1975) 51 – 62.

[La2] M.Latteux, Deux problemes decidables concernant les TUL langages. *Discrete Math.* 17 (1977) 165 – 172.

[L1] A.Lindenmayer, Mathematical models for cellular interaction in development I and II. *J.Theoret. Biol.* 18 (1968) 280 – 315.

[L2] A.Lindenmayer, Developmental systems without cellular interactions, their languages and grammars. *J. Theoret. Biol.* 30 (1971) 455 – 484.

[L3] A.Lindenmayer, Developmental algorithms for multicellular organisms: a survey of L systems. *J.Theoret. Biol.* 54 (1975) 3 – 22.

[L4] A.Lindenmayer, Models for multi-cellular development: characterization, inference and complexity of L-systems. *Lecture Notes in Computer Science* 281 (1987) 138 – 168.

[LJ] A.Lindenmayer and H.Jürgensen, Grammars of development: discrete-state models for growth, differentiation and gene expression in modular organisms. In [RSed3], 3 – 24.

[Li1] M.Linna, The D0L-ness for context-free languages is decidable. *Inform. Process. Lett.* 5 (1976) 149 – 151.

[Li2] M.Linna, The decidability of the D0L prefix problem. *Intern. J. Comput. Math.* 6 (1977) 127 – 142.

[Ma] G.Makanin, The problem of solvabillity of equations in a free semigroup. *Math. USSR Sb.* 32 (1977) 129 – 138.

[MSW1] H.Maurer, A.Salomaa and D.Wood, E0L forms. *Acta Inform.* 8 (1977) 75 – 96.

[MSW2] H.Maurer, A.Salomaa, D.Wood, L codes and number systems. *Theoret. Comput. Sci.* 22 (1983) 331 – 346.

[MSW3] H.Maurer, A.Salomaa and D.Wood, Bounded delay L codes. *Theoret. Comput. Sci.* 84 (1991) 265 – 279.

[MT] L.M.Milne-Thompson, *The Calculus of Finite Differences* Macmillan, New York (1951).

[MRS] J.Mäenpää, G.Rozenberg and A.Salomaa, Bibliography of L systems. Leiden University Computer Science Technical Report (1981).

[N] M.Nielsen, On the decidability of some equivalence problems for D0L systems. *Inform. and Control* 25 (1974) 166 – 193.

[NRSS] M.Nielsen, G.Rozenberg, A.Salomaa and S.Skyum, Nonterminals, homomorphisms and codings in different variations of 0L systems, I and II. *Acta Inform.* 3 (1974) 357-364; 4 (1974) 87 – 106.

[Nie] V.Niemi. A normal form for structurally equivalent E0L grammars. In [RSed3], 133 – 148.

[Ni] T.Nishida, Quasi-deterministic 0L systems. *Lecture Notes in Computer Science* 623 (1992) 65 – 76.

[NiS] T.Nishida and A.Salomaa, Slender 0L languages. *Theoret. Comput.Sci.,* to appear.

[OW] Th.Ottman and D.Wood, Simplifications of E0L grammars. In [RSed3], 149 – 166.

[Pa] G.Paun, Parallel communicating grammar systems of L systems. In [RSed3], 405 – 418.

[PaS] G.Paun and A.Salomaa, Decision problems concerning the thinness of D0L languages. *EATCS Bull.* 46 (1992) 171 – 181.

[PS] A.Paz and A.Salomaa, Integral sequential word functions and growth equivalence of Lindenmayer systems. *Inform. and Control* 23 (1973) 313 – 343.

[PK] P.Prusinkiewicz, L.Kari, Subapical bracketed L systems. To appear in Proceedings of the *5th International Workshop on Graph Grammars and their Applications*, Williamsburg, Virginia, 1994.

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

[R1] G.Rozenberg, T0L systems and languages. *Inform. and Control* 23 (1973) 262 – 283.

[R2] G.Rozenberg, Extension of tabled 0L systems and languages. *Internat. J. Comput. Inform. Sci.* 2 (1973) 311 – 334.

[RRS] G.Rozenberg, K.Ruohonen and A.Salomaa, Developmental systems with fragmentation. *Internat. J. Comput. Math.* 5 (1976) 177 – 191.

[RS1] G.Rozenberg and A.Salomaa, *The Mathematical Theory of L Systems.* Academic Press, New York (1980).

[RS2] G.Rozenberg and A.Salomaa, When L was young. In [RSed2], 383 – 392.

[RS3] G.Rozenberg and A.Salomaa, *Cornerstones of Undecidability.* Prentice Hall, New York (1994).

[RSed1] G.Rozenberg and A.Salomaa (eds.), *L systems.* Lecture Notes in Computer Science 15 (1974).

[RSed2] G.Rozenberg and A.Salomaa (eds.), *The Book of L.* Springer-Verlag, Berlin, Heidelberg, New York (1985).

[RSed3] G.Rozenberg and A.Salomaa (eds.), *Lindenmayer Systems.* Springer-Verlag, Berlin, Heidelberg, New York (1992).

[RSed4] G.Rozenberg and A.Salomaa (eds.), *Developments in Language Theory.* World Scientific, Singapore, New Jersey, London, Hong Kong (1994).

[Ru1] K.Ruohonen, Zeros of Z-rational functions and D0L equivalence, *Theoret. Comput. Scci.* 3 (1976) 283 – 292.

[Ru2] K.Ruohonen, The decidability of the F0L-D0L equivalence problem. *Inform. Process. Lett.* 8 (1979) 257 – 261.

[Ru3] K.Ruohonen, The inclusion problem for D0L langugaes. *Elektron. Informationsverarb. Kybernet.* 15 (1979) 535 – 548.

[Ru4] K.Ruohonen, The decidability of the D0L-DT0L equivalence problem. *J. Comput. System Sci.* 22 (1981) 42 – 52.

[S1] A.Salomaa, *Formal Languages.* Academic Press, New York (1973).

[S2] A.Salomaa, Solution of a decision problem concerning unary Lindenmayer systems. *Discrete Math.* 9 (1974) 71 – 77.

[S3] A.Salomaa, On exponential growth in Lindenmayer systems. *Indag. Math.* 35 (1973) 23 – 30.

[S4] A.Salomaa, Comparative decision problems between sequential and parallel rewriting. *Proc. Symp. Uniformly Structured Automata Logic,* Tokyo (1975) 62 – 66.

[S5] A.Salomaa, *Jewels of Formal Language Theory.* Computer Science Press, Rockville (1981).

[S6] A.Salomaa, Simple reductions between D0L language and sequence equivalence problems. *Discrete Appl. Math.* 41 (1993) 271 – 274.

[S7] A.Salomaa, Developmental models for artificial life: basics of L systems. In G.Paun (ed.) Artificial life: Grammatical Models. Black Sea University Press (1995) 22 – 32.

[SS] A.Salomaa and M.Soittola, *Automata-Theoretic Aspects of Formal Power Series.* Springer-Verlag, Berlin and New York (1978).

[SWY] K.Salomaa, D.Wood and S.Yu, Complexity of E0L structural equivalence. *Lecture Notes in Computer Science.* 841 (1994) 587 – 596.

[SY] K.Salomaa and S.Yu, Decidability of structural equivalence of E0L grammars. *Theoret. Comput. Sci.* 82 (1991) 131 – 139.

[SmS] W.Smith, A.Schweitzer, DNA computers in vitro and in vivo. NECI Research Report, Mar.31, 1995.

[So1] M.Soittola, Remarks on D0L growth sequences. *RAIRO* 10 (1976) 23 – 34.

[So2] M.Soittola, Positive rational sequences. *Theoret. Comput. Sci.* 2 (1976) 317 – 322.

[Sz] A.Szilard, Growth Functions of Lindenmayer Systems, Tech. Rep., Comput. Sci, Dep., Univ. of Western Ontario (1971).

[vL] J.van Leeuwen, The membership question for ET0L languages is polynomially complete. *Inform. Process. Lett.* 3 (1975) 138 – 143.

[V] P.Vitany, Structure of growth in Lindenmayer Systems. *Indag. Math.* 35 (1973) 247 – 253.

[Yo1] T.Yokomori, Graph-controlled systems – an extension of 0L systems. In [RSed2], 461 – 471.

[Yo2] T.Yokomori, Inductive inference of 0L languages. In [RSed3], 115 – 132.